INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

DRAFT H.263

(2 May, 1996)

LINE TRANSMISSION OF NON-TELEPHONE SIGNALS

VIDEO CODING FOR LOW BITRATE COMMUNICATION

DRAFT ITU-T Recommendation H.263

---

* Contact Person: Karel Rijkse
tel: +31 70 332 8588
fax: +31 70 332 6477

FOREWORD

The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the International Telecommunication Union. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, established the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

ITU-T Recommendation H.263 was prepared by the ITU-T Study Group 15 (199x-199x) and was approved by the WTSC (Place, Month xx-xx, 199x).

NOTES

CONTENTS

Page

Summary

This Recommendation specifies a coded representation that can be used for compressing the moving picture component of audio-visual services at low bitrates. The basic configuration of the video source coding algorithm is based on ITU-T Recommendation H.261 and is a hybrid of inter-picture prediction to utilize temporal redundancy and transform coding of the remaining signal to reduce spatial redundancy. The source coder can operate on five standardised picture formats: sub-QCIF, QCIF, CIF, 4CIF and 16CIF.

The decoder has motion compensation capability, allowing optional incorporation of this technique in the coder. Half pixel precision is used for the motion compensation, as opposed to Recommendation H.261 where full pixel precision and a loopfilter are used. Variable length coding is used for the symbols to be transmitted.

In addition to the basic video source coding algorithm, four negotiable coding options are included for improved performance: Unrestricted Motion Vectors, Syntax-based Arithmetic Coding, Advanced Prediction and PB-frames. All these options can be used together or separately.

Recommendation H.263

VIDEO CODING FOR LOW BITRATE COMMUNICATION

(Place, 199x)

1. Scope

This Recommendation specifies a coded representation that can be used for compressing the moving picture component of audio-visual services at low bitrates. The basic configuration of the video source coding algorithm is based on ITU-T Recommendation H.261. Four negotiable coding options are included for improved performance.

2. References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

[1] ITU-T Recommendation H.223 (1995): "Multiplexing protocol for low bitrate multimedia communication"

[2] ITU-T Recommendation H.242 (1993): "System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s"

[3] ITU-T Recommendation H.245 (1995): "Control protocol for multimedia communication"

[4] ITU-T Recommendation H.261 (1993): "Video codec for audiovisual services at px64 kbit/s"

[5] ITU-T Recommendation H.262 (1995): "Generic coding of moving pictures and associated audio: video" (ISO/IEC 13818-2)

[6] ITU-T Recommendation H.320 (1993): "Narrow-band ISDN visual telephone systems and terminal equipment"

[7] ITU-T Recommendation H.324 (1995): "Terminal for low bitrate multimedia communication"

3. Brief Specification

An outline block diagram of the codec is given in FIGURE 1/H.263.

3.1. Video input and output

To permit a single Recommendation to cover use in and between regions using 625- and 525-line television standards, the source coder operates on pictures based on a common intermediate format (CIF). The standards of the input and output television signals, which may, for example, be composite or component, analogue or digital and the methods of performing any necessary conversion to and from the source coding format are not subject to recommendation.

3.2. Digital output and input

The video coder provides a self-contained digital bit stream which may be combined with other multi-facility signals (for example as defined in Recommendation H.223). The video decoder performs the reverse process.

3.3. Sampling frequency

Pictures are sampled at an integer multiple of the video line rate. This sampling clock and the digital network clock are asynchronous.

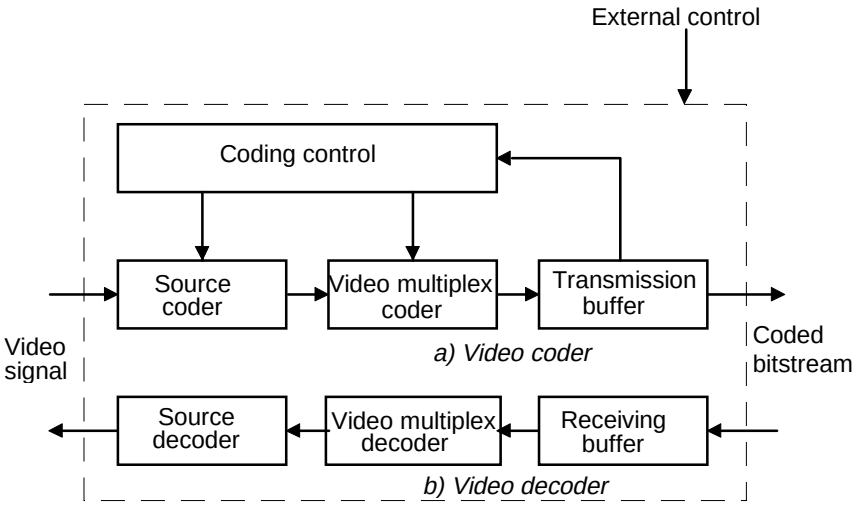


FIGURE 1/H.263

Outline block diagram of the video codec

3.4. Source coding algorithm

A hybrid of inter-picture prediction to utilize temporal redundancy and transform coding of the remaining signal to reduce spatial redundancy is adopted. The decoder has motion compensation capability, allowing optional incorporation of this technique in the coder. Half pixel precision is used for the motion compensation, as opposed to Recommendation H.261 where full pixel precision and a loopfilter are used. Variable length coding is used for the symbols to be transmitted.

In addition to the core H.263 coding algorithm, four negotiable coding options are included that will be described in the subsequent subsections. All these options can be used together or separately.

3.4.1. Unrestricted Motion Vector mode

In this optional mode motion vectors are allowed to point outside the picture. The edge pixels are used as prediction for the "not existing" pixels. With this mode a significant gain is achieved if there is movement across the edges of the picture, especially for the smaller picture formats (see also Annex D). Additionally, this mode includes an extension of the motion vector range so that larger motion vectors can be used. This is especially useful in case of camera movement.

3.4.2. Syntax-based Arithmetic Coding mode

In this optional mode arithmetic coding is used instead of variable length coding. The SNR and reconstructed pictures will be the same, but significantly fewer bits will be produced (see also Annex E).

3.4.3. Advanced Prediction mode

In this optional mode overlapped block motion compensation (OBMC) is used for the luminance part of P-pictures (see also Annex F). Four 8x8 vectors instead of one 16x16 vector are used for some of the macroblocks in the picture. The encoder has to decide which type of vectors to use. Four vectors use more bits, but give better prediction. The use of this mode generally gives a considerable improvement. Especially a subjective gain is achieved because OBMC results in less blocking artifacts.

3.4.4. PB-frames mode

A PB-frame consists of two pictures being coded as one unit. The name PB comes from the name of picture types in Recommendation H.262 where there are P-pictures and B-pictures. Thus a PB-frame consists of one P-picture which is predicted from the previous decoded P-picture and one B-picture which is predicted from both the previous decoded P-picture and the P-picture currently being decoded. The name B-picture was chosen because parts of B-pictures may be bidirectionally predicted from the past and future pictures. With this coding option, the picture rate can be increased considerably without increasing the bitrate much.

3.5. Bit rate

The transmission clock is provided externally. The video bitrate may be variable. In this Recommendation no constraints on the video bitrate are given; constraints will be given by the terminal or the network.

3.6. Buffering

The encoder shall control its output bitstream to comply with the requirements of the hypothetical reference decoder defined in Annex B. Video data shall be provided on every valid clock cycle. This can be ensured by MCBPC stuffing (see TABLE 4/H.263 and TABLE 5/H.263) or, when forward error correction is used, also by forward error correction stuffing frames (see Annex H).

The number of bits created by coding any single picture shall not exceed a maximum value specified by the parameter BPPmaxKb which is measured in units of 1024 bits. The minimum allowable value of the BPPmaxKb parameter depends on the largest source picture format that has been negotiated for use in the bitstream (see TABLE 1/H.263). An encoder may use a larger value for BPPmaxKb than as specified in TABLE 1/H.263, provided the larger value is first negotiated by external means, for example Recommendation H.245.

TABLE 1/H.263

BPPmaxKb for each of the source picture formats

Source format	BPPmaxKb
sub-QCIF	64
QCIF	64
CIF	256
4CIF	512
16CIF	1024

3.7. Symmetry of transmission

The codec may be used for bidirectional or unidirectional visual communication.

3.8. Error handling

Error handling should be provided by external means (for example Recommendation H.223). If it is not provided by external means (for example in Recommendation H.221) the optional error correction code and framing as described in Annex H can be used.

A decoder can send a command to encode one or more GOBs of its next picture in INTRA mode with coding parameters such as to avoid buffer overflow. A decoder can also send a command to transmit only non-empty GOB headers. The transmission method for these signals is by external means (for example Recommendation H.245).

3.9. Multipoint operation

Features necessary to support switched multipoint operation are included in Annex C.

4. Source Coder

4.1. Source format

The source coder operates on non-interlaced pictures occurring 30 000/1001 (approximately 29.97) times per second. The tolerance on picture frequency is ± 50 ppm.

Pictures are coded as luminance and two colour difference components (Y, C_B and C_R). These components and the codes representing their sampled values are as defined in CCIR Recommendation 601.

Black = 16

White = 235

Zero colour difference = 128

Peak colour difference = 16 and 240.

These values are nominal ones and the coding algorithm functions with input values of 1 through to 254.

There are five standardised picture formats: sub-QCIF, QCIF, CIF, 4CIF and 16CIF. For each of these picture formats, the luminance sampling structure is dx pixels per line, dy lines per picture in an orthogonal arrangement. Sampling of each of the two colour difference components is at dx/2 pixels per line, dy/2 lines per picture, orthogonal. The values of dx, dy, dx/2 and dy/2 are given in TABLE 2/H.263 for each of the picture formats.

TABLE 2/H.263

Number of pixels per line and number of lines for each of the H.263 picture formats

Picture Format	number of pixels for luminance (dx)	number of lines for luminance (dy)	number of pixels for chrominance (dx/2)	number of lines for chrominance (dy/2)
sub-QCIF	128	96	64	48
QCIF	176	144	88	72
CIF	352	288	176	144
4CIF	704	576	352	288
16CIF	1408	1152	704	576

For each of the picture formats, colour difference samples are sited such that their block boundaries coincide with luminance block boundaries as shown in FIGURE 2/H.263. The pixel aspect ratio is the same for each of these picture formats and is the same as defined for QCIF and CIF in Recommendation H.261: (4/3)*(288/352). The picture area covered by all picture formats except the sub-QCIF picture format has an aspect ratio of 4:3.

All decoders shall be able to operate using sub-QCIF. All decoders shall also be able to operate using QCIF. Some decoders may also operate with CIF, 4CIF or 16CIF. Encoders shall be able to operate with one of the formats sub-QCIF and QCIF. The encoders determine which of these two formats are used, and are not obliged to be able to operate with both. Some encoders can also operate with CIF, 4CIF or 16CIF. Which formats can be handled by the decoder is signalled by external means, for example Recommendation H.245. For a complete overview of possible picture formats and video coding algorithms refer to the terminal description, for example Recommendation H.324.

Note – For CIF, the number of pixels per line is compatible with sampling the active portions of the luminance and colour difference signals from 525- or 625-line sources at 6.75 and 3.375 MHz respectively. These frequencies have a simple relationship to those in CCIR Recommendation 601.

Means shall be provided to restrict the maximum picture rate of encoders by having a minimum number of non-transmitted pictures between transmitted ones. Selection of this minimum number shall be by external means (for example Recommendation H.245). For the calculation of the minimum number of non-transmitted pictures in PB-frames mode, the P-picture and the B-picture of a PB-frames unit are taken as two separate pictures.

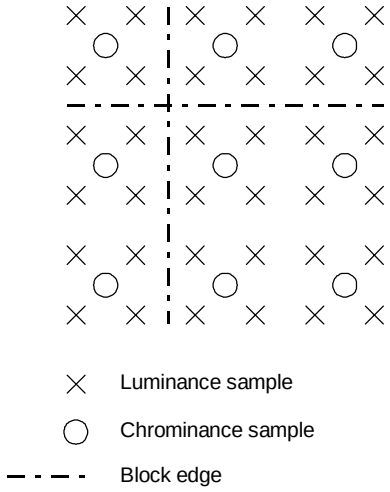
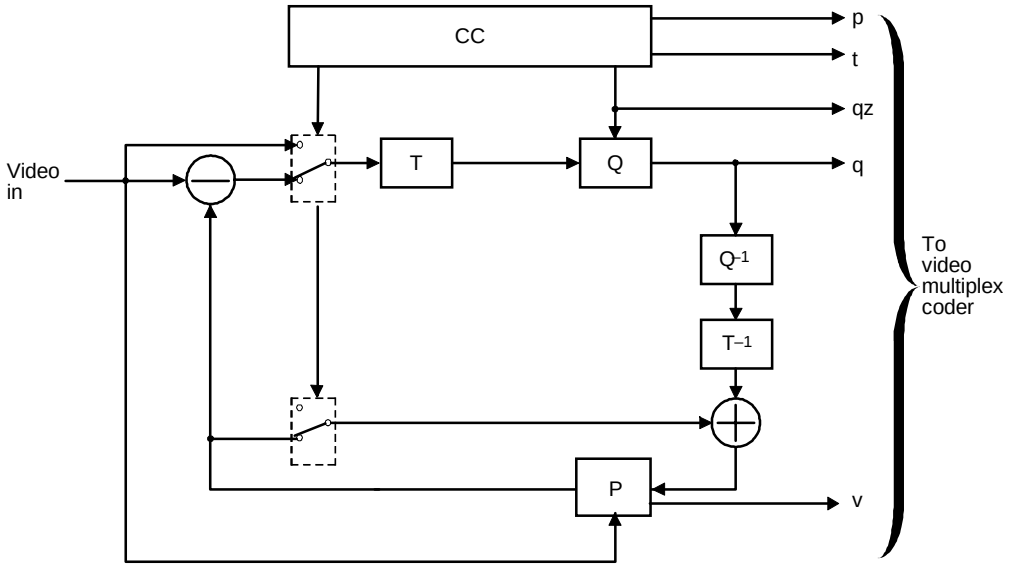


FIGURE 2/H.263

Positioning of luminance and chrominance samples

4.2. Video source coding algorithm

The source coder is shown in generalized form in FIGURE 3/H.263. The main elements are prediction, block transformation and quantization.



T Transform
Q Quantizer
P Picture Memory with motion compensated variable delay

CC Coding control
p Flag for INTRA/INTER
t Flag for transmitted or not
qz Quantizer indication
q Quantizing index for transform coefficients
v Motion vector

FIGURE 3/H.263

Source coder

4.2.1. GOBs, macroblocks and blocks

Each picture is divided into groups of blocks (GOBs). A group of blocks (GOB) comprises of k*16 lines, depending on the picture format (k = 1 for sub-QCIF, QCIF and CIF; k = 2 for 4CIF; k = 4 for 16CIF). The number of GOBs per picture is 6 for sub-QCIF, 9 for QCIF, and 18 for CIF, 4CIF and 16CIF. The GOB numbering is done by use of vertical scan of the GOBs, starting with the upper GOB (number 0) and ending with the lower GOB. An example of the arrangement of GOBs in a picture is given for the CIF picture format in FIGURE 4/H.263. Data for each GOB consists of a GOB header (may be empty) followed by data for macroblocks. Data for GOBs is transmitted per GOB in increasing GOB number.

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

FIGURE 4/H.263

Arrangement of Group of Blocks in a CIF picture

Each GOB is divided into macroblocks. A macroblock relates to 16 pixels by 16 lines of Y and the spatially corresponding 8 pixels by 8 lines of C_B and C_R. Further, a macroblock consists of four luminance blocks and the two spatially corresponding colour difference blocks as shown in FIGURE 5/H.263. Each luminance or chrominance block relates to 8 pixels by 8 lines of Y, C_B or C_R. A GOB comprises one macroblock row for sub-QCIF, QCIF and CIF, two macroblock rows for 4CIF and four macroblock rows for 16CIF.

