INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.263

(02/98)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving video

---

Video coding for low bit rate communication

ITU-T Recommendation H.263

(Previously CCITT Recommendation)

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Characteristics of transmission channels used for other than telephone purposes	H.10–H.19
Use of telephone-type circuits for voice-frequency telegraphy	H.20–H.29
Telephone circuits or cables used for various types of telegraph transmission or simultaneous transmission	H.30–H.39
Telephone-type circuits used for facsimile telegraphy	H.40–H.49
Characteristics of data signals	H.50–H.99
CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	**H.260–H.279**
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.399

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION H.263

VIDEO CODING FOR LOW BIT RATE COMMUNICATION

Summary

This Recommendation specifies a coded representation that can be used for compressing the moving picture component of audio-visual services at low bit rates. The basic configuration of the video source coding algorithm is based on 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 standardized video source formats: sub-QCIF, QCIF, CIF, 4CIF and 16CIF, and can also operate using a broad range of custom video formats.

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, sixteen negotiable coding options are included for improved compression performance and the support of additional capabilities. Additional supplemental information may also be included in the bitstream for enhanced display capability and for external usage.

Source

ITU-T Recommendation H.263 was revised by ITU-T Study Group 16 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 6th of February 1998.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. 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, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Recommendation H.263

VIDEO CODING FOR LOW BIT RATE COMMUNICATION

(revised in 1998)

1 Scope

This Recommendation specifies a coded representation that can be used for compressing the moving picture component of audio-visual services at low bit rates. The basic configuration of the video source coding algorithm is based on Recommendation H.261. Sixteen negotiable coding options are included for improved performance and increased functionality. This Recommendation contains Version 2 of Recommendation H.263, which is fully compatible with the original Recommendation, adding only optional features to the original Version 1 content of the Recommendation.

2 References

The following 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.

[1] ITU-R Recommendation BT.601-5 (1995), *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios.*

Reference [1] is referenced herein to define the (Y, C_B, C_R) colour space and its 8-bit integer representation for pictures used by video codecs designed according to this Recommendation. (Reference [1] is not used to define any other aspects of this Recommendation.)

The following additional ITU-T Recommendations are mentioned for illustration purposes in this text; however, they do not 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.

[2] ITU-T Recommendation H.223 (1996), *Multiplexing protocol for low bit rate multimedia communication.*

[3] ITU-T Recommendation H.242 (1997), *System for establishing communication between audiovisual terminals using digital channels up to 2 Mbit/s.*

[4] ITU-T Recommendation H.245 (1998), *Control protocol for multimedia communication.*

[5] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at $p \times 64$ kbit/s.*

[6] ITU-T Recommendation H.262 (1995) | ISO/IEC 13818-2:1996, *Information technology – Generic coding of moving pictures and associated audio information: Video.*

[7] ITU-T Recommendation H.324 (1998), *Terminal for low bit rate multimedia communication.*

3 Brief specification

An outline block diagram of the codec is given in Figure 1.

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 standardized source formats on which the source coder operates are based on a Common Intermediate Format (CIF). It is also possible using external negotiation (for example, Recommendation H.245), to enable the use of a wide range of optional custom source formats. 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 bitstream 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.

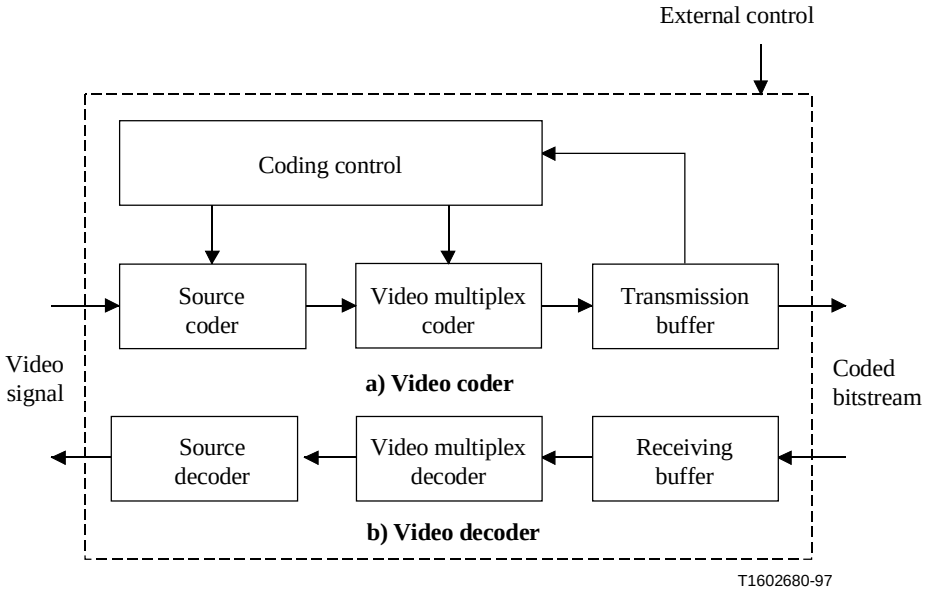


Figure 1/H.263 – Outline block diagram of the video codec

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, sixteen negotiable coding options can be used, either together or separately (subject to certain restrictions). Additional supplemental information may also be included in the bitstream for enhanced display capability and for external usage. A forward error correction method for application to the resulting video bitstream is provided for use when necessary. The negotiable coding options, forward error correction, and supplemental information usage are described in the subsequent subclauses.

3.4.1 Continuous Presence Multipoint and Video Multiplex mode

In this optional mode, up to four separate video "Sub-Bitstreams" can be sent within the same video channel. This feature is designed for use in continuous presence multipoint application or other situations in which separate logical channels are not available, but the use of multiple video bitstreams is desired (see also Annex C).

3.4.2 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 "non-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 and large picture formats.

3.4.3 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).

3.4.4 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 8×8 vectors instead of one 16×16 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. A subjective gain is achieved because OBMC results in less blocking artifacts.

3.4.5 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 substantially increasing the bit rate (see also Annex G). However, an Improved PB-frames mode is also provided (see also Annex M). The original PB-frames mode is retained herein only for purposes of compatibility with systems made prior to the adoption of the Improved PB-frames mode.

3.4.6 Forward Error Correction

A forward error correction method is specified for use when necessary for the protection of the video bitstream for use when appropriate. The method provided for forward error correction is the same BCH code method specified also in Recommendation H.261 (see also Annex H).

3.4.7 Advanced INTRA Coding mode

In this optional mode, INTRA blocks are first predicted from neighbouring INTRA blocks prior to coding (see also Annex I). Separate Variable Length Code (VLC) tables are defined for the INTRA blocks. The technique is applied to INTRA-macroblocks within INTRA pictures and to INTRA-macroblocks within INTER pictures. This mode significantly improves the compression performance over the INTRA coding of the core H.263 syntax.

3.4.8 Deblocking Filter mode

In this optional mode, a filter is applied across the 8×8 block edge boundaries of decoded I- and P-pictures to reduce blocking artifacts (see also Annex J). The purpose of the filter is to mitigate the occurrence of block edge artifacts in the decoded picture. The filter affects the picture that is used for the prediction of subsequent pictures and thus lies within the motion prediction loop.

3.4.9 Slice Structured mode

In this optional mode, a "slice" layer is substituted for the GOB layer of the bitstream syntax (see also Annex K). The purposes of this mode are to provide enhanced error resilience capability, to make the bitstream more amenable to use with an underlying packet transport delivery, and to minimize video delay. A slice is similar to a GOB, in that it is a layer of the syntax that lies between the picture layer and the macroblock layer. However, the use of a slice layer allows a flexible partitioning of the picture, in contrast with the fixed partitioning and fixed transmission order required by the GOB structure.

3.4.10 Supplemental enhancement information

Additional supplemental information may be included in the bitstream to signal enhanced display capability or to provide information for external usage (see also Annex L). This supplemental information can be used to signal a full-picture or partial-picture freeze or freeze-release request with or without resizing, can also be used to tag particular pictures or sequences of pictures within the video stream for external usage, and can be used to convey chroma key information for video compositing. The supplemental information may be present in the bitstream even though the decoder may not be capable of providing the enhanced capability to use it, or even to properly interpret it – simply discarding the supplemental information is allowable by decoders unless a requirement to provide the requested capability has been negotiated by external means.

3.4.11 Improved PB-frames mode

This optional mode represents an improvement compared to the PB-frames mode option (see also Annexes G and M). The main difference between the two modes is that in the Improved PB-frames mode, each B-block may be forward predicted using a separate motion vector or backward predicted with a zero vector. This significantly improves coding efficiency in situations in which downscaled P-vectors are not good candidates for B-prediction. The backward prediction is particularly useful when there is a scene cut between the previous P-frame and the PB-frame.

3.4.12 Reference Picture Selection mode

An optional mode is provided which improves the performance of real-time video communication over error-prone channel by allowing temporal prediction from pictures other than the most recently-sent reference picture (see also Annex N). This mode can be used with back-channel status messages which are sent back to the encoder to inform it about whether its bitstream is being properly received. In error-prone channel environments, this mode allows the encoder to optimize its video encoding for the conditions of the channel.

3.4.13 Temporal, SNR and Spatial Scalability mode

In this optional mode, there is support for temporal, SNR, and spatial scalability (see also Annex O). Scalability implies that a bitstream is composed of a base layer and one or more associated enhancement layers. The base layer is a separately decodable bitstream. The enhancement layers can be decoded in conjunction with the base layer to increase perceived quality by either increasing the picture rate, increasing the picture quality, or increasing the picture size. SNR scalability refers to enhancement information to increase the picture quality without increasing picture resolution. Spatial scalability refers to enhancement information to increase the picture quality by increasing picture resolution either horizontally, vertically, or both. There is also support for temporal scalability by the use of B-pictures. A B-picture is a scalability enhancement containing pictures that can be bidirectionally predicted from two pictures in a reference layer, one temporally previous to the current picture and one temporally subsequent. B-pictures allows enhancement layer information to be used to increase perceived quality by increasing the picture rate of the displayed enhanced video sequence. This mode can be useful for heterogenous networks with varying bandwidth capacity and also in conjunction with error correction schemes.

3.4.14 Reference Picture Resampling mode

A syntax is provided for supporting an optional mode for which the reference picture used for video image prediction is processed by a resampling operation prior to its use in forming a predictor for the current input picture (see also Annex P). This allows efficient dynamic selection of an appropriate image resolution for video encoding, and can also support picture warping for use as a global motion compensator or special-effect generator.

3.4.15 Reduced-Resolution Update mode

An optional mode is provided which allows reduced-resolution updates to a reference picture having a higher resolution (see also Annex Q). This mode is expected to be used when encoding a highly active scene, and allows an encoder to increase the picture rate at which moving parts of a scene can be represented, while maintaining a higher resolution representation in more static areas of the scene.

3.4.16 Independent Segment Decoding mode

An optional mode is provided which allows a picture to be constructed without any data dependencies which cross GOB or multi-GOB video picture segments or slice boundaries (see also Annex R). This mode provides error robustness by preventing the propagation of erroneous data across the boundaries of the video picture segment areas.

3.4.17 Alternative INTER VLC mode

An optional mode is provided which improves the efficiency of INTER picture coding when significant changes are evident in the picture (see also Annex S). This efficiency improvement is obtained by allowing a VLC code originally designed for INTRA pictures to be used for some INTER picture coefficients as well.

3.4.18 Modified Quantization mode

An optional mode is provided which improves the bit-rate control ability for encoding, reduces chrominance quantization error, expands the range of representable DCT coefficients, and places certain restrictions on coefficient values (see also Annex T). This mode modifies the semantics of the differential quantization step size parameter of the bitstream by broadening the range of step size changes that can be specified. It also reduces the quantization step size used for chrominance data. The range of DCT coefficient levels is broadened to ensure that any possible coefficient value can be

encoded to within the accuracy allowed by the step size. Certain restrictions are also placed on coefficients in this mode to increase error detection performance and minimize decoder complexity.

3.5 Bit rate

The transmission clock is provided externally. The video bit rate may be variable. In this Recommendation no constraints on the video bit rate 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 Tables 7 and 8) 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 picture size that has been negotiated for use in the bitstream (see Table 1). The picture size is measured as the picture width times height for the luminance (Y) component, measured in pixels. An encoder may use a larger value for BPPmaxKb than as specified in Table 1, provided the larger value is first negotiated by external means, for example Recommendation H.245.

When the Temporal, SNR, and Spatial Scalability mode (Annex O) is employed, the number of bits sent for each picture in each enhancement layer shall not exceed the maximum value specified by BPPmaxKb.

Table 1/H.263 – Minimum BPPmaxKb for different source picture formats

Y picture size in pixels	Minimum BPPmaxKb
up to 25 344 (or QCIF)	64
25 360 to 101 376 (or CIF)	256
101 392 to 405 504 (or 4CIF)	512
405 520 and above	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 (or slices if Annex K is employed) 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 if the Slice Structured mode (see Annex K) is not in use. 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 having a source format which is defined in terms of:

1) the picture format, as determined by the number of pixels per line, the number of lines per picture, and the pixel aspect ratio; and

2) the timing between pictures, as determined by the Picture Clock Frequency (PCF). For example, the Common Intermediate Format (CIF) has 352 pixels per line, 288 lines, a pixel aspect ratio of 12:11, and a picture clock frequency of 30 000/1001 pictures per second.

The source coder operates on non-interlaced pictures occurring at a Picture Clock Frequency (PCF) of 30 000/1001 (approximately 29.97) times per second, termed the CIF PCF. It is also possible to negotiate the use of an optional custom PCF by external means. A custom PCF is given by 1 800 000/(clock divisor * clock conversion factor) where clock divisor can have values of 1 through 127 and clock conversion factor can either be 1000 or 1001. The tolerance on the picture clock 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 ITU-R Recommendation BT.601-5.

- 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 standardized picture formats: sub-QCIF, QCIF, CIF, 4CIF and 16CIF. It is also possible to negotiate a custom picture format. For all 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 for each of the standardized picture formats.

Table 2/H.263 – Number of pixels per line and number of lines for each of the standardized 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 all picture formats, colour difference samples are sited such that their block boundaries coincide with luminance block boundaries as shown in Figure 2. The pixel aspect ratio is the same for each of the standardised picture formats and is the same as defined for QCIF and CIF in Recommendation H.261: (288/3):(352/4), which simplifies to 12:11 in relatively prime numbers. The picture area covered by all of the standardized picture formats except the sub-QCIF picture format has an aspect ratio of 4:3.

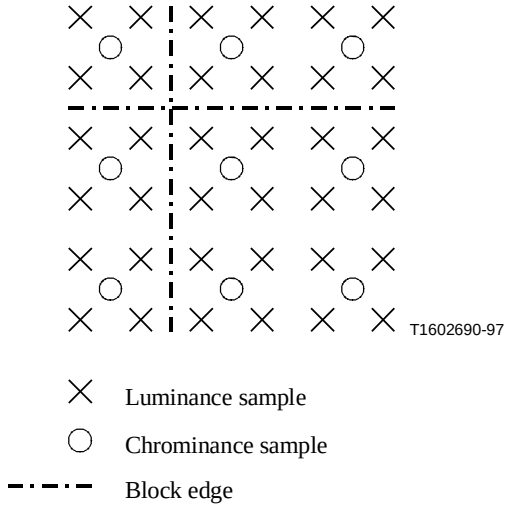


Figure 2/H.263 – Positioning of luminance and chrominance samples

Custom picture formats can have a custom pixel aspect ratio as described in Table 3, if the custom pixel aspect ratio use is first negotiated by external means. Custom picture formats can have any number of lines and any number of pixels per line, provided that the number of lines is divisible by four and is in the range [4, ..., 1152], and provided that the number of pixels per line is also divisible by four and is in the range [4, ..., 2048]. For picture formats having a width or height that is not divisible by 16, the picture is decoded in the same manner as if the width or height had the next larger size that would be divisible by 16 and then the picture is cropped at the right and the bottom to the proper width and height for display purposes only.

Table 3/H.263 – Custom pixel aspect ratios

Pixel aspect ratio	Pixel width:Pixel height
Square	1:1
CIF	12:11
525-type for 4:3 picture	10:11
CIF for 16:9 picture	16:11
525-type for 16:9 picture	40:33
Extended PAR	m:n, m and n are relatively prime

All decoders and encoders shall be able to operate using the CIF picture clock frequency. Some decoders and encoders may also support custom picture clock frequencies. All decoders shall be able to operate using the sub-QCIF picture format. All decoders shall also be able to operate using the QCIF picture format. Some decoders may also operate with CIF, 4CIF or 16CIF, or custom picture formats. Encoders shall be able to operate with one of the picture formats sub-QCIF and QCIF. The

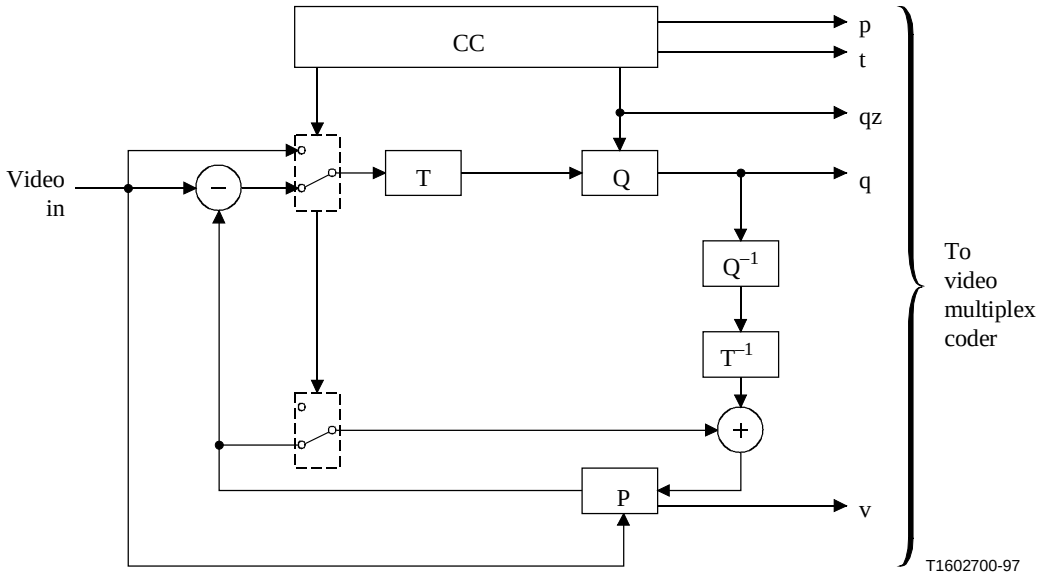
encoders determine which of these two formats are used, and are not obliged to be able to operate with both. Some encoders can also operate with CIF, 4CIF, 16CIF, or custom picture formats. Which optional formats and which picture clock frequencies can be handled by the decoder is signalled by external 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 for practical purposes 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 ITU-R Recommendation BT.601-5.

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.

4.2 Video source coding algorithm

The source coder is shown in generalized form in Figure 3. 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, slices, macroblocks and blocks

Each picture is divided either into Groups Of Blocks (GOBs) or into slices.

A Group Of Blocks (GOB) comprises of up to k * 16 lines, where k depends on the number of lines in the picture format and depends on whether the optional Reduced-Resolution Update mode is in use (see Annex Q). The dependencies are shown in Table 4. If the number of lines is less than or equal to 400 and the optional Reduced-Resolution Update mode is not in use, then k = 1. If the number of lines is less than or equal to 800 and the optional Reduced-Resolution Update mode is in use or the number of lines is more than 400, then k = 2. If the number of lines is more than 800, then k = 4. When using custom picture sizes, the number of lines in the last (bottom-most) GOB may be less than k * 16 if the number of lines in the picture is not divisible by k * 16. However, every GOB in each of the standardized picture formats has k * 16 lines, as the number of lines in each standardized picture format is an integer multiple of k * 16. Thus, for example, if the optional Reduced-Resolution mode is not in use, 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 bottom-most GOB. An example of the arrangement of GOBs in a picture is given for the CIF picture format in Figure 4. 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.

Table 4/H.263 – Parameter k for GOB size definition

Number of Lines dy	Value of k when not in RRU mode	Value of k when in RRU mode
4,…,400	1	2
404,…,800	2	2
804,…,1152	4	4

The Slice Structured mode is described in Annex K. Slices are similar to GOBs in that they are a multi-macroblock layer of the syntax, but slices have a more flexible shape and usage than GOBs, and slices may appear in the bitstream in any order, under certain conditions.

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

Figure 4/H.263 – Arrangement of Groups of Blocks in a CIF picture

Each GOB is divided into macroblocks. The macroblock structure depends on whether the optional Reduced-Resolution Update (RRU) mode is in use (see Annex Q). Unless in RRU mode, each 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. Each luminance or chrominance block thus relates to 8 pixels by 8 lines of Y, C_B or C_R. Unless in RRU mode, a GOB comprises one macroblock row for sub-QCIF, QCIF and CIF, two macroblock rows for 4CIF and four macroblock rows for 16CIF.

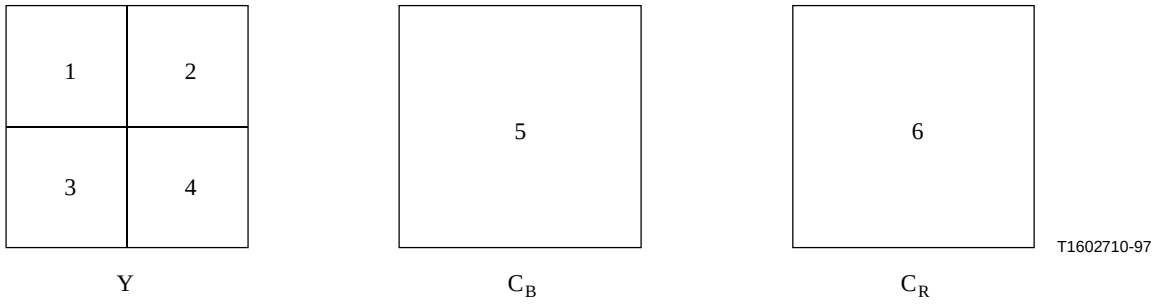


Figure 5/H.263 – Arrangement of blocks in a macroblock

When in RRU mode, a macroblock relates to 32 pixels by 32 lines of Y and the spatially corresponding 16 pixels by 16 lines of C_B and C_R, and each luminance or chrominance block relates to 16 pixels by 16 lines of Y, C_B or C_R. Furthermore, a GOB comprises one macroblock row for CIF and 4CIF, and two 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).

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.

4.2.2 Prediction

The primary form of prediction is inter-picture and may be augmented by motion compensation (see 4.2.3). The coding mode in which temporal prediction is applied is called INTER; the coding mode is called INTRA if no temporal 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).

In total, H.263 has seven basic picture types (of which only the first two are mandatory) which are defined primarily in terms of their prediction structure:

1) INTRA: A picture having no reference picture(s) for prediction (also called an I-picture);
2) INTER: A picture using a temporally previous reference picture (also called a P-picture);
3) PB: A frame representing two pictures and having a temporally previous reference picture (see Annex G);
4) Improved PB: A frame functionally similar but normally better than a PB-frame (see Annex M);

5) B: A picture having two reference pictures, one of which temporally precedes the B-picture and one of which temporally succeeds the B-picture and has the same picture size (see Annex O);

6) EI: A picture having a temporally simultaneous reference picture which has either the same or a smaller picture size (see Annex O); and

7) EP: A picture having two reference pictures, one of which temporally precedes the EP-picture and one of which is temporally simultaneous and has either the same or a smaller picture size (see Annex O).

As used herein, a "reference" or "anchor" picture is a picture that contains data which can be used by reference as a basis for the decoding of another picture. This use by reference is also known as "prediction", although it sometimes may actually indicate use in a reverse-temporal direction.

4.2.3 Motion compensation

The decoder will accept one vector per macroblock or, if the Advanced Prediction mode or the Deblocking Filter mode is used, one or four vectors per macroblock (see Annexes F and J). 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. Similarly, an Improved PB-frame macroblock (see Annex M) can include an additional forward motion vector. B-picture macroblocks (see Annex O) can be transmitted together with a forward and a backward motion vector, and EP-pictures can be transmitted with a forward motion vector.

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 increased. If PLUSPTYPE is absent, the range 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 PLUSPTYPE is absent and 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. If PLUSPTYPE is present, the motion vector values are less restricted (see also Annex D).

In the Reduced-Resolution Update mode, the motion vector range is enlarged to approximately double size, and each vector component is restricted to have only a half-integer or zero value. Therefore, the range of each motion vector component is [−31.5, 30.5] in the default Reduced-Resolution Update mode (see Annex Q) and a larger range if the Unrestricted Motion Vector mode is also used (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, the Advanced Prediction mode, the Advanced Prediction mode, or the Deblocking Filter mode is used (see Annexes D, F, and J), or in B- and EP-pictures of the Temporal, SNR, and Spatial Scalability mode (see Annex O).

4.2.4 Quantization

Unless the optional Advanced INTRA Coding mode or Modified Quantization mode is in use, 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 6.2. For quantization using the Advanced INTRA Coding mode, refer to Annex I. For quantization using the Modified Quantization mode, refer to Annex T.

NOTE – For the smaller quantization step sizes, the full dynamic range of the transform coefficients cannot be represented unless the optional Modified Quantization mode is in use.

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 of responding 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. A similar requirement applies when using optional EP-pictures (see Annex O), for which each macroblock shall be coded in an INTRA or upward mode at least once every 132 times when coefficients are transmitted for the macroblock.

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, slice, and EOSBS start codes shall be byte aligned, 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 syntax is arranged in a hierarchical structure with four primary layers. From top to bottom the layers are:

- Picture;
- Group of Blocks, or slice, or video picture segment;
- Macroblock;
- Block.

The syntax diagram is shown in Figure 6. A guide for interpretation of the diagram consists of the following:

1) Arrow paths show the possible flows of syntax elements. Any syntax element which has zero length is considered absent for arrow path diagramming purposes (thus, for example, there is an arrow path bypassing PSTUF despite the mandatory nature the PSTUF field since the length of the PSTUF field may be zero).

2) Abbreviations and semantics for each syntax element are as defined in later clauses.

3) The set of syntax elements and arrow paths shown using thick solid lines denotes the syntax flow of the "baseline" mode of operation, without the use of any optional enhancements. (This syntax was also present in the Version 1 of this Recommendation, and remains unchanged in any way.)

4) The set of syntax elements and arrow paths shown using thick dotted lines denotes the additional elements in the syntax flow of the optional enhancements which have been present in both Version 1 and Version 2 of this Recommendation. (This syntax remains unchanged in any way.)

5) The set of syntax elements and arrow paths shown using thin solid lines denotes the additional new elements in the syntax flow of the optional enhancements which are specific to the additional optional features added in Version 2. (This syntax was not present in Version 1.)

6) Syntax element fields shown with square-edged boundaries indicate fixed-length fields, and those with rounded boundaries indicate variable-length fields. One syntax element (DQUANT) is shown with both types of boundaries, because it can have either a variable or fixed length.

7) A fixed-length field is defined to be a field for which the length of the field is not dependent on the data in the content of the field itself. The length of this field is either always the same, or is determined by the prior data in the syntax flow.

8) The term "layer" is used to refer to any part of the syntax which can be understood and diagrammed as a distinct entity.

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-T.