The macroblock numbering is done by use of horizontal scan of the macroblock rows from left to right, starting with the upper macroblock row and ending with the lower macroblock row. Data for the macroblocks is transmitted per macroblock in increasing macroblock number. Data for the blocks is transmitted per block in increasing block number (see FIGURE 5/H.263).

The criteria for choice of mode and transmitting a block are not subject to recommendation and may be varied dynamically as part of the coding control strategy. Transmitted blocks are transformed and resulting coefficients are quantized and entropy coded.

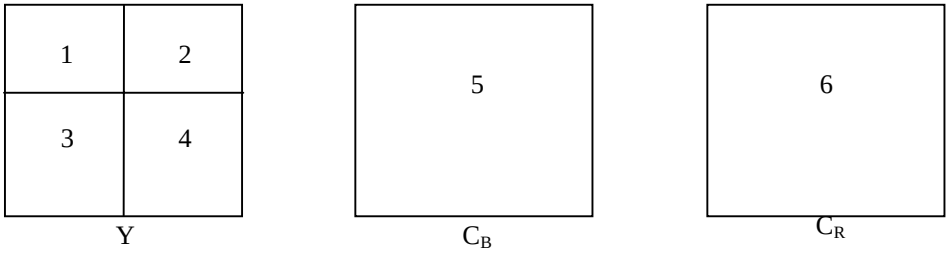


FIGURE 5/H.263

Arrangement of blocks in a macroblock

4.2.2. Prediction

The prediction is inter-picture and may be augmented by motion compensation (see § 4.2.3). The coding mode in which prediction is applied is called INTER; the coding mode is called INTRA if no prediction is applied. The INTRA coding mode can be signalled at the picture level (INTRA for I-pictures or INTER for P-pictures) or at the macroblock level in P-pictures. In the optional PB-frames mode B-pictures are always coded in INTER mode. The B-pictures are partly predicted bidirectionally (refer to Annex G).

4.2.3. Motion compensation

The decoder will accept one vector per macroblock or if the Advanced Prediction mode is used one or four vectors per macroblock (see Annex F). If the PB-frames mode is used, one additional delta vector can be transmitted per macroblock for adaptation of the motion vectors for prediction of the B-macroblock.

Both horizontal and vertical components of the motion vectors have integer or half integer values. In the default prediction mode, these values are restricted to the range [-16,15.5] (this is also valid for the forward and backward motion vector components for B-pictures). In the Unrestricted Motion Vector mode however, the maximum range for vector components is [-31.5,31.5], with the restriction that only values that are within a range of [-16,15.5] around the predictor for each motion vector component can be reached if the predictor is in the range [-15.5,16]. If the predictor is outside [-15.5,16], all values within the range [-31.5,31.5] with the same sign as the predictor plus the zero value can be reached (see also Annex D).

A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pixels in the referenced picture which are spatially to the right or below the pixels being predicted.

Motion vectors are restricted such that all pixels referenced by them are within the coded picture area, except when the Unrestricted Motion Vector mode and/or the Advanced Prediction mode is used (see Annex D and Annex F).

4.2.4. Quantization

The number of quantizers is 1 for the first coefficient of INTRA blocks and 31 for all other coefficients. Within a macroblock the same quantizer is used for all coefficients except the first one of INTRA blocks. The decision levels are not defined. The first coefficient of INTRA blocks is nominally the transform dc value uniformly quantized with a step size of 8. Each of the other 31 quantizers use equally spaced reconstruction levels with a central dead-zone around zero and with a step size of an even value in the range 2 to 62. For the exact formulas refer to section 6.2.

Note – For the smaller quantization step sizes, the full dynamic range of the transform coefficients cannot be represented.

4.3. Coding control

Several parameters may be varied to control the rate of generation of coded video data. These include processing prior to the source coder, the quantizer, block significance criterion and temporal subsampling. The proportions of such measures in the overall control strategy are not subject to recommendation.

When invoked, temporal subsampling is performed by discarding complete pictures.

A decoder can signal its preference for a certain tradeoff between spatial and temporal resolution of the video signal. The encoder shall signal its default tradeoff at the beginning of the call and shall indicate whether it is capable to respond to decoder requests to change this tradeoff. The transmission method for these signals is by external means (for example Recommendation H.245).

4.4. Forced updating

This function is achieved by forcing the use of the INTRA mode of the coding algorithm. The update pattern is not defined. To control the accumulation of inverse transform mismatch error, each macroblock shall be coded in INTRA mode at least once every 132 times when coefficients are transmitted for this macroblock in P-pictures.

4.5. Byte alignment of start codes

Byte alignment of start codes is achieved by inserting a stuffing codeword consisting of less than 8 zero-bits before the start code such that the first bit of the start code is the first (most significant) bit of a byte. A start code is therefore byte aligned if the position of its most significant bit is a multiple of 8-bits from the first bit in the H.263 bitstream. All picture start codes shall and GOB and EOS start codes may be byte aligned.

Note 1 – The number of bits spent for a certain picture is variable but always a multiple of 8 bits.

Note 2 – H.324 requires H.263 encoders to align picture start codes with the start of logical information units passed to the Adaptation Layer (AL_SDU's).

5. Syntax and Semantics

The video multiplex is arranged in a hierarchical structure with four layers. From top to bottom the layers are:

- Picture,
- Group of Blocks,
- Macroblock,
- Block.

The syntax diagram is shown in FIGURE 6/H.263. Abbreviations and semantics are defined in later sections.

Unless specified otherwise the most significant bit is transmitted first. This is bit 1 and is the left most bit in the code tables in this Recommendation. Unless specified otherwise all unused or spare bits are set to "1". Spare bits shall not be used until their functions are specified by the ITU.

5.1. Picture Layer

Data for each picture consists of a picture header followed by data for Group of Blocks, eventually followed by an end-of-sequence code and stuffing bits. The structure is shown in FIGURE 7/H.263. PSBI is only present if indicated by CPM. TRB and DBQUANT are only present if PTYPE indicates 'PB-frame'. Combinations of PSPARE and PEI may not be present.

EOS may not be present, while ESTUF may be present only if EOS is present. Picture headers for dropped pictures are not transmitted.

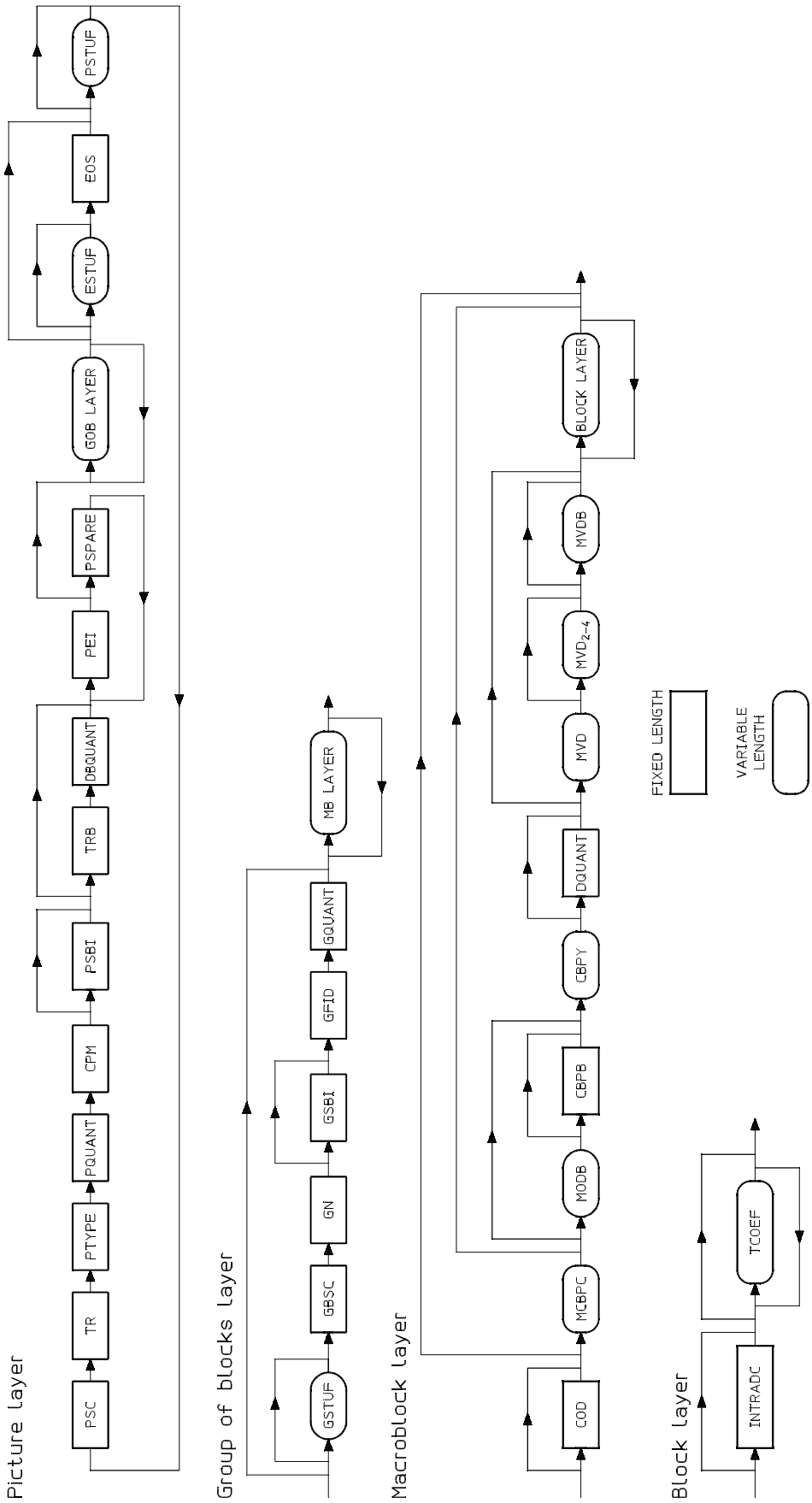

Picture layer
PSC
TR
PTYPE
PQUANT
CPM
PSBI
TRB
DBQUANT
PEI
PSPARE
GOB LAYER
ESTUF
EOS
PSTUF
Group of blocks layer
GSTUF
GBSC
GN
GSBI
GFID
GQUANT
MB LAYER
Macroblock layer
COD
MCBPC
MODB
CBPB
CBPY
DQUANT
MVD
MVD2-4
MVDB
BLOCK LAYER
FIXED LENGTH
VARIABLE LENGTH
Block layer
INTRADC
TCOEF

FIGURE 6/H.263

Syntax diagram for the video bitstream

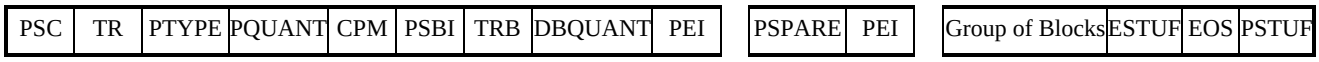


FIGURE 7/H.263

Structure of picture layer

5.1.1. Picture Start Code (PSC) (22 bits)

PSC is a word of 22 bits. Its value is 0000 0000 0000 0000 1 00000. All picture start codes shall be byte aligned. This shall be achieved by inserting PSTUF before the start code such that the first bit of the start code is the first (most significant) bit of a byte.

5.1.2. Temporal Reference (TR) (8 bits)

An 8-bit number which can have 256 possible values. It is formed by incrementing its value in the previously transmitted picture header by one plus the number of non-transmitted pictures (at 29.97 Hz) since the previously transmitted one. The arithmetic is performed with only the eight LSBs. In the optional PB-frames mode, TR only addresses P-pictures; for the temporal reference for B-pictures refer to section 5.1.7.

5.1.3. Type Information (PTYPE) (13 bits)

Information about the complete picture:

Bit 1	Always “1”, in order to avoid start code emulation.
Bit 2	Always “0”, for distinction with H.261,
Bit 3	Split screen indicator, “0” off, “1” on,
Bit 4	Document camera indicator, “0” off, “1” on,
Bit 5	Freeze Picture Release, “0” off, “1” on,
Bit 6-8	Source Format, “000” forbidden, “001” sub-QCIF, “010” QCIF, “011” CIF, “100” 4CIF, “101” 16CIF, “110” reserved, “111” reserved,
Bit 9	Picture Coding Type, “0” INTRA (I-picture), “1” INTER (P-picture),
Bit 10	Optional Unrestricted Motion Vector mode, “0” off, “1” on,
Bit 11	Optional Syntax-based Arithmetic Coding mode, “0” off, “1” on,
Bit 12	Optional Advanced Prediction mode, “0” off, “1” on,
Bit 13	Optional PB-frames mode, “0” normal I- or P-picture, “1” PB-frame.

Split screen indicator is a signal that indicates that the upper and lower half of the decoded picture could be displayed side by side. This bit has no direct effect on the encoding or decoding of the picture.

Freeze Picture Release is a signal from an encoder which responds to a request for packet retransmission (if not acknowledged) or fast update request (see also Annex C) and allows a decoder to exit from its freeze picture mode and display decoded picture in the normal manner.

If bit 6-8 indicate a different source format than in the previous picture header, the current picture shall be an I-picture.

Bit 10-13 refer to optional modes that are only used after negotiation between encoder and decoder (see also the Annexes D, E, F and G, respectively). If bit 9 is set to “0”, bit 13 shall be set to “0” as well.

5.1.4. Quantizer Information (PQUANT) (5 bits)

A fixed length codeword of 5 bits which indicates the quantizer QUANT to be used for the picture until updated by any subsequent GQUANT or DQUANT. The codewords are the natural binary representations of the values of QUANT which, being half the step sizes, range from 1 to 31.

5.1.5. Continuous Presence Multipoint (CPM) (1 bit)

A codeword of 1 bit that signals the use of the optional Continuous Presence Multipoint mode (CPM); “0” is off, “1” is on. For the use of CPM refer to Annex C.

5.1.6. Picture Sub Bitstream Indicator (PSBI) (2 bits)

A fixed length codeword of 2 bits that is only present if Continuous Presence Multipoint mode is indicated by CPM. The codewords are the natural binary representation of the sub-bitstream number for the picture header and all following information until the next Picture or GOB start code (see also Annex C).

5.1.7 Temporal Reference for B-pictures (TR_B) (3 bits)

TR_B is present if PTYPE indicates ‘PB-frame’ (see also Annex G) and indicates the number of non-transmitted pictures (at 29.97 Hz) since the last P- or I-picture and before the B-picture. The codeword is the natural binary representation of the number of non-transmitted pictures plus one. The maximum number of non-transmitted pictures is 6.

5.1.8. Quantization information for B-pictures (DBQUANT) (2 bits)

DBQUANT is present if PTYPE indicates ‘PB-frame’ (see also Annex G). In the decoding process a quantization parameter QUANT is obtained for each macroblock. With PB-frames QUANT is used for the P-block, while for the B-block a different quantization parameter BQUANT is used. QUANT ranges from 1 to 31. DBQUANT indicates the relation between QUANT and BQUANT as defined in TABLE 3/H.263. In this table, “/” means division by truncation. BQUANT ranges from 1 to 31; if the value for BQUANT resulting from TABLE 3/H.263 is greater than 31, it is clipped to 31.

TABLE 3/H.263

DBQUANT codes and relation between QUANT and BQUANT

DBQUANT	BQUANT
00	(5xQUANT)/4
01	(6xQUANT)/4
10	(7xQUANT)/4
11	(8xQUANT)/4

5.1.9. Extra Insertion Information (PEI) (1 bit)

A bit which when set to "1" signals the presence of the following optional data field.

5.1.10. Spare Information (PSPARE) (0/8/16 . . . bits)

If PEI is set to "1", then 9 bits follow consisting of 8 bits of data (PSPARE) and then another PEI bit to indicate if a further 9 bits follow and so on. Encoders shall not insert PSPARE until specified by the ITU. Decoders shall be designed to discard PSPARE if PEI is set to 1. This will allow the ITU to specify future backward compatible additions in PSPARE. If PSPARE is followed by PEI=0, PSPARE=xx000000 is prohibited in order to avoid start code emulation (x=don't care, so 4 out of 256 values are prohibited).

5.1.11 Stuffing (ESTUF) (Variable length)

A codeword of variable length consisting of less than 8 zero-bits. Encoders may insert this codeword directly before an EOS codeword. If ESTUF is present, the last bit of ESTUF shall be the last (least significant) bit of a byte, so that the start of the EOS codeword is byte aligned. Decoders shall be designed to discard ESTUF.

5.1.12. End Of Sequence (EOS) (22 bits)

A codeword of 22 bits. Its value is 0000 0000 0000 0000 1 11111. It is up to the encoder to insert this codeword or not. EOS may be byte aligned. This can be achieved by inserting ESTUF before the start code such that the first bit of the start code is the first (most significant) bit of a byte.

5.1.13. Stuffing (PSTUF) (Variable length)

A codeword of variable length consisting of less than 8 zero-bits. Encoders shall insert this codeword for byte alignment of the next PSC. The last bit of PSTUF shall be the last (least significant) bit of a byte, so that the video bitstream including PSTUF is a multiple of 8 bits from the first bit in the H.263 bitstream. Decoders shall be designed to discard PSTUF.

If for some reason the encoder stops encoding pictures for a certain time period and resumes encoding later, PSTUF shall be transmitted before the encoder stops, to prevent that the last up to 7 bits of the previous picture are not sent until the coder resumes coding.

5.2. Group of Blocks Layer

Data for each Group of Blocks (GOB) consists of a GOB header followed by data for macroblocks. The structure is shown in FIGURE 8/H.263. Each GOB contains one or more rows of macroblocks. For the first GOB in each picture (with number 0), no GOB header shall be transmitted. For all other GOBs, the GOB header may be empty, depending on the encoder strategy. A decoder can signal the remote encoder to transmit only non-empty GOB headers by external means, for example Recommendation H.245. GSTUF may be present when GBSC is present. GN, GFID and GQUANT are present when GBSC is present. GSBI is present when Continuous Presence Multipoint mode is on, as indicated in the Picture header.

FIGURE 8/H.263

Structure of GOB layer

5.2.1. Stuffing (GSTUF) (Variable length)

A codeword of variable length consisting of less than 8 zero-bits. Encoders may insert this codeword directly before an GBSC codeword. If GSTUF is present, the last bit of GSTUF shall be the last (least significant) bit of a byte, so that the start of the GBSC codeword is byte aligned. Decoders shall be designed to discard GSTUF.

5.2.2. Group of Block Start Code (GBSC) (17 bits)

A word of 17 bits. Its value is 0000 0000 0000 0000 1. GOB start codes may be byte aligned. This can be achieved by inserting GSTUF before the start code such that the first bit of the start code is the first (most significant) bit of a byte.

5.2.3. Group Number (GN) (5 bits)

A fixed length codeword of 5 bits. The bits are the binary representation of the number of the Group of Blocks. For the GOB with number 0, the GOB header including GSTUF, GBSC, GN, GSBI, GFID and GQUANT is empty; group number 0 is used in the PSC. Group number 31 is used in the EOS and the values from 18 to 30 are reserved for future use by the ITU.

5.2.4. GOB Sub Bitstream Indicator (GSBI) (2 bits)

A fixed length codeword of 2 bits that is only present if Continuous Presence Multipoint mode is indicated by CPM. The codewords are the natural binary representation of the sub-bitstream number for the GOB header and all following information until the next Picture or GOB start code (see also Annex C).

5.2.5. GOB Frame ID (GFID) (2 bits)

A fixed length codeword of 2 bits. GFID shall have the same value in every GOB header of a given picture. Moreover, if PTYPE as indicated in a picture header is the same as for the previous transmitted picture, GFID shall have the same value as in that previous picture. However, if PTYPE in a certain picture header differs from the PTYPE in the previous transmitted picture header, the value for GFID in that picture shall differ from the value in the previous picture.

5.2.6. Quantizer Information (GQUANT) (5 bits)

A fixed length codeword of 5 bits which indicates the quantizer QUANT to be used for ~~that GOB~~ the remaining part of the picture until updated by any subsequent GQUANT or DQUANT. The codewords are the natural binary representations of the values of QUANT which, being half the step sizes, range from 1 to 31.

5.3. Macroblock Layer

Data for each macroblock consists of a macroblock header followed by data for blocks. The structure is shown in FIGURE 9/H.263. COD is only present in pictures for which PTYPE indicates 'INTER', for each macroblock in these pictures. MCBPC is present when indicated by COD or when PTYPE indicates 'INTRA'. MODB is present for MB-type 0-4 if PTYPE indicates 'PB-frame'. CBPY, DQUANT, MVD and MVD_{2-4} are present when indicated by MCBPC. CBPB and MVDB are only present if indicated by MODB. Block Data is present when indicated by MCBPC and CBPY. MVD_{2-4} are

only present in Advanced Prediction mode (refer to Annex F). MODB, CBPB and MVDB are only present in PB-frames mode (refer to Annex G). For coding of the symbols in the Syntax-based Arithmetic Coding mode refer to Annex E.

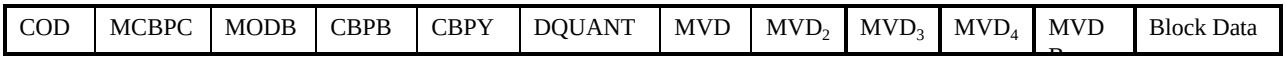


FIGURE 9/H.263

Structure of macroblock layer

5.3.1. Coded macroblock indication (COD) (1 bit)

A bit which when set to "0" signals that the macroblock is coded. If set to "1", no further information is transmitted for this macroblock; in that case the decoder shall treat the macroblock as an INTER macroblock with motion vector for the whole block equal to zero and with no coefficient data. COD is only present in pictures for which PTYPE indicates 'INTER', for each macroblock in these pictures.

Note – In Advanced Prediction mode, overlapped motion compensation is also performed if COD is set to "1".

5.3.2. Macroblock type & Coded block pattern for chrominance (MCBPC) (Variable length)

A variable length codeword giving information about the macroblock type and the coded block pattern for chrominance. The codewords for MCBPC are given in TABLE 4/H.263 and TABLE 5/H.263. MCBPC is always included in coded macroblocks.

An extra codeword is available in the tables for bit stuffing. This codeword should be discarded by decoders.

The macroblock type gives information about the macroblock and which data elements are present. Macroblock types and included elements are listed in TABLE 6/H.263 and TABLE 7/H.263.

TABLE 4/H.263

VLC table for MCBPC (for I-pictures)

Index	MB type	CBPC (56)	Number of bits	Code
0	3	00	1	1
1	3	01	3	001
2	3	10	3	010
3	3	11	3	011
4	4	00	4	0001
5	4	01	6	0000 01
6	4	10	6	0000 10
7	4	11	6	0000 11
8	Stuffing	--	9	0000 0000 1

The coded block pattern for chrominance signifies C_B and/or C_R blocks when at least one non-INTRADC transform coefficient is transmitted (INTRADC is the dc-coefficient for INTRA blocks, see section 5.4.1). $CBPC_N$ = 1 if any non-INTRADC coefficient is present for block N, else 0, for $CBPC_5$ and $CBPC_6$ in the coded block pattern. Block numbering is given in FIGURE 5/H.263. When MCBPC=Stuffing, the remaining part of the macroblock layer is skipped. In this case, the preceeding COD=0 is not related to any coded or not-coded macroblock and therefore the macroblock number is not incremented. For P-pictures, multiple stuffings are accomplished by multiple sets of COD=0 and MCBPC=Stuffing.

TABLE 5/H.263

VLC table for MCBPC (for P-pictures)

Index	MB type	CBPC (56)	Number of bits	Code
0	0	00	1	1
1	0	01	4	0011
2	0	10	4	0010
3	0	11	6	0001 01
4	1	00	3	011
5	1	01	7	0000 111
6	1	10	7	0000 110
7	1	11	9	0000 0010 1
8	2	00	3	010
9	2	01	7	0000 101
10	2	10	7	0000 100
11	2	11	8	0000 0101
12	3	00	5	0001 1
13	3	01	8	0000 0100
14	3	10	8	0000 0011

15	3	11	7	0000 011
16	4	00	6	0001 00
17	4	01	9	0000 0010 0
18	4	10	9	0000 0001 1
19	4	11	9	0000 0001 0
20	Stuffing	--	9	0000 0000 1

TABLE 6/H.263

Macroblock types and included data elements for normal pictures

Picture type	MB type	Name	COD	MCBPC	CBPY	DQUANT	MVD	MVD_{2-4}
INTER	not coded	-	X					
INTER	0	INTER	X	X	X		X	
INTER	1	INTER+Q	X	X	X	X	X	
INTER	2	INTER4V	X	X	X		X	X
INTER	3	INTRA	X	X	X			
INTER	4	INTRA+Q	X	X	X	X		
INTER	stuffing	-	X	X				
INTRA	3	INTRA		X	X			
INTRA	4	INTRA+Q		X	X	X		
INTRA	stuffing	-		X				

Note: “x” means that the item is present in the macroblock

TABLE 7/H.263

Macroblock types and included data elements for PB-frames

Picture type	MB type	Name	COD	MCBPC	MODB	CBPY	CBPB	DQUANT	MVD	MVDB	MVD_{2-4}
INTER	not coded	-	X								
INTER	0	INTER	X	X	X	X	(X)		X	(X)	
INTER	1	INTER+Q	X	X	X	X	(X)	X	X	(X)	
INTER	2	INTER4V	X	X	X	X	(X)		X	(X)	X
INTER	3	INTRA	X	X	X	X	(X)		X	(X)	
INTER	4	INTRA+Q	X	X	X	X	(X)	X	X	(X)	
INTER	stuffing	-	X	X							

Note 1: “x” means that the item is present in the macroblock

Note 2: CBPB and MVDB are only present if indicated by MODB

Note 3: B-blocks are always coded in INTER mode, even if the MB type of the PB-macroblock indicates INTRA

5.3.3 Macroblock mode for B-blocks (MODB) (Variable length)

MODB is present for MB-type 0-4 if PTYPE indicates ‘PB-frame’ and is a variable length codeword indicating whether CBPB is present (indicates that B-coefficients are transmitted for this macroblock) and/or MVDB is present. In TABLE 8/H.263 the codewords for MODB are defined.

TABLE 8/H.263

VLC table for MODB

Index	CBPB	MVDB	Number of bits	Code
0			1	0
1		X	2	10
2	X	X	2	11

Note: “x” means that the item is present in the macroblock

5.3.4. Coded block pattern for B-blocks (CBPB) (6 bits)

CBPB is only present in PB-frames mode if indicated by MODB. $CBPB_N$ = 1 if any coefficient is present for B-block N, else 0, for each bit $CBPB_N$ in the coded block pattern. Block numbering is given in FIGURE 5/H.263, the utmost left bit of CBPB corresponding with block number 1.

5.3.5. Coded block pattern for luminance (CBPY) (Variable length)

Variable length codeword giving a pattern number signifying those Y blocks in the macroblock for which at least one non-INTRADC transform coefficient is transmitted (INTRADC is the dc-coefficient for INTRA blocks, see § 5.4.1).

$CBPY_N$ = 1 if any non-INTRADC coefficient is present for block N, else 0, for each bit $CBPY_N$ in the coded block pattern. Block numbering is given in FIGURE 5/H.263, the utmost left bit of CBPY corresponding with block number 1. For a certain pattern $CBPY_N$, different codewords are used for INTER and INTRA macroblocks as defined in TABLE 10/H.263.

5.3.6. Quantizer Information (DQUANT) (2 bits)

A two bit code to define change in QUANT. In TABLE 9/H.263 the differential values for the different codewords are given. QUANT ranges from 1 to 31; if the value for QUANT after adding the differential value is less than 1 or greater than 31, it is clipped to 1 and 31 respectively.

TABLE 9/H.263

DQUANT codes and differential values for QUANT

Index	Differential value	DQUANT
0	-1	00
1	-2	01
2	1	10
3	2	11

TABLE 10/H.263

VLC table for CBPY

Index	CBPY(INTRA) (1234)	CBPY(INTER) (1234)	Number of bits	Code
0	0000	1111	4	0011
1	0001	1110	5	0010 1
2	0010	1101	5	0010 0
3	0011	1100	4	1001
4	0100	1011	5	0001 1
5	0101	1010	4	0111
6	0110	1001	6	0000 10
7	0111	1000	4	1011
8	1000	0111	5	0001 0
9	1001	0110	6	0000 11
10	1010	0101	4	0101
11	1011	0100	4	1010
12	1100	0011	4	0100
13	1101	0010	4	1000
14	1110	0001	4	0110
15	1111	0000	2	11

5.3.7. Motion vector data (MVD) (Variable length)

MVD is included for all INTER macroblocks (in PB-frames mode also for INTRA macroblocks) and consists of a variable length codeword for the horizontal component followed by a variable length codeword for the vertical component. Variable length codes are given in TABLE 11/H.263.

TABLE 11/H.263

VLC table for MVD

Index	Vector differences		Bit number	Codes
0	-16	16	13	0000 0000 0010 1
1	-15.5	16.5	13	0000 0000 0011 1
2	-15	17	12	0000 0000 0101
3	-14.5	17.5	12	0000 0000 0111
4	-14	18	12	0000 0000 1001
5	-13.5	18.5	12	0000 0000 1011
6	-13	19	12	0000 0000 1101
7	-12.5	19.5	12	0000 0000 1111
8	-12	20	11	0000 0001 001
9	-11.5	20.5	11	0000 0001 011
10	-11	21	11	0000 0001 101
11	-10.5	21.5	11	0000 0001 111
12	-10	22	11	0000 0010 001
13	-9.5	22.5	11	0000 0010 011
14	-9	23	11	0000 0010 101
15	-8.5	23.5	11	0000 0010 111
16	-8	24	11	0000 0011 001
17	-7.5	24.5	11	0000 0011 011
18	-7	25	11	0000 0011 101
19	-6.5	25.5	11	0000 0011 111
20	-6	26	11	0000 0100 001
21	-5.5	26.5	11	0000 0100 011
22	-5	27	10	0000 0100 11
23	-4.5	27.5	10	0000 0101 01
24	-4	28	10	0000 0101 11
25	-3.5	28.5	8	0000 0111
26	-3	29	8	0000 1001
27	-2.5	29.5	8	0000 1011
28	-2	30	7	0000 111
29	-1.5	30.5	5	0001 1
30	-1	31	4	0011
31	-0.5	31.5	3	011
32	0		1	1
33	0.5	-31.5	3	010
34	1	-31	4	0010
35	1.5	-30.5	5	0001 0
36	2	-30	7	0000 110
37	2.5	-29.5	8	0000 1010
38	3	-29	8	0000 1000
39	3.5	-28.5	8	0000 0110
40	4	-28	10	0000 0101 10
41	4.5	-27.5	10	0000 0101 00
42	5	-27	10	0000 0100 10
43	5.5	-26.5	11	0000 0100 010
44	6	-26	11	0000 0100 000
45	6.5	-25.5	11	0000 0011 110
46	7	-25	11	0000 0011 100
47	7.5	-24.5	11	0000 0011 010
48	8	-24	11	0000 0011 000
49	8.5	-23.5	11	0000 0010 110
50	9	-23	11	0000 0010 100
51	9.5	-22.5	11	0000 0010 010
52	10	-22	11	0000 0010 000
53	10.5	-21.5	11	0000 0001 110
54	11	-21	11	0000 0001 100
55	11.5	-20.5	11	0000 0001 010
56	12	-20	11	0000 0001 000
57	12.5	-19.5	12	0000 0000 1110
58	13	-19	12	0000 0000 1100
59	13.5	-18.5	12	0000 0000 1010
60	14	-18	12	0000 0000 1000
61	14.5	-17.5	12	0000 0000 0110
62	15	-17	12	0000 0000 0100
63	15.5	-16.5	13	0000 0000 0011 0

5.3.8. Motion vector data (MVD_{2-4}) (Variable length)

The three codewords MVD_{2-4} are included if indicated by PTYPE and by MCBPC, and consist each of a variable length codeword for the horizontal component followed by a variable length codeword for the vertical component of each vector. Variable length codes are given in TABLE 11/H.263. MVD_{2-4} are only present when in Advanced Prediction mode (refer to Annex F).