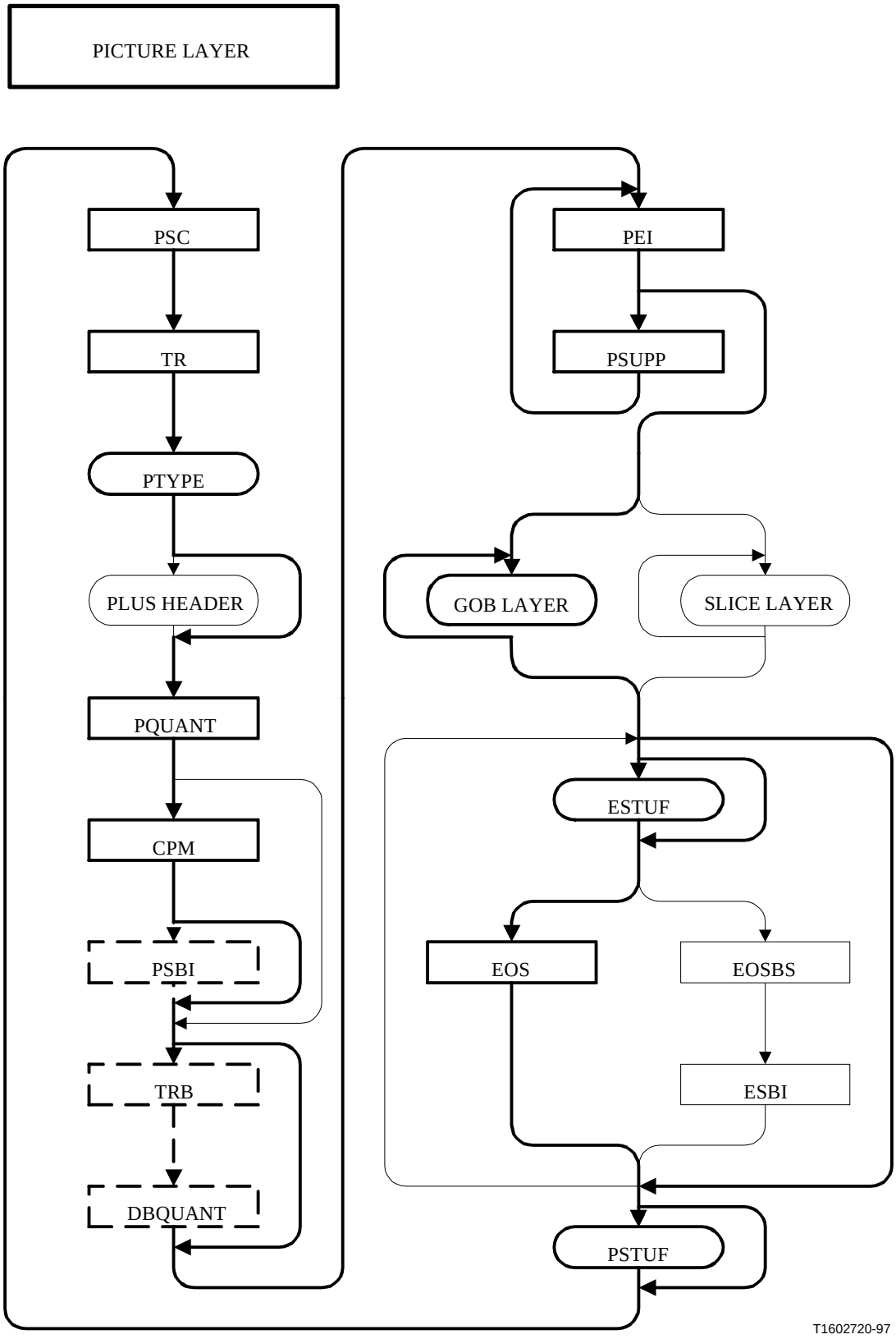


Figure 6/H.263 (part 1 of 7) – Syntax diagram for the video bitstream

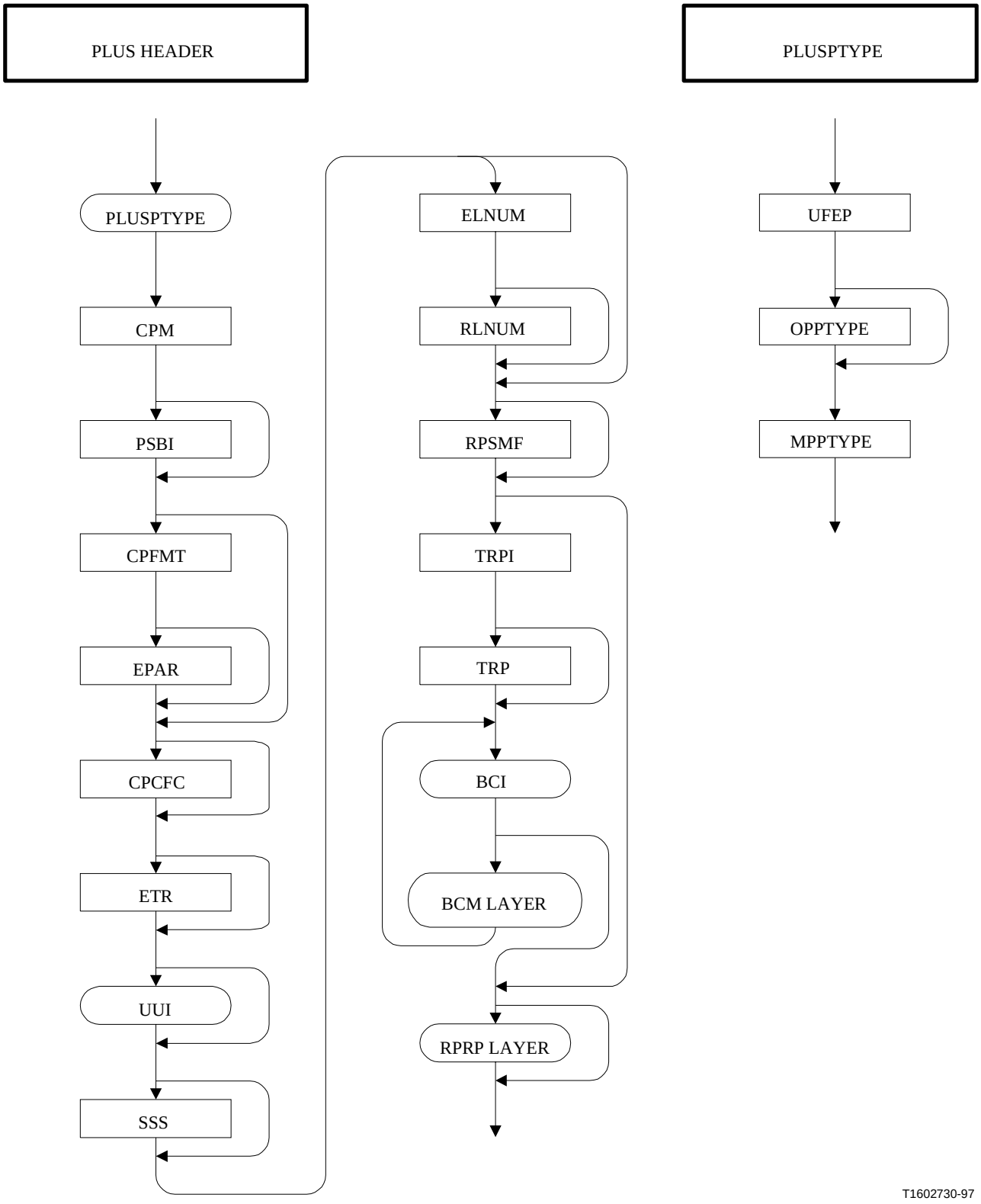


Figure 6/H.263 (part 2 of 7) – Syntax diagram for the video bitstream

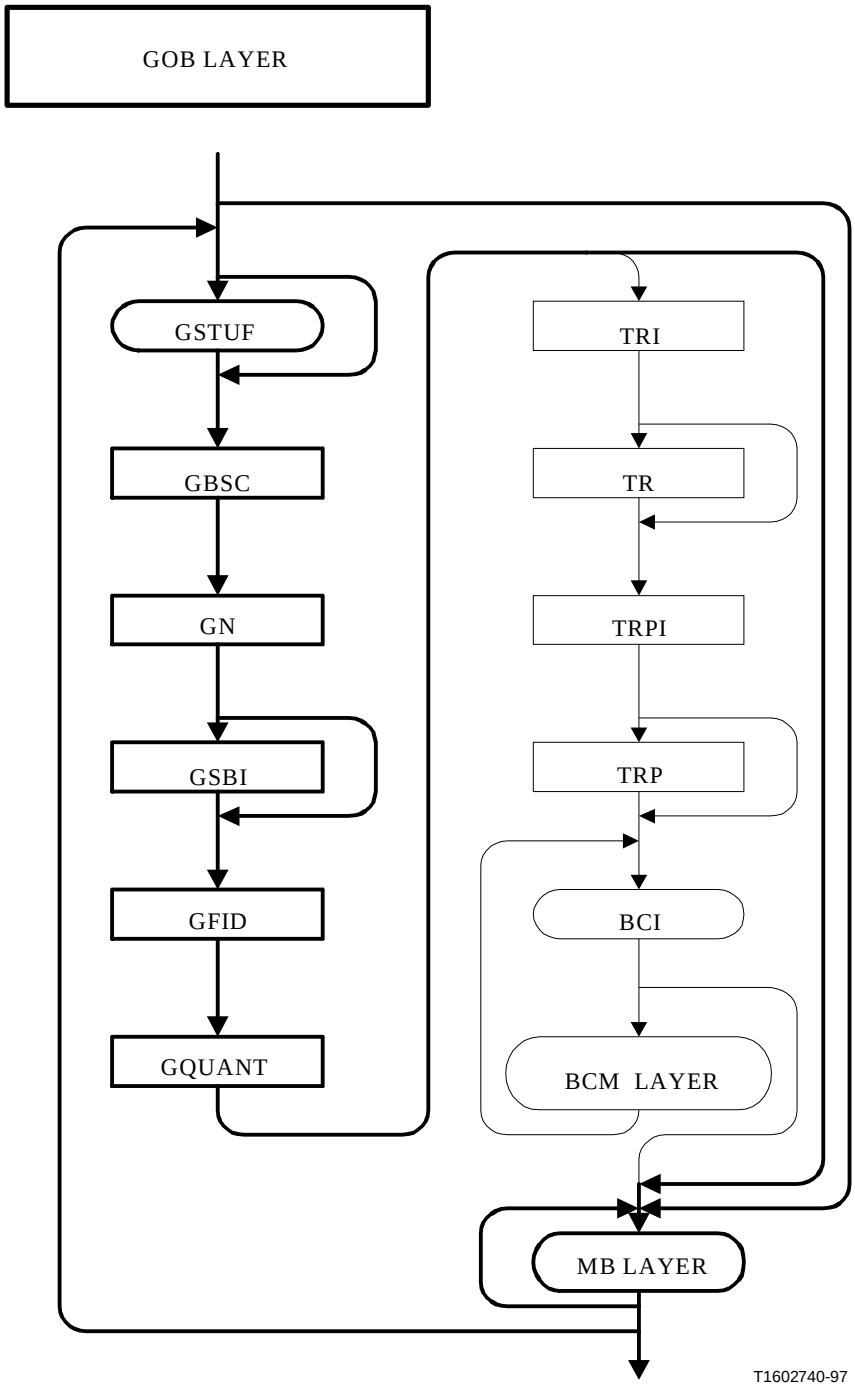


Figure 6/H.263 (part 3 of 7) – Syntax diagram for the video bitstream

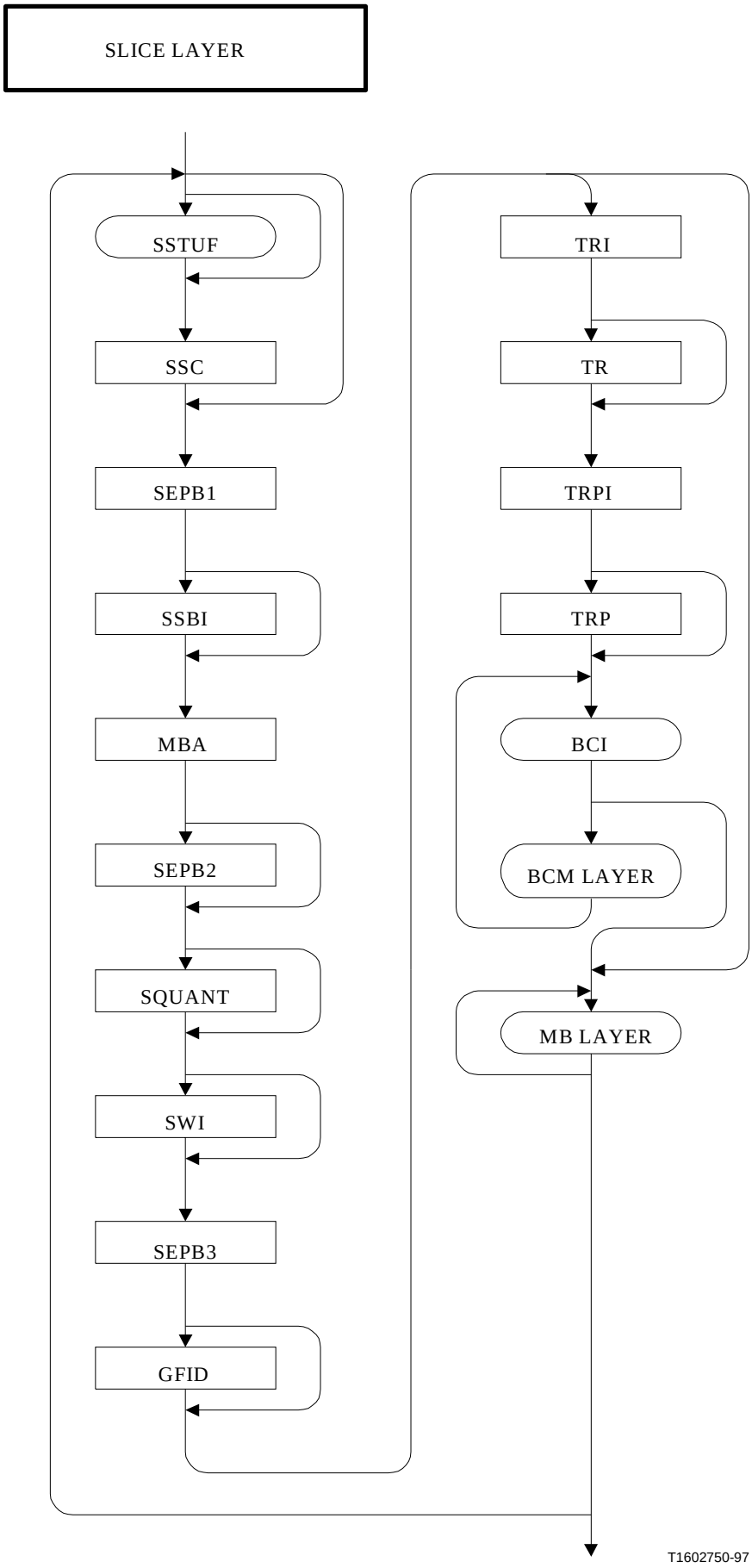


Figure 6/H.263 (part 4 of 7) – Syntax diagram for the video bitstream

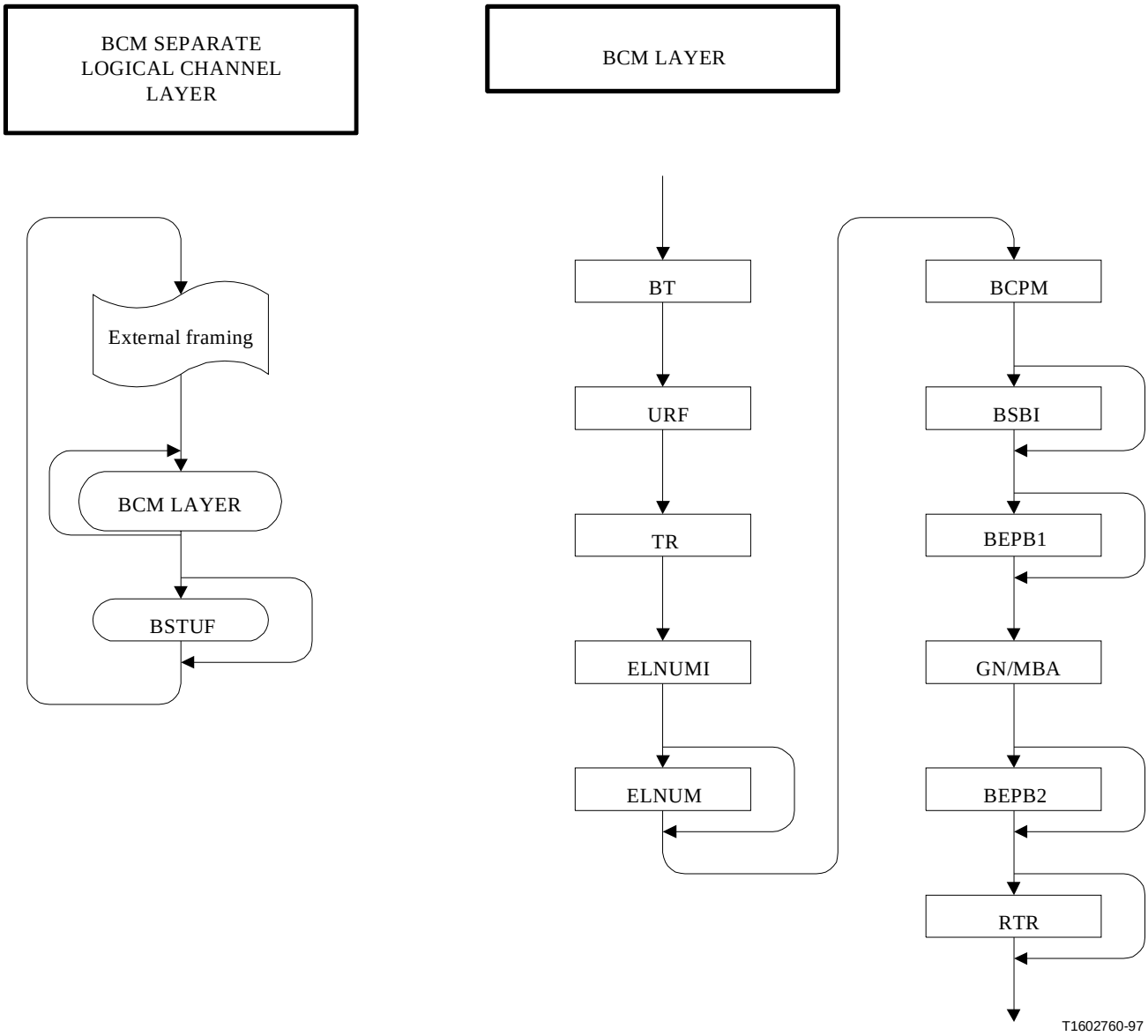


Figure 6/H.263 (part 5 of 7) – Syntax diagram for the video bitstream

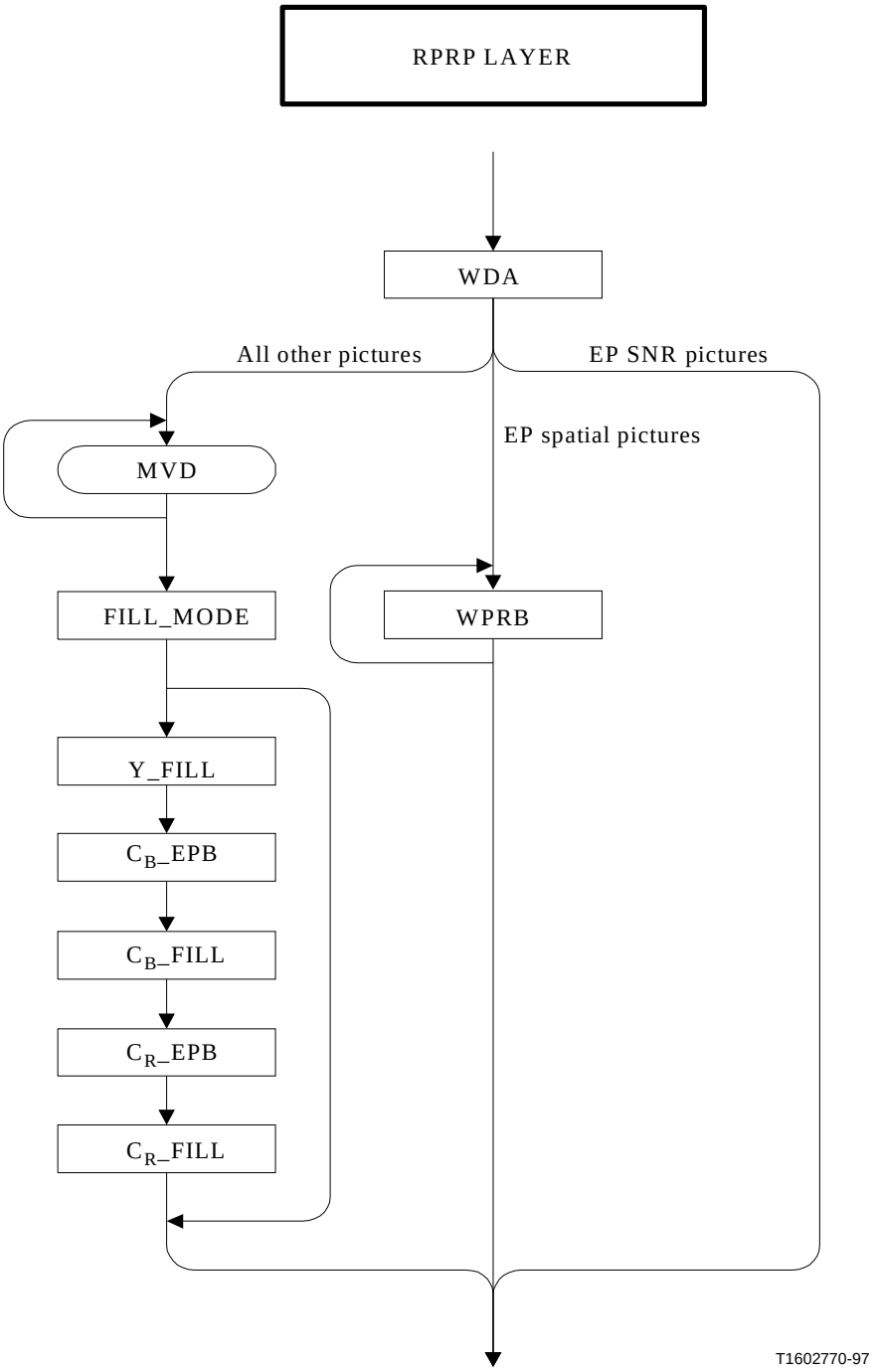


Figure 6/H.263 (part 6 of 7) – Syntax diagram for the video bitstream

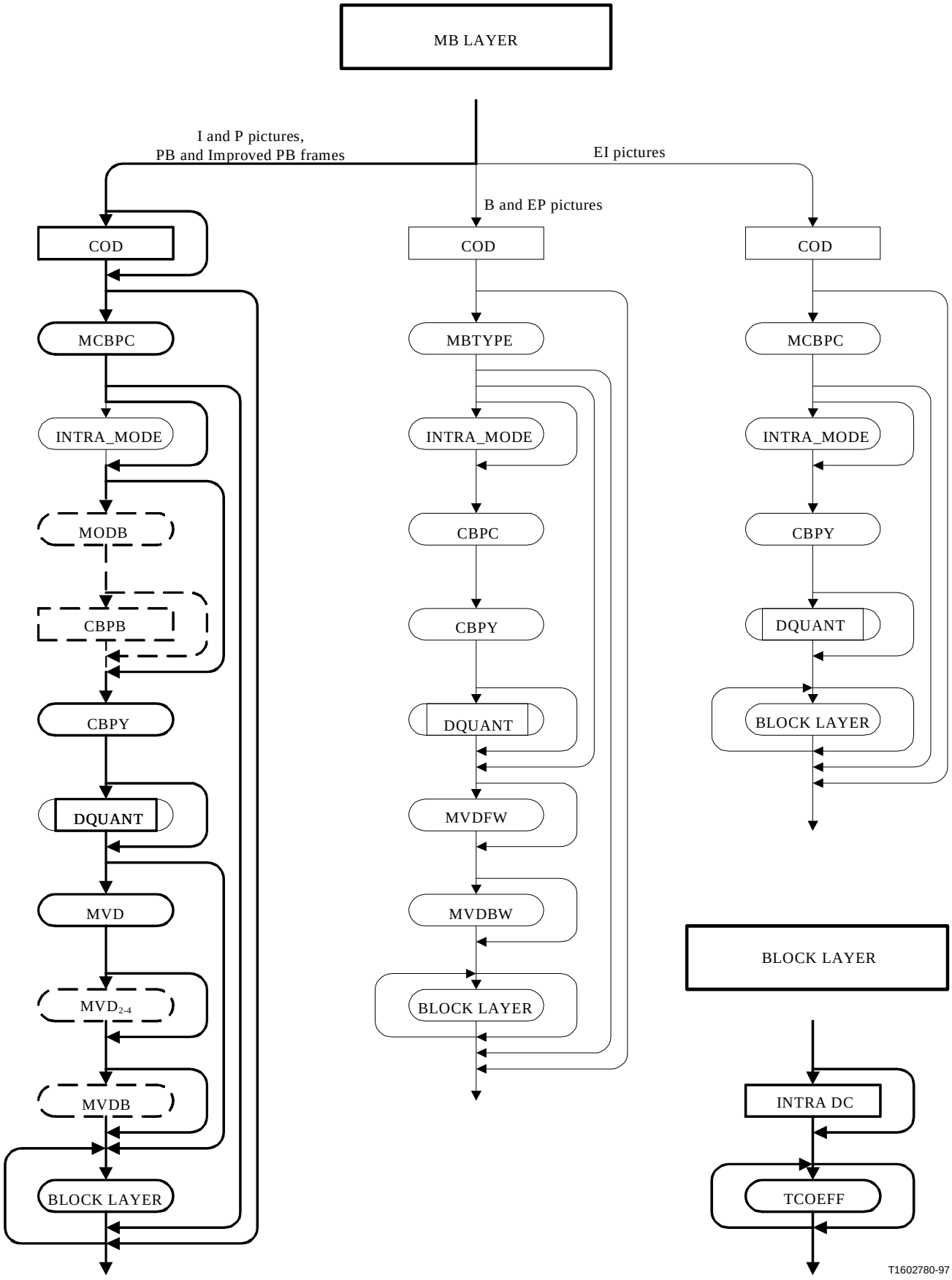


Figure 6/H.263 (part 7 of 7) – Syntax diagram for the video bitstream

5.1 Picture layer

Data for each picture consists of a picture header followed by data for Group of Blocks or for slices, eventually followed by an optional end-of-sequence code and stuffing bits. The structure is shown in Figure 7 for pictures that do not include the optional PLUSPTYPE data field. PSBI is only present if indicated by CPM. TR_B and DBQUANT are only present if PTYPE indicates use of the "PB-frame" mode (unless the PLUSPTYPE field is present and the use of DBQUANT is indicated therein).

The optional PLUSPTYPE data field is present when so indicated in bits 6-8 of PTYPE. When present, an additional set of data is included in the bitstream, which immediately follows PTYPE and precedes PQUANT. In addition, the CPM and PSBI fields are moved forward in the picture header when PLUSPTYPE is present, so that they appear immediately after PLUSPTYPE rather than being located after PQUANT. The format for the additional data following PLUSPTYPE is shown in Figure 8. All fields in this additional picture header data after PLUSPTYPE are optional, depending on whether their presence is indicated in PLUSPTYPE. When the Slice Structured mode (see Annex K) is in use, slices are substituted for GOBs in the location shown in Figure 7.

Combinations of PSUPP and PEI may not be present, and may be repeated when present. EOS and EOSBS+ESBI may not be present, while ESTUF may be present only if EOS or EOSBS is present. EOS shall not be repeated unless at least one picture start code appears between each pair of EOS codes. Picture headers for dropped pictures are not transmitted.

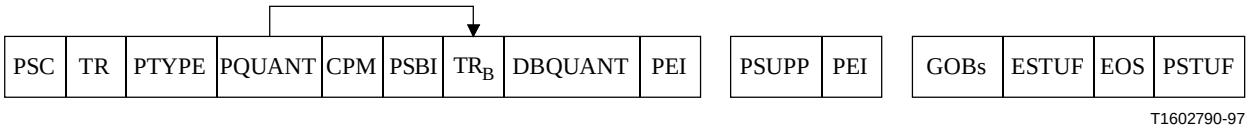


Figure 7/H.263 – Structure of picture layer (without optional PLUSPTYPE-related fields)

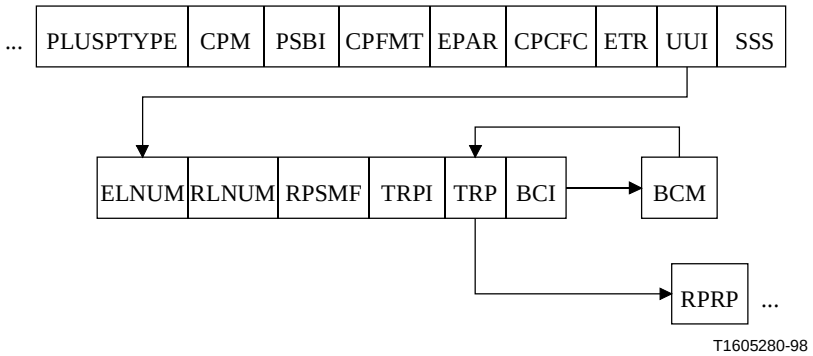


Figure 8/H.263 – Structure of optional PLUSPTYPE-related fields (located immediately after PTYPE when present)

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 bits as necessary 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)

The value of TR is formed by incrementing its value in the temporally previous reference picture header by one plus the number of skipped or non-reference pictures at the picture clock frequency since the previously transmitted one. The interpretation of TR depends on the active picture clock frequency. Under the standard CIF picture clock frequency, TR is an 8-bit number which can have 256 possible values. The arithmetic is performed with only the eight LSBs. If a custom picture clock frequency is signalled in use, Extended TR in 5.1.8 and TR form a 10-bit number where TR stores the eight Least Significant Bits (LSBs) and ETR stores the two Most Significant Bits (MSBs). The arithmetic in this case is performed with the ten LSBs. In the optional PB-frames or Improved PB-frames mode, TR only addresses P-pictures; for the temporal reference for the B-picture part of PB or Improved PB frames, refer to 5.1.22.

5.1.3 Type Information (PTYPE) (Variable Length)

Information about the complete picture:

– Bit 1: Always "1", in order to avoid start code emulation.

– Bit 2: Always "0", for distinction with Recommendation H.261.

– Bit 3: Split screen indicator, "0" off, "1" on.

– Bit 4: Document camera indicator, "0" off, "1" on.

– Bit 5: Full Picture Freeze Release, "0" off, "1" on.

– Bits 6-8: Source Format, "000" forbidden, "001" sub-QCIF, "010" QCIF, "011" CIF, "100" 4CIF, "101" 16CIF, "110" reserved, "111" extended PTYPE.

If bits 6-8 are not equal to "111", which indicates an extended PTYPE (PLUSPTYPE), the following five bits are also present in PTYPE:

– Bit 9: Picture Coding Type, "0" INTRA (I-picture), "1" INTER (P-picture).

– Bit 10: Optional Unrestricted Motion Vector mode (see Annex D), "0" off, "1" on.

– Bit 11: Optional Syntax-based Arithmetic Coding mode (see Annex E), "0" off, "1" on.

– Bit 12: Optional Advanced Prediction mode (see Annex F), "0" off, "1" on.

– Bit 13: Optional PB-frames mode (see Annex G), "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.

Full Picture Freeze 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) or picture freeze request (see also Annex L) and allows a decoder to exit from its freeze picture mode and display the decoded picture in the normal manner.

If bits 6-8 indicate a different source format than in the previous picture header, the current picture shall be an I-picture, unless an extended PTYPE is indicated in bits 6-8 and the capability to use the optional Reference Picture Resampling mode (see Annex P) has been negotiated externally (for example Recommendation H.245).

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

Bits 6-8 shall not have a value of "111" which indicates the presence of an extended PTYPE (PLUSPTYPE) unless the capability has been negotiated externally (for example Recommendation H.245) to allow the use of a custom source format or one or more of the other optional modes

available only by the use of an extended PTYPE (see Annexes I through K and M through T). Whenever bit 6-8 do not have a value of "111", all of the additional modes available only by the use of an extended PTYPE shall be considered to have been set to an "off" state and shall be inferred to remain "off" unless explicitly switched on later in the bitstream.

5.1.4 Plus PTYPE (PLUSPTYPE) (Variable Length)

A codeword of 12 or 30 bits which is present only if the presence of extended PTYPE is indicated in bits 6-8 of PTYPE.

PLUSPTYPE is comprised of up to three subfields: UFEP, OPPTYPE, and MPPTYPE. OPPTYPE is present only if UFEP has a particular value.

5.1.4.1 Update Full Extended PTYPE (UFEP) (3 bits)

A fixed length codeword of 3 bits which is present only if "extended PTYPE" is indicated in PTYPE bits 6-8. When set to "000", it indicates that only those extended PTYPE fields which need to be signalled in every picture header (MPPTYPE) are included in the current picture header. When set to "001", it indicates that all extended PTYPE fields are included in the current picture header. If the picture type is INTRA or EI, this field shall be set to "001".

In addition, if PLUSPTYPE is present in each of a continuing sequence of pictures, this field shall be set to "001" at least as often as specified by a five-second or five-picture timeout period, whichever allows a larger interval of time. More specifically, the timeout period requires UFEP = "001" to appear in the PLUSPTYPE field (if PLUSPTYPE is present in every intervening picture) of the first picture header with temporal reference indicating a time interval greater than or equal to five seconds since the last occurrence of UFEP = "001", or of the fifth picture after the last occurrence of UFEP = "001" (whichever requirement allows a longer period of time as measured by temporal reference).

Encoders should set UFEP to "001" more often in error-prone environments. Values of UFEP other than "000" and "001" are reserved.

5.1.4.2 The Optional Part of PLUSPTYPE (OPPTYPE) (18 bits)

If UFEP is "001", then the following bits are present in PLUSPTYPE:

– Bits 1-3 Source Format, "000" reserved, "001" sub-QCIF, "010" QCIF, "011" CIF, "100" 4CIF, "101" 16CIF, "110" custom source format, "111" reserved;

– Bit 4 Optional Custom PCF, "0" CIF PCF, "1" custom PCF;

– Bit 5 Optional Unrestricted Motion Vector (UMV) mode (see Annex D), "0" off, "1" on;

– Bit 6 Optional Syntax-based Arithmetic Coding (SAC) mode (see Annex E), "0" off, "1" on;

– Bit 7 Optional Advanced Prediction (AP) mode (see Annex F), "0" off, "1" on;

– Bit 8 Optional Advanced INTRA Coding (AIC) mode (see Annex I), "0" off, "1" on;

– Bit 9 Optional Deblocking Filter (DF) mode (see Annex J), "0" off, "1" on;

– Bit 10 Optional Slice Structured (SS) mode (see Annex K), "0" off, "1" on;

– Bit 11 Optional Reference Picture Selection (RPS) mode (see Annex N), "0" off, "1" on;

– Bit 12 Optional Independent Segment Decoding (ISD) mode (see Annex R), "0" off, "1" on;

– Bit 13 Optional Alternative INTER VLC (AIV) mode (see Annex S), "0" off, "1" on;

– Bit 14 Optional Modified Quantization (MQ) mode (see Annex T), "0" off, "1" on;

– Bit 15 Equal to "1" to prevent start code emulation;

– Bit 16 Reserved, shall be equal to "0";

– Bit 17 Reserved, shall be equal to "0";

– Bit 18 Reserved, shall be equal to "0".

5.1.4.3 The mandatory part of PLUSPTYPE when PLUSPTYPE present (MPPTYPE) (9 bits)

Regardless of the value of UFEP, the following 9 bits are also present in PLUSPTYPE:

– Bits 1-3 Picture Type Code:
"000" I-picture (INTRA),
"001" P-picture (INTER),
"010" Improved PB-frame (see Annex M),
"011" B-picture (see Annex O),
"100" EI-picture (see Annex O),
"101" EP-picture (see Annex O),
"110" Reserved,
"111" Reserved;

– Bit 4 Optional Reference Picture Resampling (RPR) mode (see Annex P), "0" off, "1" on;

– Bit 5 Optional Reduced-Resolution Update (RRU) mode (see Annex Q), "0" off, "1" on;

– Bit 6 Rounding Type (RTYPE) (see 6.1.2);

– Bit 7 Reserved, shall be equal to "0";

– Bit 8 Reserved, shall be equal to "0";

Bit 9 Equal to "1" to prevent start code emulation.

The encoder should control the rounding type so that P-pictures, Improved PB-frames, and EP-pictures have different values for bit 6 (Rounding Type for P-pictures) from their reference pictures for motion compensation. Bit 6 can have an arbitrary value if the reference picture is an I- or an EI-picture. Bit 6 can be set to "1" only when bits 1-3 indicate a P-picture, Improved PB-frame, or EP-picture. For other types of pictures, this bit shall always be set to "0".

5.1.4.4 The semantics of PLUSPTYPE

The mandatory part of PLUSPTYPE consists of features which are likely to be changed on a picture-by-picture basis. Primarily, these are the bits that indicate the picture type among I, P, Improved PB, B, EI, and EP. (Note that the PB-frames mode of Annex G cannot be used with PLUSPTYPE present – the Improved PB-frames mode of Annex M should be used instead.) However, they also include indications for the use of the RPR and RRU modes, as these may change from picture to picture as well.

Features that are likely to remain in use rather than being changed from one picture to another (except in obvious ways as discussed in 5.1.4.5) have been placed in the optional part of PLUSPTYPE. When UFEP is 000, the missing mode information is inferred from the type of picture and from the mode information sent in a previous PLUSPTYPE with UFEP equal to 001.

If PLUSPTYPE is present but UFEP is 000, then:

1) For a P-picture or Improved PB-frame (see Annex M), the pixel aspect ratio, picture width, and picture height are unchanged from those of the reference picture.

2) For a temporal-scalability B-picture (see Annex O) in an enhancement layer, the Reference Layer Number (RLNUM) is the same as the Enhancement Layer Number (ELNUM) if the last picture sent in the enhancement layer was an EI- or EP-picture. If the last picture sent in the enhancement layer was a B-picture, the reference layer number is the same as the reference layer number of the last B-picture. The pixel aspect ratio, picture width and picture height are unchanged from those of the temporally subsequent reference layer picture.

   Note that if temporally surrounding EI- or EP-pictures exist in the same enhancement layer as the B-picture, RLNUM (explicit or implied) shall always be equal to ELNUM. Note also that the pixel aspect ratio, picture width and picture height of a B-picture (explicit or implied) shall always be equal to those of its temporally subsequent reference layer picture.

3) For a SNR/Spatial scalability EP-picture (see Annex O), the pixel aspect ratio, picture width and picture height are unchanged from those of the temporally previous reference picture in the same enhancement layer.

5.1.4.5 Mode restrictions for certain picture types and mode inference rules

Certain modes do not apply to certain types of pictures. Particularly, these restrictions apply:

1) The following modes do not apply within I (INTRA) pictures: Unrestricted Motion Vector (see Annex D), Advanced Prediction (see Annex F), Alternative INTER VLC (see Annex S), Reference Picture Resampling (see Annex P), and Reduced-Resolution Update (see Annex Q).

2) The following modes do not apply within B-pictures (see Annex O): Syntax-based Arithmetic Coding (see Annex E), Deblocking Filter (see Annex J), and Advanced Prediction (see Annex F).

3) The following modes do not apply within EI-pictures (see Annex O): Unrestricted Motion Vector (see Annex D), Syntax-based Arithmetic Coding (see Annex E), Advanced Prediction (see Annex F), Reference Picture Resampling (see Annex P), Reduced-Resolution Update (see Annex Q), and Alternative INTER VLC (see Annex S).

4) The following modes do not apply within EP-pictures (see Annex O): Syntax-based Arithmetic Coding (see Annex E) and Advanced Prediction (see Annex F).

One or more of the modes listed in the above four-item list may have a mode flag having a value "1" in the optional part of PLUSPTYPE within a picture of a type that is prohibited for that mode (types I, B, EI, or EP). This condition is allowed and shall be interpreted subject to the mode inference rules which follow in the next paragraph.

Mode states are subject to the following mode inference rules:

1) Once a mode flag has been set to "1" in the optional part of PLUSPTYPE, the current picture and each subsequent picture in the bitstream shall be assigned a state of "on" for that mode.

2) An inferred state of "off" shall be assigned to any mode which does not apply within a picture having the current picture type code. However, each subsequent picture in the bitstream shall have an inferred state of "on" for that mode (unless this too results in an obvious conflict – which shall be resolved in the same way). In the case of layered scalable bitstreams (see Annex O), the mode state shall be inferred only from within the same layer of the bitstream.