5.3.9 Motion vector data for B-macroblock (MVDB) (Variable length)

MVDB is only present in PB-frames mode if indicated by MODB and consists of a variable length codeword for the horizontal component followed by a variable length codeword for the vertical component of each vector. Variable length codes are given in TABLE 11/H.263. For the use of MVDB refer to Annex G.

5.4. Block Layer

If not in PB-frames mode, a macroblock comprises four luminance blocks and one of each of the two colour difference blocks (see FIGURE 5/H.263). The structure of the block layer is shown in FIGURE 10/H.263. INTRADC is present for every block of the macroblock if MCBPC indicates MB type 3 or 4 (see TABLE 4/H.263 and TABLE 5/H.263). TCOEF is present if indicated by MCBPC or CBPY.

In PB-frames mode, a macroblock comprises twelve blocks. First the data for the six P-blocks is transmitted as in the default H.263 mode, then the data for the six B-blocks. INTRADC is present for every P-block of the macroblock if MCBPC indicates MB type 3 or 4 (see TABLE 4/H.263 and TABLE 5/H.263). INTRADC is not present for B-blocks. TCOEF is present for P-blocks if indicated by MCBPC or CBPY; TCOEF is present for B-blocks if indicated by CBPB.

For coding of the symbols in the Syntax-based Arithmetic Coding mode refer to Annex E.

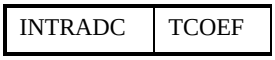


FIGURE 10/H.263

Structure of block layer

5.4.1. DC coefficient for INTRA blocks (INTRADC) (8 bits)

A codeword of 8 bits. The code 0000 0000 is not used. The code 1000 0000 is not used, the reconstruction level of 1024 being coded as 1111 1111 (see TABLE 12/H.263).

TABLE 12/H.263

Reconstruction levels for INTRA-mode DC coefficient

Index	FLC	Reconstruction level into inverse transform
0	0000 0001 (1)	8
1	0000 0010 (2)	16
2	0000 0011 (3)	24
.	. .	.
.	. .	.
126	0111 1111 (127)	1016

127	1111 1111 (255)	1024
128	1000 0001 (129)	1032
.	. .	.
.	. .	.
252	1111 1101 (253)	2024
253	1111 1110 (254)	2032

5.4.2. Transform coefficient (TCOEF) (Variable length)

The most commonly occurring EVENTs are coded with the variable length codes given in TABLE 13/H.263. The last bit "s" denotes the sign of the level, "0" for positive and "1" for negative.

An EVENT is a combination of a last non-zero coefficient indication (LAST; "0": there are more nonzero coefficients in this block, "1": this is the last nonzero coefficient in this block), the number of successive zeros preceding the coded coefficient (RUN), and the non-zero value of the coded coefficient (LEVEL).

The remaining combinations of (LAST, RUN, LEVEL) are coded with a 22 bit word consisting of 7 bits ESCAPE, 1 bit LAST, 6 bits RUN and 8 bits LEVEL. Use of this 22-bit word for encoding the combinations listed in TABLE 13/H.263 is not prohibited. For the 8-bit word for LEVEL, the codes 0000 0000 and 1000 0000 are not used. The codes for RUN and for LEVEL are given in TABLE 14/H.263.

TABLE 13/H.263

VLC table for TCOEF

INDEX	LAST	RUN	LEVEL	BITS	VLC CODE
0	0	0	1	3	10s
1	0	0	2	5	1111 s
2	0	0	3	7	0101 01s
3	0	0	4	8	0010 111s
4	0	0	5	9	0001 1111 s
5	0	0	6	10	0001 0010 1s
6	0	0	7	10	0001 0010 0s
7	0	0	8	11	0000 1000 01s
8	0	0	9	11	0000 1000 00s
9	0	0	10	12	0000 0000 111s
10	0	0	11	12	0000 0000 110s
11	0	0	12	12	0000 0100 000s
12	0	1	1	4	110s
13	0	1	2	7	0101 00s
14	0	1	3	9	0001 1110 s
15	0	1	4	11	0000 0011 11s
16	0	1	5	12	0000 0100 001s
17	0	1	6	13	0000 0101 0000s
18	0	2	1	5	1110 s
19	0	2	2	9	0001 1101 s
20	0	2	3	11	0000 0011 10s
21	0	2	4	13	0000 0101 0001s
22	0	3	1	6	0110 1s

INDEX	LAST	RUN	LEVEL	BITS	VLC CODE
58	1	0	1	5	0111 s
59	1	0	2	10	0000 1100 1s
60	1	0	3	12	0000 0000 101s
61	1	1	1	7	0011 11s
62	1	1	2	12	0000 0000 100s
63	1	2	1	7	0011 10s
64	1	3	1	7	0011 01s
65	1	4	1	7	0011 00s
66	1	5	1	8	0010 011s
67	1	6	1	8	0010 010s
68	1	7	1	8	0010 001s
69	1	8	1	8	0010 000s
70	1	9	1	9	0001 1010 s
71	1	10	1	9	0001 1001 s
72	1	11	1	9	0001 1000 s
73	1	12	1	9	0001 0111 s
74	1	13	1	9	0001 0110 s
75	1	14	1	9	0001 0101 s
76	1	15	1	9	0001 0100 s
77	1	16	1	9	0001 0011 s
78	1	17	1	10	0000 1100 0s
79	1	18	1	10	0000 1011 1s
80	1	19	1	10	0000 1011 0s

23	0	3	2	10	0001 0001 1s
24	0	3	3	11	0000 0011 01s
25	0	4	1	6	0110 0s
26	0	4	2	10	0001 0001 0s
27	0	4	3	13	0000 0101 0010s
28	0	5	1	6	0101 1s
29	0	5	2	11	0000 0011 00s
30	0	5	3	13	0000 0101 0011s
31	0	6	1	7	0100 11s
32	0	6	2	11	0000 0010 11s

81	1	20	1	10	0000 1010 1s
82	1	21	1	10	0000 1010 0s
83	1	22	1	10	0000 1001 1s
84	1	23	1	10	0000 1001 0s
85	1	24	1	10	0000 1000 1s
86	1	25	1	11	0000 0001 11s
87	1	26	1	11	0000 0001 10s
88	1	27	1	11	0000 0001 01s
89	1	28	1	11	0000 0001 00s
90	1	29	1	12	0000 0100 100s

TABLE 13/H.263

VLC table for TCOEF *(concluded)*

INDEX	LAST	RUN	LEVEL	BITS	VLC CODE
33	0	6	3	13	0000 0101 0100s
34	0	7	1	7	0100 10s
35	0	7	2	11	0000 0010 10s
36	0	8	1	7	0100 01s
37	0	8	2	11	0000 0010 01s
38	0	9	1	7	0100 00s
39	0	9	2	11	0000 0010 00s
40	0	10	1	8	0010 110s
41	0	10	2	13	0000 0101 0101s
42	0	11	1	8	0010 101s
43	0	12	1	8	0010 100s
44	0	13	1	9	0001 1100 s
45	0	14	1	9	0001 1011 s
46	0	15	1	10	0001 0000 1s
47	0	16	1	10	0001 0000 0s
48	0	17	1	10	0000 1111 1s
49	0	18	1	10	0000 1111 0s
50	0	19	1	10	0000 1110 1s
51	0	20	1	10	0000 1110 0s
52	0	21	1	10	0000 1101 1s
53	0	22	1	10	0000 1101 0s
54	0	23	1	12	0000 0100 010s
55	0	24	1	12	0000 0100 011s
56	0	25	1	13	0000 0101 0110s
57	0	26	1	13	0000 0101 0111s

INDEX	LAST	RUN	LEVEL	BITS	VLC CODE
91	1	30	1	12	0000 0100 101s
92	1	31	1	12	0000 0100 110s
93	1	32	1	12	0000 0100 111s
94	1	33	1	13	0000 0101 1000s
95	1	34	1	13	0000 0101 1001s
96	1	35	1	13	0000 0101 1010s
97	1	36	1	13	0000 0101 1011s
98	1	37	1	13	0000 0101 1100s
99	1	38	1	13	0000 0101 1101s
100	1	39	1	13	0000 0101 1110s
101	1	40	1	13	0000 0101 1111s
102	ESCAPE			7	0000 011

TABLE 14/H.263

FLC table for RUNS and LEVELS

Index	Run	Code
0	0	000 000
1	1	000 001
2	2	000 010
.	.	.
.	.	.
63	63	111 111

Index	Level	Code
-	-128	FORBIDDEN
0	-127	1000 0001
.	.	.
125	-2	1111 1110
126	-1	1111 1111
-	0	FORBIDDEN
127	1	0000 0001
128	2	0000 0010
.	.	.
253	127	0111 1111

6. Decoding process

6.1. Motion compensation

In this section, the motion compensation for the default H.263 prediction mode is described. For a description of motion compensation in the Unrestricted Motion Vector mode refer to Annex D. For a description of motion compensation in the Advanced Prediction mode refer to Annex F.

6.1.1 Differential motion vectors

The macroblock vector is obtained by adding predictors to the vector differences indicated by MVD (see TABLE 11/H.263). For differential coding with four vectors per macroblock refer to Annex F. In case of one vector per macroblock, the candidate predictors for the differential coding are taken from three surrounding macroblocks as indicated in FIGURE 11/H.263. The predictors are calculated separately for the horizontal and vertical components.

In the special cases at the borders of the current GOB or picture the following decision rules are applied in increasing order:

1. When the corresponding macroblock was coded in INTRA mode (if not in PB-frames mode) or was not coded (COD=1), the candidate predictor is set to zero.
2. The candidate predictor MV1 is set to zero if the corresponding macroblock is outside the picture (at the left side).
3. Then, the candidate predictors MV2 and MV3 are set to MV1 if the corresponding macroblocks are outside the picture (at the top) or outside the GOB (at the top) if the GOB header of the current GOB is non-empty;
4. Then, the candidate predictor MV3 is set to zero if the corresponding macroblock is outside the picture (at the right side).
5. ~~When the corresponding macroblock was coded in INTRA mode (if not in PB-frames mode) or was not coded (COD=1), the candidate predictor is set to zero.~~

For each component, the predictor is the median value of the three candidate predictors for this component.

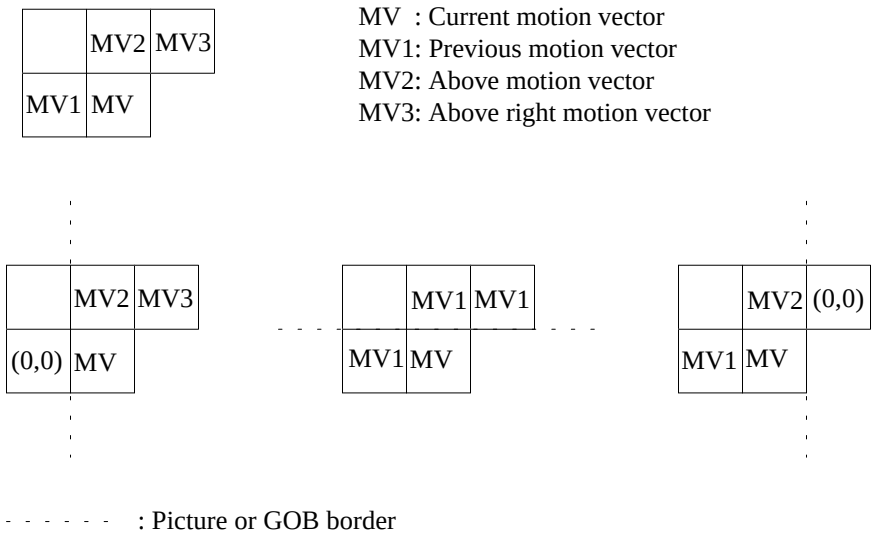


FIGURE 11/H.263

Motion vector prediction

Advantage is taken of the fact that the range of motion vector component values is constrained. Each VLC word for MVD represents a pair of difference values. Only one of the pair will yield a macroblock vector component falling within the permitted range [-16,15.5]. A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pixels in the previous picture which are spatially to the right or below the pixels being predicted.

The motion vector is used for all pixels in all four luminance blocks in the macroblock. Motion vectors for both chrominance blocks are derived by dividing the component values of the macroblock vector by two, due to the lower chrominance format. The component values of the resulting quarter pixel resolution vectors are modified towards the nearest half pixel position as indicated in TABLE 15/H.263.

TABLE 15/H.263

Modification of quarter pixel resolution chrominance vector components

quarter pixel position	0	1/4	1/2	3/4	1
resulting position	0	1/2	1/2	1/2	1

6.1.2 Interpolation for subpixel prediction

Half pixel values are found using bilinear interpolation as described in FIGURE 12/H.263. "/" indicates division by truncation.

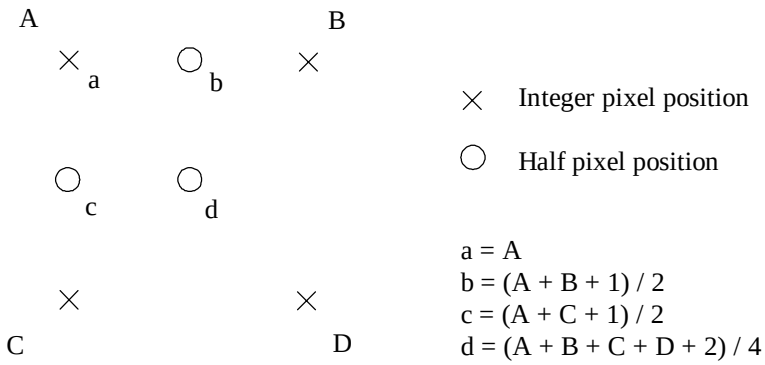


FIGURE 12/H.263

Halfpixel prediction by bilinear interpolation

6.2. Coefficients decoding

6.2.1 Inverse Quantization

If LEVEL = “0”, the reconstruction level REC = “0”. The reconstruction level of INTRADC is given by TABLE 12/H.263. The reconstruction levels of all non-zero coefficients other than the INTRADC one are given by the following formulas:

$|REC| = QUANT \cdot (2 \cdot |LEVEL| + 1)$ if QUANT = “odd”

$|REC| = QUANT \cdot (2 \cdot |LEVEL| + 1) - 1$ if QUANT = “even”

Note that this process disallows even valued numbers. This has been found to prevent accumulation of IDCT mismatch errors. After calculation of |REC|, the sign is added to obtain REC: $REC = sign(LEVEL) \cdot |REC|$

Sign(LEVEL) is given by the last bit of the TCOEF code (see TABLE 13/H.263) or by TABLE 14/H.263.

6.2.2 Clipping of reconstruction levels

After inverse quantization, the reconstruction levels of all coefficients other than the INTRADC one are clipped to the range -2048 to 2047.

6.2.3 Zig-zag positioning

The quantized transform coefficients are placed into an 8x8 block according to the sequence given in FIGURE 13/H.263. Coefficient 1 is the dc-coefficient.