3) The inference of state shall continue until a picture in the same layer that either contains the optional part of PLUSPTYPE or does not contain PLUSPTYPE at all is sent. If a new picture containing the optional part of PLUSPTYPE is sent, the state sent in the new message shall override the old state. If a picture is sent which does not contain PLUSPTYPE (a picture in which bits 6-8 of PTYPE is not "111"), a state of "off" shall be assigned to all modes not explicitly set to "on" in the PTYPE field, and all modes shall continue to have an inferred state of "off" until a new picture containing the optional part of PLUSPTYPE is sent.

4) Two modes do not require mode state inference, since the mode flags for these modes appear in the mandatory part of PLUSPTYPE. These are the Reference Picture Resampling mode (Annex P) and the Reduced-Resolution Update mode (Annex Q). The mode flag for either of these modes shall not be set unless the current picture allows the use of the mode. For example, the Reduced-Resolution Update mode bit shall not be set in an INTRA picture.

5.1.4.6 Mode interaction restrictions

Certain modes cannot be used in combination with certain other modes.

1) The Syntax-based Arithmetic Coding mode (see Annex E) shall not be used with the Alternative INTER VLC mode (see Annex S) or the Modified Quantization mode (see Annex T).

2) If PLUSPTYPE is present, the Unrestricted Motion Vector mode (see Annex D) shall not be used with the Syntax-based Arithmetic Coding mode (see Annex E).

3) The Independent Segment Decoding mode (see Annex R) shall not be used with the Reference Picture Resampling mode (see Annex P).

4) The Independent Segment Decoding mode (see Annex R) shall not be used with the Slice Structured mode without the simultaneous use of the Rectangular Slice submode of the Slice Structured mode (see Annex K).

5.1.4.7 The picture header location of CPM (1 bit) and PSBI (2 bits)

The location of the CPM and PSBI fields in the picture header depends on whether or not PLUSPTYPE is present (see 5.1.20 and 5.1.21). If PLUSPTYPE is present, then CPM follows immediately after PLUSPTYPE in the picture header. If PLUSPTYPE is not present, then CPM follows immediately after PQUANT in the picture header. PSBI always follows immediately after CPM (if CPM = "1").

5.1.5 Custom Picture Format (CPFMT) (23 bits)

A fixed length codeword of 23 bits that is present only if the use of a custom picture format is signalled in PLUSPTYPE and UFEP is ’001’. When present, CPFMT consists of:

– Bits 1-4 Pixel Aspect Ratio Code: A 4-bit index to the PAR value in Table 5. For extended PAR, the exact pixel aspect ratio shall be specified in EPAR (see 5.1.6);

– Bits 5-13 Picture Width Indication: Range [0, ..., 511]; Number of pixels per line = (PWI + 1) * 4;

– Bit 14 Equal to "1" to prevent start code emulation;

– Bits 15-23 Picture Height Indication: Range [1, ..., 288]; Number of lines = PHI * 4.

Table 5/H.263 – PAR code definition

PAR code	Pixel aspect ratios
0000	Forbidden
0001	1:1 (Square)
0010	12:11 (CIF for 4:3 picture)
0011	10:11 (525-type for 4:3 picture)
0100	16:11 (CIF stretched for 16:9 picture)
0101	40:33 (525-type stretched for 16:9 picture)
0110-1110	Reserved
1111	Extended PAR

5.1.6 Extended Pixel Aspect Ratio (EPAR) (16 bits)

A fixed length codeword of 16 bits that is present only if CPFMT is present and extended PAR is indicated therein. When present, EPAR consists of:

– Bits 1-8 PAR Width: "0" is forbidden. The natural binary representation of the PAR width;

– Bits 9-16 PAR Height: "0" is forbidden. The natural binary representation of the PAR height.

The PAR Width and PAR Height shall be relatively prime.

5.1.7 Custom Picture Clock Frequency Code (CPCFC) (8 bits)

A fixed length codeword of 8 bits that is present only if PLUSPTYPE is present and UFEP is 001 and a custom picture clock frequency is signalled in PLUSPTYPE. When present, CPCFC consists of:

– Bit 1 Clock Conversion Code: "0" indicates a clock conversion factor of 1000 and "1" indicates 1001;

– Bits 2-8 Clock Divisor: "0" is forbidden. The natural binary representation of the value of the clock divisor.

The custom picture clock frequency is given by 1 800 000/(clock divisor * clock conversion factor) Hz.

The temporal reference counter shall count in units of the inverse of the picture clock frequency, in seconds. When the PCF changes from that specified for the previous picture, the temporal reference for the current picture is measured in terms of the prior PCF, so that the new PCF takes effect only for the temporal reference interpretation of future pictures.

5.1.8 Extended Temporal Reference (ETR) (2 bits)

A fixed length codeword of 2 bits which is present only if a custom picture clock frequency is in use (regardless of the value of UFEP). It is the two MSBs of the 10-bit number defined in 5.1.2.

5.1.9 Unlimited Unrestricted Motion Vectors Indicator (UUI) (Variable length)

A variable length codeword of 1 or 2 bits that is present only if the optional Unrestricted Motion Vector mode is indicated in PLUSPTYPE and UFEP is 001. When UUI is present it indicates the effective limitation of the range of the motion vectors being used.

– UUI = "1" The motion vector range is limited according to Tables D.1 and D.2.

– UUI = "01" The motion vector range is not limited except by the picture size.

5.1.10 Slice Structured Submode bits (SSS) (2 bits)

A fixed length codeword of 2 bits which is present only if the optional Slice Structured mode (see Annex K) is indicated in PLUSPTYPE and UFEP is 001. If the Slice Structured mode is in use but UFEP is not 001, the last values sent for SSS shall remain in effect.

– Bit 1 Rectangular Slices, "0" indicates free-running slices, "1" indicates rectangular slices;

– Bit 2 Arbitrary Slice Ordering, "0" indicates sequential order, "1" indicates arbitrary order.

5.1.11 Enhancement Layer Number (ELNUM) (4 bits)

A fixed length codeword of 4 bits which is present only if the optional Temporal, SNR, and Spatial Scalability mode is in use (regardless of the value of UFEP). The particular enhancement layer is identified by an enhancement layer number, ELNUM. Picture correspondence between layers is achieved via the temporal reference. Picture size is either indicated within each enhancement layer using the existing source format fields or is inferred by the relationship to the reference layer. The first enhancement layer above the base layer is designated as Enhancement Layer Number 2, and the base layer has number 1.

5.1.12 Reference Layer Number (RLNUM) (4 bits)

A fixed length codeword of 4 bits which is present only if the optional Temporal, SNR, and Spatial Scalability mode is in use (see Annex O) and UFEP is 001. The layer number for the pictures used as reference anchors is identified by a Reference Layer Number (RLNUM). Time correspondence between layers is achieved via the temporal reference.

Note that for B-pictures in an enhancement layer having temporally surrounding EI- or EP-pictures which are present in the same enhancement layer, RLNUM shall be equal to ELNUM (see Annex O).

5.1.13 Reference Picture Selection Mode Flags (RPSMF) (3 bits)

A fixed length codeword of 3 bits that is present only if the Reference Picture Selection mode is in use and UFEP is 001. When present, RPSMF indicates which type of back-channel messages are needed by the encoder. If the Reference Picture Selection mode is in use but RPSMF is not present, the last value of RPSMF that was sent shall remain in effect.

– 100: neither ACK nor NACK signals needed;

– 101: need ACK signals to be returned;

– 110: need NACK signals to be returned;

– 111: need both ACK and NACK signals to be returned;

– 000-011: Reserved.

5.1.14 Temporal Reference for Prediction Indication (TRPI) (1 bit)

A fixed length codeword of 1 bit that is present only if the optional Reference Picture Selection mode is in use (regardless of the value of UFEP). When present, TRPI indicates the presence of the following TRP field:

– 0: TRP field is not present;

– 1: TRP field is present.

TRPI shall be 0 whenever the picture header indicates an I- or EI-picture.

5.1.15 Temporal Reference for Prediction (TRP) (10 bits)

When present (as indicated in TRPI), TRP indicates the Temporal Reference which is used for prediction of the encoding, except for in the case of B-pictures. For B-pictures, the picture having the temporal reference TRP is used for the prediction in the forward direction. (Prediction in the reverse-temporal direction always uses the immediately temporally subsequent picture.) TRP is a ten-bit number. If a custom picture clock frequency was not in use for the reference picture, the two MSBs of TRP are zero and the LSBs contain the eight-bit TR found in the picture header of the reference picture. If a custom picture clock frequency was in use for the reference picture, TRP is a ten-bit number consisting of the concatenation of ETR and TR from the reference picture header.

When TRP is not present, the most recent temporally previous anchor picture shall be used for prediction, as when not in the Reference Picture Selection mode. TRP is valid until the next PSC, GSC, or SSC.

5.1.16 Back-Channel message Indication (BCI) (Variable length)

A variable length field of one or two bits that is present only if the optional Reference Picture Selection mode is in use. When set to "1", this signals the presence of the following optional video Back-Channel Message (BCM) field. "01" indicates the absence or the end of the video back-channel message field. Combinations of BCM and BCI may not be present, and may be repeated when present. BCI shall be set to "01" if the videomux submode of the optional Reference Picture Selection mode is not in use.

5.1.17 Back-Channel Message (BCM) (Variable length)

The Back-Channel message with syntax as specified in N.4.2, which is present only if the preceding BCI field is present and is set to "1".

5.1.18 Reference Picture Resampling Parameters (RPRP) (Variable length)

A variable length field that is present only if the optional Reference Picture Resampling mode bit is set in PLUSPTYPE. This field carries the parameters of the Reference Picture Resampling mode (see Annex P). Note that the Reference Picture Resampling mode can also be invoked implicitly by the occurrence of a picture header for an INTER coded picture having a picture size which differs from that of the previous encoded picture, in which case the RPRP field is not present and the Reference Picture Resampling mode bit is not set.

5.1.19 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.

5.1.20 Continuous Presence Multipoint and Video Multiplex (CPM) (1 bit)

A codeword of 1 bit that signals the use of the optional Continuous Presence Multipoint and Video Multiplex mode (CPM); "0" is off, "1" is on. For the use of CPM, refer to Annex C. CPM follows immediately after PLUSPTYPE if PLUSPTYPE is present, but follows PQUANT in the picture header if PLUSPTYPE is not present.

5.1.21 Picture Sub-Bitstream Indicator (PSBI) (2 bits)

A fixed length codeword of 2 bits that is only present if Continuous Presence Multipoint and Video Multiplex 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). PSBI follows immediately after CPM if CPM is "1" (the location of CPM and PSBI in the picture header depend on whether PLUSPTYPE is present).

5.1.22 Temporal Reference for B-pictures in PB-frames (TR_B) (3/5 bits)

TR_B is present if PTYPE or PLUSPTYPE indicates "PB-frame" or "Improved PB-frame" (see also Annexes G and M) and indicates the number of non-transmitted or non-reference pictures (at 29.97 Hz or the custom picture clock frequency indicated in CPCFC) since the last P- or I-picture or P-part of a PB-or Improved PB-frame and before the B-picture part of the PB- or Improved PB-frame. The codeword is the natural binary representation of the number of non-transmitted pictures plus one. It is 3 bits long for standard CIF picture clock frequency and is extended to 5 bits when a custom picture clock frequency is in use. The maximum number of non-transmitted pictures is 6 for the standard CIF picture clock frequency and 30 when a custom picture clock frequency is used.

5.1.23 Quantization information for B-pictures in PB-frames (DBQUANT) (2 bits)

DBQUANT is present if PTYPE or PLUSPTYPE indicates "PB-frame" or "Improved PB-frame" (see also Annexes G and M). 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 6. In this table, "/" means division by truncation. BQUANT ranges from 1 to 31; if the value for BQUANT resulting from Table 6 is greater than 31, it is clipped to 31.

Table 6/H.263 – DBQUANT codes and relation between QUANT and BQUANT

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

5.1.24 Extra Insertion Information (PEI) (1 bit)

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

5.1.25 Supplemental Enhancement Information (PSUPP) (0/8/16 ... bits)

If PEI is set to "1", then 9 bits follow consisting of 8 bits of data (PSUPP) and then another PEI bit to indicate if a further 9 bits follow and so on. Encoders shall use PSUPP as specified in Annex L.

Decoders which do not support the extended capabilities described in Annex L shall be designed to discard PSUPP if PEI is set to 1. This enables backward compatibility for the extended capabilities of Annex L so that a bitstream which makes use of the extended capabilities can also be used without alteration by decoders which do not support those capabilities.

5.1.26 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. Encoders shall insert this codeword as necessary to attain mandatory byte alignment directly before an EOSBS 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 or EOSBS codeword is byte aligned. Decoders shall be designed to discard ESTUF. See Annex C for a description of EOSBS and its use.

5.1.27 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 EOS code such that the first bit of the EOS code is the first (most significant) bit of a byte. EOS shall not be repeated unless at least one picture start code appears between each pair of EOS codes.

5.1.28 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 9. 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 CPM is "1" in the Picture header.

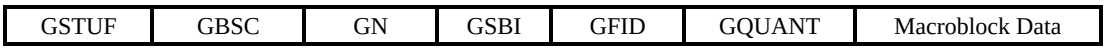


Figure 9/H.263 – Structure of GOB layer

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 a 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; as group number 0 is used in the PSC. Group numbers 1 through 17 are used in GOB headers of the standard picture formats. Group numbers 1 through 24 are used in GOB headers of custom picture formats. Group numbers 16 through 28 are emulated in the slice header (see Annex K) when CPM = "0", and Group Numbers 25-27 and 29 are emulated in the slice header (see Annex K) when CPM = "1". Group number 31 is used in the EOS code, and group number 30 is used in the EOSBS code.

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

A fixed length codeword of 2 bits that is only present if CPM is "1" in the picture header. 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 (or slice) 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, provided PLUSPTYPE is not present. 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.

If PLUSPTYPE is present, the value of GFID shall be the same as that for the previous picture (in the same layer) if the PTYPE, and PLUSPTYPE, and all of the present fields among CPFMT, EPAR, CPCFC, SSS, ELNUM, RLNUM, UUI, RPSMF, and RPRP remain in effect as for the previous picture; otherwise, GFID shall be different from that for 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 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 10. COD is only present in pictures that are not of type 'INTRA', for each macroblock in these pictures. MCBPC is present when indicated by COD or when the picture is of type 'INTRA'. MODB is present for MB-type 0-4 if PTYPE indicates "PB-frame". CBPY, DQUANT, MVD and $MVD_{2\text{-}4}$ 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. $MVD_{2\text{-}4}$ are only present when in Advanced Prediction mode (refer to Annex F) or Deblocking Filter mode (refer to Annex J). 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. For coding of the macroblock layer in B-, EI-, and EP-pictures, see Annex O.

COD	MCBPC	MODB	CBPB	CBPY	DQUANT	MVD	MVD_2	MVD_3	MVD_4	MVDB	Block Data

Figure 10/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 that are not of type "INTRA", for each macroblock in these pictures.

NOTE – In Advanced Prediction mode (see Annex F), overlapped block motion compensation is also performed if COD is set to "1"; and in Deblocking Filter mode (see Annex J), the deblocking filter can also affect the values of some pixels of macroblocks having COD is set to "1".

5.3.2 Macroblock type & Coded Block Pattern for Chrominance (MCBPC) (Variable length)

MCBPC is a variable length codeword giving information about the macroblock type and the coded block pattern for chrominance. The codewords for MCBPC are given in Tables 7 and 8. MCBPC is always included in coded macroblocks.

Table 7/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

Table 8/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
21	5	00	11	0000 0000 010
22	5	01	13	0000 0000 0110 0
23	5	10	13	0000 0000 0111 0
24	5	11	13	0000 0000 0111 1

An extra codeword is available in the tables for bit stuffing. This codeword should be discarded by decoders. If an Improved PB-frame is indicated by MPPTYPE bits 1-3 and Custom Source Format is indicated in OPPTYPE bits 1-3, then MBA shall not indicate stuffing before the first macroblock of the picture (in order to prevent start code emulation).

The macroblock type gives information about the macroblock and which data elements are present. Macroblock types and included elements are listed in Tables 9 and 10. Macroblock type 5 (indices 21-24 in Table 8) shall not be present unless an extended PTYPE (PLUSPTYPE) is present in the picture header and either the Advanced Prediction mode (see Annex F) or the Deblocking Filter mode (see Annex J) is in use, and shall not be present for the first macroblock of a picture. Also, encoders shall not allow an MCBPC code for macroblock type 5 to immediately follow seven consecutive zeros in the bitstream (as can be caused by particular INTRADC codes followed by COD = 0), in order to prevent start code emulation. Codes for macroblock type 5 can be preceded by stuffing when necessary to fulfill this requirement (for macroblocks other than the first of a picture).

Table 9/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	5	INTER4V+Q	X	X	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 10/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	5	INTER4V+Q	X	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 macroblock type of the PB-macroblock indicates INTRA.

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 5.4.1), unless the optional Advanced INTRA Coding mode is in use. $CBPC_N = 1$ if any non-INTRADC coefficient is present for block N, else 0, for $CBPC_5$ and $CBPC_6$ in the coded block pattern. If Advanced INTRA Coding is in use, the usage is similar, but the INTRADC coefficient is indicated in the same way as the other coefficients (see Annex I). Block numbering is given in Figure 5. When MCBPC = Stuffing, the remaining part of the macroblock layer is skipped. In this case, the preceding 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. See Tables 7 and 8.

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 11 the codewords for MODB are defined. MODB is coded differently for Improved PB-frames, as specified in Annex M.

Table 11/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, 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), unless the Advanced INTRA Coding mode is in use. If Advanced INTRA Coding is in use, INTRADC is indicated in the same manner as the other coefficients (see Annex I).

$CBPY_N$ = 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, 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 13.

5.3.6 Quantizer Information (DQUANT) (2 bits/Variable Length)

If the Modified Quantization mode is not in use, DQUANT is a two-bit code to define a change in QUANT. In Table 12 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. If the Modified Quantization mode is in use, DQUANT is a variable length codeword as specified in Annex T.

Table 12/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 13/H.263 – VLC table for CBPY

Index	CBPY(INTRA) (12, 34)	CBPY(INTER) (12, 34)	Number of bits	Code
0	00 00	11 11	4	0011
1	00 01	11 10	5	0010 1
2	00 10	11 01	5	0010 0
3	00 11	11 00	4	1001
4	01 00	10 11	5	0001 1
5	01 01	10 10	4	0111
6	01 10	10 01	6	0000 10
7	01 11	10 00	4	1011
8	10 00	01 11	5	0001 0
9	10 01	01 10	6	0000 11
10	10 10	01 01	4	0101
11	10 11	01 00	4	1010
12	11 00	00 11	4	0100
13	11 01	00 10	4	1000
14	11 10	00 01	4	0110
15	11 11	00 00	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 14. If the Unrestricted Motion Vector mode is used and PLUSPTYPE is present, motion vectors are coded using Table D.3 instead of Table 14 (see Annex D).

Table 14/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

Table 14/H.263 – VLC table for MVD (*concluded*)

Index	Vector	Differences	Bit number	Codes
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 ($MVD_{2\text{-}4}$) (Variable length)

The three codewords $MVD_{2\text{-}4}$ 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 14. $MVD_{2\text{-}4}$ are only present when in Advanced Prediction mode (see Annex F) or Deblocking Filter mode (see Annex J).

5.3.9 Motion Vector Data for B-macroblock (MVDB) (Variable length)

MVDB is only present in PB-frames or Improved 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 14. For the use of MVDB, refer to Annexes G and M.

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). The structure of the block layer is shown in Figure 11. INTRADC is present for every block of the macroblock if MCBPC indicates macroblock type 3 or 4 (see Tables 7 and 8). 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 macroblock type 3 or 4 (see Tables 7 and 8). 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 11/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 15).

Table 15/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 16. 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 non-zero coefficients in this block, "1": this is the last non-zero 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 16 is not prohibited. For the 8-bit word for LEVEL, the code 0000 0000 is forbidden, and the code 1000 0000 is forbidden unless the Modified Quantization mode is in use (see Annex T). The codes for RUN and for LEVEL are given in Table 17.

Table 16/H.263 – VLC table for TCOEF

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
0	0	0	1	3	10s
1	0	0	2	5	1111s
2	0	0	3	7	0101 01s
3	0	0	4	8	0010 111s
4	0	0	5	9	0001 1111s
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 1110s
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	1110s
19	0	2	2	9	0001 1101s
20	0	2	3	11	0000 0011 10s
21	0	2	4	13	0000 0101 0001s
22	0	3	1	6	0110 1s
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
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 1100s
45	0	14	1	9	0001 1011s

Table 16/H.263 – VLC table for TCOEF (*continued*)

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
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
58	1	0	1	5	0111s
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 1010s
71	1	10	1	9	0001 1001s
72	1	11	1	9	0001 1000s
73	1	12	1	9	0001 0111s
74	1	13	1	9	0001 0110s
75	1	14	1	9	0001 0101s
76	1	15	1	9	0001 0100s
77	1	16	1	9	0001 0011s
78	1	17	1	10	0000 1100 0s
79	1	18	1	10	0000 1011 1s
80	1	19	1	10	0000 1011 0s
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
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

Table 16/H.263 – VLC table for TCOEF (*concluded*)

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
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 17/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	see text
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 subclause, 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. For a description of motion compensation in the Reduced-Resolution Update mode refer to Annex Q.

6.1.1 Differential motion vectors

The macroblock vector is obtained by adding predictors to the vector differences indicated by MVD (see Table 14 and Table D.3). 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 12. The predictors are calculated separately for the horizontal and vertical components.

In the special cases at the borders of the current GOB, slice, 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 with bidirectional prediction) 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 or the slice (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; or outside the slice when in Slice Structured mode.

4) Then, the candidate predictor MV3 is set to zero if the corresponding macroblock is outside the picture (at the right side).

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

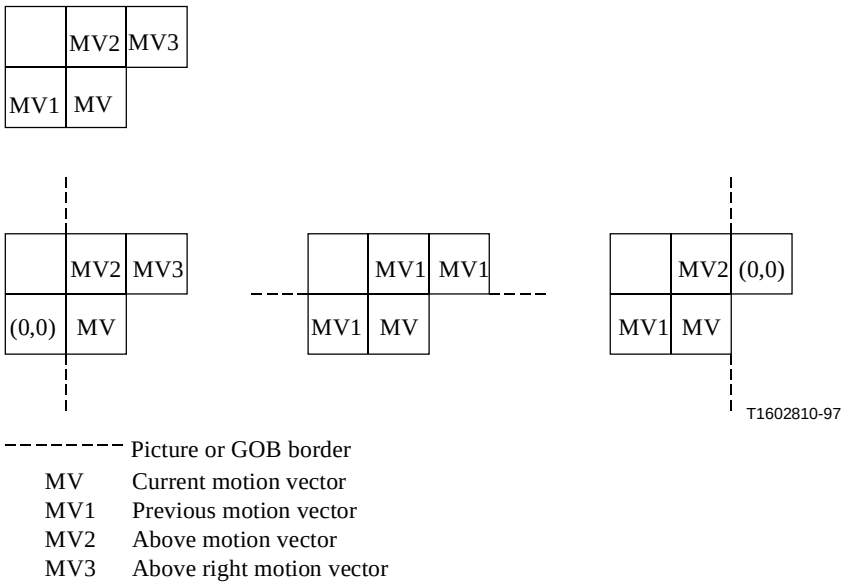


Figure 12/H.263 – Motion vector prediction

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. If the unrestricted motion vector mode (see Annex D) is used, the decoding of motion vectors shall be performed as specified in D.2.

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 18.

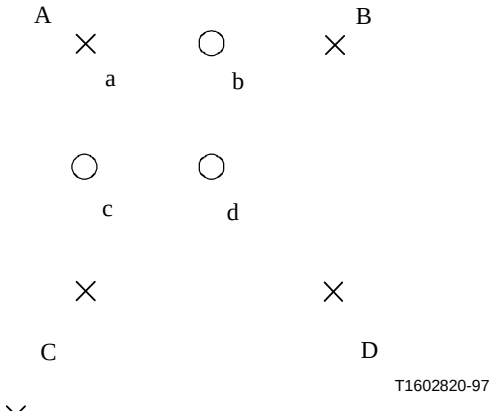
Table 18/H.263 – Modification of quarter-pixel resolution chrominance vector components

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 13. "/" indicates division by truncation.

The value of RCONTROL is equal to the value of the rounding type (RTYPE) bit (bit 6) in MPPTYPE (see 5.1.4.3), when the Source Format field (bits 6-8) in PTYPE indicates "extended PTYPE". Otherwise RCONTROL has an implied value of 0. Regardless of the RTYPE bit, the value of RCONTROL is set to 0 for the B-part of Improved PB frames (see Annex M).



a = A
b = (A + B + 1 – RCONTROL) / 2
c = (A + C + 1 – RCONTROL) / 2
d = (A + B + C + D + 2 – RCONTROL) / 4

Figure 13/H.263 – Half-pixel prediction by bilinear interpolation

6.2 Coefficients decoding

6.2.1 Inverse quantization

The inverse quantization process is described in this subclause, except for when the optional Advanced INTRA Coding mode is in use (see Annex I). If LEVEL = "0", the reconstruction level REC = "0". The reconstruction level of INTRADC is given by Table 15. The reconstruction levels of all non-zero coefficients other than the INTRADC one are given by the following formulas:

$$|REC| = QUANT \cdot (2 \cdot |LEVEL| + 1) \qquad \text{if QUANT = "odd"}$$

$$|REC| = QUANT \cdot (2 \cdot |LEVEL| + 1) - 1 \qquad \text{if QUANT = "even"}$$

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) · |REC|

Sign(LEVEL) is given by the last bit of the TCOEF code (see Table 16) or by Table 17.

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 Zigzag positioning

The quantized transform coefficients are placed into an 8 × 8 block according to the sequence given in Figure 14, unless the optional Advanced INTRA Coding mode is in use (see Annex I). 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 14/H.263 – Zigzag positioning of quantized transform coefficients

6.2.4 Inverse transform

After inverse quantization and zigzag of coefficients, the resulting 8 × 8 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:

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

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

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. For the summation in the Reduced-Resolution Update mode, refer to Annex Q.

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

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.

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

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

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 cosine 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 A.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

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 picture clock frequency, 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 3.6). The value of B is defined as follows:

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

where PCF is the effective picture clock frequency, and R_{max} is the maximum video bit rate during the connection in bits per second. The effective picture clock frequency is the standard CIF picture clock frequency unless a custom PCF is specified in the CPCFC field of the picture header. 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 bit rate 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 picture clock intervals (1000/PCF ms). If at least one complete coded picture is in the buffer, then all the data for the earliest picture in bitstream order is instantaneously removed (e.g. at t_{n+1} in Figure B.1). 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, unless the optional Temporal, SNR, and Spatial Scalability mode is in use, a complete coded picture is one normal I- or P-picture, or a PB-frame or Improved PB-frame.

When the Temporal, SNR, and Spatial Scalability mode is in use (see Annex O), each enhancement layer is given an additional HRD, for which a complete coded picture is one EI-, EP-, or B-picture. The base layer buffer shall hold the bits of the picture header as they arrive, until a sufficient amount of the picture header has arrived to determine whether the picture is a base layer or enhancement layer picture, and the number of the enhancement layer. When it can be determined that the arriving picture belongs to an enhancement layer, all bits for that picture shall be instantly transferred to the appropriate enhancement layer HRD, and any later bits which arrive continue to be placed into the enhancement layer HRD until a sufficient amount of some new picture header has arrived to determine that the bitstream should again be re-routed into another HRD buffer. The process of enhancement layer identification is instantaneous and asynchronous, independent of the picture clock interval checking times.

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

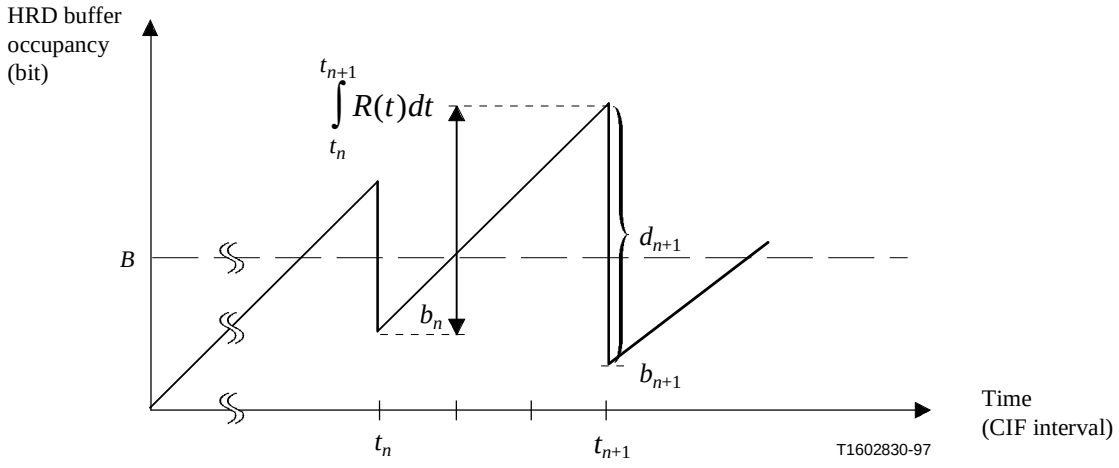
$$d_{n+1} \geq b_n + \int_{t_n}^{t_{n+1}} R(t)dt - B$$

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 bit rate 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 B.1/H.263 – HRD buffer occupancy

ANNEX C

Considerations for multipoint

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). Note that a similar command may also be sent using supplemental enhancement information within the picture header of the video bitstream (see L.4).

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 and Video Multiplexing (CPM)

NOTE – Not used for Recommendation H.324.

In this Recommendation a negotiable Continuous Presence Multipoint and Video Multiplexing mode is provided in which up to four independent H.263 bitstreams can be multiplexed as independent "Sub-Bitstreams" in one new video bitstream with use of the PSBI, GSBI, SSBI, and ESBI fields. Capability exchange for this mode is done by external means (for example Recommendation H.242). Reference Picture Selection back-channel data for responding to independent Sub-Bitstreams is supported using the BCPM and BSBI fields.

When in CPM mode, the CPM field shall be set to "1" in each of the independent H.263 Sub-Bitstreams. Sub-Bitstreams are identified by a stream identifier number using the Sub-Bitstream Indicators (SBIs) in the Picture and GOB or slice and EOSBS headers of each H.263 bitstream. The SBI indicates the number of the H.263 bitstream to which that header and all following information until the next Picture or GOB header or slice header in the composed video bitstream belongs.

Each Sub-Bitstream is considered as a normal H.263 bitstream and shall therefore comply with 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 preceding 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. Similarly, the mode state inference rules when using an extended picture type (PLUSPTYPE) in the picture header and all other aspects of the video bitstream operation shall operate independently and separately for each Sub-Bitstream.

As a matter of convention, the Sub-Bitstream with the lowest Sub-Bitstream identifier number (sent in SBI) is considered to have the highest priority in situations in which some conflict of resource

requirements may necessitate a choice of priority (unless a different priority convention is established by external means).

In order to mark the end of each Sub-Bitstream of the CPM mode, a syntax is provided as shown in Figure C.1, provided the ability to send this added syntax is first negotiated by external means (while CPM operation was defined in Version 1 of this Recommendation, the end of Sub-Bitstream syntax was added in Version 2 and thus is not considered a part of Version 1 CPM operation). The end of Sub-Bitstream syntax (ESTUF + EOSBS + ESBI) marks the end of each Sub-Bitstream, rather than the end of the entire stream as done by an EOS code.

NOTE – No ability to negotiate CPM Sub-Bitstream operation as defined herein for Recommendation H.263 was adopted into any ITU-T Recommendations for terminals (such as Recommendation H.324) prior to the creation of Version 2 of this Recommendation. Thus any external negotiation of CPM operation adopted into a future H-Series terminal Recommendation shall imply support for the end of Sub-Bitstream syntax unless otherwise specified in the H-Series terminal Recommendation.

There are three parts to the end of Sub-Bitstream syntax. Following mandatory byte alignment using ESTUF, an EOSBS codeword of 23 bits is sent (corresponding to a GOB header with GN = 30, which is otherwise unused in the syntax, followed by a single zero-valued bit which is reserved for future use). The EOSBS codeword is then followed by a two-bit ESBI codeword indicating which Sub-Bitstream is affected. This pair of codewords signifies that the sending of data for the associated Sub-Bitstream has ended and that any subsequent data sent for the same Sub-Bitstream shall be entirely independent from that which came before the EOSBS. In particular, the next picture for the Sub-Bitstream after the EOSBS code shall not be an INTER picture or any other picture type which can use forward temporal prediction (an I- or EI-picture is allowed, but a P-picture, PB-frame, Improved PB-frame, B-picture, or EP-picture is not).

The syntax of EOSBS and ESBI is described in the following subclauses. ESTUF is described in 5.1.26.

Figure C.1/H.263 – Syntax diagram for End of Sub-Bitstream Indicators

C.4.1 End Of Sub-Bitstream code (EOSBS) (23 bits)

The EOSBS code is a codeword of 23 bits. Its value is 0000 0000 0000 0000 1 11110 0. It is up to the encoder whether to insert this codeword or not. EOSBS should not be sent unless at least one picture header has previously been sent for the same Sub-Bitstream indicated in the following ESBI field, and shall not be sent unless the capability to send EOSBS has been negotiated by external means. EOSBS codes shall be byte aligned. This is achieved by inserting ESTUF before the EOSBS start code such that the first bit of the EOSBS start code is the first (most significant) bit of a byte (see 5.1.26).

The EOSBS code indicates that the sending of data for the indicated Sub-Bitstream has stopped and the Sub-Bitstream has been declared over, until re-started by issuance of another picture start code for that Sub-Bitstream. Subsequent pictures with the same Sub-Bitstream identifier number (ESBI) shall be completely independent of and shall not depend in any way on the pictures which were sent prior to the EOSBS code.