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

FIGURE 13/H.263

Zig-zag positioning of quantized transform coefficients

6.2.4 Inverse transform

After inverse quantization and zigzag of coefficients, the resulting 8x8 blocks are processed by a separable two-dimensional inverse discrete cosine transform of size 8 by 8. The output from the inverse transform ranges from -256 to +255 after clipping to be represented with 9 bits. The transfer function of the inverse transform is given by:

$$f(x,y) = 1/4 \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u,v)\cos[\pi(2x+1)u/16]\cos[\pi(2y+1)v/16]$$

with $u, v, x, y = 0, 1, 2, \ldots, 7$

where x,y = spatial coordinates in the pixel domain,

u,v = coordinates in the transform domain,

$C(u) = 1/\sqrt{2}$ for $u = 0$, otherwise 1,

$C(v) = 1/\sqrt{2}$ for $v = 0$, otherwise 1.

Note – Within the block being transformed, $x = 0$ and $y = 0$ refer to the pixel nearest the left and top edges of the picture respectively.

The arithmetic procedures for computing the inverse transform are not defined, but should meet the error tolerance specified in Annex A.

6.3 Reconstruction of blocks

6.3.1 Summation

After motion compensation and coefficients decoding (inverse transform included), a reconstruction is formed for each luminance and chrominance block. For INTRA blocks, the reconstruction is equal to the result of the inverse transformation. For INTER blocks, the reconstruction is formed by summing the prediction and the result of the inverse transformation. The summation is performed on a pixel basis.

6.3.2 Clipping

To prevent quantization distortion of transform coefficient amplitudes causing arithmetic overflow in the encoder and decoder loops, clipping functions are inserted. The clipper operates after the summation of prediction and reconstructed prediction error on resulting pixel values less than 0 or greater than 255, changing them to 0 and 255 respectively.

Annex A

Inverse transform accuracy specification

(This annex forms an integral part of this Recommendation)

A.1 Generate random integer pixel data values in the range –L to +H according to the random number generator given below ("C" version). Arrange into 8 by 8 blocks. Data set of 10 000 blocks should each be generated for (L = 256, H = 255), (L = H = 5) and (L = H = 300).

A.2 For each 8 by 8 block, perform a separable, orthonormal, matrix multiply, forward discrete cosine transform using at least 64-bit floating point accuracy.

$$F(u,v) = 1/4C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos[\pi(2x+1)u/16]\cos[\pi(2y+1)v/16]$$

with $u, v, x, y = 0, 1, 2, \ldots, 7$

where x,y = spatial coordinates in the pixel domain,

u,v = coordinates in the transform domain,

$C(u) = 1/\sqrt{2}$ for $u = 0$, otherwise 1,

$C(v) = 1/\sqrt{2}$ for $v = 0$, otherwise 1.

A.3 For each block, round the 64 resulting transformed coefficients to the nearest integer values. Then clip them to the range –2048 to +2047. This is the 12-bit input data to the inverse transform.

A.4 For each 8 by 8 block of 12-bit data produced by § A.3, perform a separable, orthonormal, matrix multiply, inverse discrete transform (IDCT) using at least 64-bit floating point accuracy. Round the resulting pixels to the nearest integer and clip to the range –256 to +255. These blocks of 8×8 pixels are the reference IDCT output data.

A.5 For each 8 by 8 block produced by § A.3, apply the IDCT under test and clip the output to the range –256 to +255. These blocks of 8×8 pixels are the test IDCT output data.

A.6 For each of the 64 IDCT output pixels, and for each of the 10,000 block data sets generated above, measure the peak, mean and mean square error between the reference and the test data.

A.7 For any pixel, the peak error should not exceed 1 in magnitude.

For any pixel, the mean square error should not exceed 0.06.

Overall, the mean square error should not exceed 0.02.

For any pixel, the mean error should not exceed 0.015 in magnitude.

Overall, the mean error should not exceed 0.0015 in magnitude.

A.8 All zeros in shall produce all zeros out.

A.9 Re-run the measurements using exactly the same data values of step 1, but change the sign on each pixel.

"C" program for random number generation

```
/* L and H shall be long, that is 32 bits */
long rand  (L,H)
long        L,H;
{
   static long   randx = 1;              /* long is 32 bits       */
   static double z = (double) 0x7fffffff;

   long    i,j;
   double  x;                            /* double is 64 bits     */

   randx = (randx * 1103515245) + 12345;
   i = randx & 0x7ffffffe;               /* keep 30 bits          */
   x = ( (double)i ) /  z;               /* range 0 to 0.99999 ... */
   x *= (L+H+1);                         /* range 0 to < L+H+1    */
   j = x;                                /* truncate to integer   */
   return(j - L);                        /* range -L to H         */
}
```

Annex B

Hypothetical Reference Decoder

(This annex forms an integral part of this Recommendation)

The Hypothetical Reference Decoder (HRD) is defined as follows.

B.1 The HRD and the encoder have the same clock frequency as well as the same CIF rate, and are operated synchronously.

B.2 The HRD receiving buffer size is (B + BPPmaxKb * 1024 bits) where (BPPmaxKb * 1024) is the maximum number of bits per picture that has been negotiated for use in the bitstream (see section 3.6). The value of B is defined as follows:

$$B = 4 \cdot R_{max} / 29.97$$

where 29.97 is in Hz and R_{max} is the maximum video bitrate during the connection in bits per second. This value for B is a minimum. An encoder may use a larger value for B, provided the larger number is first negotiated by external means, for example Recommendation H.245.

The value for R_{max} depends on the system configuration (for example GSTN or ISDN, single or multi-link) and may be equal to the maximum bitrate supported by the physical link. Negotiation of R_{max} is done by external means, for example Recommendation H.245.

B.3 The HRD is initially empty.

B.4 The HRD buffer is examined at CIF intervals (1000 / 29.97 ms). If at least one complete coded picture is in the buffer then all the data for the earliest picture is instantaneously removed (e.g. at t_{n+1} in FIGURE 14/H.263). Immediately after removing the above data the buffer occupancy must be less than B. This is a requirement on the coder output bitstream including coded picture data and MCBPC and STUF stuffing but not error correction framing bits, fill indicator (Fi), fill bits or error correction parity information described in Annex H.

For the purposes of this definition, a complete coded picture is a normal I- or P-picture or a PB-frame.

To meet this requirement the number of bits for the $(n+1)$th coded picture d_{n+1} must satisfy:

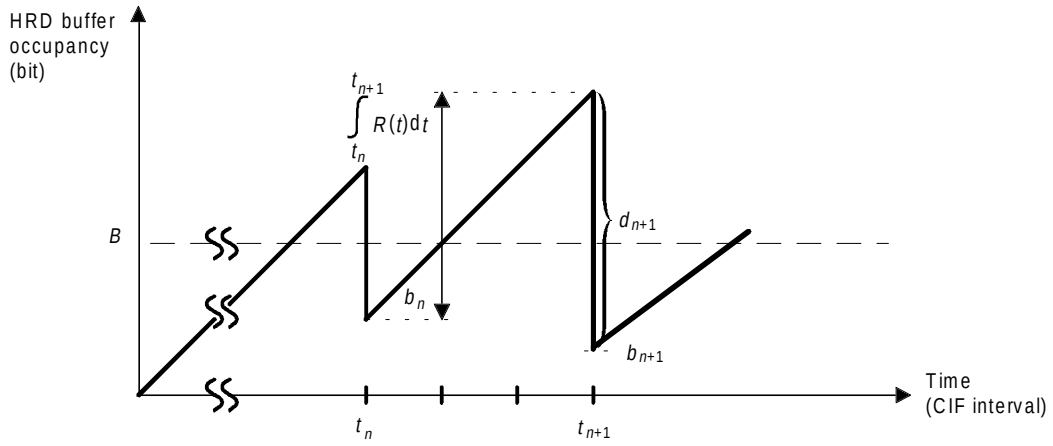
$$d_{n+1} \geq b_n + \int_{t_n}^{t_{n+1}} R(t)dt - B$$

where

b_n is the buffer occupancy just after time t_n;

t_n is the time the nth coded picture is removed from the HRD buffer;

R(t) is the video bitrate at time t.



NOTE – Time ($t_{n+1} - t_n$) is an integer number of CIF picture periods (1/29.97, 2/29.97, 3/29.97, ...).

FIGURE 14/H.263

HRD buffer occupancy

Annex C

Considerations for Multipoint

(This annex forms an integral part of this Recommendation)

The following facilities are provided to support switched multipoint operation.

C.1. *Freeze picture request*

Causes the decoder to freeze its displayed picture until a freeze picture release signal is received or a time-out period of at least six seconds has expired. The transmission of this signal is by external means (for example Recommendation H.245).

C.2. *Fast update request*

Causes the encoder to encode its next picture in INTRA mode with coding parameters such as to avoid buffer overflow. The transmission method for this signal is by external means (for example Recommendation H.245).

C.3. *Freeze picture release*

A signal from an encoder which has responded to a fast update request and allows a decoder to exit from its freeze picture mode and display decoded pictures in the normal manner. This signal is transmitted by PTYPE (see § 5.1.3) in the picture header of the first picture coded in response to the fast update request.

C.4. *Continuous Presence Multipoint (CPM*) (not used for H.324)

In H.263 a negotiable Continuous Presence Multipoint mode is provided in which up to four independent H.263 QCIF bitstreams can be multiplexed as independent 'Sub-Bitstreams' in one new video bitstream with use of the PSBI and GSBI fields. Capability exchange for this mode is done by external means (for example Recommendation H.242).

When in CPM mode, the CPM field shall be set to "1" in each of the independent H.263 bitstreams. Sub-Bitstream indicators (SBIs) in the Picture and GOB headers of each H.263 bitstream indicate the number of the H.263 bitstream to which that header and all following information until the next Picture or GOB header in the composed video bitstream belong.

Each Sub-Bitstream is considered as a normal H.263 bitstream and shall therefore comply to the capabilities that are exchanged by external means. The information for the different H.263 bitstreams is not transmitted in any special predefined order, an SBI can have any value independent from preceeding SBIs and the picture rates for the different H.263 bitstreams may be different. The information in each individual bitstream is also completely independent from the information in the other bitstreams. For example, the GFID codewords in one Sub-Bitstream are not influenced by GFID or PTYPE codewords in other Sub-Bitstreams.

Annex D

Unrestricted Motion Vector mode

(This annex forms an integral part of this Recommendation)

This annex describes the optional Unrestricted Motion Vector mode of H.263. The capability of this mode of H.263 is signalled by external means (for example Recommendation H.245). The use of this mode is indicated in PTYPE.

D.1. Motion vectors over picture boundaries

In the default prediction mode of H.263, motion vectors are restricted such that all pixels referenced by them are within the coded picture area (see section 4.2.3). In the Unrestricted Motion Vector mode however this restriction is removed and therefore motion vectors *are* allowed to point outside the picture. When a pixel referenced by a motion vector is outside the coded picture area, an edge pixel is used instead. This edge pixel is found by limiting the motion vector to the last full pixel position inside the coded picture area. Limitation of the motion vector is performed on a pixel basis and separately for each component of the motion vector.

For example, if the Unrestricted Motion Vector mode is used for a QCIF picture, the referenced pixel value for the luminance component is given by the following formula:

$$Rumv(x, y) = R(x', y'),$$

where

x, y, x', y' = spatial coordinates in the pixel domain,

$Rumv(x, y)$ = pixel value of the referenced picture at (x, y) when in Unrestricted Motion Vector mode,

$R(x', y')$ = pixel value of the referenced picture at (x', y') when in Unrestricted Motion Vector mode,

$$x' = \begin{cases} 0 & \text{if } x < 0 \\ 175 & \text{if } x > 175 \\ x & \text{otherwise,} \end{cases}$$

$$y' = \begin{cases} 0 & \text{if } y < 0 \\ 143 & \text{if } y > 143 \\ y & \text{otherwise,} \end{cases}$$

and the coded picture area of R(x', y') is $0 <= x' <= 175$, $0 <= y' <= 143$. The given boundaries are integer pixel positions; however, (x',y') can also be a half pixel position within these boundaries.

D.2. Extension of the motion vector range

In the default prediction mode, the values for both horizontal and vertical components of the motion vectors are restricted to the range [-16,15.5] (this is also valid for the forward and backward motion vector components for B-pictures). In the

Unrestricted Motion Vector mode however, the maximum range for vector components is [-31.5,31.5], with the restriction that only values that are within a range of [-16,15.5] around the predictor for each motion vector component can be reached if the predictor is in the range [-15.5,16]. If the predictor is outside [-15.5,16], all values within the range [-31.5,31.5] with the same sign as the predictor plus the zero value can be reached. So, if MV_C is the motion vector component and P_C is the predictor for it, then

$$\begin{aligned} -31.5 \le MV_C \le 0 \quad & \text{if} \quad -31.5 \le P_C \le -16 \\ -16 + P_C \le MV_C \le 15.5 + P_C \quad & \text{if} \quad -15.5 \le P_C \le 16 \\ 0 \le MV_C \le 31.5 \quad & \text{if} \quad 16.5 \le P_C \le 31.5 \end{aligned}$$

In the Unrestricted Motion Vector mode, the interpretation of TABLE 11/H.263 for MVD, $MVD_{2\text{-}4}$ and MVDB is as follows:

- If the predictor for the motion vector component is in the range [-15.5,16], only the first column of vector differences applies;
- If the predictor for the motion vector component is outside the range [-15.5,16], the vector difference from TABLE 11/H.263 shall be used that results in a vector component inside the range [-31.5,31.5] with the same sign as the predictor (including zero).

The predictor for MV_D and $MVD_{2\text{-}4}$ is defined as the median value of the vector components MV1, MV2 and MV3 as defined in section 6.1.1 and F.2. For MVDB the predictor $P_C = (TR_B \times MV) / TR_D$ where MV represents a vector component for an 8*8 luminance block in a P-picture (see also section G.4).

Annex E

Syntax-based Arithmetic Coding mode

(This annex forms an integral part of this Recommendation)

E.1. Introduction

In the variable length coding/decoding (VLC/VLD) as described in section 5 of this Recommendation, a symbol is VLC encoded using a specific table based on the syntax of the coder. This table typically stores lengths and values of the VLC code words. The symbol is mapped to an entry of the table in a table look-up operation, and then the binary code word specified by the entry is sent out normally to a buffer for transmitting to the receiver. In VLD decoding, the received bitstream is matched entry by entry in a specific table based on the syntax of the coder. This table must be the same as the one used in the encoder for encoding the current symbol. The matched entry in the table is then mapped back to the corresponding symbol which is the end result of the VLD decoder and is then used for recovering the video pictures. This VLC/VLD process implies that each symbol must be encoded into a fixed integral number of bits. Removing this restriction of fixed integral number of bits for symbols can lead to reductions of resulting bitrates, which can be achieved by arithmetic coding.

This Annex describes the optional Syntax-based Arithmetic Coding (SAC) mode of H.263. In this mode, all the corresponding variable length coding/decoding operations of H.263 are replaced with arithmetic coding/decoding operations. The capability of this mode of H.263 is signalled by external means (for example Recommendation H.245). The use of this mode is indicated in PTYPE.