Control and other information associated with the video bitstream in general without specification of which Sub-Bitstreams these codes are meant to apply (such as a Freeze Picture Request or a Fast Update Request sent in H.242) should be presumed to apply to all of the *active* Sub-Bitstreams only. A Sub-Bitstream is considered active if at least one picture start code has been received for the

Sub-Bitstream and the last data sent which was applicable to that Sub-Bitstream was not an EOS or EOSBS + ESBI.

C.4.2 Ending Sub-Bitstream Indicator (ESBI) (2 bits)

ESBI is a fixed length codeword of two bits which follows immediately after EOSBS. It indicates the Sub-Bitstream number of the ending Sub-Bitstream. Its value is the natural two-bit binary representation of the Sub-Bitstream number.

ANNEX D

Unrestricted Motion Vector mode

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

The range of motion vectors and the VLC table used for coding the motion vector differences for the Unrestricted Motion Vector mode depend on whether the PLUSPTYPE field of the picture header is present. When PLUSPTYPE is present, the range of motion vectors also depends on the picture size and the value of the UUI field in the picture header.

D.1 Motion vectors over picture boundaries

In the default prediction mode of this Recommendation, motion vectors are restricted such that all pixels referenced by them are within the coded picture area (see 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 $0 \le x' \le 175$, $0 \le y' \le 143$. The given boundaries are integer pixel positions; however, (x′, y′) can also be a half-pixel position within these boundaries.

D.1.1 Restrictions for motion vector values

If PLUSPTYPE is present in the picture header, the motion vector values are restricted such that no element of the 16 × 16 or (8 × 8) region that is selected shall have a horizontal or vertical distance more than 15 pixels outside the coded picture area. Note that this is a smaller extrapolation range than when PLUSPTYPE is not present.

NOTE 1 – When PLUSPTYPE is absent, the extrapolation range is a maximum of 31.5 pixels outside the coded picture area when the Unrestricted Motion Vector mode is used, and 16 pixels outside the coded picture area when the Advanced Prediction mode (see Annex F) is used without the Unrestricted Motion Vector mode.

NOTE 2 – When the Advanced Prediction mode (see Annex F) is in use, the motion vector for each 16 × 16 or (8 × 8) region affects a larger areas, due to overlapped block motion compensation. This can cause the effective extrapolation range to increase for the "remote" motion vectors of the Advanced Prediction mode, since the amount of overlapping (4 pixels, or 8 pixels when the Reduced-Resolution Update mode is also in use) adds to the amount of extrapolation required (even though the range of values allowed for each motion vector remains the same as when the Advanced Prediction mode is not in use).

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 extended.

If the PLUSPTYPE field is not present in the picture header, the motion vector range is extended to [−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:

$$-31.5 \le MV_c \le 0 \qquad \text{if } -31.5 \le P_c \le -16$$

$$-16 + P_c \le MV_c \le 15.5 + P_c \qquad \text{if } -15.5 \le P_c \le 16$$

$$0 \le MV_c \le 31.5 \qquad \text{if } 16.5 \le P_c \le 31.5$$

In the Unrestricted Motion Vector mode, the interpretation of Table 14 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 14 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 MVD and $MVD_{2\text{-}4}$ is defined as the median value of the vector components MV1, MV2 and MV3 as defined in 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 G.4).

If PLUSPTYPE is present, the motion vector range does not depend on the motion vector prediction value. If the UUI field is set to "1" the motion vector range depends on the picture format. For standardized picture formats up to CIF the range is [−32, 31.5], for those up to 4CIF the range is

[−64, 63.5], and for those up to 16CIF the range is [−128, 127.5], and for even larger custom picture formats the range is [−256, 255.5]. The horizontal and vertical motion vector ranges may be different for custom picture formats. The horizontal and vertical ranges are specified in Tables D.1 and D.2.

Table D.1/H.263 – Horizontal motion vector range when PLUSPTYPE present and UUI = 1

Picture width	Horizontal motion vector range
4, …, 352	[−32, 31.5]
356, …, 704	[−64, 63.5]
708, …, 1408	[−128, 127.5]
1412, …, 2048	[−256, 255.5]

Table D.2/H.263 – Vertical motion vector range when PLUSPTYPE present and UUI = 1

Picture height	Vertical motion vector range
4, …, 288	[−32, 31.5]
292, …, 576	[−64, 63.5]
580, …, 1152	[−128, 127.5]

In the Reduced-Resolution Update mode, the specified range applies to the pseudo motion vectors. This implies that the resulting actual motion vector range is enlarged to approximately double size (see also Annex Q).

If UUI is set to "01", the motion vectors are not limited except by their distance to the coded area border as explained in D.1.1. The same limitation applies to the actual motion vectors (not just the pseudo motion vectors) in the Reduced-Resolution Update mode.

To encode the motion vectors when PLUSPTYPE is present, Table D.3 is used to encode the difference between the motion vector and the motion vector prediction. Every entry in Table D.3 has a single value (in contrast to Table 14). The motion vector range and the use of Table D.3 to encode motion vector data apply to all picture types when PLUSPTYPE is present.

Motion vectors differences are always coded as a pair of a horizontal and a vertical component. If a pair equals (0.5, 0.5) six consecutive zeros are produced. To prevent start code emulation, this occurrence shall be followed by one bit set to "1". This corresponds to sending one additional zero motion vector component.

Table D.3/H.263 – Motion Vector Table used when PLUSPTYPE present

Absolute value of vector difference in half-pixel units	Number of bits	Codes
0	1	1
1	3	0s0
"x_0"+2 (2:3)	5	0$x_0$1s0
"x_1x_0"+4 (4:7)	7	0$x_1$1$x_0$1s0
"$x_2x_1x_0$"+8 (8:15)	9	0$x_2$1$x_1$1$x_0$1s0
"$x_3x_2x_1x_0$"+16 (16:31)	11	0$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_4x_3x_2x_1x_0$"+32 (32:63)	13	0$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_5x_4x_3x_2x_1x_0$"+64 (64:127)	15	0$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_6x_5x_4x_3x_2x_1x_0$"+128 (128:255)	17	0$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_7x_6x_5x_4x_3x_2x_1x_0$"+256 (256:511)	19	0$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_8x_7x_6x_5x_4x_3x_2x_1x_0$"+512 (512:1023)	21	0$x_8$1$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0$"+1024 (1024:2047)	23	0$x_9$1$x_8$1$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_{10}x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0$"+2048 (2048:4095)	25	0x_{10}1$x_9$1$x_8$1$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0

Table D.3 is a regularly constructed reversible table. Each row represents an interval of motion vector differences in half-pixel units. The bits "…x_1x_0" denote all bits following the leading "1" in the binary representation of the absolute value of the motion vector difference. The bit "s" denotes the sign of the motion vector difference, "0" for positive and "1" for negative. The binary representation of the motion vector difference is interleaved with bits that indicate if the code continues or ends. For example, the motion vector difference –13 has sign s = 1 and binary representation $1x_2x_1x_0$ = 1101. It is encoded as 0 $x_2$1 $x_1$1 $x_0$1 s0 = 0 11 01 11 10. The 0 in the second position of the last group of two bits indicates the end of the code.

ANNEX E

Syntax-based Arithmetic Coding mode

E.1 Introduction

In the Variable Length Coding/Decoding (VLC/VLD) as described in clause 5, 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 bit rates, which can be achieved by arithmetic coding.

This annex describes the optional Syntax-based Arithmetic Coding (SAC) mode of this Recommendation. In this mode, all the corresponding variable length coding/decoding operations of this Recommendation are replaced with arithmetic coding/decoding operations. The capability of this H.263 mode 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 language.

```
#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 language.

```
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 this Recommendation, 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 E.1.

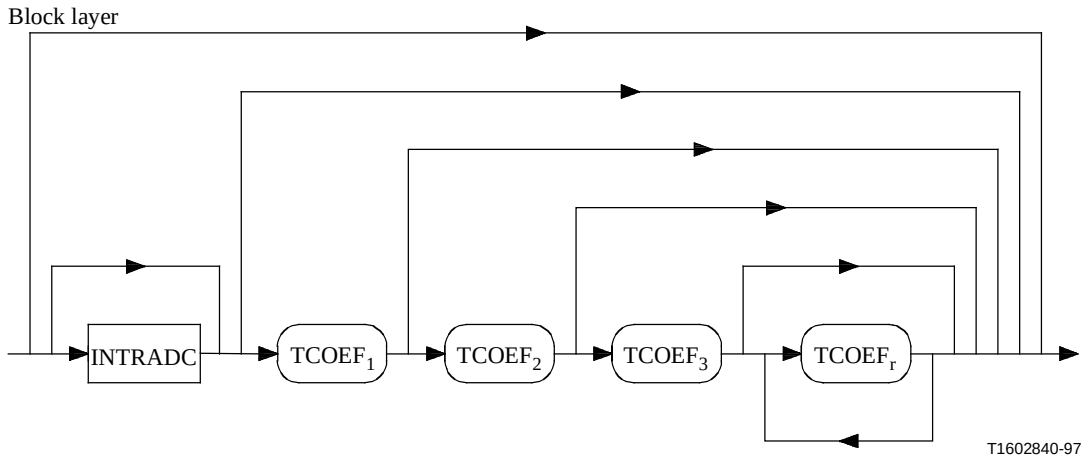


Figure E.1/H.263 – Structure of SAC Block layer

In Figure E.1, TCOEF1, TCOEF2, TCOEF3 and TCOEFr are Last-Run-Level symbols as defined in 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 Header layer symbols

The header layers of the syntax are considered to be those syntactical elements above the block and macroblock layers (see Figure 6 and the syntax specification in the text). The header levels of the basic Version 1 syntax can form three possible strings, (PSTUF)--PSC--TR--PTYPE--PQUANT--CPM--(PSBI)--(TRB-DBQUANT)--PEI--(PSUPP--PEI--...), (GSTUF)--GBSC--GN--(GSBI)--GFID--GQUANT, and (ESTUF)--(EOS)--(PSTUF). In the revised syntax of Version 2, the header levels of the syntax can have other structures (see Figure 6 and the syntax specification in the text). The header level syntax strings are directly sent to PSC_FIFO as in the normal VLC table mode of this Recommendation at encoder side, and are directly sent out from PSC_FIFO in the decoder after a legal PSC, GBSC, SSC, EOS, or EOSBS is detected.

If a header is not the first in a video session, the arithmetic encoder needs to be reset before sending the header by calling the following procedure. This procedure shall also be called at the end of a video session if (ESTUF)--EOS [or (ESTUF)--EOSBS for the Sub-Bitstream for which the last header was sent] is not sent.

```
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 Macroblock and Block layer symbols

Models for the macroblock and block layer symbols are included in E.8. The indices as given in the VLC tables of clause 5 are used for the indexing of the integers in the models.

The model for COD in P-pictures is named cumf_COD. The index for COD being "0" is 0, and is 1 for COD being "1". The model for MCBPC in P-pictures is named cumf_MCBPC_no4MVQ unless PLUSPTYPE is present in the picture header and either the Advanced Prediction mode (Annex F) or the Deblocking Filter mode (Annex J) is in use, in which case the model is named cumf_MCBPC_4MVQ The indexes for MCBPC are defined in Table 7 for I-pictures and Table 8 for P-pictures. The model for MCBPC in I-pictures is named by cumf_MCBPC_intra.

The model for MODB is cumf_MODB_G if Annex G is used or cumf_MODB_M if Annex M is used. The indexes for MODB are defined in Table 11 or Table M.1, respectively. 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 Tables 13 and 12, 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 Tables 14 and 15, 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 16. 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 17.

The model for INTRA_MODE is cumf_INTRA_AC_DC. The indexing is defined in Table I.1.

E.8 SAC models

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

int cumf_MCBPC_no4MVQ[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_4MVQ[26]={16383, 6880, 6092, 5178, 4916, 3965, 3880, 3795, 3768, 1491, 1190, 889, 655, 442, 416, 390, 360, 337, 334, 331, 327, 326, 88, 57, 26, 0};

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

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

int cumf_MODB_M[7] = {16383, 6717, 4568, 2784, 1370, 655, 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};

int cumf_INTRA_AC_DC[4]={16383, 9229, 5461, 0};

ANNEX F

Advanced Prediction mode

F.1 Introduction

This annex describes the optional Advanced Prediction mode of this Recommendation, 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 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 this Recommendation, one motion vector per macroblock is used except when in Advanced Prediction mode or Deblocking Filter 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 $MVD_{2\text{-}4}$ (see also 5.3.7 and 5.3.8).

The vectors are obtained by adding predictors to the vector differences indicated by MVD and $MVD_{2\text{-}4}$ in a similar way as when only one motion vector per macroblock is present, according to the decision rules given in 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 F.1. 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 (this definition is given in the upper left of the four sub-figures of Figure F.1).

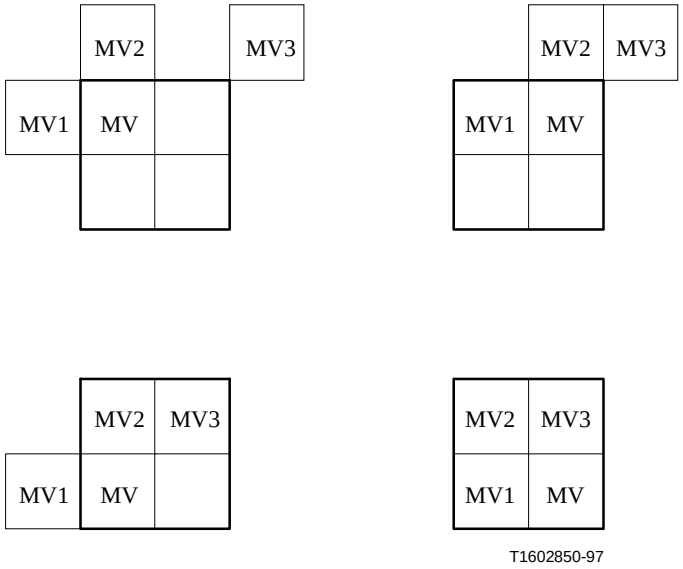


Figure F.1/H.263 – Redefinition of the candidate predictors MV1, MV2 and MV3 for each of the luminance blocks in a macroblock

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. 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 F.1.

Table F.1/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 6.1.2. In Advanced Prediction mode, the prediction for luminance is obtained by overlapped motion compensation as described in F.3. 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.

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 F.3). 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 F.4).

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

Let *(m,n)* be an integer block index in a picture, as given by:

$$m = x / 8 \qquad \text{and} \qquad n = y / 8,$$

where "/" denotes division with truncation.

Let *(i,j)* be an integer pixel location in an 8×8 block, given by:

$$i = x - m \cdot 8 \qquad \text{and} \qquad j = y - n \cdot 8,$$

resulting in:

$$(x,y) = (m \cdot 8 + i, n \cdot 8 + j).$$

Let $(MV^k{}_x, MV^k{}_y)$ be a motion vector, which may contain full-pixel or half pixel offsets, with $k = 0$, 1, or 2. For example, $(MV^k{}_x, MV^k{}_y)$ can be equal to (–7.0, 13.5). Here, $(MV^0{}_x, MV^0{}_y)$ denotes the motion vector for the current block *(m,n)*, $(MV^1{}_x, MV^1{}_y)$ 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 *(m,n)* as defined above.

Then the creation of each pixel, *P(x,y)*, in an 8×8 luminance prediction block with block index *(m,n)* is governed by the following equation:

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

where *q(x,y)*, *r(x,y)* and *s(x,y)* are the prediction values taken from the referenced picture as defined by:

$$q(x,y) = p(x + MV^0{}_x, y + MV^0{}_y),$$

$$r(x,y) = p(x + MV^1{}_x, y + MV^1{}_y),$$

$$s(x,y) = p(x + MV^2{}_x, y + MV^2{}_y),$$

where $p(x + MV^k{}_x, y + MV^k{}_y)$ is the prediction value at position $(x + MV^k{}_x, y + MV^k{}_y)$ in the referenced picture. Note that $(x + MV^k{}_x, y + MV^k{}_y)$ may be outside the picture and can be at a full or half-pixel position. In cases using half-pixel motion vectors, $p(x + MV^k{}_x, y + MV^k{}_y)$ refers to the value obtained after the interpolation process described in 6.1.2 is applied.

The matrices $H_0(i,j)$, $H_1(i,j)$ and $H_2(i,j)$ are defined in Figures F.2, F.3 and F.4, where *(i, j)* denotes the column and row, respectively, of the matrix.

When neither the Slice Structured mode (see Annex K) nor the Independent Segment Decoding mode (see Annex R) are in use, remote motion vectors from other video picture segments are used in the same way as remote motion vectors inside the current GOB. If either the Slice Structured mode or the Independent Segment Decoding mode are in use, the remote motion vectors corresponding to blocks from other video picture segments are set to the motion vector of the current block, regardless of the other conditions described in the next paragraph. (See Annex R for the definition of a video picture segment.)

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, 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, if the current block is at the bottom of the macroblock (for block number 3 or 4, see Figure 5), 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 Figures F.2, F.3 and F.4.

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 F.2/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 F.3/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 F.4/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

G.1 Introduction

This annex describes the optional PB-frames mode of this Recommendation. 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 G.1.

An improved version of the PB-frames mode, termed the "Improved PB-frames mode," is described in Annex M. The PB-frames mode as described in this annex is retained herein only for purposes of compatibility with systems designed prior to the adoption of the Improved PB-frames mode. For this reason, the PB-frames mode as described in this annex cannot be used with the additional features of the syntax which require the use of PLUSPTYPE.

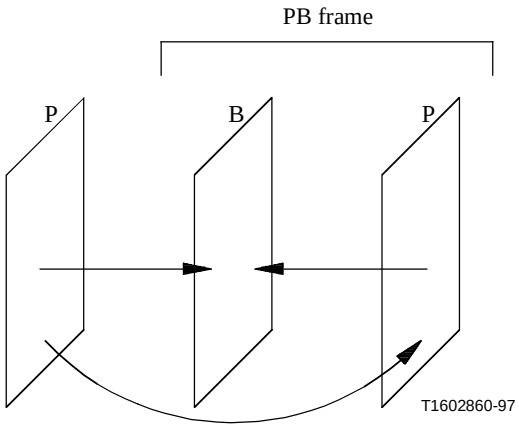


Figure G.1/H.263 – Prediction in PB-frames mode

G.2 PB-frames and INTRA blocks

When PB-frames are used, the coding mode INTRA has the following meaning (see also 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 $MVD_{2\text{-}4}$ are never used for INTRA (see also Table 10). 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 5.4). The structure of the block layer is shown in Figure 11. INTRADC is present for every P-block of the macroblock if MCBPC indicates macroblock type 3 or 4 (see Tables 7 and 8). 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 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 the temporal reference TR (or the combined extended temporal reference ETR and temporal reference TR in an Improved PB frame when a custom picture clock frequency is in use) from the last picture header (see 5.1.2). If TR_D is negative, then $TR_D = TR_D + d$ where d = 256 for CIF picture frequency and 1024 for any custom picture clock frequency.
- TR_B: See 5.1.2.

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:

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

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

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

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 F.1. 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 F.1.

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 subclause a block means an 8 × 8 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 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 bidirectional 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 G.2 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).

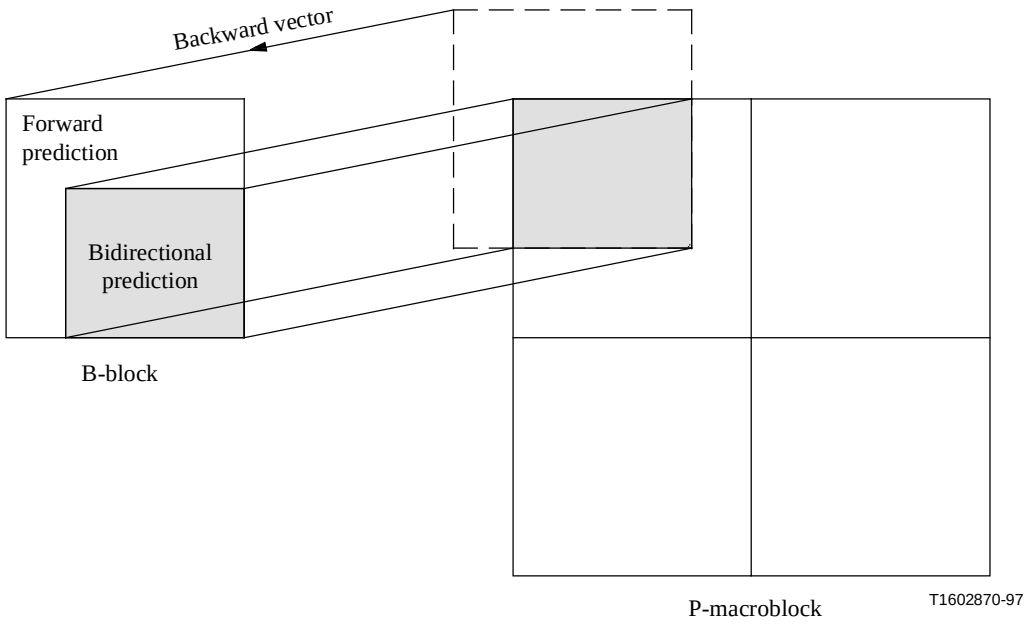


Figure G.2/H.263 – Forward and bidirectional prediction for a B-block

Bidirectional 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 language.

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

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 Recommendation H.324. Both the framing and the forward error correction code are the same as in Recommendation 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 H.1). 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 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.

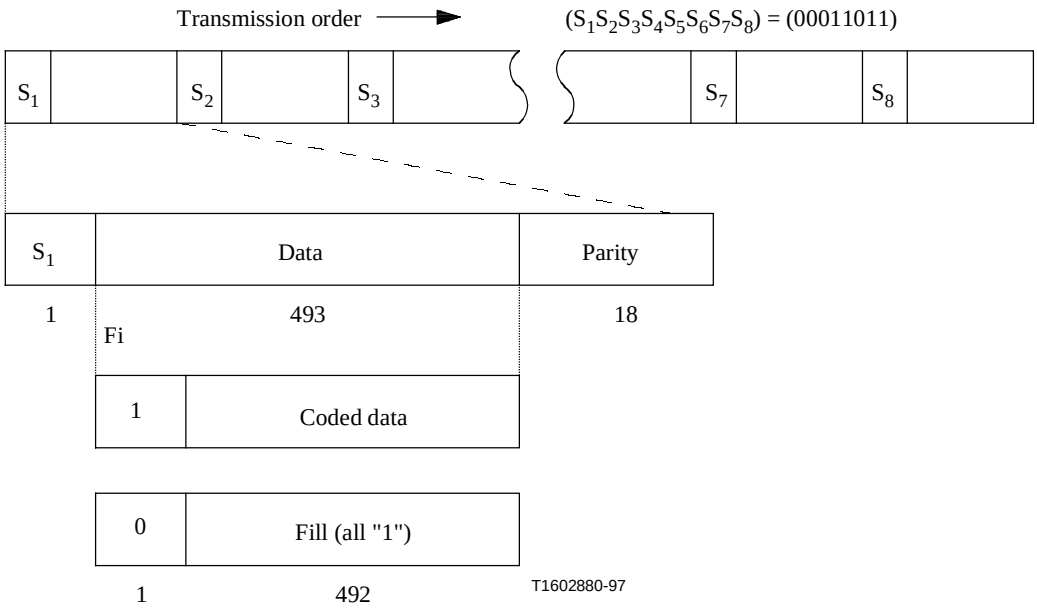


Figure H.1/H.263 – Error correcting frame

ANNEX I

Advanced INTRA Coding mode

This annex describes the optional Advanced INTRA Coding mode of this Recommendation. The capability of this H.263 mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

I.1 Introduction

This optional mode alters the decoding of macroblocks of type "INTRA" (macroblocks of other types are not affected). The coding efficiency of INTRA macroblocks is improved by using:

1) INTRA-block prediction using neighbouring INTRA blocks for the same component (Y, C_B, or C_R);

2) modified inverse quantization for INTRA coefficients; and

3) a separate VLC for INTRA coefficients.

A particular INTRA-coded block may be predicted from the block above the current block being decoded, from the block to the left of the current block being decoded, or from both. Special cases exist for situations in which the neighbouring blocks are not INTRA coded or are not in the same video picture segment. Block prediction always uses data from the same luminance or color difference component (Y, C_B, or C_R) as the block being decoded. In prediction, DC coefficients are always predicted in some manner. The first row of AC coefficients may be predicted from those in the block above, or the first column of AC coefficients may be predicted from those in the block to the left, or only the DC coefficient may be predicted as an average from the block above and the block to the left, as signalled on a macroblock-by-macroblock basis. The remaining AC coefficients are never predicted. Inverse quantization of the INTRADC coefficient is modified to allow a varying quantization step size, unlike in the main text of this Recommendation where a fixed step size of 8 is used for INTRADC coefficients. Inverse quantization of all INTRA coefficients is performed without a "dead-zone" in the quantizer reconstruction spacing.

I.2 Syntax

When using the Advanced INTRA Coding mode, the macroblock layer syntax is altered as specified in Figure I.1. The syntax as shown in Figure I.1 is the same as that defined in 5.3 except for the insertion of an additional INTRA_MODE field for INTRA macroblocks. INTRA_MODE is present only when MCBPC indicates a macroblock of type INTRA (macroblock type 3 or 4). The prediction mode is coded using the variable length code shown in Table I.1. One prediction mode is transmitted per INTRA macroblock.

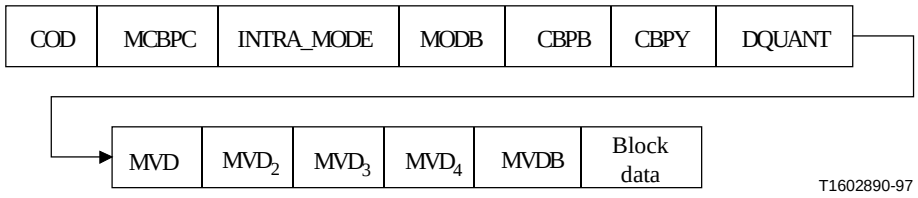


Figure I.1/H.263 – Structure of macroblock layer

Table I.1/H.263 – VLC for INTRA_MODE

Index	Prediction Mode	VLC
0	0 (DC Only)	0
1	1 (Vertical DC & AC)	10
2	2 (Horizontal DC & AC)	11

I.3 Decoding process

Two scans in addition to the zigzag scan are employed. The two added scans are shown in Figure I.2 parts a) and b), and the zigzag scan is shown in Figure 14.

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

a) Alternate-Horizontal scan

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

b) Alternate-Vertical scan (as in Recommendation H.262)

Figure I.2/H.263 – Alternate DCT scanning patterns for Advanced INTRA coding

For INTRA-coded blocks, if Prediction Mode = 0, the zigzag scan shown in Figure 14 is selected for all blocks in the macroblock, otherwise, the prediction direction is used to select a scan for the macroblock.

Prediction Mode = 1 uses the vertically adjacent block in forming a prediction. This prediction mode is designed for INTRA blocks which are dominated by stronger horizontal frequency content, so the vertically adjacent block is used to predict the horizontal frequency content of the current block, with a prediction of zero for all coefficients representing vertical AC content. Then the scanning pattern is chosen to scan the stronger horizontal frequencies prior to the vertical ones, using the Alternate-Horizontal scan.

Prediction Mode = 2 uses the horizontally adjacent block in forming a prediction. This prediction mode is designed for INTRA blocks which are dominated by stronger vertical frequency content, so the horizontally adjacent block is used to predict the vertical frequency content of the current block, with a prediction of zero for all coefficients representing horizontal AC content. Then the scanning pattern is chosen to scan the stronger vertical frequencies prior to the horizontal ones, using the Alternate-Vertical scan.

For non-INTRA blocks, the 8×8 blocks of transform coefficients are scanned with "zigzag" scanning as shown in Figure 14.

A separate VLC table is used for all INTRADC and INTRA AC coefficients. This table is specified in Table I.2. Note that the VLC codeword entries used in Table I.2 are the same as those used in the normal TCOEF table (Table 16) used when Advanced INTRA coding is not in use, but with a different interpretation of LEVEL and RUN (without altering LAST).

Table I.2/H.263 – VLC for INTRA TCOEF

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
0	0	0	1	3	10s
1	0	1	1	5	1111s
2	0	3	1	7	0101 01s
3	0	5	1	8	0010 111s
4	0	7	1	9	0001 1111s
5	0	8	1	10	0001 0010 1s
6	0	9	1	10	0001 0010 0s
7	0	10	1	11	0000 1000 01s
8	0	11	1	11	0000 1000 00s
9	0	4	3	12	0000 0000 111s
10	0	9	2	12	0000 0000 110s
11	0	13	1	12	0000 0100 000s
12	0	0	2	4	110s
13	0	1	2	7	0101 00s
14	0	1	4	9	0001 1110s
15	0	1	5	11	0000 0011 11s
16	0	1	6	12	0000 0100 001s
17	0	1	7	13	0000 0101 0000s
18	0	0	3	5	1110s
19	0	3	2	9	0001 1101s
20	0	2	3	11	0000 0011 10s
21	0	3	4	13	0000 0101 0001s
22	0	0	5	6	0110 1s
23	0	4	2	10	0001 0001 1s
24	0	3	3	11	0000 0011 01s
25	0	0	4	6	0110 0s

Table I.2/H.263 – VLC for INTRA TCOEF (*continued*)

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
26	0	5	2	10	0001 0001 0s
27	0	5	3	13	0000 0101 0010s
28	0	2	1	6	0101 1s
29	0	6	2	11	0000 0011 00s
30	0	0	25	13	0000 0101 0011s
31	0	4	1	7	0100 11s
32	0	7	2	11	0000 0010 11s
33	0	0	24	13	0000 0101 0100s
34	0	0	8	7	0100 10s
35	0	8	2	11	0000 0010 10s
36	0	0	7	7	0100 01s
37	0	2	4	11	0000 0010 01s
38	0	0	6	7	0100 00s
39	0	12	1	11	0000 0010 00s
40	0	0	9	8	0010 110s
41	0	0	23	13	0000 0101 0101s
42	0	2	2	8	0010 101s
43	0	1	3	8	0010 100s
44	0	6	1	9	0001 1100s
45	0	0	10	9	0001 1011s
46	0	0	12	10	0001 0000 1s
47	0	0	11	10	0001 0000 0s
48	0	0	18	10	0000 1111 1s
49	0	0	17	10	0000 1111 0s
50	0	0	16	10	0000 1110 1s
51	0	0	15	10	0000 1110 0s
52	0	0	14	10	0000 1101 1s
53	0	0	13	10	0000 1101 0s
54	0	0	20	12	0000 0100 010s
55	0	0	19	12	0000 0100 011s
56	0	0	22	13	0000 0101 0110s
57	0	0	21	13	0000 0101 0111s
58	1	0	1	5	0111s
59	1	14	1	10	0000 1100 1s
60	1	20	1	12	0000 0000 101s
61	1	1	1	7	0011 11s
62	1	19	1	12	0000 0000 100s
63	1	2	1	7	0011 10s
64	1	3	1	7	0011 01s
65	1	0	2	7	0011 00s
66	1	5	1	8	0010 011s
67	1	6	1	8	0010 010s
68	1	4	1	8	0010 001s
69	1	0	3	8	0010 000s
70	1	9	1	9	0001 1010s
71	1	10	1	9	0001 1001s
72	1	11	1	9	0001 1000s
73	1	12	1	9	0001 0111s
74	1	13	1	9	0001 0110s

Table I.2/H.263 – VLC for INTRA TCOEF (*concluded*)

INDEX	LAST	RUN	|LEVEL|	BITS	VLC CODE
75	1	8	1	9	0001 0101s
76	1	7	1	9	0001 0100s
77	1	0	4	9	0001 0011s
78	1	17	1	10	0000 1100 0s
79	1	18	1	10	0000 1011 1s
80	1	16	1	10	0000 1011 0s
81	1	15	1	10	0000 1010 1s
82	1	2	2	10	0000 1010 0s
83	1	1	2	10	0000 1001 1s
84	1	0	6	10	0000 1001 0s
85	1	0	5	10	0000 1000 1s
86	1	4	2	11	0000 0001 11s
87	1	3	2	11	0000 0001 10s
88	1	1	3	11	0000 0001 01s
89	1	0	7	11	0000 0001 00s
90	1	2	3	12	0000 0100 100s
91	1	1	4	12	0000 0100 101s
92	1	0	9	12	0000 0100 110s
93	1	0	8	12	0000 0100 111s
94	1	21	1	13	0000 0101 1000s
95	1	22	1	13	0000 0101 1001s
96	1	23	1	13	0000 0101 1010s
97	1	7	2	13	0000 0101 1011s
98	1	6	2	13	0000 0101 1100s
99	1	5	2	13	0000 0101 1101s
100	1	3	3	13	0000 0101 1110s
101	1	0	10	13	0000 0101 1111s
102	ESCAPE			7	0000 011

Depending on the value of INTRA_MODE, either one or eight coefficients are prediction residuals that must be added to a predictor as described below. Figure I.3 shows three 8 × 8 blocks of final reconstructed DCT levels for the same component (Y, C_B, or C_R), labelled RecA'(u,v), RecB'(u,v) and RecC'(u,v), where u and v are column (horizontal) and row (vertical) indices, respectively. The reconstruction process differs from the processing described in 6.2.1. INTRADC residuals are reconstructed differently by use of a variable step size, as opposed to using Table 15, and then a predictor is added to the residual values to obtain the final coefficient reconstruction value. INTRA coefficients other than INTRADC are also reconstructed differently than in 6.2.1, using a reconstruction spacing without a "dead-zone" and then in some cases adding a predictor to obtain the final coefficient reconstruction value. The block may contain both DC and AC prediction residuals.

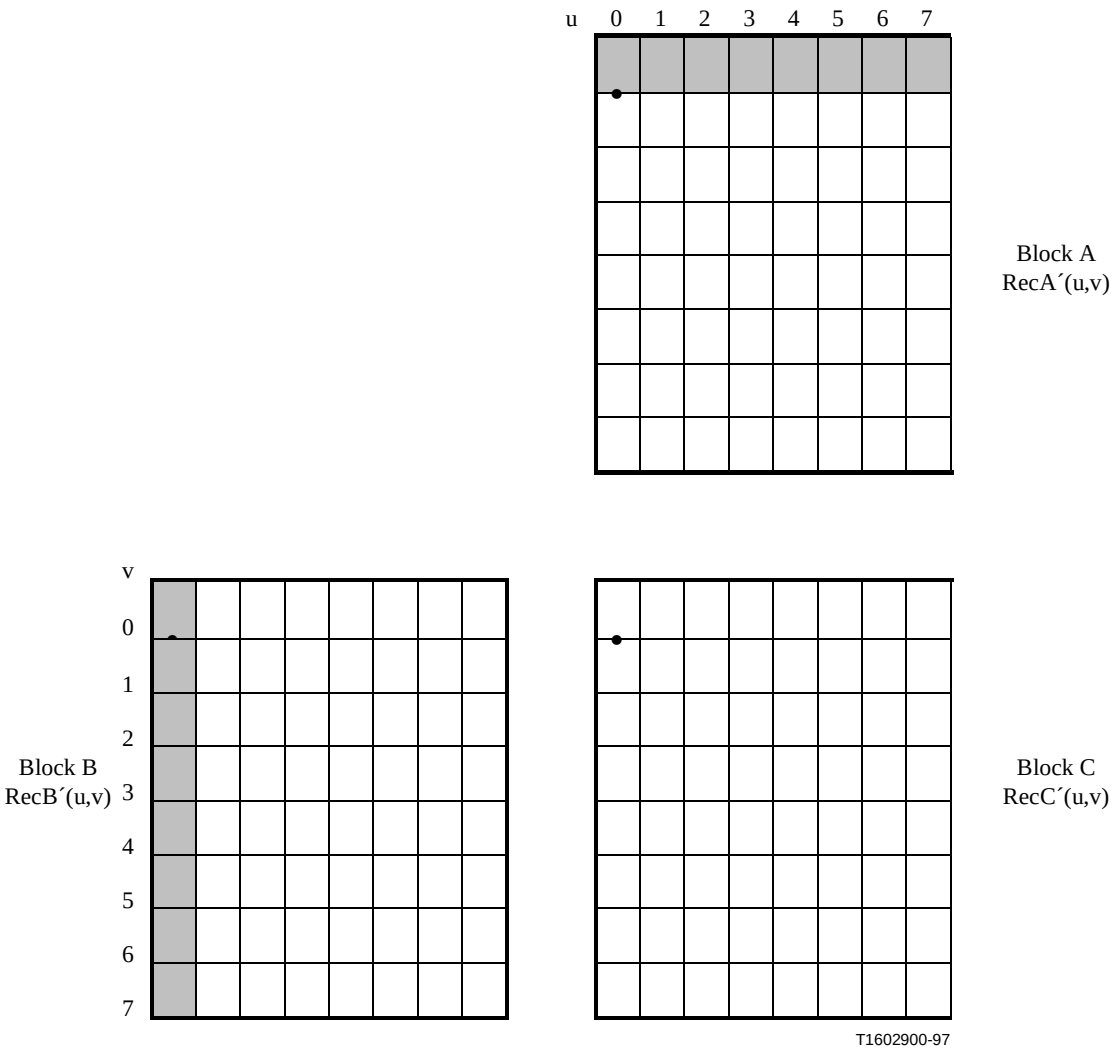


Figure I.3/H.263 – Three neighbouring blocks in the DCT domain

The definitions of MCBPC and CBPY are changed when Advanced INTRA coding is in use. When Advanced INTRA coding is in use, INTRADC transform coefficients are no longer handled as a separate case, but are instead treated in the same way as the AC coefficients in regard to MCBPC and CBPY. This means that a zero INTRADC will not be coded as a LEVEL, but will simply increase the run for the following AC coefficients.

The inverse quantization process for the B-part of an Improved PB frame (see Annex M) is not altered by the use of the Advanced INTRA coding mode.

Define RecC(u,v) to be the reconstructed coefficient *residuals* of the current block. For all INTRA coefficients, the reconstructed residual value is obtained by:

$$RecC(u,v) = 2 * QUANT * LEVEL(u,v) \qquad u = 0, \ldots, 7, v = 0, \ldots, 7.$$

NOTE – LEVEL(u,v) denotes a quantity having both a magnitude and a sign in the above equation.

Define RecC´(u,v) to be the *final* reconstructed coefficient values of the current block (after adjustments for prediction, oddification as described below, and clipping). The final reconstructed coefficient values RecC´(u,v) are recovered by adding RecC(u,v) to the appropriate prediction as signalled in the INTRA_MODE field, altering the least-significant bit if needed for oddification of the DC coefficient, and clipping.

RecA´(u,v) denotes the *final* reconstructed coefficient values for the block immediately above the current block. RecB´(u,v) denotes the *final* reconstructed coefficient values of the block immediately to the left of the current block.

The ability to use the reconstructed coefficient values for blocks A and B in the prediction of the coefficient values for block C depends on whether blocks A and B are in the same video picture segment as block C. A block is defined to be "in the same video picture segment" as another block only if the following conditions are fulfilled:

1) The relevant block is within the boundary of the picture, and

2) If not in Slice Structured mode (see Annex K), the relevant block is either within the same GOB or no GOB header is present for the current GOB, and

3) If in Slice Structured mode, the relevant block is within the same slice.

The block C to be decoded is predicted only from INTRA blocks within the same video picture segment as block C, as shown below.

If Prediction Mode = 0 is used (DC prediction only) and blocks A and B are both INTRA blocks within the same video picture segment as block C, then the DC coefficient of block C is predicted from the average (with truncation) of the DC coefficients of blocks A and B. If only one of the two blocks A and B is an INTRA block within the same video picture segment as block C, then the DC coefficient of only this one of the two blocks is used as the predictor for Prediction Mode = 0. If neither of the two blocks A and B are INTRA blocks within the same video picture segment as block C, then the prediction uses the value of 1024 as the predictor for the DC coefficient.

If Prediction Mode = 1 or 2 (vertical DC & AC or horizontal DC & AC prediction) and the referenced block (block A or block B) is not an INTRA block within the same video picture segment as block C, then the prediction uses the value of 1024 as the predictor for the DC coefficient and the value of 0 as the predictor for the AC coefficients of block C.

A process of "oddification" is applied to the DC coefficient in order to minimize the impact of IDCT mismatch errors. Certain values of coefficients can cause a round-off error mismatch between different IDCT implementations, especially certain values of the (0,0), (0,4), (4,0), and (4,4) coefficients. For example, a DC coefficient of 8k + 4 for some integer k results in an inverse-transformed block having a constant value k + 0.5, for which slight errors can cause rounding in different directions on different implementations.

Define the clipAC() function to indicate clipping to the range –2048 to 2047. Define the clipDC() function to indicate clipping to the range 0 to 2047. Define the oddifyclipDC(x) function as:

```
If (x is even) {
    result = clipDC(x+1)
} else {
    result = clipDC(x)
}
```

The reconstruction for each INTRA prediction mode is then specified as follows, in which the operator "/" is defined as division by truncation:

Mode 0: DC prediction only.

```
RecC´(u,v) = clipAC( RecC(u,v) )                (u,v) ≠ (0,0), u = 0, …, 7, v = 0, …, 7.
If (block A and block B are both INTRA coded and are both in the same video picture segment as
block C) {
    tempDC = RecC(0,0) + ( RecA´(0,0)  + RecB´(0,0)  ) / 2
} else {
```

```
    If (block A is INTRA coded and is in the same video picture segment as block C) {
      tempDC = RecC(0,0) + RecA´(0,0)
    } else {
      If (block B is INTRA coded and is in the same video picture segment as block C) {
        tempDC = RecC(0,0) + RecB´(0,0)
      } else {
        tempDC = RecC(0,0) + 1024
      }
    }
  }
  RecC'(0,0) = oddifyclipDC( tempDC )
```

Mode 1: DC and AC prediction from the block above.

```
  If (block A is INTRA coded and is in the same video picture segment as block C) {
    tempDC = RecC(0,0) + RecA´(0,0)
    RecC´(u,0) = clipAC( RecC(u,0) + RecA´(u,0) )        u = 1, …, 7,
    RecC´(u,v) = clipAC( RecC(u,v) )                     u = 0, …, 7, v = 1, …, 7.
  } else {
    tempDC = RecC(0,0) + 1024
    RecC´(u,v) = clipAC( RecC(u,v) )                     (u,v) ≠ (0,0), u = 0, …,7, v = 0, …, 7
  }
  RecC´(0,0) = oddifyclipDC( tempDC )
```

Mode 2: DC and AC prediction from the block to the left.

```
  If (block B is INTRA coded and is in the same video picture segment as block C) {
    tempDC = RecC(0,0) + RecB´(0,0)
    RecC´(0,v) = clipAC( RecC(0,v) + RecB´(0,v) )        v = 1, …, 7,
    RecC´ (u,v) = clipAC( RecC(u,v) )                    u = 1, …, 7, v = 0, …, 7.
  } else {
    tempDC = RecC(0,0) + 1024
    RecC´(u,v) = clipAC( RecC(u,v) )                     (u,v) ≠ (0,0), u = 0, …, 7, v = 0, …, 7
  }
  RecC'(0,0) = oddifyclipDC( tempDC )
```

ANNEX J

Deblocking Filter mode

J.1 Introduction

This annex describes the use of an optional block edge filter within the coding loop. The main purpose of the block edge filter is to reduce blocking artifacts. The filtering is performed on 8 × 8 block edges. Motion vectors may have either 8 × 8 or 16 × 16 resolution (see J.2). The processing described in this annex applies only for the P-, I-, EP-, or EI-pictures or the P-picture part of an Improved PB-frame. (Possible filtering of B-pictures or the B-picture part of an Improved PB-frame is not a matter for standardization; however, some type of filtering is recommended for improved picture quality.) The capability of this mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

J.2 Relation to UMV and AP modes (Annexes D and F)

The use of the deblocking filter mode has similar effects on picture quality as Overlapped Block Motion Compensation (OBMC) as defined in Annex F when used alone. When both techniques are used together, a further improvement in picture quality can be obtained. The Advanced Prediction mode (see also Annex F) consists of three elements:

1) four motion vectors per macroblock as defined in F.2;

2) overlapped motion compensation for luminance as defined in F.3;

3) motion vectors over picture boundaries as defined in D.1.

In order for the Deblocking Filter mode to be able to provide maximum performance when complexity considerations may prevent use of the OBMC part of the Advanced Prediction mode, the Deblocking Filter mode includes the ability to use four motion vectors per macroblock and motion vectors over picture boundaries.

In summary, the three options defined in Annexes D, F and J contain the following five coding elements:

1) motion vectors over picture boundaries (D.1);

2) extension of motion vector range (D.2);

3) four motion vectors per macroblock (F.2);

4) overlapped motion compensation for luminance (F.3);

5) deblocking edge filter (J.3).

Table J.1 indicates which of the five elements are turned on depending on which of the three options defined in Annexes D, F and J are turned on.

Table J.1/H.263 – Feature Elements for UMV, AP, and DF modes

Unrestricted Motion Vector mode	Advanced Prediction mode	Deblocking Filter mode	Motion vectors over picture boundaries	Extension of motion vector range	Four motion vectors per macroblock	Overlapped motion compensation for luminance	Deblocking edge filter
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
OFF	OFF	ON	ON	OFF	ON	OFF	ON
OFF	ON	OFF	ON	OFF	ON	ON	OFF
OFF	ON	ON	ON	OFF	ON	ON	ON
ON	OFF	OFF	ON	ON	OFF	OFF	OFF
ON	OFF	ON	ON	ON	ON	OFF	ON
ON	ON	OFF	ON	ON	ON	ON	OFF
ON	ON	ON	ON	ON	ON	ON	ON

J.3 Definition of the deblocking edge filter

The filter operations are performed across 8 × 8 block edges at the encoder as well as on the decoder side. The reconstructed image data (the sum of the prediction and reconstructed prediction error) are clipped to the range of 0 to 255 as described in 6.3.2. Then the filtering is applied, which alters the picture that is to be stored in the picture store for future prediction. The filtering operations include an additional clipping to ensure that resulting pixel values stay in the range 0...255. No filtering is performed across a picture edge, and when the Independent Segment Decoding mode is in use, no

filtering is performed across slice edges when the Slice Structured mode is in use (see Annexes K and R) or across the top boundary of GOBs having GOB headers present when the Slice Structured mode is not in use (see Annex R). Chrominance as well as luminance data are filtered.

When the mode described in this annex is used together with the Improved PB-frames mode of Annex M, the backward prediction of the B-macroblock is based on the reconstructed P-macroblock (named P_{REC} in G.5) after the clipping operation but before the deblocking edge filter operations. The forward prediction of the B-macroblock is based on the filtered version of the previous decoded picture (the same picture data that are used for prediction of the P-macroblock).

The deblocking filter operates using a set of four (clipped) pixel values on a horizontal or vertical line of the reconstructed picture, denoted as A, B, C and D, of which A and B belong to one block called block1 and C and D belong to a neighbouring block called block2 which is to the right of or below block1. Figure J.1 shows examples for the position of these pixels.

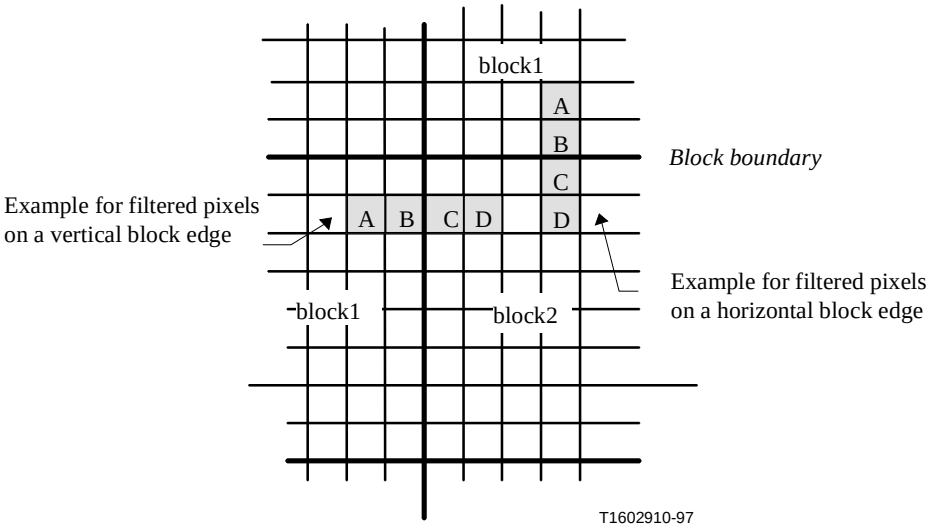


Figure J.1/H.263 – Examples of positions of filtered pixels

One or both of the following conditions must be fulfilled in order to apply the filter across a particular edge:

– Condition 1: block1 belongs to a coded macroblock (COD==0 || MB-type == INTRA); or

– Condition 2: block2 belongs to a coded macroblock (COD==0 || MB-type == INTRA).

If filtering is to be applied across the edge, A, B, C, and D shall be replaced by A1, B1, C1, D1 where:

B1 = clip(B + d1)

C1 = clip(C – d1)

A1 = A – d2

D1 = D + d2

d = (A–4B+4C–D) / 8

d1 = UpDownRamp(d, STRENGTH)

d2 = clipd1((A – D) / 4, d1/2)

UpDownRamp(x, STRENGTH) = SIGN(x) * (MAX(0, abs(x)–MAX(0, 2*(abs(x) – STRENGTH))))

STRENGTH depends on QUANT and determines the amount of filtering. The relation between STRENGTH and QUANT is given in Table J.2.

QUANT = quantization parameter used for block2 if block2 belongs to a coded macroblock, or

QUANT = quantization parameter used for block1 if block2 does not belong to a coded macroblock (but block1 does).

Table J.2/H.263 – Relationship between QUANT and STRENGTH of filter

QUANT	STRENGTH	QUANT	STRENGTH
1	1	17	8
2	1	18	8
3	2	19	8
4	2	20	9
5	3	21	9
6	3	22	9
7	4	23	10
8	4	24	10
9	4	25	10
10	5	26	11
11	5	27	11
12	6	28	11
13	6	29	12
14	7	30	12
15	7	31	12
16	7		

The function clip(x) is defined according to 6.3.2, and the function clipd1(x, lim) clips x to the range ± abs(lim). The symbol "/" denotes division by truncation toward zero.

Figure J.2 shows how the value of d1 varies as a function of d. As a result, the filter has an effect only if d is smaller than 2*STRENGTH (and different from zero). This is to prevent the filtering of strong true edges in the picture content. However, if the Reduced-Resolution Update mode is in use, STRENGTH is set to infinity, and as a result, the value of d1 is always equal to the value of d (see Q.7.2).

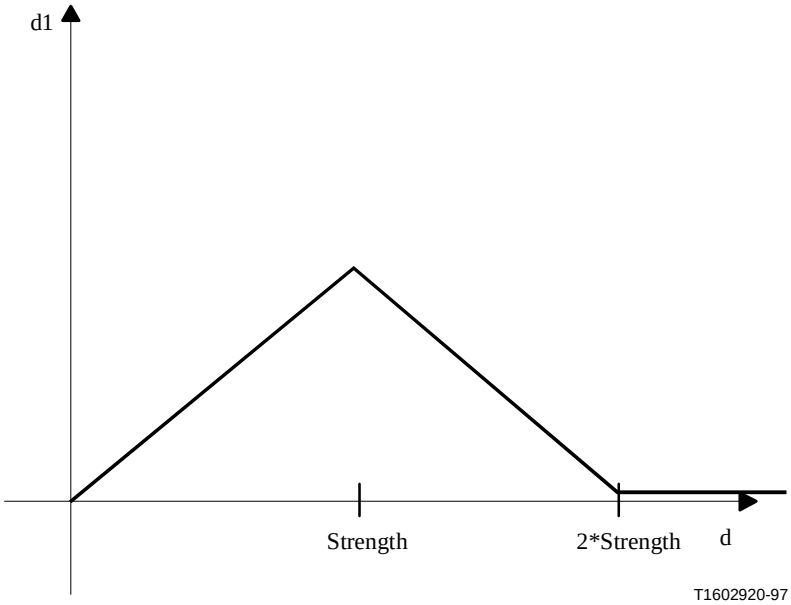


Figure J.2/H.263 – Parameter d1 as a function of parameter d for deblocking filter mode

The definition of d1 is designed to ensure that small mismatches between the encoder and decoder will remain small and will not build up over multiple pictures of a video sequence. This would be a problem, for example, with a condition that simply switches the filter on or off, because a mismatch of only ±1 for d could then cause the filter to be switched on at the encoder side and off at the decoder side, or vice versa.

Due to rounding effects, the order of edges where filtering is performed must be specified.

Filtering across horizontal edges:

Basically this process is assumed to take place first. More precisely, the pixels $\begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix}$ that are used in filtering across a horizontal edge shall not have been influenced by previous filtering across a vertical edge.

Filtering across vertical edges:

Before filtering across a vertical edge using pixels (A, B, C, D), all modifications of pixels (A, B, C, D) resulting from filtering across a horizontal edge shall have taken place.

Note that if one or more of the pixels (A, B, C, D) taking part in a filtering process are outside of a picture, no filtering takes place. Also, if the Independent Segment Decoding mode is in use (see Annex R) and one or more of the pixels (A, B, C, D) taking part in a filtering process are in different video picture segments (see I.3 concerning when a block is considered to be in the same video picture segment), no filtering is performed.

ANNEX K

Slice Structured mode

K.1 Introduction

This annex describes the optional Slice Structured mode of this Recommendation. The capability of this H.263 mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header. In order to facilitate optimal usage in a number of environments, this mode contains two submodes which are also capable of being signalled by external means (for example, Recommendation H.245). These two submodes are used to indicate whether or not rectangular slices will be used and/or whether the slices will be transmitted in sequential order or sent in an arbitrary order.

A slice is defined as a slice header followed by consecutive macroblocks in scanning order. An exception is for the slice that immediately follows the picture start code in the bitstream for a picture (which is not necessarily the slice starting with macroblock 0). In this case only part of the slice header is transmitted as described in K.2. The slice layer defines a video picture segment, and is used in place of the GOB layer in this optional mode. A slice video picture segment starts at a macroblock boundary in the picture and contains a number of macroblocks. Different slices within the same picture shall not overlap with each other, and every macroblock shall belong to one and only one slice.

This mode contains two submodes, which are signalled in the SSS field of the picture header:

1) The Rectangular Slice submode (RS): When RS is in use, the slice shall occupy a rectangular region of width specified by the SWI parameter of the slice header in units of macroblocks, and contains a number of macroblocks in scanning order within the rectangular region. When the Rectangular Slice submode is not in use, the SWI field is not present in the slice header and a slice contains a number of macroblocks in scanning order within the picture as a whole.

2) The Arbitrary Slice Ordering submode (ASO): When ASO is in use, the slices may appear in any order within the bitstream. When ASO is not in use, the slices must be sent in the (unique) order for which the MBA field of the slice header is strictly increasing from each slice to each subsequent slice in the picture.

Slice boundaries are treated differently from simple macroblock boundaries in order to allow slice header locations within the bitstream to act as resynchronization points for bit error and packet loss recovery and in order to allow out-of-order slice decoding within a picture. Thus, no data dependencies can cross the slice boundaries within the current picture, except for the Deblocking Filter mode which, when in use without the Independent Segment Decoding mode, filters across the boundaries of the blocks in the picture. However, motion vectors within a slice can cause data dependencies which cross the slice boundaries in the reference picture used for prediction purposes, unless the optional Independent Segment Decoding mode is in use.

The following rules are adopted to ensure that slice boundary locations can act as resynchronization points and to ensure that slices can be sent out of order without causing additional decoding delays:

1) The prediction of motion vector values are the same as if a GOB header were present (see 6.1.1), preventing the use of motion vectors of blocks outside the current slice for the prediction of the values of motion vectors within the slice.

2) The Advanced INTRA Coding mode (see Annex I) treats the slice boundary as if it were a picture boundary with respect to the prediction of INTRA block DCT coefficient values.

3) The assignment of remote motion vectors for use in overlapped block motion compensation within the Advanced Prediction mode also prevents the use of motion vectors of blocks outside the current slice for use as remote motion vectors (see F.3).

K.2 Structure of slice layer

The structure of the slice layer of the syntax is shown in Figure K.1 for all slices except the slice that immediately follows the picture start code in the bitstream for a picture. For the slice following the picture start code, only the emulation prevention bits (SEPB1, SEPB3, and conditionally SEPB2 as specified below), the MBA field and, when in the RS submode, also the SWI field are included.

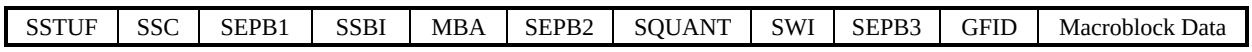


Figure K.1/H.263 – Structure of slice layer

Refer to 5.2.5 for GFID, and to 5.3 for description of Macroblock layer.

K.2.1 Stuffing (SSTUF) (Variable length)

A codeword of variable length consisting of less than 8 bits. Encoders shall insert this codeword directly before an SSC codeword whenever needed to ensure that SSC is byte aligned. If SSTUF is present, the last bit of SSTUF shall be the last (least significant) bit of a byte, so that the start of the SSC codeword is byte aligned. Decoders shall be designed to discard SSTUF. Notice that 0 is used for stuffing within SSTUF.

K.2.2 Slice Start Code (SSC) (17 bits)

A word of 17 bits. Its value is 0000 0000 0000 0000 1. Slice start codes shall be byte aligned. This can be achieved by inserting SSTUF before the start code such that the first bit of the start code is the first (most significant) bit of a byte. The slice start code is not present for the slice which follows the picture start code.

K.2.3 Slice Emulation Prevention Bit 1 (SEPB1) (1 bit)

A single bit always having the value of "1", which is included to prevent start code emulation.

K.2.4 Slice Sub-Bitstream Indicator (SSBI) (4 bits)

A codeword of length four bits which is present only when CPM = "1" in the picture header. SSBI indicates the Sub-Bitstream number for the slice for Continuous Presence Multipoint and Video Multiplex operation, as described in Annex C. The mapping from the value of SSBI to the Sub-Bitstream number is shown in Table K.1. SSBI is not present for the slice which follows the picture start code.

Table K.1/H.263 – Values of SSBI and associated Sub-Bitstream numbers

Sub-Bitstream number	SSBI field value	GN value emulated
0	1001	25
1	1010	26
2	1011	27
3	1101	29

K.2.5 Macroblock Address (MBA) (5/6/7/9/11/12/13/14 bits)

A codeword having a length that is dependent on the current picture size and whether the Reduced-Resolution Update mode is in effect (see Annex Q). The bits are the binary representation of the macroblock number of the first macroblock in the current slice as counted from the beginning of the picture in scanning order, starting with macroblock number 0 at the upper left corner. MBA uniquely identifies which macroblock in the picture the current slice starts with. Codeword lengths for this codeword are provided in Table K.2. For custom picture sizes, the field width is given by the first entry in the table that has an equal or larger number of macroblocks, and the maximum value is the number of macroblocks in the current picture minus one. When in the Reduced-Resolution Update mode, the relevant picture size is the lower resolution update picture size rather than the picture size indicated in the picture header (see Annex Q).

Table K.2/H.263 – Specification of MBA parameter

	Default		RRU mode	
Picture format	**Max value**	**Field width**	**Max value**	**Field width**
sub-QCIF	47	6	11	5
QCIF	98	7	29	6
CIF	395	9	98	7
4CIF	1583	11	395	9
16CIF	6335	13	1583	11
2048×1152	9215	14	2303	12

K.2.6 Slice Emulation Prevention Bit 2 (SEPB2) (1 bit)

A single bit always having the value "1", which is included under certain conditions to prevent start code emulation. For slices other than the one following the picture start code, SEPB2 is included only if the MBA field width is greater than 11 bits and CPM = "0" in the picture header, or if the MBA field width is greater than 9 bits and CPM = "1" in the picture header. For the slice following the picture start code, SEPB2 is included only if the Rectangular Slice submode is in use.

K.2.7 Quantizer Information (SQUANT) (5 bits)

A fixed length codeword of five bits which indicates the quantizer QUANT to be used for that slice until updated by any subsequent DQUANT. The codewords are the natural binary representations of the values of QUANT which, being half the step sizes, range from 1 to 31. SQUANT is not present for the slice which follows the picture start code.

K.2.8 Slice Width Indication in Macroblocks (SWI) (3/4/5/6/7 bits)

A codeword which is present only if the Rectangular Slice submode is active, and having a length which depends on the current picture size and whether the Reduced-Resolution Update mode is active, as specified in Table K.3. For custom picture sizes, the field width is given by the next standard format size which is equal or larger in width (QCIF, CIF, …), and the maximum value is the total number of macroblocks across the picture minus one. The last row of the table indicates the field width for picture sizes wider than 16CIF. SWI refers to the width of the current rectangular slice having its first macroblock (upper left) specified by MBA. The calculation of the actual slice width is given by

$$\text{Actual Slice Width} = \text{SWI} + 1$$

When in the Reduced-Resolution Update mode, the relevant picture size is that of the lower resolution picture size for the update information, rather than the picture size indicated in the picture header.

Table K.3/H.263 – Specification of SWI parameter

	Default		RRU mode	
Picture format	**Max value**	**Field width**	**Max value**	**Field width**
sub-QCIF	7	4	3	3
QCIF	10	4	5	3
CIF	21	5	10	4
4CIF	43	6	21	5
16CIF	87	7	43	6
1412...2048 pixels wide	127	7	63	6

K.2.9 Slice Emulation Prevention Bit 3 (SEPB3) (1 bit)

A single bit always having the value of "1" in order to prevent start code emulation.

ANNEX L

Supplemental Enhancement Information Specification

L.1 Introduction

This annex describes the format of the supplemental enhancement information sent in the PSUPP field of the picture layer of this Recommendation. The capability of a decoder to provide any or all of the enhanced capabilities described in this annex may be signalled by external means (for example Recommendation H.245). Decoders which do not provide the enhanced capabilities may simply discard any PSUPP information bits that appear in the bitstream. The presence of this supplemental enhancement information is indicated in PEI, and an additional PEI bit is inserted between every octet of PSUPP data, as described in 5.1.24 and 5.1.25.

In this annex, a distinction is made between the "decoded picture" and the "displayed picture". For purposes of this annex, the "displayed picture" is a picture having the same picture format as specified for the current picture by the picture layer of the video bitstream syntax. The "displayed picture" is constructed as described in this annex from the decoded picture, the prior displayed picture, the supplementary enhancement information described herein, and in some cases partly from a background picture which is externally controlled.

L.2 PSUPP format

The PSUPP data consists of a four-bit function type indication FTYPE, followed by a four-bit parameter data size specification DSIZE, followed by DSIZE octets of function parameter data, optionally followed by another function type indication, and so on. One function type indication value is defined as an escape code to provide for future extensibility to allow definition of more than fifteen different functions. A decoder which receives a function type indication which it does not support can discard the function parameter data for that function and then check for a subsequent

function type indication which may be supported. The defined FTYPE values are shown in Table L.1.

Table L.1/H.263 – FTYPE function type values

0	Reserved
1	Do Nothing
2	Full-Picture Freeze Request
3	Partial-Picture Freeze Request
4	Resizing Partial-Picture Freeze Request
5	Partial-Picture Freeze-Release Request
6	Full-Picture Snapshot Tag
7	Partial-Picture Snapshot Tag
8	Video Time Segment Start Tag
9	Video Time Segment End Tag
10	Progressive Refinement Segment Start Tag
11	Progressive Refinement Segment End Tag
12	Chroma Key Information
13	Reserved
14	Reserved
15	Extended Function Type

L.3 Do Nothing

No action is requested by the Do Nothing function. This function is used to prevent start code emulation. Whenever the last five or more bits of the final octet of the previous PSUPP octet are all zero and no additional PSUPP function requests are to be sent, the Do Nothing function shall be inserted into PSUPP to prevent the possibility of start code emulation. The Do Nothing function may also be sent when it is not required by rule expressed in the previous sentence. DSIZE shall be zero for the Do Nothing function.

L.4 Full-Picture Freeze Request

The full-picture freeze request function indicates that the contents of the entire prior displayed video picture shall be kept unchanged, without updating the displayed picture using the contents of the current decoded picture. The displayed picture shall then remain unchanged until the freeze picture release bit in the current PTYPE or in a subsequent PTYPE is set to 1, or until timeout occurs, whichever comes first. The request shall lapse due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of another full-picture freeze request prior to or upon expiration of the timeout period (e.g. repeating the request in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be zero for the full-picture freeze request function.

L.5 Partial-Picture Freeze Request

The partial-picture freeze request function indicates that the contents of a specified rectangular area of the prior displayed video picture should be kept unchanged, without updating the specified area of the displayed picture using the contents of the current decoded picture. The specified area of the displayed picture shall then remain unchanged until the freeze picture release bit in the current

PTYPE or in a subsequent PTYPE is set to 1, until a partial-picture freeze-release request affecting the specified area is received, until the source format specified in a picture header differs from that of previous picture headers, or until timeout occurs, whichever comes first. Any change in the picture source format shall act as a freeze release for all active partial-picture freeze requests. The request shall lapse due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical partial-picture freeze request prior to or upon expiration of the timeout period (e.g. repeating the request in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 4 for the partial-picture freeze request. The four octets of PSUPP that follow contain the horizontal and vertical location of the upper left corner of the frozen picture rectangle, and the width and height of the rectangle, respectively, using eight bits each and expressed in units of eight pixels. For example, a 24-pixel wide and 16 pixel tall area in the upper left corner of the video display is specified by the four parameters (0, 0, 3, 2).

L.6 Resizing Partial-Picture Freeze Request

The resizing partial-picture freeze request function indicates that the contents of a specified rectangular area of the prior displayed video picture should be resized to fit into a smaller part of the displayed video picture, which should then be kept unchanged, without updating the specified area of the displayed picture using the contents of the current decoded picture. The specified area of the displayed picture shall then remain unchanged until the freeze release bit in the current PTYPE or in a subsequent PTYPE is set to 1, until a partial-picture freeze-release request affecting the specified area is received, until the source format specified in a picture header differs from that of previous picture headers, or until timeout occurs, whichever comes first. Any change in the picture source format shall act as a freeze release for all active resizing partial-picture freeze requests. The request shall lapse due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of a partial-picture freeze request for the affected area of the displayed picture prior to or upon expiration of the timeout period (e.g. issuing a partial-picture freeze request in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 8 for the resizing partial-picture freeze request. The eight octets of PSUPP data that follow contain 32 bits used to specify the rectangular region of the affected area of the displayed picture, and then 32 bits used to specify the corresponding rectangular region of the affected area of the decoded picture. The width and height of the rectangular region in the decoded picture shall both be equal to 2^i times the width and height specified for the rectangular region in the displayed picture, where *i* is an integer in the range of 1 to 8. The location and size of each of these two rectangular regions is specified using the same format as such a region is specified in the partial-picture freeze request function.

L.7 Partial-Picture Freeze-Release Request

The partial-picture freeze-release request function indicates that the contents of a specified rectangular area of the displayed video picture shall be updated by the current and subsequent decoded pictures. DSIZE shall be equal to 4 for the partial-picture freeze-release request. The four octets of PSUPP data that follow specify a rectangular region of the displayed picture in the same format as such a region is specified in the partial-picture freeze request function.

L.8 Full-Picture Snapshot Tag

The full-picture snapshot tag function indicates that the current picture is labelled for external use as a still-image snapshot of the video content. DSIZE shall be equal to 4 for the full-picture snapshot

tag function. The four octets of PSUPP data that follow specify a snapshot identification number for external use.

L.9 Partial-Picture Snapshot Tag

The partial-picture snapshot tag function indicates that a specified rectangular area of the current picture is labelled for external use as a still-image snapshot of the video content. DSIZE shall be equal to 8 for the partial-picture snapshot tag function. The first four octets of PSUPP data that follow specify a snapshot identification number for external use, and the remaining four octets of PSUPP data that follow specify a rectangular region of the decoded picture in the same format as such a region is specified in the partial-picture freeze request function.

L.10 Video Time Segment Start Tag

The video time segment start tag function indicates that the beginning of a specified sub-sequence of video data is labelled as a useful section of video content for external use, starting with the current picture. The tagged sub-sequence of video data shall continue until stopped by the receipt of a matching video time segment end tag function or until timeout, whichever comes first. The tagged sub-sequence shall end due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical video time segment start tag function prior to or upon expiration of the timeout period (e.g. repeating the video time segment start tag function in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a video time segment identification number for external use.

L.11 Video Time Segment End Tag

The video time segment end tag function indicates that the end of a specified sub-sequence of video data is labelled as a useful section of video content for external use, ending with the previous picture. DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a video time segment identification number for external use.

L.12 Progressive Refinement Segment Start Tag

The progressive refinement segment start tag function indicates the beginning of a specified sub-sequence of video data which is labelled as the current picture followed by a sequence of zero or more pictures of refinement of the quality of the current picture, rather than as a representation of a continually moving scene. The tagged sub-sequence of video data shall continue until stopped by the receipt of a matching progressive refinement segment end tag function or until timeout, whichever comes first. The tagged sub-sequence shall end due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical progressive refinement segment start tag function prior to or upon expiration of the timeout period (e.g. repeating the progressive refinement start tag function in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a progressive refinement segment identification number for external use.

L.13 Progressive Refinement Segment End Tag

The progressive refinement segment end tag function indicates the end of a specified sub-sequence of video data which is labelled as an initial picture followed by a sequence of zero or more pictures

of the refinement of the quality of the initial picture, and ending with the previous picture. DSIZE shall be equal to 4 for the video time segment start tag function. The four octets of PSUPP data that follow specify a progressive refinement identification number for external use.

L.14 Chroma Keying Information

The Chroma Keying Information Function (CKIF) indicates that the "chroma keying" technique is used to represent "transparent" and "semi-transparent" pixels in the decoded video pictures. When being presented on the display, "transparent" pixels are not displayed. Instead, a background picture which is either a prior reference picture or is an externally controlled picture is revealed. Semi-transparent pixels are displayed by blending the pixel value in the current picture with the corresponding value in the background picture. One octet is used to indicate the keying color value for each component (Y, C_B, or C_R) which is used for chroma keying. To represent pixels that are to be "semi-transparent", two threshold values, denoted as T_1 and T_2, are used. Let α denote the transparency of a pixel; $\alpha = 255$ indicates that the pixel is opaque, and $\alpha = 0$ indicates that the pixel is transparent. For other values of α, the resulting value for a pixel should be a weighted combination of the pixel value in the current picture and the pixel value from the background picture (which is specified externally). The values of α may be used to form an image that is called an "alpha map." Thus, the resulting value for each component may be:

$$[\alpha \cdot X + (255 - \alpha) \cdot Z] / 255$$

where X is the decoded pixel component value (for Y, C_B, or C_R), and Z is the corresponding pixel component value from the background picture.

The α value can be calculated as follows. First, the distance of the pixel color from the key color value is calculated:

$$d = A_Y (X_Y - K_Y)^2 + A_B (X_B - K_B)^2 + A_R - K_R)^2$$

in which X_Y, X_B, and X_R are the Y, C_B, and C_R values of the decoded pixel color, K_Y, K_B, and K_R are the corresponding key color parameters, and A_Y, A_B, and A_R. are keying flag bits which indicate which color components are used as keys. Once the distance d is calculated, the α value may be computed as specified in the following pseudo-code:

```
for each pixel
   if      (d<T1) then α = 0;
   else if (d>T2) then α = 255;
   else α = [255·(d-T1)]/(T2-T1)
```

However, the precise method for performing the chroma keying operation in the decoder is not specified herein, since normative specification of the method is not needed for interoperability. The process described here is provided for illustration purposes in order to convey the intended interpretation of the data parameters.

Since the derived α value is simply a function of X_Y, X_B, and X_R, a Look-Up Table (LUT) can be built to achieve the above operation. Such an LUT has $2^{8\times N}$ entries corresponding to all pixel values, where N is the number of color components used as keys. Each entry in the LUT would then contain the corresponding α value.

DSIZE shall be in the range of 1 to 9 (inclusive) for chroma keying information, according to the amount of data sent with the CKIF. No more than one CKIF shall be sent with a picture.

The first octet following the DSIZE octet shall contain the representation order of the current picture – streams having a lower representation order are assumed to form the background picture for streams having a higher representation order.

If DSIZE is greater than one, the next octet after the representation order octet shall be used to send six flag bits defined as:

bit 1: A_Y: A flag bit indicating the presence of a K_Y key parameter for luminance *Y* values

bit 2: A_B: A flag bit indicating the presence of a K_B key parameter for chroma C_B values

bit 3: A_R: A flag bit indicating the presence of a K_R key parameter for chroma C_R values

bit 4: A_1: A flag bit indicating the presence of a T_1 threshold parameter for transparency

bit 5: A_2: A flag bit indicating the presence of a T_2 threshold parameter for opacity

bit 6: RPB: A flag bit indicating the use of the reference picture as a background picture

bit 7: reserved

bit 8: reserved

DSIZE shall be equal to 1 or shall be equal to 2 plus the number of flag bits among A_Y, A_B, and A_R which are set to 1, plus 2 times the number of bits among A_1 and A_2 which are set to 1. If DSIZE is greater than 1, then an additional octet shall be sent to specify the value of each color component for each of the flag bits A_Y, A_B, and A_R which are set to one and two additional octets shall be sent to specify each of the flagged threshold values among T_1 and T_2. These octets shall follow in the same order as the flag bits.

If DSIZE is equal to 1, or if all three keying color flag bits A_Y, A_B, and A_R, are zero, the keying flag bits A_Y, A_B, and A_R and keying colours K_Y, K_B, and K_R which were used for the previous keyed picture should also be used for the current picture. If no previous values have been sent for the video sequence, the default keying flag bits A_Y = 1, A_B = 1, and A_R = 1 and the default key colours K_Y = 50, K_B = 220, and K_R = 100 should be used as the previous values.

If DSIZE is equal to 1, or if both of the keying threshold flag bits A_1 and A_2 are zero, the keying threshold values T_1 and T_2 which were used for the previous keyed picture should also be used for the current picture. If no previous values have been sent for the video sequence, the default threshold values T_1 = 48 and T_2 = 75 should be used as the previous values.

In the portion where the pixels are "semi-transparent" (i.e. when $T_1 < d < T_2$), the decoded pixels typically contain the chroma key color in components where chroma keys are used. This may result in certain color artifacts. To address this problem, these pixel values may be adjusted before they are blended with the background color. Such a correction process may be applied to color components being used in the chroma key operation as indicated by the flag bits. The process is as follows:

$$X' = K + (T_2 / d)(X - K)$$

where *X* is the original decoded pixel component value, and *X'* is the corrected value.

Since the adjusted pixel values X'_Y, X'_B, and X'_R are functions of X_Y, X_B, and X_R, the color correction can be achieved by using an LUT. This LUT would have 2^{8N} entries corresponding to all pixel values, where N is the number of color components used as keys. Each entry would then contain the corresponding corrected values.

If the Reference Picture Background (RPB) flag bit is set to "1", this indicates that the temporally previous reference picture (prior to any Annex P resampling performed for the current picture) should be held as the (opaque) background of the current picture and of all subsequent chroma keyed pictures, until replaced by the arrival of another picture having an RPB flag set to "1". If the current picture has no temporally previous reference picture (i.e if the current picture is a picture of type

INTRA or EI), the picture to which the RPB flag bit refers is the picture which would normally have been the reference picture if the current picture were of type INTER or EP as appropriate. If the RPB flag bit is set to "0", this indicates that background should remain as previously controlled (either under external control or using a reference picture which was previously stored upon receipt of a prior picture having RPB set to "1").

The use of the chroma keying which is invoked by the issuance of the chroma keying information function shall start with the current picture and shall continue until a subsequent picture of type INTRA or EI occurs or until a timeout period expires, whichever comes first. The use of chroma keying shall end due to timeout after five seconds or five pictures, whichever is a longer period of time. The timeout can be prevented by the issuance of an identical chroma keying information function prior to or upon expiration of the timeout period (e.g. repeating the chroma keying information function in the header of the first picture with temporal reference indicating a time interval greater than or equal to five seconds since issuance, or in the header of the fifth picture after issuance). The encoder shall send sufficient information with the chroma keying information function for complete resynchronization to occur with each picture of type INTRA or EI and within each timeout interval (it shall not rely on using stored or default values of the key colours or thresholds).

L.15 Extended Function Type

The extended function type indication is used to signal that the following PSUPP octet contains an extended function. The usage of extended functions is reserved for the ITU to have a later ability to define a larger number of backward-compatible PSUPP data functions. DSIZE shall be equal to zero for the extended function type indication. In order to allow backward compatibility of future use of the extended function type indication, decoders shall treat the second set of four bits in the octet which follows the extended function type indication as a DSIZE value indicating the number of subsequent octets of PSUPP that are to be skipped for extended function parameter data, which may be followed by additional FTYPE indications.

ANNEX M

Improved PB-frames mode

M.1 Introduction

This annex describes an optional Improved PB-frames mode of this Recommendation. It is therefore considered to be advantageous to use the present Improved PB-frames mode instead of the PB-frames mode defined in Annex G. The capability of this mode is signalled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

Most parts of this option are similar to the PB-frame option defined in Annex G.

To avoid confusion with B-pictures as defined in Annex O, the terms B-picture, B-macroblock, and B-block will not be used in this annex. Instead, we will use the notation B_{PB} to mean the "B-part" of an Improved PB-frame. When reference is made to Annex G, B-picture and B-block shall be read as B_{PB}-picture and B_{PB}-block.

The main difference between the PB-frames mode and the Improved PB-frames mode is that in the Improved PB-frames mode, the B_{PB}-macroblock has available a forward and a backward prediction mode in addition to the bidirectional prediction mode. In this annex, MVDB (when present) refers to a forward motion vector. (Note that, in Annex G, MVDB was used to refer to an enhancement of the

downscaled forward and backward vectors for bidirectional prediction, rather than a distinct forward motion vector.)

All the differences from Annex G are identified in this annex. When nothing is indicated, it means that the same procedure as described in Annex G is used.

M.2 B_{PB}-macroblock prediction modes

There are three different ways of coding a B_{PB}-macroblock. The different coding modes are signalled by the parameter MODB. The B_{PB}-macroblock coding modes are:

M.2.1 Bidirectional prediction

In the bidirectional prediction mode, prediction uses the reference pictures before and after the B_{PB}-picture (in the case of a sequence of Improved PB-frames, this means the P-picture part of the temporally previous Improved PB-frame and the P-picture part of the current Improved PB-frame). This prediction is equivalent to the prediction defined in Annex G when $MV_D = 0$. Notice that in this mode (and only in this mode), Motion Vector Data (MVD) of the PB-macroblock must be included even if the P-macroblock is INTRA coded. (Notice the difference between MVD – motion vector data – and MV_D – delta vector – defined in Annex G.)

M.2.2 Forward prediction

In the forward prediction mode, the vector data contained in MVDB are used for forward prediction from the previous reference picture (an INTRA or INTER picture, or the P-picture part of a PB or Improved PB-frame). This means that there is always only one 16×16 vector for the B_{PB}-macroblock in this prediction mode.

A simple predictor is used for coding of the forward motion vector. The rule for this predictor is that if the current macroblock is not at the far left edge of the picture or slice and the macroblock to the left has a forward motion vector, then the predictor of the forward motion vector for the current macroblock is set to the value of the forward motion vector of the block to the left; otherwise, the predictor is set to zero. The difference between the predictor and the desired motion vector is then VLC coded in the same way as vector data to be used for the P-picture (MVD).

Concerning motion vectors over picture boundaries defined in D.1, the described technique also applies for the forward B_{PB}-vector if this feature is in use (this applies for forward as well as for bidirectional prediction mode).

M.2.3 Backward prediction

In the backward prediction mode, the prediction of the B_{PB} macroblock is identical to P_{REC} (defined in G.5). No motion vector data is used for the backward prediction.

M.3 Calculation of vectors for bidirectional prediction of a the B-macroblock

In case bidirectional prediction is used, the scaled forward and backward vectors are calculated as described in Annex G when $MV_D = 0$.

M.4 MODB table

A new definition for MODB (replacing Table 11) is shown in Table M.1. It indicates the possible coding modes for a B_{PB}-block.

Table M.1/H.263 – MODB table for Improved PB-frames mode

Index	CBPB	MVDB	Number of bits	Code	Coding mode
0			1	0	Bidirectional prediction
1	x		2	10	Bidirectional prediction
2		x	3	110	Forward prediction
3	x	x	4	1110	Forward prediction
4			5	11110	Backward prediction
5	x		5	11111	Backward prediction
NOTE – The symbol "x" indicates that the associated syntax element is present.					

ANNEX N

Reference Picture Selection mode

N.1 Introduction

This annex describes the optional Reference Picture Selection mode of this Recommendation, which operates using a modified interframe prediction method called "NEWPRED." The capability of this H.263 mode is signalled by external means (for example, Recommendation H.245). The amount of additional picture memory accommodated in the decoder may also be signalled by external means to help the memory management at the encoder. This mode can use backward channel messages sent from a decoder to an encoder to inform the encoder which part of which pictures have been correctly decoded at the decoder. The use of this mode is indicated in the PLUSPTYPE field of the picture header. This mode has two back-channel mode switches which define whether a backward channel is used and what kind of messages are returned on that backward channel from the decoder, and has another submode defined in terms of the channel for the backward channel messages.

The two back-channel mode switches of this mode determine the type of messages sent on the back-channel, specifying whether ACK (Acknowledgment messages), or NACK (Non-Acknowledgment messages) are sent. Together, the two switches define four basic methods of operation:

1) NEITHER, in which no back-channel data is returned from the decoder to the encoder;
2) ACK, in which the decoder returns only acknowledgment messages;
3) NACK, in which the decoder returns only non-acknowledgment messages; and
4) ACK+NACK, in which the decoder returns both acknowledgment and non-acknowledgment messages.

The specific type of messages to be sent as outlined above is indicated in the picture header.

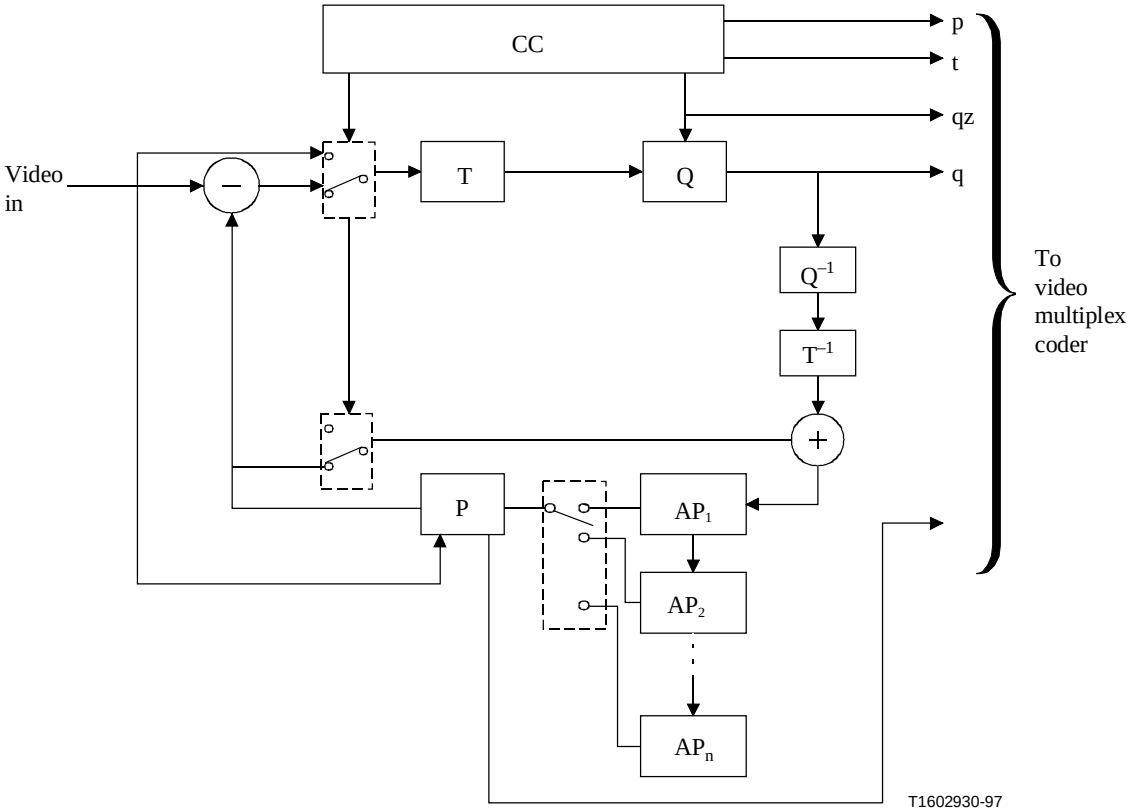
There are also two methods of operation in terms of the channel for backward channel messages:

1) Separate Logical Channel mode: This method of operation delivers back-channel data through a separate logical channel in the multiplex layer of the system, and
2) VideoMux mode: This method of operation delivers back-channel data for received video within the forward video data of a video stream of encoded data.

This annex specifies a syntax for the backward channel messages as well as for forward-channel data.

N.2 Video source coding algorithm

The source coder of this mode is shown in generalized form in Figure N.1. This figure shows a structure which uses a number of picture memories. The source coder may select one of the picture memories to suppress the temporal error propagation due to the inter-frame coding. The Independent Segment Decoding mode (see Annex R), which treats boundaries of GOBs with non-empty headers or slices as picture boundaries, can be used to avoid error propagation due to motion compensation across the boundaries of the GOBs or slices when this mode is applied to a smaller unit than a picture, such as a GOB or slice. The information to signal which picture is selected for prediction is included in the encoded bitstream. The strategy used by the encoder to select the picture to be used for prediction is out of the scope of this Recommendation.



T	Transform
Q	Quantizer
P	Picture Memory with motion compensated variable delay
AP	Additional Picture Memory
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 N.1/H.263 – Source coder for NEWPRED

N.3 Channel for back-channel messages

This mode has two methods of operation in terms of the type of channel for back-channel messages. One is the separate logical channel mode and the other is the videomux mode. The separate logical channel mode is a preferred mode and delivers the back-channel message defined in N.4.2 through the dedicated logical channel. The videomux mode is prepared for the system which cannot set up

the separate extra channel for the back-channel messages due to the restriction of the number of channels' combinations. The videomux mode delivers the back-channel messages through the same logical channel with the forward video data in the opposite direction.

N.3.1 Separate logical channel mode

The separate logical channel mode delivers back-channel messages through a dedicated logical channel opened only for the purpose of the back-channel messages. The association mechanism with the forward channel which delivers video data is provided by external means (for example, Recommendation H.245). Separate logical channel operation requires an external framing mechanism for synchronization of the messages within the back channel, as the back-channel syntax defined herein contains no synchronization flag words.

N.3.2 Videomux mode

The videomux mode delivers the back-channel messages through the same logical channel with the forward video data in the opposite direction. The syntax of the multiplexed bitstream is described in N.4.1. The back-channel messages may be inserted by using the Back-Channel message Indication (BCI) in the GOB or slice header.

N.4 Syntax

N.4.1 Forward channel

The syntax for the forward-channel data which conveys the compressed video signal is altered only in the Group of Blocks (GOB) or slice layer.

The syntax of the GOB layer is illustrated in Figure N.2. The fields of TRI, TR, TRPI, TRP, BCI, and BCM are added to Figure 9.

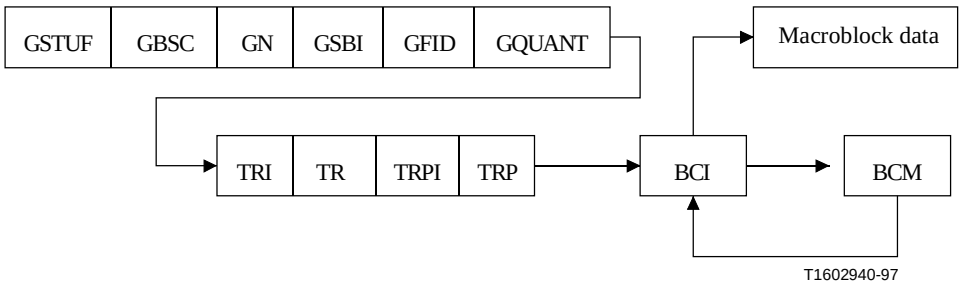


Figure N.2/H.263 – Structure of GOB layer for NEWPRED

When the optional Slice Structured mode (see Annex K) is in use, the syntax of the slice layer is modified in the same way as the GOB layer. The syntax is illustrated Figure N.3.

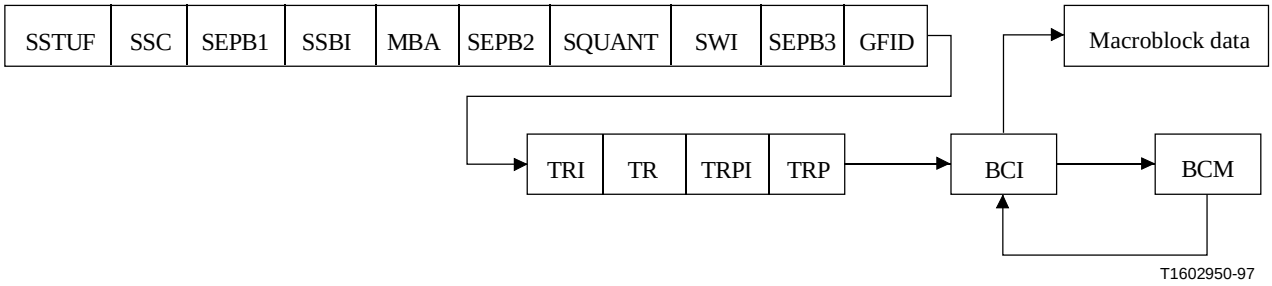


Figure N.3/H.263 – Structure of slice layer for NEWPRED

N.4.1.1 Temporal Reference Indicator (TRI) (1 bit)

TRI indicates whether or not the following TR field is present.

0: TR field is not present.

1: TR field is present.

N.4.1.2 Temporal Reference (TR) (8/10 bits)

When present, TR is an eight-bit number unless a custom picture clock frequency is in use, in which case it is a ten-bit number consisting of the concatenation of ETR and TR of the picture header.

N.4.1.3 Temporal Reference for Prediction Indicator (TRPI) (1 bit)

TRPI indicates whether or not the following TRP field is present.

0: TRP field is not present.

1: TRP field is present.

TRPI shall be equal to zero whenever the picture is an I- or EI-picture.

N.4.1.4 Temporal Reference for Prediction (TRP) (10 bits)

When present (as indicated in TRPI), TRP indicates the Temporal Reference which is used for prediction of the encoding, except in the case of B-pictures and the B-picture part of an Improved PB-frame. For B-pictures or the B-picture part of an Improved PB-frame, the picture having the temporal reference TRP is used for the prediction in the forward direction. (Prediction in the reverse-temporal direction always uses the immediately temporally subsequent picture.) TRP is a ten-bit number. If a custom picture clock frequency was not in use for the reference picture, the two MSBs of TRP are zero and the LSBs contain the eight-bit TR found in the picture header of the reference picture. If a custom picture clock frequency was in use for the reference picture, TRP is a ten-bit number consisting of the concatenation of ETR and TR from the reference picture header.

When TRP is not present, the most recent temporally previous anchor picture shall be used for prediction, as when not in the Reference Picture Selection mode. TRP is valid until the next PSC, GSC, or SSC.

N.4.1.5 Back-Channel message Indication (BCI) (Variable length)

This field contains one or two bits; when set to "1", it signals the presence of the following video Back-Channel Message (BCM) field. Otherwise, the field value is "01", which indicates the absence or the end of the video back-channel message field. Combinations of BCM and BCI may not be present, and may be repeated when present. BCI shall always be set to "01" if the videomux mode is not in use.

N.4.1.6 Back-Channel Message (BCM) (Variable length)

The Back-Channel Message having syntax as defined in N.4.2, which is present only if the preceding BCI field is set to "1".

N.4.2 Back-Channel Message (BCM) syntax

The syntax for the back-channel which conveys the acknowledgment/non-acknowledgment messages is illustrated in Figure N.4. This message is returned from a decoder to an encoder in order to tell whether a forward-channel data was correctly decoded or not.

Figure N.4/H.263 – Structure of Back-Channel Message (BCM) syntax for NEWPRED

N.4.2.1 Back-channel message Type (BT) (2 bits)

Back-channel message type indicates if the corresponding part of the encoded message is correctly decoded or not. Which type of message is required for the encoder is indicated in the picture header of the forward channel.

00: Reserved for future use.

01: Reserved for future use.

10: NACK. This indicates the erroneous decoding of the corresponding part of the forward-channel data.

11: ACK. This indicates the correct decoding of the corresponding part of the forward-channel data.

N.4.2.2 Unreliable Flag (URF) (1 bit)

The Unreliable Flag is set to 1 when a reliable value for TR or GN/MBA is not available to the decoder. (When BT is NACK, a reliable TR may not be available at the decoder.)

0: Reliable.

1: Unreliable.

N.4.2.3 Temporal Reference (TR) (10 bits)

Temporal reference contains the TR information of the video picture segment for which the ACK/NACK is indicated in the back-channel message.

NOTE – The meaning of the term "video picture segment" as used herein is defined in Annex R. If a custom picture clock frequency was not in use for the reference picture, the two MSBs of TR are zero and the LSBs contain the eight-bit TR found in the picture header of the reference picture. If a custom picture clock frequency was in use for the reference picture, TR is a ten-bit number consisting of the concatenation of ETR and TR from the reference picture header.

N.4.2.4 Enhancement Layer Number Indication (ELNUMI) (1 bit)

The enhancement layer number indication is "0" unless the optional Temporal, SNR and Spatial Scalability mode (Annex O) is used in the forward-channel data and some enhancement layers of the forward channel are combined on one logical channel and the back-channel message refers to an enhancement layer (rather than the base layer), in which case the enhancement layer number indication shall be "1".

N.4.2.5 Enhancement Layer Number (ELNUM) (4 bits)

Enhancement layer number is present if and only if ELNUMI is "1", in which case it contains the layer number of the enhancement layer referred to in the back-channel message.

N.4.2.6 BCPM (1 bit)

BCPM is "0" unless the CPM mode is used in the forward-channel data, in which case it is "1". If BCPM is "1", this indicates that BSBI is present.

N.4.2.7 Back-channel Sub-Bitstream Indicator (BSBI) (2 bits)

A fixed length codeword of 2 bits that is only present if BCPM is "1". The BSBI is the natural binary representation of the appropriate Sub-Bitstream number in the forward-channel data for which the ACK/NACK message is indicated in the back-channel message as described in 5.2.4 and in Annex C.

N.4.2.8 Back-channel Emulation Prevention Bit 1 (BEPB1) (1 bit)

A field which is present if and only if the videomux mode is in use. This field is always set to "1" to prevent a start code emulation.

N.4.2.9 GOB Number/Macroblock Address (GN/MBA) (5/6/7/9/11/12/13/14 bits)

A GOB number or macroblock address is present in this field. If the optional Slice Structured mode (see Annex K) is not in use, this field contains the GOB number of the beginning of the video picture segment for which the NACK/ACK message is indicated in the back-channel message. If the optional Slice Structured mode is in use, this field contains the macroblock address of the beginning of the slice for which the NACK/ACK message is indicated in the back-channel message. The length of this field is the length specified elsewhere in this Recommendation for GN or MBA.

N.4.2.10 Back-channel Emulation Prevention Bit 2 (BEPB2) (1 bit)

A field which is present if and only if the videomux mode is in use. This field is always set to "1" to prevent a start code emulation.

N.4.2.11 Requested Temporal Reference (RTR) (10 bits)

Requested temporal reference is present only if BT is NACK. RTR indicates the requested temporal reference of the GOB or slice associated with the NACK. Typically, it is the TR of the last correctly decoded video picture segment of the corresponding position at the decoder. If a custom picture clock frequency was not in use for the requested reference picture, the two MSBs of RTR are zero and the LSBs contain the eight-bit TR found in the picture header of the requested reference picture. If a custom picture clock frequency was in use for the requested reference picture, RTR is a ten-bit number consisting of the concatenation of ETR and TR from the requested reference picture header.

N.4.2.12 Stuffing (BSTUF) (Variable length)

This field is present if and only if the separate logical channel mode is in use and the back-channel message is the last in an external frame. BSTUF consists of a codeword of variable length consisting of zero or more bits of value "0". This field is only present at the end of an external frame.

N.5 Decoder process

The decoder of this mode may need an additional number of picture memories to store the correctly decoded video signals and their Temporal Reference (TR) information. The decoder uses the stored picture for which the TR is TRP as the reference picture for inter-frame decoding instead of the last decoded picture, if the TRP field exists in the forward-channel data. When the picture for which the

TR is TRP is not available at the decoder, the decoder may send a forced INTRA update signal to the encoder by external means (for example, Recommendation H.245). Unless a different frame storage policy is negotiated by external means, correctly decoded video picture segments shall be stored into memory for use as later reference pictures on a first-in, first-out basis as shown in Figure N.1 (except for B-pictures, which are not used as reference pictures), and video picture segments which are detected as having been incorrectly decoded should not replace correctly decoded ones in this memory area.

An Acknowledgment message (ACK) and a Non-Acknowledgment message (NACK) are defined as back-channel messages. An ACK may be returned when the decoder decodes a video picture segment successfully. NACKs may be returned when the decoder fails to decode a video picture segment, and may continue to be returned until the decoder gets the expected forward channel data which includes the requested TRP or an INTRA update. Which types of message shall be sent is indicated in the RPSMF field of the picture header of the forward-channel data.

In a usage scenario known as "Video Redundancy Coding", the Reference Picture Selection mode may be used by some encoders in a manner in which more than one representation is sent for the pictured scene at the same temporal instant (usually using different reference pictures). In such a case in which the Reference Picture Selection mode is in use and in which adjacent pictures in the bitstream have the same temporal reference, the decoder shall regard this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and shall decode and use the first such received picture while discarding the subsequent redundant picture(s).

ANNEX O

Temporal, SNR, and Spatial Scalability mode

This annex describes the optional mode of this Recommendation in support of Temporal, SNR, and Spatial Scalability. This mode may also be used in conjunction with error control schemes. The capability of this mode and the extent to which its features are supported is signaled by external means (for example, Recommendation H.245). The use of this mode is indicated in PLUSPTYPE.

O.1 Overview

Scalability allows for the decoding of a sequence at more than one quality level. This is done by using a hierarchy of pictures and enhancement pictures partitioned into one or more layers. There are three types of pictures used for scalability: B-, EI-, and EP-pictures, as explained below. Each of these has an enhancement layer number ELNUM that indicates to which layer it belongs, and a reference layer number RLNUM that indicates which layer is used for its prediction. The lowest layer is called the base layer, and has layer number 1.

Scalability is achieved by three basic methods: temporal, SNR, and spatial enhancement.

O.1.1 Temporal scalability

Temporal scalability is achieved using bidirectionally predicted pictures, or B-pictures. B-pictures allow prediction from either or both a previous and subsequent reconstructed picture in the reference layer. This property generally results in improved compression efficiency as compared to that of P-pictures. These B-pictures differ from the B-picture part of a PB- (or Improved PB-) frame (see Annexes G and M) in that they are separate entities in the bitstream: they are not syntactically intermixed with a subsequent P- (or EP-) picture.

B-pictures (and the B-part of PB- or Improved PB-frames) are not used as reference pictures for the prediction of any other pictures. This property allows for B-pictures to be discarded if necessary without adversely affecting any subsequent pictures, thus providing temporal scalability. Figure O.1 illustrates the predictive structure of P- and B-pictures.

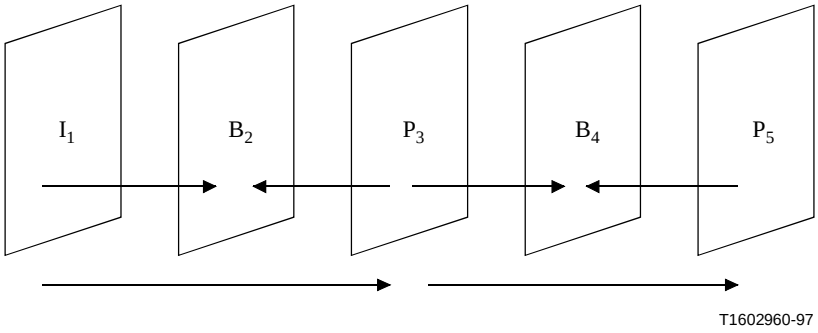


Figure O.1/H.263 – Illustration of B-picture prediction dependencies

The location of B-pictures in the bitstream is in a data-dependence order rather than in strict temporal order. (This rule is consistent with the ordering of other pictures in the bitstream, but for all picture types other than the B-picture, no such conflict arises between the data-dependence order and the temporal order.) For example, if the pictures of a video sequence were numbered 1, 2, 3, …, then the bitstream order of the encoded pictures would be I_1, P_3, B_2, P_5, B_4, …, where the subscript refers to the original picture number (as illustrated in Figure O.1).

There is no limit to the number of B-pictures that may be inserted between pairs of reference pictures in the reference layer (other than what is necessary to prevent temporal ambiguity from overflows of the temporal reference field in the picture header). However, a maximum number of such pictures may be signaled by external means (for example, Recommendation H.245).

The picture height, width, and pixel aspect ratio of a B-picture shall always be equal to those of its temporally subsequent reference layer picture.

Motion vectors are allowed to extend beyond the picture boundaries of B-pictures.

O.1.2 SNR scalability

The other basic method to achieve scalability is through spatial/SNR enhancement. Spatial scalability and SNR scalability are equivalent except for the use of interpolation as is described shortly. Because compression introduces artifacts and distortions, the difference between a reconstructed picture and its original in the encoder is (nearly always) a nonzero-valued picture, containing what can be called the coding error. Normally, this coding error is lost at the encoder and never recovered. With SNR scalability, these coding error pictures can also be encoded and sent to the decoder, producing an enhancement to the decoded picture. The extra data serves to increase the signal-to-noise ratio of the video picture, and hence, the term SNR scalability. Figure O.2 illustrates the data flow for SNR scalability. The vertical arrows from the lower layer illustrate that the picture in the enhancement layer is predicted from a reconstructed approximation of that picture in the reference (lower) layer.

If prediction is only formed from the lower layer, then the enhancement layer picture is referred to as an EI-picture. It is possible, however, to create a modified bidirectionally predicted picture using both a prior enhancement layer picture and a temporally simultaneous lower layer reference picture. This type of picture is referred to as an EP-picture or "Enhancement" P-picture. The prediction flow for EI- and EP-pictures is shown in Figure O.2. (Although not specifically shown in Figure O.2, an

EI-picture in an enhancement layer may have a P-picture as its lower layer reference picture, and an EP-picture may have an I-picture as its lower-layer enhancement picture.)

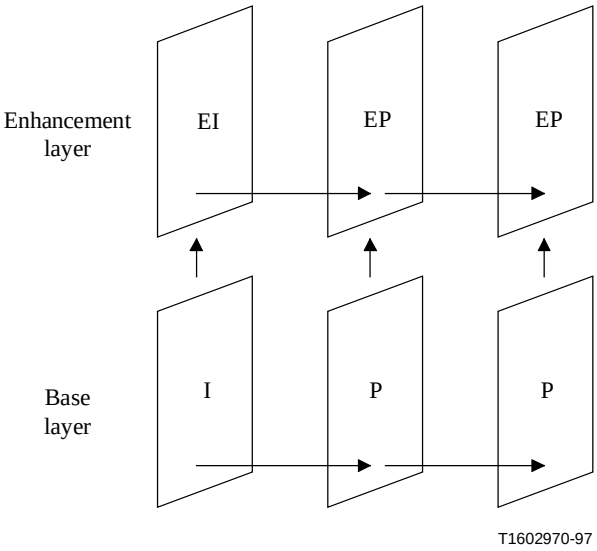


Figure O.2/H.263 – Illustration of SNR scalability

For both EI- and EP-pictures, the prediction from the reference layer uses no motion vectors. However, as with normal P-pictures, EP-pictures use motion vectors when predicting from their temporally prior reference picture in the same layer.

O.1.3 Spatial scalability

The third and final scalability method in the Temporal, SNR, and Spatial Scalability mode is spatial scalability, which is closely related to SNR scalability. The only difference is that before the picture in the reference layer is used to predict the picture in the spatial enhancement layer, it is interpolated by a factor of two either horizontally or vertically (1-D spatial scalability), or both horizontally and vertically (2-D spatial scalability). The interpolation filters for this operation are defined in O.6. For a decoder to be capable of some forms of spatial scalability, it may also need to be capable of custom picture formats. For example, if the base layer is sub-QCIF (128 × 96), the 2-D spatial enhancement layer picture would be 256 × 192, which does not correspond to a standard picture format. Another example would be if the base layer were QCIF (176 × 144), with the standard pixel aspect ratio of 12:11. A 1-D horizontal spatial enhancement layer would then correspond to a picture format of 352 × 144 with a pixel aspect ratio of 6:11. Thus a custom picture format would have to be used for the enhancement layer in these cases. An example which does not require a custom picture format would be the use of a QCIF base layer with a CIF 2-D spatial enhancement layer. Spatial scalability is illustrated in Figure O.3.

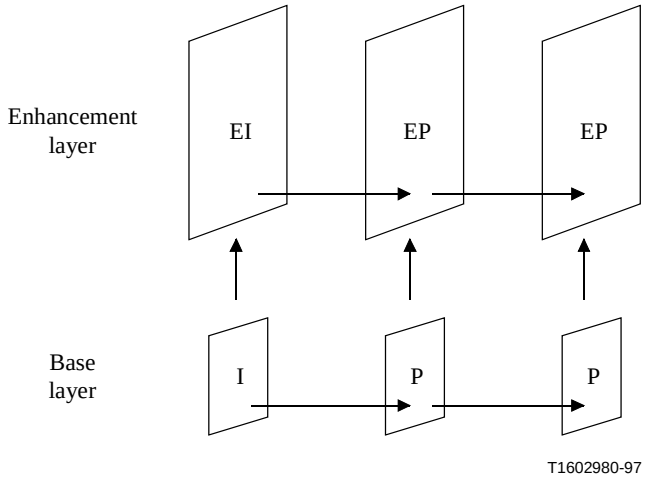


Figure O.3/H.263 – Illustration of spatial scalability

Other than requiring an upsampling process to increase the size of the reference layer picture prior to its use as a reference for the encoding process, the processing and syntax for a spatial scalability picture is functionally identical to that for an SNR scalability picture.

Since there is very little syntactical distinction between pictures using SNR scalability and pictures using spatial scalability, the pictures used for either purpose are called EI- and EP-pictures.

The picture in the base layer which is used for upward prediction in an EI- or EP-picture may be an I-picture, a P-picture, or the P-part of a PB- or Improved PB-frame (but shall not be a B-picture or the B-part of a PB- or Improved PB-frame).

O.1.4 Multilayer scalability

It is possible not only for B-pictures to be temporally inserted between pictures of types I, P, PB, and Improved PB, but also between pictures of types EI and EP, whether these consist of SNR or spatial enhancement pictures. It is also possible to have more than one SNR or spatial enhancement layer in conjunction with a base layer. Thus, a multilayer scalable bitstream can be a combination of SNR layers, spatial layers, and B-pictures. The size of a picture cannot decrease, however, with increasing layer number. It can only stay the same or increase by a factor of two in one or both dimensions. Figure O.4 illustrates a multilayer scalable bitstream.

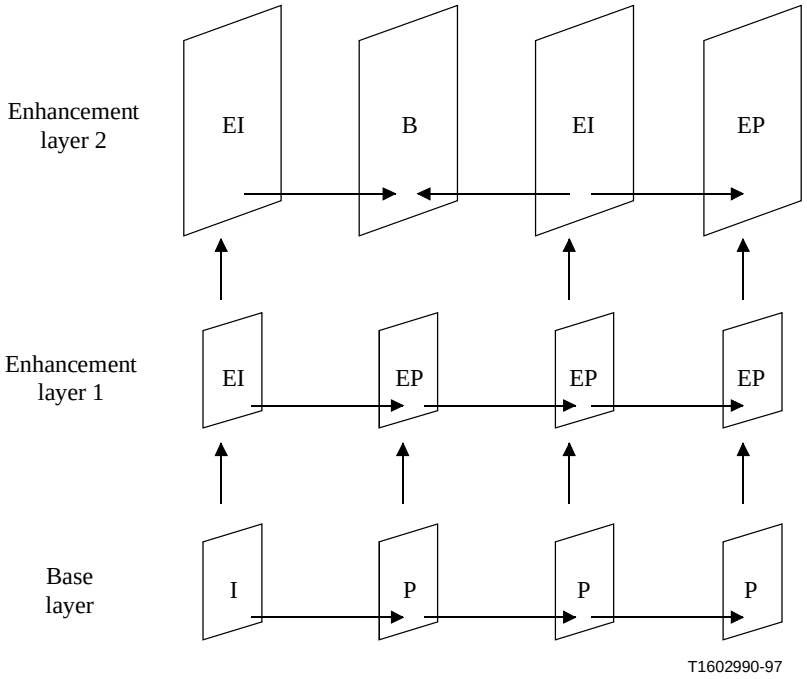


Figure O.4/H.263 – Illustration of multilayer scalability

In the case of multilayer scalability, the picture in a reference layer which is used for upward prediction in an EI- or EP-picture may be a I-, P-, EI-, or EP-picture, or may be the P-part of a PB- or Improved PB-frame in the base layer (but shall not be a B-picture or the B-part of a PB- or Improved PB-frame).

As with the two-layer case, B-pictures may occur in any layer. However, any picture in an enhancement layer which is temporally simultaneous with a B-picture in its reference layer must be a B-picture or the B-picture part of an PB- or Improved PB-frame. This is to preserve the disposable nature of B-pictures. Note, however, that B-pictures may occur in layers that have no corresponding picture in lower layers. This allows an encoder to send enhancement video with a higher picture rate than the lower layers.

The enhancement layer number and the reference layer number for each enhancement picture (B-, EI-, or EP-) are indicated in the ELNUM and RLNUM fields, respectively, of the picture header (when present). See the inference rules described in 5.1.4.4 for when these fields are not present. If a B-picture appears in an enhancement layer in which temporally surrounding SNR or spatial scalability pictures also appear, the Reference Layer Number (RLNUM) of the B-picture shall be the same as the Enhancement Layer Number (ELNUM).

The picture height, width, and pixel aspect ratio of a B-picture shall always be equal to those of its temporally subsequent reference layer picture.

O.2 Transmission order of pictures

Pictures which are dependent on other pictures shall be located in the bitstream after the pictures on which they depend.

The bitstream syntax order is specified such that for reference pictures (i.e. picture having types I, P, EI, or EP, or the P-part of PB or Improved PB), the following two rules shall be obeyed:

1) All reference pictures with the same temporal reference shall appear in the bitstream in increasing enhancement layer order (since each lower layer reference picture is needed to decode the next higher layer reference picture).

2) All temporally simultaneous reference pictures as discussed in item 1) above shall appear in the bitstream prior to any B-pictures for which any of these reference pictures is the first temporally subsequent reference picture in the reference layer of the B-picture (in order to reduce the delay of decoding all reference pictures which may be needed as references for B-pictures).