E.2. Specification of SAC encoder

In SAC mode, a symbol is encoded by using a specific array of integers (or a model) based on the syntax of the coder and by calling the following procedure which is specified in C.

```
#define   q1     16384
#define   q2     32768
#define   q3     49152
#define   top    65535

static long      low, high, opposite_bits, length;

void   encode_a_symbol(int index, int cumul_freq[ ])
{
       length = high - low + 1;
       high = low - 1 + (length * cumul_freq[index]) / cumul_freq[0];
       low += (length * cumul_freq[index+1]) / cumul_freq[0];
       for ( ; ; ) {
          if (high < q2) {
             send out a bit “0” to PSC_FIFO;
             while (opposite_bits > 0) {
                send out a bit “1” to PSC_FIFO;
                opposite_bits--;
             }
          }
          else if (low >= q2) {
             send out a bit “1” to PSC_FIFO;
             while (opposite_bits > 0) {
                send out a bit “0” to PSC_FIFO;
                opposite_bits--;
             }
             low -= q2;
```

```
        high -= q2;
    }
```

```
            else if (low >= q1 && high < q3) {
                opposite_bits += 1;
                low -= q1;
                high -= q1;
            }
            else break;

            low *= 2;
            high = 2*high+1;
        }
}
```

The values of `low`, `high` and `opposite_bits` are initialized to `0`, `top` and `0`, respectively. `PSC_FIFO` is a FIFO for buffering the output bits from the arithmetic encoder. The model is specified through `cumul_freq[ ],` and the symbol is specified using its index in the model.

E.3. Specification of SAC decoder

In SAC decoder, a symbol is decoded by using a specific model based on the syntax and by calling the following procedure which is specified in C.

```
static long  low, high, code_value, bit, length, index, cum;

int    decode_a_symbol(int cumul_freq[ ])
{
        length = high - low + 1;
        cum = (-1 + (code_value - low + 1) * cumul_freq[0]) / length;
        for (index = 1; cumul_freq[index] > cum; index++);
        high = low - 1 + (length * cumul_freq[index-1]) / cumul_freq[0];
        low += (length * cumul_freq[index]) / cumul_freq[0];
        for ( ; ; ) {
            if (high < q2) ;
            else if (low >= q2) {
                code_value -= q2;
                low -= q2;
                high -= q2;
            }
            else if (low >= q1 && high < q3) {
                code_value -= q1;
                low -= q1;
                high -= q1;
            }
            else break;

            low *= 2;
            high = 2*high + 1;
            get bit from PSC_FIFO;
            code_value = 2*code_value + bit;
        }
        return (index-1);
}
```

Again the model is specified through `cumul_freq[ ]`. The decoded symbol is returned through its index in the model. `PSC_FIFO` is a FIFO for buffering the incoming bitstream. The decoder is initialized to start decoding an arithmetic coded bitstream by calling the following procedure.

```
void   decoder_reset( )
{
       code_value = 0;
       low = 0;
       high = top;
       for (int i = 1;   i <= 16;   i++) {
          get bit from PSC_FIFO;
          code_value = 2 * code_value + bit;
       }
}
```

E.4. Syntax

As in VLC table mode of H.263, the syntax of the symbols is partitioned into four layers: Picture, Group of Blocks, Macroblock and Block. The syntax of the top three layers remain exactly the same. The syntax of the Block layer also remains quite similar, but is illustrated in FIGURE 15/H.263.

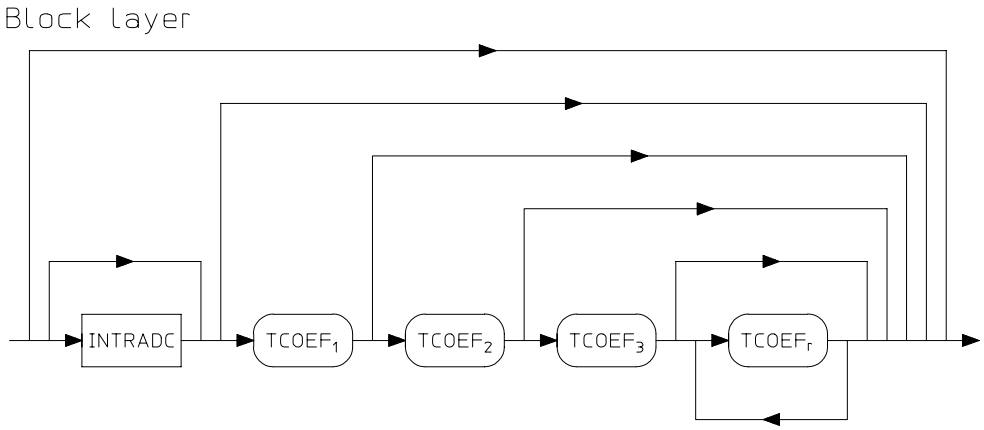


FIGURE 15/H.263

Structure of SAC Block layer

In FIGURE 15/H.263, TCOEF1, TCOEF2, TCOEF3 and TCOEFr are Last-Run-Level symbols as defined in section 5.4.2, and are the possible 1st, 2nd, 3rd and rest of the symbols, respectively. TCOEF1, TCOEF2, TCOEF3 and TCOEFr are only present when one, two, three or more coefficients are present in the block layer, respectively.

E.5. PSC_FIFO

PSC_FIFO in encoder or decoder is a FIFO of size > 17 bits. In PSC_FIFO of encoder, illegal emulations of PSC and GBSC are located and are avoided by stuffing a “1” after each successive appearance of 14 “0”s (which are not part of PSC or

GBSC). In PSC_FIFO of the decoder, the first “1” after each string of 14 “0”s is deleted; if instead a string of 14 “0”s is followed by a “0”, it indicates that a legal PSC or GBSC is detected. The exact location of the PSC or GBSC is determined by the next “1” following the string of zeros.

E.6. Fixed Length Symbols

Fixed length symbols form three possible strings, PSC--TR--PTYPE--PQUANT--CPM--(PSBI)--(TRB-DBQUANT)--PEI--(PSPARE--PEI--...), (GSTUF)--GBSC--GN--(GSBI)--GFID--GQUANT, and (ESTUF)--(EOS)--(PSTUF). These strings are directly sent to PSC_FIFO as in the normal VLC table mode of H.263 at encoder side, and are directly sent out from PSC_FIFO in the decoder after a legal PSC or GBSC or EOS is detected.

If a fixed length symbol string is not the first in a video session, the arithmetic encoder needs to be reset before sending the fixed length symbol string by calling the following procedure. This procedure shall also be called at the end of a video session or before EOS.

```
void   encoder_flush( )
{
       opposite_bits++;
       if (low < q1) {
          send out a bit “0” to PSC_FIFO;
          while (opposite_bits > 0) {
             send out a bit “1” to PSC_FIFO;
             opposite_bits--;
          }
       }
       else {
          send out a bit “1” to PSC_FIFO;
          while (opposite_bits > 0) {
             send out a bit “0” to PSC_FIFO;
             opposite_bits--;
          }
       }
       low = 0;
       high = top;
}
```

In the decoder, after each fixed length symbol string, procedure **`decoder_reset`** is called.

E.7. Non-Fixed Length Symbols

Models for the non-fixed length symbols are included in section E8. The indices as given in the VLC tables of section 5 are used for the indexing of the integers in the models.

The models for COD and MCBPC in P-pictures are named by cumf_COD and cumf_MCBPC. The index for COD being “0” is 0, and is 1 for COD being “1”. The indexes for MCBPC are defined in TABLE 4/H.263 for I-pictures and TABLE 5/H.263 for P-pictures. The model for MCBPC in I-pictures is named by cumf_MCBPC_intra.

The model for MODB is cumf_MODB. The indexes for MODB are defined in TABLE 8/H.263. The model for CBPBn, n=1,2,...,4, is cumf_YCBPB, and the model for CBPBn, n=5,6, is cumf_UVCBPB, with index 0 for CBPBn=0 and index 1 for CBPBn=1.

The model for CBPY is cumf_CBPY in INTER macroblocks and cumf_CBPY_intra in INTRA macroblocks. The model for DQUANT is cumf_DQUANT. The indexing for CBPY and DQUANT is defined in TABLE 10/H.263 and TABLE 9/H.263, respectively.

The model for MVD, $MVD_{2\text{-}4}$ and MVDB is cumf_MVD and the model for INTRADC is cumf_INTRADC. The indexing is defined in TABLE 11/H.263 and TABLE 12/H.263, respectively.

A non-escaped TCOEF consists of a symbol for TCOEF1/2/3/r followed by a symbol, SIGN, for the sign of the TCOEF. The models for TCOEF1, TCOEF2, TCOEF3 and TCOEFr in INTER blocks are cumf_TCOEF1, cumf_TCOEF2, cumf_TCOEF3, cumf_TCOEFr. The models for INTRA blocks are cumf_TCOEF1_intra, cumf_TCOEF2_intra, cumf_TCOEF3_intra, cumf_TCOEFr_intra. For all TCOEFs the indexing is defined in TABLE 13/H.263. The model for SIGN is cumf_SIGN. The indexing for SIGN is 0 for positive sign and 1 for negative sign.

The models for LAST, RUN, LEVEL after ESCAPE are cumf_LAST (cumf_LAST_intra), cumf_RUN (cumf_RUN_intra), cumf_LEVEL (cumf_LEVEL_intra) for INTER (INTRA) blocks. The indexing for LAST is 0 for LAST=0 and 1 for LAST=1, while the indexing for RUN and LEVEL is defined in TABLE 14/H.263.

E.8. SAC Models

int cumf_COD[3]={16383, 6849, 0};

int cumf_MCBPC[22]={16383, 4105, 3088, 2367, 1988, 1621, 1612, 1609, 1608, 496, 353, 195, 77, 22, 17, 12, 5, 4, 3, 2, 1, 0};

int cumf_MCBPC_intra[10]={16383, 7410, 6549, 5188, 442, 182, 181, 141, 1, 0};

int cumf_MODB[4]={16383, 6062, 2130, 0};

int cumf_YCBPB[3]={16383, 6062, 0};

int cumf_UVCBPB[3]={16383, 491, 0};

int cumf_CBPY[17]={16383, 14481, 13869, 13196, 12568, 11931, 11185, 10814, 9796, 9150, 8781, 7933, 6860, 6116, 4873, 3538, 0};

int cumf_CBPY_intra[17]={16383, 13619, 13211, 12933, 12562, 12395, 11913, 11783, 11004, 10782, 10689, 9928, 9353, 8945, 8407, 7795, 0};

int cumf_DQUANT[5]={16383, 12287, 8192, 4095, 0};

int cumf_MVD[65]={16383, 16380, 16369, 16365, 16361, 16357, 16350, 16343, 16339, 16333, 16326, 16318, 16311, 16306, 16298, 16291, 16283, 16272, 16261, 16249, 16235, 16222, 16207, 16175, 16141, 16094, 16044, 15936, 15764, 15463, 14956, 13924, 11491, 4621, 2264, 1315, 854, 583, 420, 326, 273, 229, 196, 166, 148, 137, 123, 114, 101, 91, 82, 76, 66, 59, 53, 46, 36, 30, 26, 24, 18, 14, 10, 5, 0};

int cumf_INTRADC[255]={16383, 16380, 16379, 16378, 16377, 16376, 16370, 16361, 16360, 16359, 16358, 16357, 16356, 16355, 16343, 16238, 16237, 16236, 16230, 16221, 16220, 16205, 16190, 16169, 16151, 16130, 16109, 16094,

16070, 16037, 16007, 15962, 15938, 15899, 15854, 15815, 15788, 15743, 15689, 15656, 15617, 15560, 15473, 15404, 15296, 15178, 15106, 14992, 14868, 14738, 14593, 14438, 14283, 14169, 14064, 14004, 13914, 13824, 13752, 13671, 13590, 13515, 13458, 13380, 13305, 13230, 13143, 13025, 12935, 12878, 12794, 12743, 12656, 12596, 12521, 12443, 12359, 12278, 12200, 12131, 12047, 12002, 11948, 11891, 11828, 11744, 11663, 11588, 11495, 11402, 11288, 11204, 11126, 11039, 10961, 10883, 10787, 10679, 10583, 10481, 10360, 10227, 10113, 9961, 9828, 9717, 9584, 9485, 9324, 9112, 9019, 8908, 8766, 8584, 8426, 8211, 7920, 7663, 7406, 7152, 6904, 6677, 6453, 6265, 6101, 5904, 5716, 5489, 5307, 5056, 4850, 4569, 4284, 3966, 3712, 3518, 3342, 3206, 3048, 2909, 2773, 2668, 2596, 2512, 2370, 2295, 2232, 2166, 2103, 2022, 1956, 1887, 1830, 1803, 1770, 1728, 1674, 1635, 1599, 1557, 1500, 1482, 1434, 1389, 1356, 1317, 1284, 1245, 1200, 1179, 1140, 1110, 1092, 1062, 1044, 1035, 1014, 1008, 993, 981, 954, 936, 912, 894, 876, 864, 849, 828, 816, 801, 792, 777, 756, 732, 690, 660, 642, 615, 597, 576, 555, 522, 489, 459, 435, 411, 405, 396, 387, 375, 360, 354, 345, 344, 329, 314, 293, 278, 251, 236, 230, 224, 215, 214, 208, 199, 193, 184, 178, 169, 154, 127, 100, 94, 73, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 20, 19, 18, 17, 16, 15, 9, 0};

int cumf_TCOEF1[104]={16383, 13455, 12458, 12079, 11885, 11800, 11738, 11700, 11681, 11661, 11651, 11645, 11641, 10572, 10403, 10361, 10346, 10339, 10335, 9554, 9445, 9427, 9419, 9006, 8968, 8964, 8643, 8627, 8624, 8369, 8354, 8352, 8200, 8192, 8191, 8039, 8036, 7920, 7917, 7800, 7793, 7730, 7727, 7674, 7613, 7564, 7513, 7484, 7466, 7439, 7411, 7389, 7373, 7369, 7359, 7348, 7321, 7302, 7294, 5013, 4819, 4789, 4096, 4073, 3373, 3064, 2674, 2357, 2177, 1975, 1798, 1618, 1517, 1421, 1303, 1194, 1087, 1027, 960, 890, 819, 758, 707, 680, 656, 613, 566, 534, 505, 475, 465, 449, 430, 395, 358, 335, 324, 303, 295, 286, 272, 233, 215, 0};

int cumf_TCOEF2[104]={16383, 13582, 12709, 12402, 12262, 12188, 12150, 12131, 12125, 12117, 12113, 12108, 12104, 10567, 10180, 10070, 10019, 9998, 9987, 9158, 9037, 9010, 9005, 8404, 8323, 8312, 7813, 7743, 7726, 7394, 7366, 7364, 7076, 7062, 7060, 6810, 6797, 6614, 6602, 6459, 6454, 6304, 6303, 6200, 6121, 6059, 6012, 5973, 5928, 5893, 5871, 5847, 5823, 5809, 5796, 5781, 5771, 5763, 5752, 4754, 4654, 4631, 3934, 3873, 3477, 3095, 2758, 2502, 2257, 2054, 1869, 1715, 1599, 1431, 1305, 1174, 1059, 983, 901, 839, 777, 733, 683, 658, 606, 565, 526, 488, 456, 434, 408, 380, 361, 327, 310, 296, 267, 259, 249, 239, 230, 221, 214, 0};

int cumf_TCOEF3[104]={16383, 13532, 12677, 12342, 12195, 12112, 12059, 12034, 12020, 12008, 12003, 12002, 12001, 10586, 10297, 10224, 10202, 10195, 10191, 9223, 9046, 8999, 8987, 8275, 8148, 8113, 7552, 7483, 7468, 7066, 7003, 6989, 6671, 6642, 6631, 6359, 6327, 6114, 6103, 5929, 5918, 5792, 5785, 5672, 5580, 5507, 5461, 5414, 5382, 5354, 5330, 5312, 5288, 5273, 5261, 5247, 5235, 5227, 5219, 4357, 4277, 4272, 3847, 3819, 3455, 3119, 2829, 2550, 2313, 2104, 1881, 1711, 1565, 1366, 1219, 1068, 932, 866, 799, 750, 701, 662, 605, 559, 513, 471, 432, 403, 365, 336, 312, 290, 276, 266, 254, 240, 228, 223, 216, 206, 199, 192, 189, 0};

int cumf_TCOEFr[104]={16383, 13216, 12233, 11931, 11822, 11776, 11758, 11748, 11743, 11742, 11741, 11740, 11739, 10203, 9822, 9725, 9691, 9677, 9674, 8759, 8609, 8576, 8566, 7901, 7787, 7770, 7257, 7185, 7168, 6716, 6653, 6639, 6276, 6229, 6220, 5888, 5845, 5600, 5567, 5348, 5327, 5160, 5142, 5004, 4900, 4798, 4743, 4708, 4685, 4658, 4641, 4622, 4610, 4598, 4589, 4582, 4578, 4570, 4566, 3824, 3757, 3748, 3360, 3338, 3068, 2835, 2592, 2359, 2179, 1984, 1804, 1614, 1445, 1234, 1068, 870, 739, 668, 616, 566, 532, 489, 453, 426, 385, 357, 335, 316, 297, 283, 274, 266, 259, 251, 241, 233, 226, 222, 217, 214, 211, 209, 208, 0};

int cumf_TCOEF1_intra[104]={16383, 13383, 11498, 10201, 9207, 8528, 8099, 7768, 7546, 7368, 7167, 6994, 6869, 6005, 5474, 5220, 5084, 4964, 4862, 4672, 4591, 4570, 4543, 4397, 4337, 4326, 4272, 4240, 4239, 4212, 4196, 4185, 4158, 4157, 4156, 4140, 4139, 4138, 4137, 4136, 4125, 4124, 4123, 4112, 4111, 4110, 4109, 4108, 4107, 4106, 4105, 4104, 4103, 4102, 4101, 4100, 4099, 4098, 4097, 3043, 2897, 2843, 1974, 1790, 1677, 1552, 1416, 1379, 1331, 1288, 1251, 1250, 1249, 1248, 1247, 1236, 1225, 1224, 1223, 1212, 1201, 1200, 1199, 1198, 1197, 1196, 1195, 1194, 1193, 1192, 1191, 1190, 1189, 1188, 1187, 1186, 1185, 1184, 1183, 1182, 1181, 1180, 1179, 0};

int cumf_TCOEF2_intra[104]={16383, 13242, 11417, 10134, 9254, 8507, 8012, 7556, 7273, 7062, 6924, 6839, 6741, 6108, 5851, 5785, 5719, 5687, 5655, 5028, 4917, 4864, 4845, 4416, 4159, 4074, 3903, 3871, 3870, 3765, 3752, 3751, 3659, 3606, 3580, 3541, 3540, 3514, 3495, 3494, 3493, 3474, 3473, 3441, 3440, 3439, 3438, 3425, 3424, 3423, 3422, 3421, 3420, 3401, 3400, 3399, 3398, 3397, 3396, 2530, 2419, 2360, 2241, 2228, 2017, 1687, 1576, 1478, 1320, 1281, 1242, 1229, 1197, 1178, 1152, 1133, 1114, 1101, 1088, 1087, 1086, 1085, 1072, 1071, 1070, 1069, 1068, 1067, 1066, 1065, 1064, 1063, 1062, 1061, 1060, 1059, 1058, 1057, 1056, 1055, 1054, 1053, 1052, 0};

int cumf_TCOEF3_intra[104]={16383, 12741, 10950, 10071, 9493, 9008, 8685, 8516, 8385, 8239, 8209, 8179, 8141, 6628, 5980, 5634, 5503, 5396, 5327, 4857, 4642, 4550, 4481, 4235, 4166, 4151, 3967, 3922, 3907, 3676, 3500, 3324, 3247, 3246, 3245, 3183, 3168, 3084, 3069, 3031, 3030, 3029, 3014, 3013, 2990, 2975, 2974, 2973, 2958, 2943, 2928, 2927, 2926, 2925, 2924, 2923, 2922, 2921, 2920, 2397, 2298, 2283, 1891, 1799, 1591, 1445, 1338, 1145, 1068, 1006, 791, 768, 661, 631, 630, 615, 592, 577, 576, 561, 546, 523, 508, 493, 492, 491, 476, 475, 474, 473, 472, 471, 470, 469, 468, 453, 452, 451, 450, 449, 448, 447, 446, 0};

int cumf_TCOEFr_intra[104]={16383, 12514, 10776, 9969, 9579, 9306, 9168, 9082, 9032, 9000, 8981, 8962, 8952, 7630, 7212, 7053, 6992, 6961, 6940, 6195, 5988, 5948, 5923, 5370, 5244, 5210, 4854, 4762, 4740, 4384, 4300, 4288, 4020, 3968, 3964, 3752, 3668, 3511, 3483, 3354, 3322, 3205, 3183, 3108, 3046, 2999, 2981, 2974, 2968, 2961, 2955, 2949, 2943, 2942, 2939, 2935, 2934, 2933, 2929, 2270, 2178, 2162, 1959, 1946, 1780, 1651, 1524, 1400, 1289, 1133, 1037, 942, 849, 763, 711, 591, 521, 503, 496, 474, 461, 449, 442, 436, 426, 417, 407, 394, 387, 377, 373, 370, 367, 366, 365, 364, 363, 362, 358, 355, 352, 351, 350, 0};

int cumf_SIGN[3]={16383, 8416, 0};

int cumf_LAST[3]={16383, 9469, 0};

int cumf_LAST_intra[3]={16383, 2820, 0};

int cumf_RUN[65]={16383, 15310, 14702, 13022, 11883, 11234, 10612, 10192, 9516, 9016, 8623, 8366, 7595, 7068, 6730, 6487, 6379, 6285, 6177, 6150, 6083, 5989, 5949, 5922, 5895, 5828, 5774, 5773, 5394, 5164, 5016, 4569, 4366, 4136, 4015, 3867, 3773, 3692, 3611, 3476, 3341, 3301, 2787, 2503, 2219, 1989, 1515, 1095, 934, 799, 691, 583, 435, 300, 246, 206, 125, 124, 97, 57, 30, 3, 2, 1, 0};

int cumf_RUN_intra[65]={16383, 10884, 8242, 7124, 5173, 4745, 4246, 3984, 3034, 2749, 2607, 2298, 966, 681, 396, 349, 302, 255, 254, 253, 206, 159, 158, 157, 156, 155, 154, 153, 106, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0};

int cumf_LEVEL[255]={16383, 16382, 16381, 16380, 16379, 16378, 16377, 16376, 16375, 16374, 16373, 16372, 16371, 16370, 16369, 16368, 16367, 16366, 16365, 16364, 16363, 16362, 16361, 16360, 16359, 16358, 16357, 16356, 16355, 16354, 16353, 16352, 16351, 16350, 16349, 16348, 16347, 16346, 16345, 16344, 16343, 16342, 16341, 16340, 16339, 16338, 16337, 16336, 16335, 16334, 16333, 16332, 16331, 16330, 16329, 16328, 16327, 16326, 16325, 16324, 16323, 16322, 16321, 16320, 16319, 16318, 16317, 16316, 16315, 16314, 16313, 16312, 16311, 16310, 16309, 16308, 16307, 16306, 16305, 16304, 16303, 16302, 16301, 16300, 16299, 16298, 16297, 16296, 16295, 16294, 16293, 16292, 16291, 16290, 16289, 16288, 16287, 16286, 16285, 16284, 16283, 16282, 16281, 16280, 16279, 16278, 16277, 16250, 16223, 16222, 16195, 16154, 16153, 16071, 15989, 15880, 15879, 15878, 15824, 15756, 15674, 15606, 15538, 15184, 14572, 13960, 10718, 7994, 5379, 2123, 1537, 992, 693, 611, 516, 448, 421, 380, 353, 352, 284, 257, 230, 203, 162, 161, 160, 133, 132, 105, 104, 103, 102, 101, 100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0};

int cumf_LEVEL_intra[255]={16383, 16379, 16378, 16377, 16376, 16375, 16374, 16373, 16372, 16371, 16370, 16369, 16368, 16367, 16366, 16365, 16364, 16363, 16362, 16361, 16360, 16359, 16358, 16357, 16356, 16355, 16354, 16353, 16352, 16351, 16350, 16349, 16348, 16347, 16346, 16345, 16344, 16343, 16342, 16341, 16340, 16339, 16338, 16337, 16336, 16335, 16334, 16333, 16332, 16331, 16330, 16329, 16328, 16327, 16326, 16325, 16324, 16323, 16322, 16321, 16320, 16319, 16318, 16317, 16316, 16315, 16314, 16313, 16312, 16311, 16268, 16267, 16224, 16223, 16180, 16179, 16136, 16135, 16134, 16133, 16132, 16131, 16130, 16129, 16128, 16127, 16126, 16061, 16018, 16017, 16016, 16015, 16014, 15971, 15970, 15969, 15968, 15925, 15837, 15794, 15751, 15750, 15749, 15661, 15618, 15508, 15376, 15288, 15045, 14913, 14781, 14384, 13965, 13502, 13083, 12509, 12289, 12135, 11892, 11738, 11429, 11010, 10812, 10371,

9664, 9113, 8117, 8116, 8028, 6855, 5883, 4710, 4401, 4203, 3740, 3453, 3343, 3189, 2946, 2881, 2661, 2352, 2132, 1867, 1558, 1382, 1250, 1162, 1097, 1032, 967, 835, 681, 549, 439, 351, 350, 307, 306, 305, 304, 303, 302, 301, 300, 299, 298, 255, 212, 211, 210, 167, 166, 165, 164, 163, 162, 161, 160, 159, 158, 115, 114, 113, 112, 111, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0};

Annex F

Advanced Prediction mode

(This annex forms an integral part of this Recommendation)

F.1. Introduction.

This annex describes the optional Advanced Prediction mode of H.263, including overlapped block motion compensation and the possibility of four motion vectors per macroblock. The capability of this mode is signalled by external means (for example Recommendation H.245). The use of this mode is indicated in PTYPE. In the Advanced Prediction mode, motion vectors are allowed to cross picture boundaries as is the case in the Unrestricted Motion Vector mode (for the description of this technique refer to Annex D, section D.1). The extended motion vector range feature of the Unrestricted Motion Vector mode is not automatically included in the Advanced Prediction mode, and only is active if the Unrestricted Motion Vector mode is selected. If the Advanced Prediction mode is used in combination with the PB-frames mode, overlapped motion compensation is only used for prediction of the P-pictures, not for the B-pictures.

F.2. Four motion vectors per macroblock

In H.263, one motion vector per macroblock is used except when in Advanced Prediction mode. In this mode, the one/four vectors decision is indicated by the MCBPC codeword for each macroblock. If only one motion vector is transmitted for a certain macroblock, this is defined as four vectors with the same value. If MCBPC indicates that four motion vectors are transmitted for the current macroblock, the information for the first motion vector is transmitted as the codeword MVD and the information for the three additional motion vectors is transmitted as the codewords MVD_{2-4} (see also section 5.3.7 and 5.3.8).

The vectors are obtained by adding predictors to the vector differences indicated by MVD and MVD_{2-4} in a similar way as when only one motion vector per macroblock is present, according to the decision rules given in section 6.1.1. Again the predictors are calculated separately for the horizontal and vertical components. However, the candidate predictors MV1, MV2 and MV3 are redefined as indicated in FIGURE 16/H.263. If only one vector per macroblock is present, MV1, MV2 and MV3 are defined as for the 8*8 block numbered 1 in FIGURE 5/H.263 (this definition is given in the upper left of the four sub-figures of FIGURE 16/H.263).

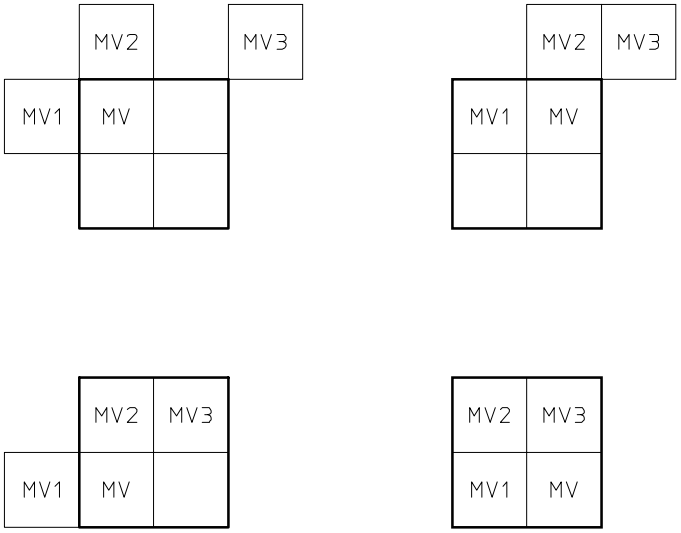


FIGURE 16/H.263

Redefinition of the candidate predictors MV1, MV2 and MV3 for each of the luminance blocks in a macroblock

If four vectors are used, each of the motion vectors is used for all pixels in one of the four luminance blocks in the macroblock. The numbering of the motion vectors is equivalent to the numbering of the four luminance blocks as given in FIGURE 5/H.263. Motion vector MVD_{CHR} for both chrominance blocks is derived by calculating the sum of the four luminance vectors and dividing this sum by 8; the component values of the resulting sixteenth pixel resolution vectors are modified towards the nearest half pixel position as indicated in TABLE 16/H.263.

TABLE 16/H.263

Modification of sixteenth pixel resolution chrominance vector components

sixteenth pixel position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	/16
resulting position	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2	2	/2

Half pixel values are found using bilinear interpolation as described in section 6.1.2. In Advanced Prediction mode, the prediction for luminance is obtained by overlapped motion compensation as described in section F3. The prediction for chrominance is obtained by applying the motion vector MVD_{CHR} to all pixels in the two chrominance blocks (as it is done in the default prediction mode).

F.3. Overlapped motion compensation for luminance

Each pixel in an 8*8 luminance prediction block is a weighted sum of three prediction values, divided by 8 (with rounding). In order to obtain the three prediction values, three motion vectors are used: the motion vector of the current luminance block, and two out of four "remote" vectors:

- the motion vector of the block at the left or right side of the current luminance block;
- the motion vector of the block above or below the current luminance block.

Remote motion vectors from other GOBs are used in the same way as remote motion vectors inside the current GOB.

For each pixel, the remote motion vectors of the blocks at the two nearest block borders are used. This means that for the upper half of the block the motion vector corresponding to the block above the current block is used, while for the lower half of the block the motion vector corresponding to the block below the current block is used (see FIGURE 18/H.263). Similarly, for the left half of the block the motion vector corresponding to the block at the left side of the current block is used, while for the right half of the block the motion vector corresponding to the block at the right side of the current block is used (see FIGURE 19/H.263).

Let (x,y) be a position in a picture measured in integer pixel units,

let (i,j) be a position in a picture measured in half pixel units, and

let MV^n_x and MV^n_y be motion vector components measured in half pixel units,

then ~~T~~the creation of each pixel, ~~$p(i,j$~~$P(x,y)$, in an 8*8 luminance prediction block is governed by the following equation:

$$~~p(i,j)~~P(x,y) = (q(i,j) \times H_0(~~i,j~~x,y) + r(i,j) \times H_1(~~i,j~~x,y) + s(i,j) \times H_2(~~i,j~~x,y) + 4)/8,$$

where $q(i,j)$, $r(i,j)$ and $s(i,j)$ are the ~~pixels~~ prediction values taken from the referenced picture as defined by

$$q(i,j) = p(~~i~~ 2*x + MV^0_x, ~~j~~ 2*y + MV^0_y),$$
$$r(i,j) = p(~~i~~ 2*x + MV^1_x, ~~j~~ 2*y + MV^1_y),$$
$$s(i,j) = p(~~i~~ 2*x + MV^2_x, ~~j~~ 2*y + MV^2_y),$$

where $p(i,j)$ is the prediction value at position (i,j) in the referenced picture (in Unrestricted Motion Vector mode, (i,j) may be outside the picture). Note that $p(i,j)$ can be at a full or half pixel position (see also section 6.1.2).

~~Here,~~ (MV^0_x, MV^0_y) denotes the motion vector for the current block, (MV^2_x, MV^2_y) denotes the motion vector of the block either above or below, and (MV^2_x, MV^2_y) denotes the motion vector either to the left or right of the current block as defined above.

The matrices $H_0(~~i,j~~x,y)$, $H_1(~~i,j~~x,y)$ and $H_2(~~i,j~~x,y)$ are defined in FIGURE 17/H.263, FIGURE 18/H.263 and FIGURE 19/H.263, where ~~(i, j)~~ (x,y) denotes the column and row, respectively, of the matrix.

If one of the surrounding macroblocks was not coded, the corresponding remote motion vector is set to zero. If one of the surrounding blocks was INTRA coded ~~in INTRA mode~~, the corresponding remote motion vector is replaced by the motion vector for the current block except when in PB-frames mode. In this case (INTRA block in PB-frame mode), the INTRA block's motion vector is used (also see Annex G). If the current block is at the border of the picture and therefore a surrounding block is not present, the corresponding remote motion vector is replaced by the current motion vector. In all cases~~addition~~, if the current block is at the bottom of the macroblock (for block number 3 or 4, see FIGURE 5/H.263), the

remote motion vector corresponding with an 8*8 luminance block in the macroblock below the current macroblock is replaced by the motion vector for the current block.

The weighting values for the prediction are given in FIGURE 17/H.263, FIGURE 18/H.263 and FIGURE 19/H.263.

4	5	5	5	5	5	5	4
5	5	5	5	5	5	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	4

FIGURE 17/H.263

Weighting values, H_0, for prediction with motion vector of current luminance block

2	2	2	2	2	2	2	2
1	1	2	2	2	2	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	2	2	2	2	1	1
2	2	2	2	2	2	2	2

FIGURE 18/H.263

Weighting values, H_1, for prediction with motion vectors of the luminance blocks on top or bottom of current luminance block

2	1	1	1	1	1	1	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	1	1	1	1	1	1	2

FIGURE 19/H.263

Weighting values, H_2, for prediction with motion vectors of the luminance blocks to the left or right of current luminance block

Annex G

PB-frames mode

(This annex forms an integral part of this Recommendation)

G.1. Introduction.

This annex describes the optional PB-frames mode of H.263. The capability of this mode is signalled by external means (for example Recommendation H.245). The use of this mode is indicated in PTYPE.

A PB-frame consists of two pictures being coded as one unit. The name PB comes from the name of picture types in Recommendation H.262 where there are P-pictures and B-pictures. Thus a PB-frame consists of one P-picture which is predicted from the previous decoded P-picture and one B-picture which is predicted both from the previous decoded P-picture and the P-picture currently being decoded. The name B-picture was chosen because parts of B-pictures may be bidirectionally predicted from the past and future pictures. The prediction process is illustrated in FIGURE 20/H.263.

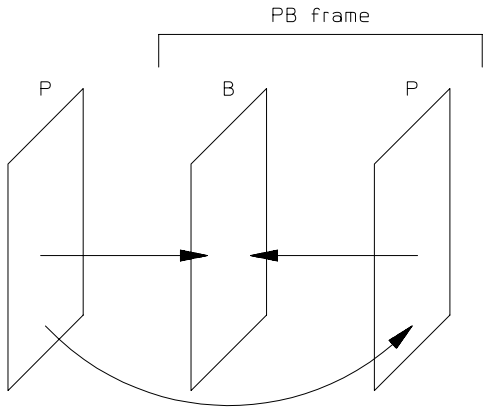


FIGURE 20/H.263

Prediction in PB-frames mode

G.2. PB-frames and INTRA blocks

When PB-frames are used the coding mode INTRA has the following meaning (see also section 5.3.2):

- The P-blocks are INTRA coded.
- The B-blocks are INTER coded with prediction as for an INTER block.

If PB-frames are used, motion vector data (MVD) is included also for INTRA macroblocks in pictures for which PTYPE indicates 'INTER'. In this case the vector is used for the B-blocks only. The codewords MVD_{2-4} are never used for INTRA (see also TABLE 7/H.263). When in both the Advanced Prediction mode and the PB-frames mode, and one of the surrounding blocks was coded in INTRA mode, the corresponding remote motion vector is not replaced by the motion vector for the current block. Instead, the remote 'INTRA' motion vector is used.

G.3. Block Layer

In a PB-frame, a macroblock comprises twelve blocks. First the data for the six P-blocks is transmitted as in the default H.263 mode, then the data for the six B-blocks (see also section 5.4). The structure of the block layer is shown in FIGURE 10/H.263. INTRADC is present for every P-block of the macroblock if MCBPC indicates MB type 3 or 4 (see TABLE 4/H.263 and TABLE 5/H.263). INTRADC is not present for B-blocks. TCOEF is present for P-blocks if indicated by MCBPC or CBPY; TCOEF is present for B-blocks if indicated by CBPB.

G.4. Calculation of vectors for the B-picture in a PB-frame

The vectors for the B-picture are calculated as follows (see also section 6.1.1). Assume we have a vector component MV in half pixel units to be used in the P-picture (MV represents a vector component for an 8*8 luminance block; if only one vector per macroblock is transmitted, MV has the same value for each of the four 8*8 luminance blocks). For prediction of the B-picture we need both forward and backward vector components MV_F and MV_B. These forward and backward vector components are derived from MV and eventually enhanced by a delta vector given by MVDB.

TR_D: The increment of temporal reference TR from the last picture header (see section 5.1.2). If TR_D is negative then $TR_D = TR_D + 256$.

TR_B: See section 5.1.7.

Assume that MV_D is the delta vector component given by MVDB and corresponding with vector component MV. If MVDB is not present, MV_D is set to zero. If MVDB is present, the same MV_D given by MVDB is used for each of the four luminance B-blocks within the macroblock.

Now MV_F and MV_B are given in half pixel units by the following formulas:

$$MV_F = (TR_B \times MV) \ / \ TR_D + MV_D$$

$$MV_B = ((TR_B - TR_D) \times MV) \ / \ TR_D \qquad \text{if } MV_D \text{ is equal to 0}$$

$$MV_B = MV_F - MV \qquad \text{if } MV_D \text{ is unequal to 0}$$

where "/" means division by truncation. It is assumed that the scaling reflects the actual position in time of P- and B-pictures. Advantage is taken of the fact that the range of values for MV_F is constrained. Each VLC word for MVDB represents a pair of difference values. Only one of the pair will yield a value for MV_F falling within the permitted range (default [-16,15.5]; in Unrestricted Motion Vector mode [-31.5,31.5]). The formulas for MV_F and MV_B are also used in the case of INTRA blocks where the vector data is used only for predicting B-blocks.

For chrominance blocks, MV_F is derived by calculating the sum of the four corresponding luminance MV_F vectors and dividing this sum by 8; the resulting sixteenth pixel resolution vector components are modified towards the nearest half pixel position as indicated in TABLE 16/H.263. MV_B for chrominance is derived by calculating the sum of the four corresponding luminance MV_B vectors and dividing this sum by 8; the resulting sixteenth pixel resolution vector components are modified towards the nearest half pixel position as indicated in TABLE 16/H.263.

A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pixels in the referenced picture which are spatially to the right or below the pixels being predicted.

G.5. Prediction of a B-block in a PB-frame

In this section a block means an 8x8 block. The following procedure applies for luminance as well as chrominance blocks. First, the forward and backward vectors are calculated. It is assumed that the P-macroblock (luminance and chrominance) is first decoded, reconstructed and clipped (see section 6.3.2). This macroblock is called P_{REC}. Based on P_{REC} and the prediction for P_{REC}, the prediction for the B-block is calculated.

The prediction of the B-block has two modes that are used for different parts of the block:

- For pixels where the backward vector - MV_B -points inside P_{REC}, use bi-directional prediction. This is obtained as the average of the forward prediction using MV_F relative to the previous decoded picture, and the backward prediction using MV_B relative to P_{REC}. The average is calculated by dividing the sum of the two predictions by two (division by truncation).
- For all other pixels, forward prediction using MV_F relative to the previous decoded picture is used.

FIGURE 21/H.263 indicates which part of a block is predicted bidirectionally (shaded part of the B-block) and which part with forward prediction only (rest of the B-block).

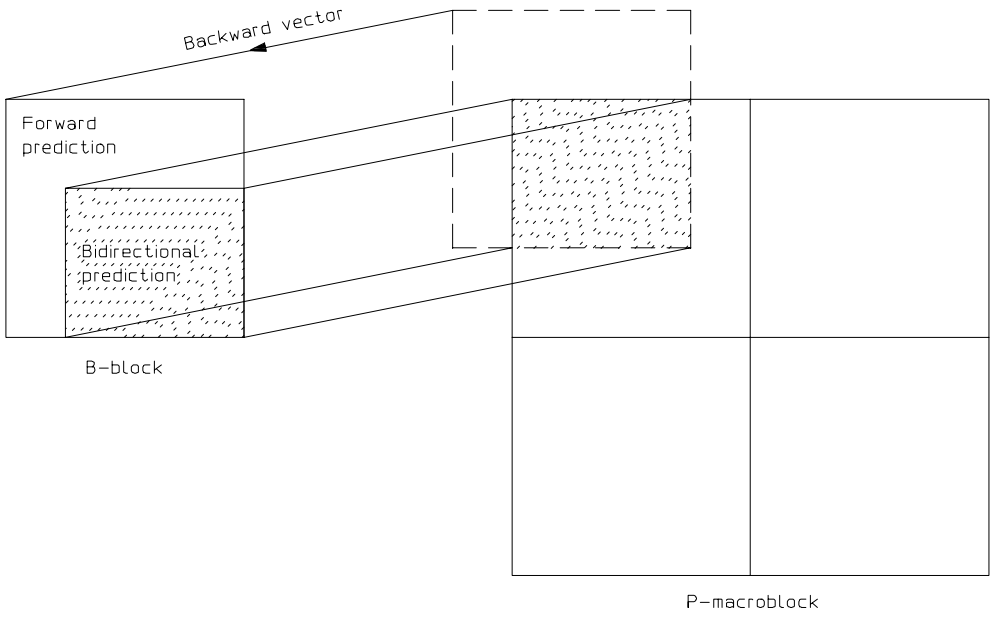


FIGURE 21/H.263

Forward and bidirectional prediction for a B-block

Bi-directional prediction is used for pixels where the backward vector - MV_B -points inside P_{REC}. These pixels are defined by the following procedures which are specified in C.

Definitions:	`nh`:	horizontal position of block within a macroblock (0 or 1).
	`nv`:	vertical position of block within a macroblock (0 or 1).
	`mh(nh,nv)`:	horizontal vector component of block `(nh,nv)` in half pixel units.
	`mv(nh,nv)`:	vertical vector component of block `(nh,nv)` in half pixel units.
	`mhc`:	horizontal chrominance vector component.
	`mvc`:	vertical chrominance vector component.

Procedure for luminance

```
for (nh = 0; nh <= 1; nh++) {
   for (nv = 0; nv <= 1; nv++) {
      for (i = nh*8 + max(0,(-mh(nh,nv)+1)/2 - nh*8);
            i <= nh*8 + min(7,15-(mh(nh,nv)+1)/2 - nh*8); i++) {
         for (j = nv*8 + max(0,(-mv(nh,nv)+1)/2 - nv*8);
               j <= nv*8 + min(7,15-(mv(nh,nv)+1)/2 - nv*8); j++) {
            predict pixel (i,j) bidirectionally
         }
      }
   }
}
```

Procedure for chrominance

```
for (i = max(0,(-mhc+1)/2); i <= min(7,7-(mhc+1)/2); i++) {
   for (j = max(0,(-mvc+1)/2); j <= min(7,7-(mvc+1)/2); j++) {
      predict pixel (i,j) bidirectionally;
   }
}
```

Pixels not predicted bidirectionally are predicted with forward prediction only.

Annex H

Forward Error Correction for coded video signal

(This annex forms an integral part of this Recommendation)

H.1. Introduction

This annex describes an optional forward error correction method (code and framing) for transmission of H.263 encoded video data. This forward error correction may be used in situations where no forward error correction is provided by external means, for example at the multiplex or system level. It is not used for H.324. Both the framing and the forward error correction code are the same as in H.261.

H.2. Error correction framing

To allow the video data and error correction parity information to be identified by a decoder an error correction framing pattern is included. This pattern consists of multiframes of eight frames, each frame comprising 1 bit framing, 1 bit fill indicator (Fi), 492 bits of coded data (or fill all 1s) and 18 bits parity (see FIGURE 22/H.263). For each multiframe the frame alignment pattern formed by the framing bits of the eight individual frames is:

$(S_1S_2S_3S_4S_5S_6S_7S_8) = (00011011)$.

The fill indicator (Fi) can be set to zero by an encoder. In this case 492 consecutive fill bits (fill all 1s) are used instead of 492 bits of coded data. This may be used for stuffing data (see section 3.6).

H.3. Error correcting code

The error correction code is a BCH (511,493) forward error correction code. Use of this by the decoder is optional. The parity is calculated against a code of 493 bits, comprising 1 bit fill indicator (Fi) and 492 bits of coded video data.

The generator polynomial is: $g(x) = (x9 + x4 + 1)(x9 + x6 + x4 + x3 + 1)$.

Example: for the input data of “01111 . . . 11” (493 bits) the resulting correction parity bits are “011011010100011011” (18 bits).

H.4. Relock time for error corrector framing

Three consecutive error correction frame alignment patterns (24 bits) should be received before frame lock is deemed to have been achieved. The decoder should be designed such that frame lock will be re-established within 34 000 bits after an error corrector framing phase change.

Note – This assumes that the video data does not contain three correctly phased emulations of the error correction framing sequence during the relocking period.

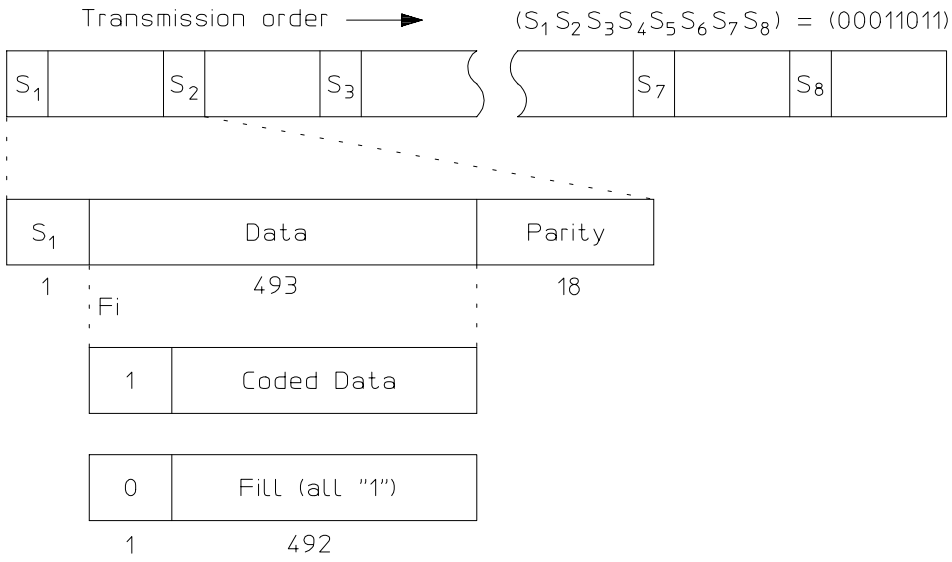


FIGURE 22/H.263

Error correcting frame

Appendix I

List of patent holders

(This appendix does not form an integral part of this Recommendation)