Then, the B-pictures with earlier temporal references shall follow (temporally ordered within each enhancement layer).

The bitstream location of each B-picture shall comply with the following rules:

1) Its bitstream location shall be after that of its first temporally subsequent reference picture in the reference layer (since the decoding of the B-picture generally depends on the prior decoding of that reference picture).

2) Its bitstream location shall be after that of all reference pictures that are temporally simultaneous with the first temporally subsequent reference picture in the reference layer (in order to reduce the delay of decoding all reference pictures which may be needed as references for B-pictures).

3) Its bitstream location shall precede the location of any additional temporally subsequent pictures other than B-pictures in its reference layer (since to allow otherwise would increase picture-storage memory requirements for the reference layer pictures).

4) Its bitstream location shall be after that of all EI- and EP-pictures that are temporally simultaneous with the first temporally subsequent reference picture.

5) Its bitstream location shall precede the location of all temporally subsequent pictures within its same enhancement layer (since to allow otherwise would introduce needless delay and increase picture-storage memory requirements for the enhancement layer).

Figure O.5 illustrates two allowable picture transmission orders given by the rules above for the layering structure shown therein (with numbers in dotted-line boxes indicating the bitstream order, separated by commas for the two alternatives).

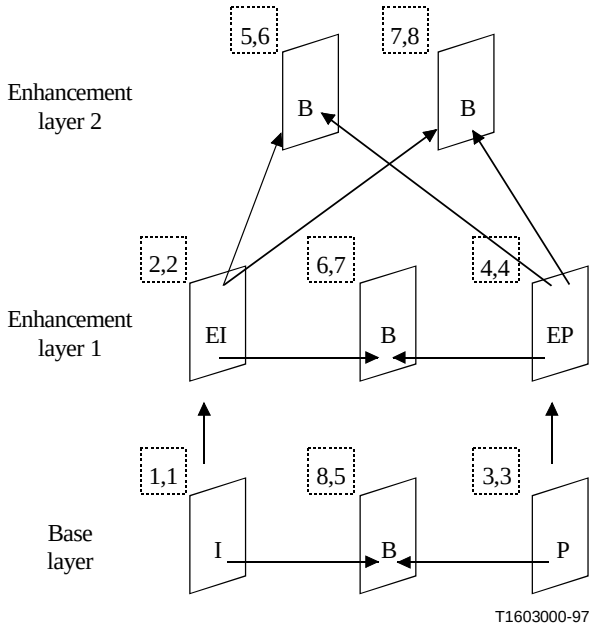


Figure O.5/H.263 – Example of picture transmission order

O.3 Picture layer syntax

The Enhancement Layer Number (ELNUM) (see 5.1.11) is always present in any enhancement (B, EI, or EP) picture and shall not be present in I- or P-pictures, or in PB- or Improved PB-frames. The Reference Layer Number (RLNUM) (see 5.1.12) is present for some enhancement pictures and is inferred for others, as described in 5.1.12.

There is exactly one base layer, and it has ELNUM and RLNUM equal to 1. The RLNUM gives the enhancement layer number of the forward and backward reference pictures for B-pictures, for the upward reference picture for EI- and EP-pictures. The reference pictures of the base layer may consist of I, PB, Improved PB, and P-pictures, none of which include ELNUM or RLNUM in the picture header (their implied values are 1).

For B-pictures, RLNUM must be less or equal to ELNUM, while for EI- and EP-pictures, RLNUM must be smaller than ELNUM.

ELNUM may be different from the layer number used at the system level. Since B-pictures have no other pictures dependent on themselves, they may even be put in a separate enhancement layer by system components external to this Recommendation (for example, Recommendations H.245 and H.223). Moreover, it is up to the implementor as to whether the enhancement pictures are sent in separate video channels or remain multiplexed together with the base layer pictures.

As stated in 5.1.4.5, the Deblocking Filter mode (see Annex J) does not apply within B-pictures. This is because B-pictures are not used for the prediction of any other pictures, and thus the application of a deblocking filter to these pictures is entirely a post-processing technique which is outside the scope of this Recommendation. However, the use of some type of deblocking filter for B-pictures is encouraged, and a filter such as that described in Annex J may in fact be a good design to use for that purpose.

The Temporal Reference (TR) (see 5.1.2) is defined exactly as for I- and P-pictures.

There shall be no TR_B (see 5.1.22) or DBQUANT (see 5.1.23) fields in the picture header of B-, EI- or EP-pictures.

O.4 Macroblock layer syntax

The macroblock layer syntax for B- and EP-pictures is the same, since each uses two reference pictures in a similar way. However, the interpretation varies slightly depending on the picture type. Figure O.6 indicates the B and EP syntax. The MBTYPE field indicates whether there is direct-mode prediction, forward prediction, backward/upward prediction, or bidirectional prediction. The MBTYPE is defined differently for B- and EP-pictures as described below.

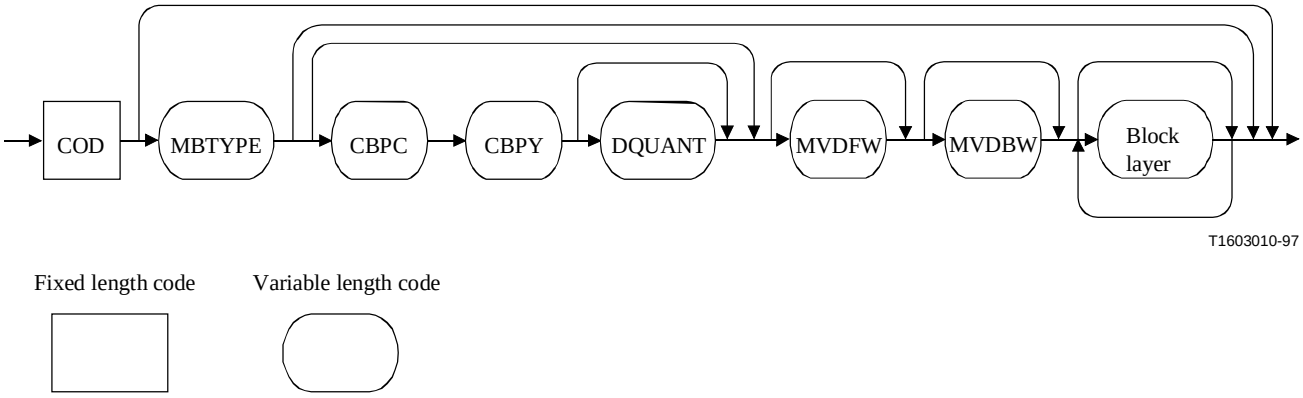


Figure O.6/H.263 – Macroblock syntax for EP- and B-pictures

The direct prediction mode is only available for B-pictures. It is a bidirectional prediction mode similar to the bidirectional mode in Improved PB-frames mode (Annex M). The only difference is that there is no restriction on which pixels can be backward predicted since the complete backward prediction image is known at the decoder. Bidirectional mode uses separate motion vectors for forward and backward prediction. In both direct and bidirectional mode, the prediction pixel values are calculated by averaging the forward and backward prediction pixels. The average is calculated by dividing the sum of the two predictions by two (division by truncation). In direct mode, when there are four motion vectors in the reference macroblock, then all four motion vectors are used as in Improved PB-frames mode (Annex M).

For B-pictures, forward prediction means prediction from a previous reference picture in the reference layer. Backward prediction means prediction from a temporally subsequent reference picture in the reference layer.

For EP-pictures, forward prediction means prediction from a previous EI- or EP-picture in the same layer, while upward prediction is used to mean prediction from the temporally simultaneous (possibly interpolated) reference picture in the reference layer. No motion vector is used for the upward prediction (which is syntactically in the same place as backward prediction for B-pictures), although one can be used for the forward prediction.

The macroblock syntax for EI-pictures is slightly different. As shown in Figure O.7, the MBTYPE and CBPC are combined into an MCBPC field. No forward prediction is used, only upward prediction from the temporally simultaneous reference picture in the reference layer. No motion vector is used.

In B- and EP-pictures, motion vectors over picture boundaries may be used as described in D.1 (although extension of the motion vector range as described in D.2 is only active if the Unrestricted Motion Vector mode is also in use).

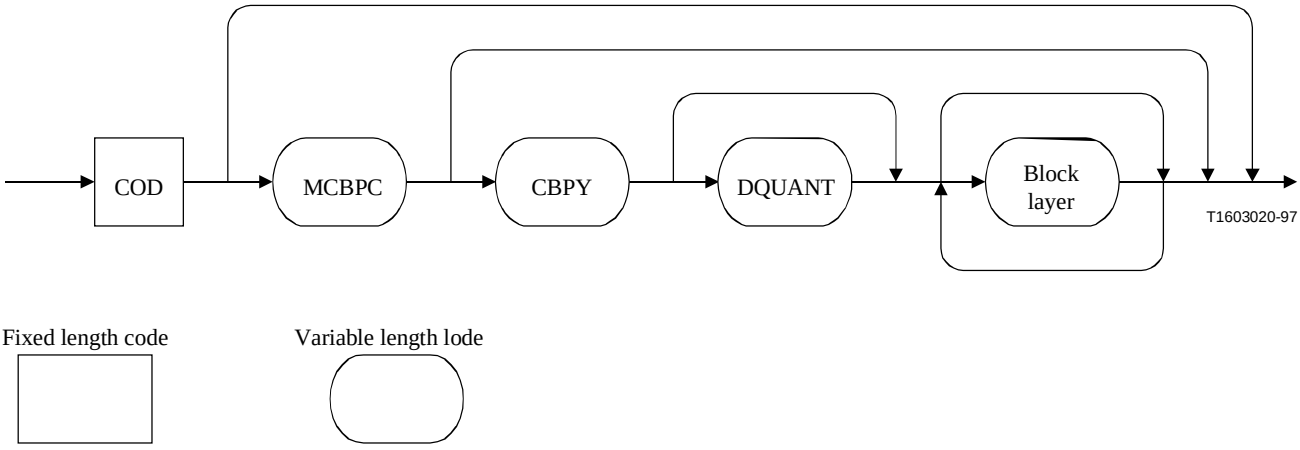


Figure O.7/H.263 – Macroblock syntax for EI-pictures

O.4.1 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 and the macroblock is handled as "skipped" as described below.

O.4.2 MBTYPE/MCBPC (VLC)

There are different MBTYPE tables for B- and EP-pictures. For EI-pictures, there is instead a MCBPC table. Table O.1 is the MBTYPE table for B-pictures. Table O.2 is the MBTYPE table for EP-pictures. Table O.3 is the MCBPC table for EI-pictures.

For B-pictures, the "Direct (skipped)" prediction type indicates that neither MBTYPE, nor any data, is transmitted in the macroblock and that the decoder derives forward and backward motion vectors and the corresponding bidirectional prediction. This is signaled by the COD bit. The "Forward (no texture)", the "Backward (no texture)", and the "Bi-dir (no texture)" prediction types for a B-picture indicate forward, backward, and bi-directional prediction with no coefficients, and with one transmitted motion vector for forward and backward prediction, and two transmitted motion vectors for bidirectional prediction.

For EP-pictures, the "Forward (skipped)" prediction type indicates that no additional data is sent for the macroblock, so that the decoder should use forward prediction with a zero motion vector and no coefficients. The "Upward (no texture)" and "Bi-dir (no texture)" prediction types for an EP-picture indicate upward and bidirectional prediction with no coefficients, and with zero motion vector(s).

For EI-pictures, the "Upward (skipped)" prediction type indicates that no additional data is sent for the macroblock, so that the decoder should use upward prediction with a zero motion vector and no coefficients.

Table O.1/H.263 – MBTYPE VLC codes for B-pictures

Index	Prediction type	MVDFW	MVDBW	CBPC + CBPY	DQUANT	MBTYPE	Bits
–	Direct (skipped)					(COD = 1)	0
0	Direct			X		11	2
1	Direct + Q			X	X	0001	4
2	Forward (no texture)	X				100	3
3	Forward	X		X		101	3
4	Forward + Q	X		X	X	0011 0	5
5	Backward (no texture)		X			010	3
6	Backward		X	X		011	3
7	Backward + Q		X	X	X	0011 1	5
8	Bi-dir (no texture)	X	X			0010 0	5
9	Bi-dir	X	X	X		0010 1	5
10	Bi-dir + Q	X	X	X	X	0000 1	5
11	INTRA			X		0000 01	6
12	INTRA + Q			X	X	0000 001	7
13	Stuffing					0000 0000 1	9

Table O.2/H.263 – MBTYPE VLC codes for EP-pictures

Index	Prediction type	MVDFW	MVDBW	CBPC + CBPY	DQUANT	MBTYPE	Bits
–	Forward (skipped)					(COD = 1)	0
0	Forward	X		X		1	1
1	Forward + Q	X		X	X	001	3
2	Upward (no texture)					010	3
3	Upward			X		011	3
4	Upward + Q			X	X	0000 1	5
5	Bi-dir (no texture)					0001 0	5
6	Bi-dir	X		X		0001 1	5
7	Bi-dir + Q	X		X	X	0000 01	6
8	INTRA			X		0000 001	7
9	INTRA + Q			X	X	0000 0001	8
10	Stuffing					0000 0000 1	9

Table O.3/H.263 – MCBPC VLC codes for EI-pictures

Index	Prediction type	Code block pattern (56)	CBPY	DQUANT	MCBPC	Bits
–	Upward (skipped)				(COD = 1)	0
0	Upward	00	X		1	1
1	Upward	01	X		001	3
2	Upward	10	X		010	3
3	Upward	11	X		011	3
4	Upward + Q	00	X	X	0001	4
5	Upward + Q	01	X	X	0000 001	7
6	Upward + Q	10	X	X	0000 010	7
7	Upward + Q	11	X	X	0000 011	7
8	INTRA	00	X		0000 0001	8
9	INTRA	01	X		0000 1001	8
10	INTRA	10	X		0000 1010	8
11	INTRA	11	X		0000 1011	8
12	INTRA+Q	00	X	X	0000 1100	8
13	INTRA+Q	01	X	X	0000 1101	8
14	INTRA+Q	10	X	X	0000 1110	8
15	INTRA+Q	11	X	X	0000 1111	8
16	Stuffing				0000 0000 1	9

O.4.3 Coded Block Pattern for Chrominance (CBPC) (Variable length)

When present, CBPC indicates the coded block pattern for chrominance blocks as described in Table O.4. CBPC is present only for EP- and B-pictures, when its presence is indicated by MBTYPE (see Tables O.1 and O.2).

Table O.4/H.263 – CBPC VLC codes

Index	Code block pattern (56)	CBPC	Bits
0	00	0	1
1	01	10	2
2	10	111	3
3	11	110	3

O.4.4 Coded Block Pattern for Luminance (CBPY) (Variable length)

When present, CBPY indicates which blocks in the luminance portion of the macroblock are present. CBPY is present only when its presence is indicated by MBTYPE (see Tables O.1, O.2 and O.3). CBPY is coded as described in 5.3.5 and in Table 13. Upward predicted macroblocks in EI- and EP-pictures, bidirectional predicted macroblocks in EP-pictures, and INTRA macroblocks in EI-, EP-, and B-pictures use the CBPY definition for INTRA macroblocks, and other macroblock types in EI-, EP-, and B-pictures use the CBPY definition for INTER macroblocks.

O.4.5 Quantizer Information (DQUANT) (2 bits/Variable length)

DQUANT is used as in other picture macroblock types. See 5.3.6 and Annex T.

O.4.6 Motion vector data (MVDFW, MVDBW) (Variable length)

MVDFW is the motion vector data for the forward vector, if present. MVDBW is the motion vector data for the backward vector, if present (allowed only in B-pictures). The variable length codewords are given in Table 14, or in Table D.3 if the Unrestricted Motion Vector mode is used (see Annex D).

O.5 Motion vector decoding

O.5.1 Differential motion vectors

Motion vectors for forward, backward or bidirectionally predicted blocks are differentially encoded. To recover the macroblock motion vectors, a prediction is added to the motion vector differences. The predictions are formed in a similar manner to that described in 6.1.1, except that forward motion vectors are predicted only from forward motion vectors in surrounding macroblocks, and backward motion vectors are predicted only from backward motion vectors in surrounding macroblocks. The same decision rules apply for the special cases at picture, GOB or slice borders as described in 6.1.1. If a neighboring macroblock does not have a motion vector of the same type (forward or backward), the candidate predictor for that macroblock is zero for that motion vector type.

O.5.2 Motion vectors in direct mode

For macroblocks coded in direct mode, no vector differences are transmitted. Instead, the forward and backward motion vectors are directly computed from the temporally consecutive P-vector as described in G.4 with the restriction that MV_D is always zero. These derived vectors are not used for prediction of other motion vectors.

O.6 Interpolation filters

The method by which a picture is interpolated for 2-D spatial scalability is shown in Figures O.8 and O.9. The first figure shows the interpolation for interior pixels, while the second one shows the interpolation close to picture boundaries. This is the same technique used in Annex Q and in some cases in Annex P. The method by which a picture is interpolated for 1-D spatial scalability is shown in Figures O.10 and O.11. Figure O.10 shows the interpolation for interior pixels in the horizontal direction. The interpolation for the vertical direction is analogous. Figure O.11 shows the interpolation for pixels at picture boundaries. Again, the interpolation for the vertical direction is analogous. Again, this is the same technique used in some cases in Annex P.

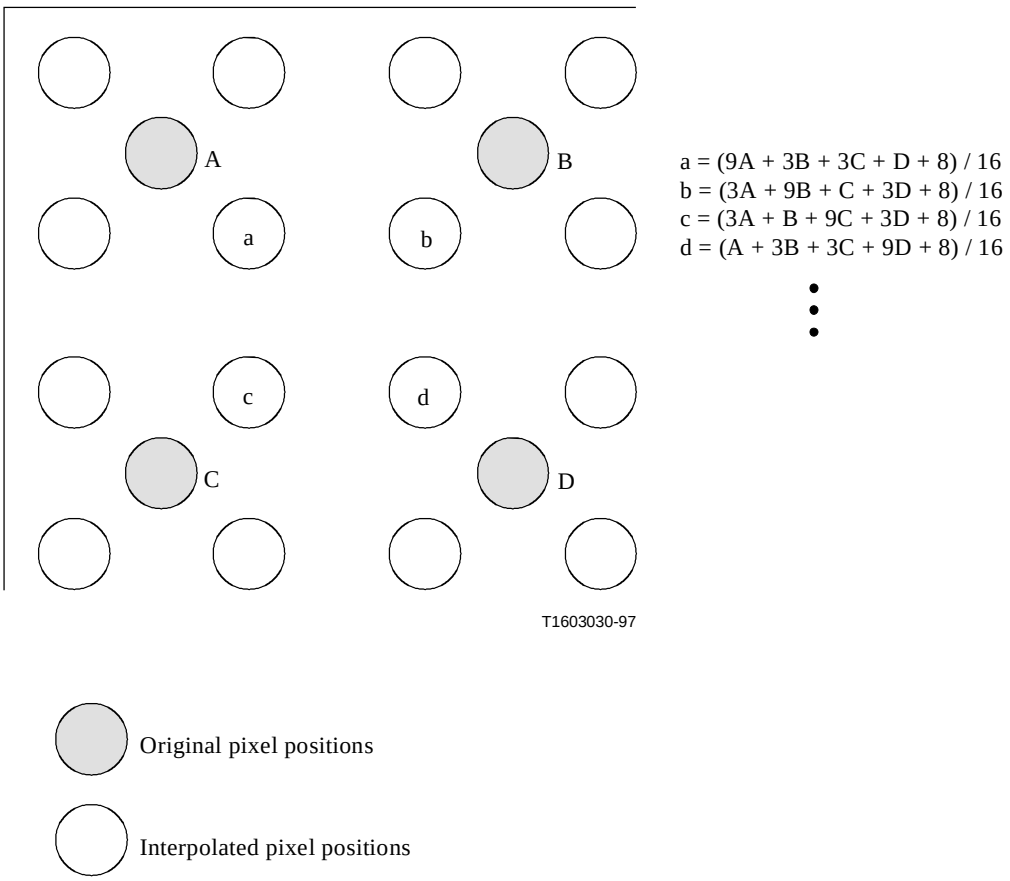


Figure O.8/H.263 – Method for interpolating pixels for 2-D scalability

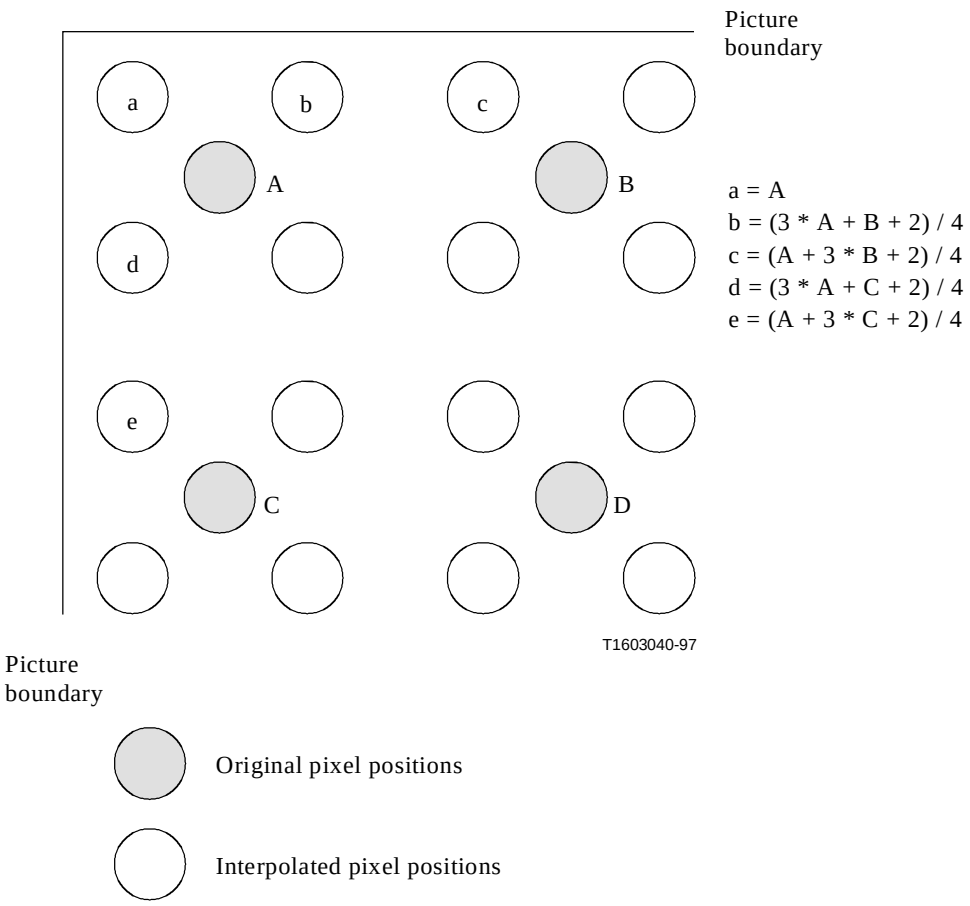


Figure O.9/H.263 – Method for 2-D interpolation at boundaries

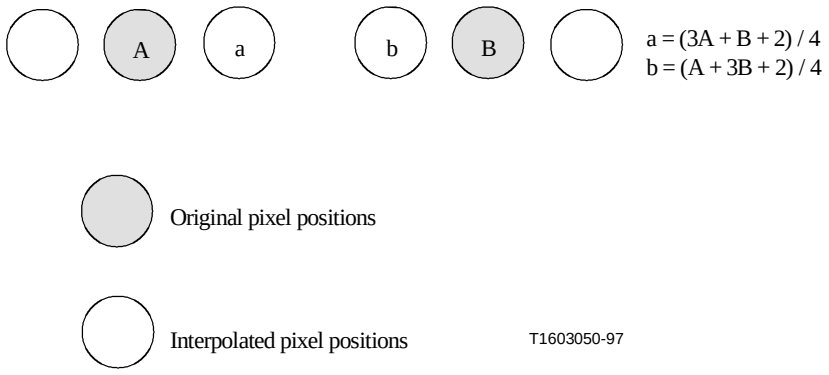


Figure O.10/H.263 – Method for interpolating pixels for 1-D scalability

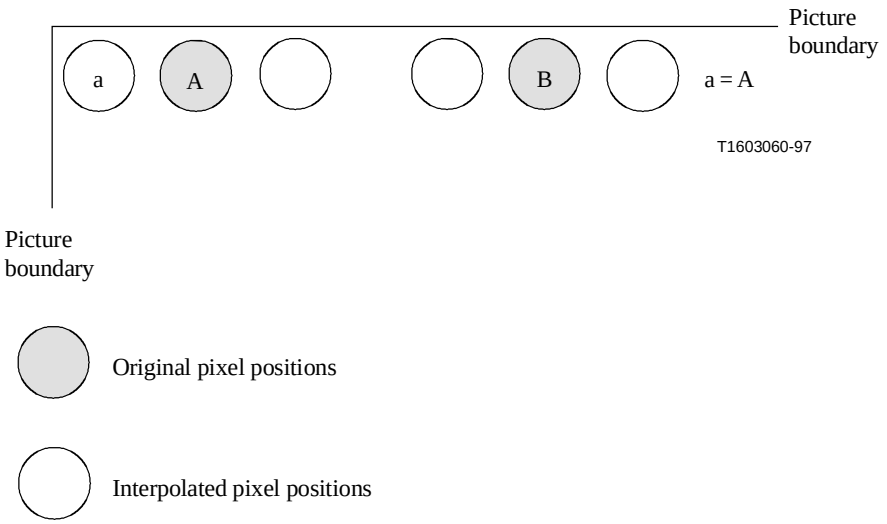


Figure O.11/H.263 – Method for 1-D interpolation at picture boundaries

ANNEX P

Reference picture resampling

P.1 Introduction

This annex describes the use and syntax of a resampling process which can be applied to the previous decoded reference picture in order to generate a "warped" picture for use in predicting the current picture. This resampling syntax can specify the relationship of the current picture to a prior picture having a different source format, and can also specify a "global motion" warping alteration of the shape, size, and location of the prior picture with respect to the current picture. In particular, the Reference Picture Resampling mode can be used to adaptively alter the resolution of pictures during encoding. A fast algorithm is used to generate bilinear interpolation coefficients. The capability to use this mode and the extent to which its features are supported is negotiated externally (for example, Recommendation H.245). This mode may be used in restricted scenarios that may be defined during capability negotiation (e.g. to support only factor-of-four picture resizing, to support only half-pixel resolution picture warping, or to support arbitrary image size resizing and displacements).

NOTE – The default image transformation between pictures of differing resolutions is defined to maintain the spatial alignment of the edges of the picture area and of the relative locations of the luminance and chrominance samples. This may have implications on the design of any resampling operations used for generating pictures of various resolutions at the encoder and for displaying pictures of various resolutions after decoding (particularly regarding shifts in spatial location caused by phase shifts induced in resampling). Also, since this mode can be used for dynamic adaptive picture resolution changes, operation with this mode may benefit from external negotiation to display the decoded picture at a higher resolution than its encoded picture size, in order to allow switching between encoded picture sizes without resizing the picture display.

If the Reference Picture Resampling bit in the PLUSPTYPE field is not set, but PLUSPTYPE is present and the picture is an INTER, B, or EP-picture or an Improved PB-frame, and the picture size differs from that of the temporally previous encoded picture, this condition invokes Reference Picture Resampling with warping parameters (see P.2.2) set equal to zero, fill mode (see P.2.3) set to *clip*, and displacement accuracy (see P.2.1) set to $\frac{1}{16}$-pixel accuracy. This causes the resampling process to easily act as a predictively-encoded change in picture resolution. In simple factor-of-four

resolution change cases such as transitions between CIF and 4CIF, the resampling process reduces to the same simple filter used for Spatial Scalability (Annex O) or for Reduced-Resolution Update (Annex Q), except for the application of rounding control.

If the picture is an EP-picture and the Reference Picture Resampling bit is set in the PLUSPTYPE field of the picture header of the reference layer, this bit shall also be set in the picture header of the EP-picture in the enhancement layer.

If a B-picture uses the Reference Picture Resampling mode, the resampling process shall be applied to the temporally previous anchor picture and not to the temporally subsequent one. The temporally previous anchor picture to which the resampling process is applied shall be the decoded picture (i.e. prior to any resampling applied by Reference Picture Resampling if this mode is also invoked for the subsequent reference picture). The temporally subsequent anchor picture shall have the same picture size as the B-picture.

If the Reference Picture Resampling mode is invoked for an Improved PB-frame, one set of warping parameters is sent and the resampled reference picture is used as the reference for both the B- and P-parts of the Improved PB-frame.

The Reference Picture Resampling is defined in terms of the displacement of the four corners of the current picture area. For the luminance field of the current picture with horizontal size *H* and vertical size *V*, four conceptual motion vectors $\underline{v}^{00}$, $\underline{v}^{H0}$, $\underline{v}^{0V}$, and $\underline{v}^{HV}$, are defined for the upper left, upper right, lower left, and lower right corners of the picture, respectively. These vectors describe how to move the corners of the current picture to map them onto the corresponding corners of the previous decoded picture, as shown in Figure P.1. The units of these vectors are the same as those in the reference picture grid. To generate a vector $\underline{v}(x, y)$ at some real-valued location (*x, y*) in the interior of the current picture, an approximation to bilinear interpolation is used, *i.e.* as in:

$$\underline{v}(x,y)=\left(1-\frac{y}{V}\right)\left[\left(1-\frac{x}{H}\right)\underline{v}^{00}+\left(\frac{x}{H}\right)\underline{v}^{H0}\right]+\left(\frac{y}{V}\right)\left[\left(1-\frac{x}{H}\right)\underline{v}^{0V}+\left(\frac{x}{H}\right)\underline{v}^{HV}\right]$$

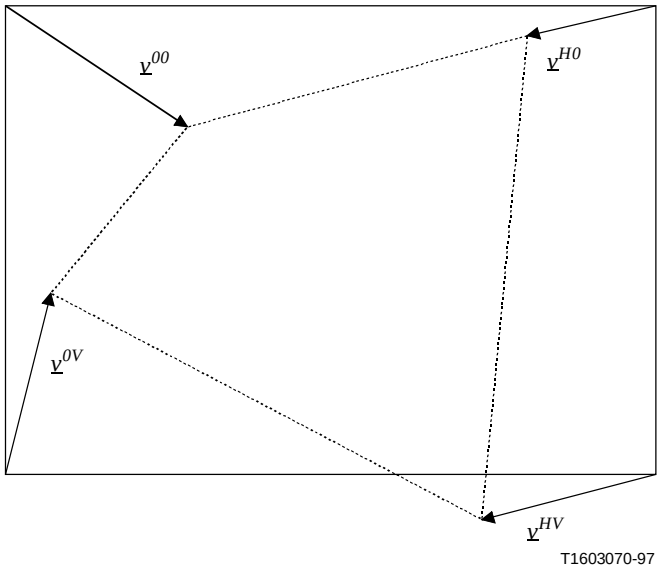


Figure P.1/H.263 – Example of conceptual motion vectors used for warping

The horizontal size *H* and vertical size *V* of the current picture, and the horizontal size H_R and vertical size V_R of the reference picture are those indicated by the picture header, regardless of

whether these values are divisible by 16 or not. If the picture width or height is not divisible by 16, the additional area shall be generated by adding pixels to the resampled picture using the same fill mode as used in the resampling process.

For the simplicity of description, the resampling vectors $\underline{r}^0$, $\underline{r}^x$, $\underline{r}^y$ and $\underline{r}^{xy}$ are defined as:

$$\underline{r}^0 = \underline{v}^{00}$$

$$\underline{r}^x = \underline{v}^{H0} - \underline{v}^{00}$$

$$\underline{r}^y = \underline{v}^{0V} - \underline{v}^{00}$$

$$\underline{r}^{xy} = \underline{v}^{00} - \underline{v}^{H0} - \underline{v}^{0V} + \underline{v}^{HV}$$

Using this definition, the equation for bilinear interpolation is rewritten as:

$$\underline{v}(x,y) = \underline{r}^0 + \left(\frac{x}{H}\right)\underline{r}^x + \left(\frac{y}{V}\right)\underline{r}^y + \left(\frac{x}{H}\right)\left(\frac{y}{V}\right)\underline{r}^{xy}$$

For warping purposes, it is assumed that the coordinates of the upper left corner of the picture area are (x, y) = (0,0) and that each pixel has unit height and width, so that the centers of the pixels lie at the points $(x,y) = \left(i_L + \frac{1}{2}, j_L + \frac{1}{2}\right)$ for i_L = 0, …, H – 1 and j_L = 0, …, V – 1, where the L subscript indicates that i_L and j_L pertain to the luminance field. (As the pixel aspect ratio is usually constant or the conversion of the aspect ratio is performed by this resampling, there is no need to consider the actual pixel aspect ratio for these purposes.) Using this convention, the x- and y-displacements at the locations of interest in the luminance field of the reference picture are:

$$v_x(i_L, j_L) = \frac{1}{HV}\left[HVr_x^0 + \left(i_L + \frac{1}{2}\right)Vr_x^x + \left(j_L + \frac{1}{2}\right)Hr_x^y + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_x^{xy}\right]$$

$$v_y(i_L, j_L) = \frac{1}{HV}\left[HVr_y^0 + \left(i_L + \frac{1}{2}\right)Vr_y^x + \left(j_L + \frac{1}{2}\right)Hr_y^y + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_y^{xy}\right]$$

Because all positions and phases must be computed relative to the center of the upper-left corner pixel, which has the coordinate (x, y) = $\left(\frac{1}{2}, \frac{1}{2}\right)$, the quantities of primary interest are:

$$x_R(i_L, j_L) - \frac{1}{2} = \left(i_L + \frac{1}{2}\right) + v_x(i_L, j_L) - \frac{1}{2}$$

$$= \frac{1}{HV}\left[HVr_x^0 + \left(i_L + \frac{1}{2}\right)(HV + Vr_x^x) + \left(j_L + \frac{1}{2}\right)Hr_x^y + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_x^{xy}\right] - \frac{1}{2}$$

$$y_R(i_L, j_L) - \frac{1}{2} = \left(i_L + \frac{1}{2}\right) + v_y(i_L, j_L) - \frac{1}{2}$$

$$= \frac{1}{HV}\left[HVr_y^0 + \left(j_L + \frac{1}{2}\right)Vr_y^x + \left(j_L + \frac{1}{2}\right)(HV + Hr_y^y) + \left(i_L + \frac{1}{2}\right)\left(j_L + \frac{1}{2}\right)r_y^{xy}\right] - \frac{1}{2}$$

Once a location has been determined in the prior decoded reference picture using an approximation of these equations, bilinear interpolation as specified later in this annex shall be used to generate a value for the resampled pixel.

Each resampling vector can be decomposed into two components, with the first component describing the geometrical warping and the second component accounting for any difference in size between the predicted picture (horizontal size *H* and vertical size *V*) and the reference picture (horizontal size H_R and vertical size V_R). This decomposition is as follows:

$$\underline{v}^{00} = \underline{v}^{00}_{warp} + \underline{v}^{00}_{size} = \underline{v}^{00}_{warp} + (0,0)$$
$$\underline{v}^{H0} = \underline{v}^{H0}_{warp} + \underline{v}^{H0}_{size} = \underline{v}^{H0}_{warp} + (H_R - H,0)$$
$$\underline{v}^{0V} = \underline{v}^{0V}_{warp} + \underline{v}^{0V}_{size} = \underline{v}^{0V}_{warp} + (0,V_R - V)$$
$$\underline{v}^{HV} = \underline{v}^{HV}_{warp} + \underline{v}^{HV}_{size} = \underline{v}^{HV}_{warp} + (H_R - H,V_R - V)$$

P.2 Syntax

Whenever the reference picture resampling bit is set in the PLUSTPYPE field of the picture header, the RPRP field of the picture header includes parameters which control the Reference Picture Resampling process. This includes a two-bit Warping Displacement Accuracy (WDA) field, may include eight warping parameters or one-bit warping parameter refinements, and includes a fill mode, as described in this subclause.

P.2.1 Warping Displacement Accuracy (WDA) (2 bits)

A two-bit warping displacement accuracy field WDA appears first in the RPRP field of the bitstream, and indicates the accuracy of the displacements for each pixel. A value of "10" indicates that the *x*- and *y*-displacements for each pixel are quantized to half-pixel accuracy. A value of "11" indicates that the displacements are quantized to $\frac{1}{16}$-pixel accuracy. The use of other values is reserved.

P.2.2 Warping parameters (Variable length)

When the Reference Picture Resampling parameters are sent for an INTER or B-picture, or an Improved PB-frame, eight warping parameters are included in the picture header using the variable length code (VLC) shown in Table D.3. For an EP-picture using SNR scalability, the warping parameters of the lower layer are used and no warping parameters are transmitted. If the Reference Picture Resampling bit is set in the PLUSPTYPE field of the picture header for an EP-picture using spatial scalability, the warping parameters of the lower layer are refined to the accuracy needed for the current layer by multiplying the warping parameter for each up-sampled dimension (warping parameters with subscript *x* and/or *y*) of the lower layer by two and adding the value of one additional bit that is sent in place of the associated warping parameter to define the least significant bit of the warping parameter.

The eight integer warping parameters (or their one-bit refinements) are sent in the following order:

$$w^0_x, w^0_y, w^x_x, w^x_y, w^y_x, w^y_y, w^{xy}_x, \text{ and } w^{xy}_y$$

When not a one-bit refinement, these parameters are sent in a similar manner as motion vector differences in the Unrestricted Motion Vector mode when PLUSPTYPE is present, using Table D.3 with no restriction on the range of the parameters (i.e. a range of –4095 to +4095). As when encoding motion vector difference pairs, an emulation prevention bit shall be added as needed after each pair of warping parameters is sent, such that if the all-zero codeword (the value +1 in half-pixel units) of Table D.3 is used for both warping parameters in the pair

(w_x^0 and w_y^0, w_x^x and w_y^y, or w_x^{xy} and w_y^{xy}) the pair of codewords is followed by a single bit equal to 1 to prevent start code emulation.

These eight warping parameters are interpreted as picture corner displacements relative to the displacements that would be induced by removing the resizing component of the resampling vectors. The warping parameters are scaled to represent half-pixel offsets in the luminance field of the current picture, and the range for the values of these paramenters is –4095 to +4095. The warping parameters are defined using the resampling vectors by the relations:

$$w_x^0 = 2r_x^0 \qquad w_y^0 = 2r_y^0$$

$$w_x^x = 2\left(r_x^x - (H_R - H)\right) \qquad w_y^x = 2r_y^x$$

$$w_x^y = 2r_x^y \qquad w_y^y = 2\left(r_y^y - (V_R - V)\right)$$

$$w_x^y = 2r_x^y \qquad w_y^{xy} = 2r_y^{xy}$$

P.2.3 Fill Mode (FILL_MODE) (2 bits)

For an INTER or B-picture or an Improved PB-frame, immediately following the VLC-coded warping parameters in the picture header are two bits which define the fill-mode action to be taken for the values of pixels for which the calculated location in the reference picture lies outside of the reference picture area. The meaning of these two bits is shown in Table P.1 and their location is shown in Figure P.2. For an EP-picture, the fill-mode action is the same as that for the reference layer, and the two fill-mode bits are not sent.

Table P.1/H.263 – Fill mode bits/action

fill-mode bits	fill action
0 0	*color*
0 1	*black*
1 0	*gray*
1 1	*clip*

If the fill mode is *clip*, the coordinates of locations in the prior reference picture are independently limited as in the Unrestricted Motion Vector mode so that pixel values outside of the prior reference picture area are estimated by extrapolation from the values of pixels at the image border. If the fill mode is *black*, luminance samples outside of the prior reference picture area are assigned a value of Y = 16 and chrominance samples are assigned a value of $C_B = C_R = 128$. If the fill mode is *gray*, luminance and chrominance values are assigned a value of $Y = C_B = C_R = 128$. If the fill mode is *color*, then additional fields are sent to specify a fill color, as described in the next subclause.

P.2.4 Fill Color Specification (Y_FILL, C_B_EPB, C_B_FILL, C_R_EPB, C_R_FILL) (26 bits)

If the fill mode is *color* and the picture is not an EP-picture, then the fill-mode bits are followed in the bitstream by three eight-bit integers, *Y_fill*, C_B_ *fill*, and C_R_ *fill*, which specify a fill color precisely. Between these three eight-bit integers are two emulation prevention bits (C_B_EPB and C_R_EPB) each of which is equal to 1. The format of this color specification, which is present only when the fill mode is *color*, is shown in Figure P.2. Each eight-bit integer field is sent using its

natural representation. For an EP-picture, the fill-mode action (and fill color) is the same as that for the reference layer, and the fill color specification is not sent.

FILL_MODE	Y_FILL	C_B_EPB	C_B_FILL	C_R_EPB	C_R_FILL

Figure P.2/H.263 – Format of fill mode and fill color specification data

P.3 Resampling algorithm

The method described in this subclause shall be mathematically equal in result to that used to generate the samples of the resampled reference picture. Using the integer warping parameters $w_x^0, w_y^0, w_x^x, w_y^x, w_x^y, w_y^y, w_x^{xy}$, and w_y^{xy} integer parameters u_x^{00}, u_y^{00}, u_x^{H0}, u_y^{H0}, u_x^{0V}, u_y^{0V}, u_x^{HV}, and u_y^{HV} which denote the *x*- and *y*-displacements at the corners of the luminance field in $\frac{1}{32}$-pixel accuracy (the actual displacements are obtained by dividing these values by 32) are defined as:

$$u_x^{00} = 16w_x^0 \qquad u_y^{00} = 16w_y^0$$

$$u_x^{H0} = 16\left(w_x^0 + w_x^x + 2\left(H_R - H\right)\right) \qquad u_y^{H0} = 16\left(w_y^0 + w_y^x\right)$$

$$u_x^{0V} = 16\left(w_x^0 + w_x^y\right) \qquad u_y^{0V} = 16\left(w_y^0 + w_y^y + 2\left(V_R - V\right)\right)$$

$$u_x^{HV} = 16\left(w_x^0 + w_x^x + w_x^y + w_x^{xy} + 2\left(H_R - H\right)\right) \qquad u_y^{HV} = 16\left(w_y^0 + w_y^x + w_y^y + w_y^{xy} + 2\left(V_R - V\right)\right)$$

Next, *H´* and *V´*, which denote the horizontal and vertical size of the *virtual frame*, are defined as the smallest integers that satisfy the following condition:

$$H' \geq H, V' \geq V,\ H' = 2^m,\ V' = 2^n,\ m \text{ and } n \text{ are positive integers}$$

By applying bilinear extrapolation to the corner vectors of the luminance field, the integer parameters u_x^{LT}, u_y^{LT}, u_x^{RT}, u_y^{RT}, u_x^{LB}, u_y^{LB}, u_x^{RB}, and u_y^{RB} which denote the *x*- and *y*-displacements of the luminance field at the *virtual points* (*x*, *y*) = (0, 0), (*H´*, 0), (0, *V´*), and (*H´*, *V´*) in $\frac{1}{32}$-pixel accuracy (the actual displacements are obtained by dividing these values by 32) are defined as:

$$u_x^{LT} = u_x^{00} \qquad u_y^{LT} = u_y^{00}$$

$$u_x^{RT} = \left((H - H')u_x^{00} + H'u_x^{H0}\right) // H \qquad u_y^{RT} = \left((H - H')u_y^{00} + H'u_y^{H0}\right) // H$$

$$u_x^{LB} = \left((V - V')u_x^{00} + V'u_x^{0V}\right) // V \qquad u_y^{LB} = \left((V - V')u_y^{00} + V'u_y^{0V}\right) // V$$

$$u_x^{RB} = \left((V - V')\left((H - H')u_x^{00} + H'u_x^{H0}\right) + V'\left((H - H')u_x^{0V} + H'u_x^{HV}\right)\right) // (HV)$$

$$u_y^{RB} = \left((V - V')\left((H - H')u_y^{00} + H'u_y^{H0}\right) + V'\left((H - H')u_y^{0V} + H'u_y^{HV}\right)\right) // (HV)$$

where "//" denotes integer division that rounds the quotient to the nearest integer, and rounds half integer values away from 0.

In the remainder of this annex, it is assumed that the centers of the pixels lie at the points $(x, y) = \left(i + \frac{1}{2}, j + \frac{1}{2} \right)$ for both the luminance and chrominance fields. Integer parameters *i, j* are defined as:

- $i = 0, \ldots, H - 1$ and $j = 0, \ldots, V - 1$ for luminance, and
- $i = 0, \ldots, \frac{H}{2} - 1$ and $j = 0, \ldots, \frac{V}{2} - 1$ for chrominance.

This implies that different coordinate systems are used for luminance and chrominance as shown in Figure P.3. Using the coordinate system for chrominance, the integer parameters u_x^{LT}, u_y^{LT}, u_x^{RT}, u_y^{RT}, u_x^{LB}, u_y^{LB}, u_x^{RB}, and u_y^{RB} defined above can also be regarded as the *x*- and *y*-displacements of the chrominance field at virtual points (*x*, *y*) = (0, 0), (*H´*/2, 0), (0, *V´*/2), and (*H´*/2, *V´*/2) in $\frac{1}{64}$-pixel accuracy (the actual displacements are obtained by dividing these values by 64). Using these parameters and an additional parameter *S*, which is defined as 2 for luminance and as 1 for chrominance, the resampling algorithm for the luminance and chrominance fields are defined using common equations.

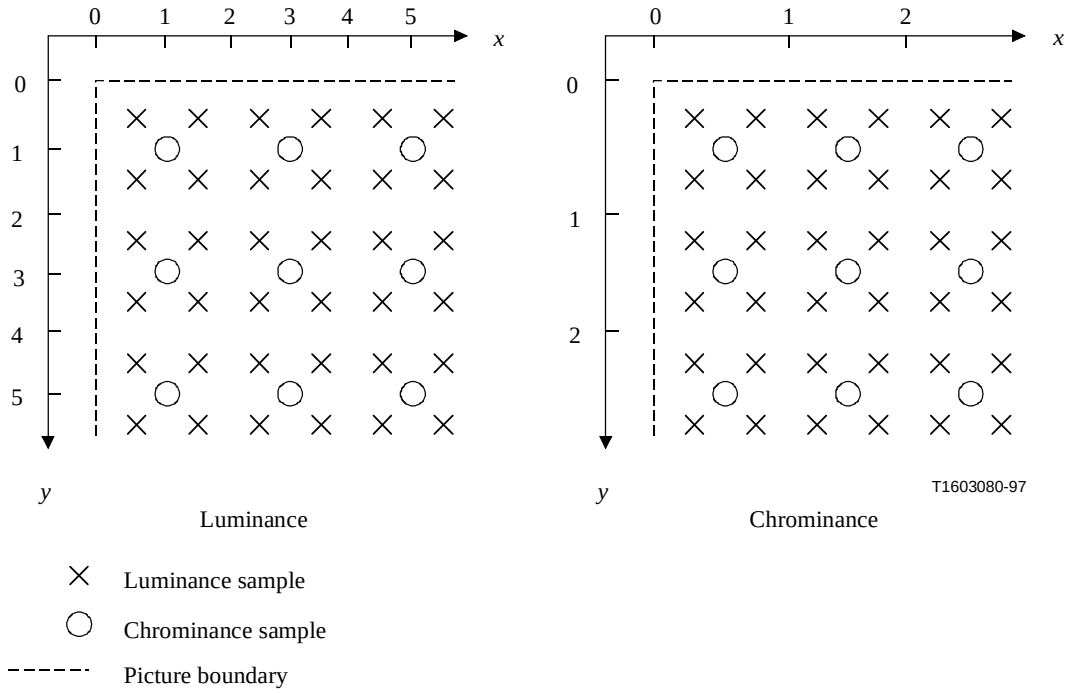


Figure P.3/H.263 – Coordinate systems for luminance and chrominance fields

The integer parameters $u_x^L(j)$, $u_y^L(j)$, $u_x^R(j)$, and $u_y^R(j)$ which denote the *x*- and *y*-displacements of the picture field at (*x*, *y*) = $(0, j + \frac{1}{2})$ and $(SH´/2, j + \frac{1}{2})$ in $\frac{s}{64}$-pixel accuracy (the actual displacements are obtained by dividing these values by 64/*S*) are defined using one dimensional linear interpolation as:

$$u_x^L(j)=\left((SV'-2j-1)u_x^{LT}+(2j+1)u_x^{LB}\right)//(SV') \quad u_y^L(j)=\left((SV'-2j-1)u_y^{LT}+(2j+1)u_y^{LB}\right)//(SV')$$

$$u_x^R(j)=\left((SV'-2j-1)u_x^{RT}+(2j+1)u_x^{RB}\right)//(SV') \quad u_y^R(j)=\left((SV'-2j-1)u_y^{RT}+(2j+1)u_y^{RB}\right)//(SV')$$

where "//" denotes integer division that rounds the quotient to the nearest integer, and rounds half integer values away from 0.

Finally, the parameters that specify the transformed position in the reference picture become:

$$I_R(i,j)=Pi+\left((SH'-2i-1)u_x^L(j)+(2i+1)u_x^R(j)+32H'/P\right)///(64H'/P)$$

$$J_R(i,j)=Pj+\left((SH'-2i-1)u_y^L(j)+(2i+1)u_y^R(j)+32H'/P\right)///(64H'/P)$$

$$i_R(i,j)=I_R(i,j)///P \qquad j_R(i,j)=J_R(i,j)///P$$

$$\varnothing_x=I_R(i,j)-\left(I_R(i,j)///P\right)P \qquad \varnothing_y=J_R(i,j)-\left(J_R(i,j)///P\right)P$$

where

"///" integer division with truncation towards the negative infinity;

"/" integer division (in this case resulting in no loss of accuracy);

P accuracy of *x*- and *y*-displacements (*P* = 2 when WDA = "10" and *P* = 16 when WDA = "11" or is absent, see P.2.1 for the definition of WDA);

$\left(\frac{I_R(i,j)}{P}+\frac{1}{2},\frac{J_R(i,j)}{P}+\frac{1}{2}\right)$ (*x*, *y*) location of the transformed position (both $I_R(i,j)$ and $J_R(i,j)$ are integers);

$\left(i_R(i,j)+\frac{1}{2}, j_R(i,j)+\frac{1}{2}\right)$ (*x*, *y*) location of the sampling point near the transformed position (both $i_R(i,j)$ and $j_R(i,j)$ are integers);

$\left(\varnothing_x,\varnothing_y\right)$ bilinear interpolation coefficients of the transformed position (both $\varnothing_x$ and $\varnothing_y$ are integers).

The computation of this equation can be simplified by replacing the divisions by shift operations, since $64H'/P = 2^{m+2}$ when *P* = 16 and $64H'/P = 2^{m+5}$ when *P* = 2.

Using these parameters, the sample value, $E_P(i,j)$, of the pixel located at $(x,y)=\left(i+\frac{1}{2}, j+\frac{1}{2}\right)$ in the resampled picture is obtained using bilinear interpolation by:

$$E_P(i,j)=\Big((P-\varnothing_y)\big((P-\varnothing_x)E_R(i_R,j_R)+\varnothing_x E_R(i_R+1,j_R)\big)$$

$$+\varnothing_y\big((P-\varnothing_x)E_R(i_R,j_R+1)+\varnothing_x E_R(i_R+1,j_R+1)\big)+P^2/2-1+RCRPR\Big)/P^2$$

where "/" denotes division by truncation. i_R and j_R are simplified notations for $i_R(i,j)$ and $j_R(i,j)$, and $E_R(i_R,j_R)$ denotes the sample value of the pixel located at $(x,y)=\left(i_R+\frac{1}{2}, j_R+\frac{1}{2}\right)$ in the reference picture after extrapolation using the proper fill mode if necessary. The value of parameter RCRPR is defined as follows:

- For a B-picture (or the B-part of an Improved PB-frame) which has a P-picture as its temporally subsequent anchor picture, RCRPR is equal to the rounding type (RTYPE) bit in

MPPTYPE (see 5.1.4.3) of this temporally subsequent P-picture. This implies that for Improved PB-frames, RCRPR has the same value for the P-part and the B-part.

- For other types of pictures, RCRPR is equal to the RTYPE bit of the current picture.

P.4 Example of implementation

In this subclause, an implementation example of the algorithm described in the previous subclause is provided as pseudo-code.

P.4.1 Displacements of virtual points

When a large picture is coded, a straightforward implementation of the equation for obtaining parameters u_x^{RB} and u_y^{RB} shown in P.3 may necessitate the usage of variables requiring more that 32 bits for their binary representation. For systems which cannot easily use 64-bit integer or floating point registers, an example of an algorithm which does not require variables with more than 32 bits for the calculation of u_x^{RB} and u_y^{RB} is shown below.

Since *H*, *V*, *H´*, and *V´* are divisible by 4, the definition of u_x^{RB} can be rewritten as:

$$u_x^{RB} = \Big((V_Q - V_Q')\big((H_Q - H_Q')u_x^{00} + H_Q' u_x^{H0}\big) + V_Q'\big((H_Q - H_Q')u_x^{0V} + H_Q' u_x^{HV}\big)\Big) // A$$

where $H_Q = H/4$, $V_Q = V/4$, $H_Q' = H'/4$, $V_Q' = V'/4$, $A = H_Q V_Q$, and "//" denotes integer division that rounds the quotient to the nearest integer, and rounds half integer values away from 0. Next, parameters T_T and T_B are defined as:

$$T_T = (H_Q - H_Q')u_x^{00} + H_Q' u_x^{H0}$$
$$T_B = (H_Q - H_Q')u_x^{0V} + H_Q' u_x^{HV}$$

for simplicity of description. Using operator "///" which denotes integer division with truncation towards the negative infinity, and operator "%" which is defined as *a* % *b* = *a* – (*a* /// *b*) *b*, the value of u_x^{RB} can be obtained via the following pseudo-code:

```
q = (VQ - VQ´)*(TT /// A)+ VQ´*(TB /// A) + ((VQ - VQ´)*(TT % A)+ VQ´
    *(TB % A)) /// A;
r = ((VQ - VQ´)*(TT % A)+ VQ´ *(TB % A)) % A;
if (q < 0)
  uxRB = q+(r+(A-1)/2)/A;
else
  uxRB = q+(r+A/2)/A;
```

The value of u_y^{RB} can also be calculated using this algorithm.

P.4.2 Resampling algorithm

To simplify the description of the algorithm, a function `prior_sample` is defined. Its purpose is to generate a pixel value for any integer location (i_p, j_p) relative to the prior reference picture sampling grid:

```
clip(xmin, x, xmax) {
  if (x < xmin) {
    return xmin;
  } else if (x > xmax) {
    return xmax;
```

```
        } else {
          return x;
        }
      }
prior_sample (ip, jp) {
    if (FILL_MODE = clip) {
      ic = clip (0, ip, S*HR /2-1);
      jc = clip (0, jp, S*VR /2-1);
      return prior_ref[ic, jc];
    } else {
      if ((ip < 0) OR (ip > S*HR /2-1) or (jp < 0) OR (jp > S*VR /2-1) {
        return fill_value;
      } else {
        return prior_ref[ip,jp];
      }
    }
}
```

In the pseudo-code, `prior_ref[`*i, j*`]` indicates the sample in column *i* and row *j* in the temporally previous reference picture.

Next, a filter function that implements the bilinear interpolation described in P.3 is defined. It is assumed that all arguments of the following function are integers and that the bilinear interpolation coefficients, $Ø_x$ and $Ø_y$, are quantized in the range 0, …, *P*–1 (inclusive).

```
filter(x0, y0, Øx, Øy) {
  return [(P–Øy)·((P–Øx)·prior_sample(x0, y0)+Øx·prior_sample(x0 +1, y0))+
          Øy·((P–Øx)·prior_sample(x0, y0 + 1)+Øx·prior_sample(x0 +1, y0 +1))+
          P2 / 2–1+ RCRPR] / P2;
}
```

Finally, the method for warping the reference picture to generate a prediction for the current picture can be specified in terms of these functions. The pixels of the prediction picture can be generated in a raster scan order. It is assumed that the values $u_x^L(j)$, $u_y^L(j)$, $u_x^R(j)$, and $u_y^R(j)$ are already calculated and loaded into variables u_x^L, u_y^L, u_x^R, and u_y^R. Defining parameter *D* as $D = 64H'/P$ and noting that $H' = 2^m$, the sample values of the pixels in the *j* th line of the resampled field (the topmost line is defined as the 0th line) is obtained by the following pseudo-code:

```
a_x^i = D * P + 2 * (u_x^R – u_x^L);
a_y^i = 2 * (u_y^R – u_y^L);
a_x = u_x^L * S * 2^m + (u_x^R – u_x^L); + D / 2;
a_y = j * D * P + u_y^L * S * 2^m + (u_y^R – u_y^L); + D / 2;
for (i = 0; i < S * H / 2; i++) {
      I_R = a_x /// D;
      J_R = a_y /// D;
      i_R = I_R /// P;
```

```
        jR = JR /// P;
        Øx = IR – (iR * P);
        Øy = JR – (jR * P);
        new_ref[i, j] = filter(iR, jR, Øx, Øy);
        ax += axi;
        ay += ayi;
}
```

where all the variables used in this code are integer variables and `new_ref[i, j]` indicates the sample generated for column *i* and row *j* in the resampled reference picture. According to the definition of the parameters, all the divisions in this code can be replaced by binary shift operations. For example, when *P* = 16:

```
IR = ax /// D;
JR = ay /// D;
iR = IR /// P;
jR = JR /// P;
Øx = IR – (iR  * P);
Øy = JR – (jR  * P);
```

can be rewritten, assuming that a_x, a_y, I_R, and J_R are binary-coded integer variables in two's complement representation, as:

```
IR = ax >> (m+2);
JR = ay >> (m+2);
iR = IR >> 4;
jR = JR >> 4;
Øx = IR & 15;
Øy = JR & 15;
```

where ">> N_{shift}" denotes a right arithmetic binary shift by N_{shift} bits (N_{shift} is a positive integer), and "&" denotes a bit-wise AND operation.

P.5 Factor-of-4 resampling

Factor-of-4 resampling, which converts both the horizontal and vertical size of the picture by a factor of 2 or $\frac{1}{2}$, is a special case of the resampling algorithm described in P.3. The simplified description of the resampling algorithm for this special case is provided in this subclause.

The value of the parameter RCRPR used in Figures P.4 to P.6 is determined by the Rounding Type (RTYPE) bit in MPPTYPE (see 5.1.4.3) as described in P.3. Additionally, "/" in the figures indicates division by truncation.

P.5.1 Factor-of-4 upsampling

The pixel value interpolation method used in factor-of-4 upsampling for internal pixels are shown in Figure P.4. By assuming the existence of pixels outside the picture according to the selected fill mode (see P.2.3 and P.2.4), the same interpolation method is applied for boundary pixels. The interpolation method for boundary pixels when *clip* is selected as the fill mode is shown in Figure P.5. Since precise factor-of-4 upsampling requires x- and y-displacements with at least

$\frac{1}{4}$-pixel accuracy, the Warping Displacement Accuracy (WDA) field specified in P.2.1 must be set to "11" or the resampling must be implicitly invoked in order to use this upsampling method.

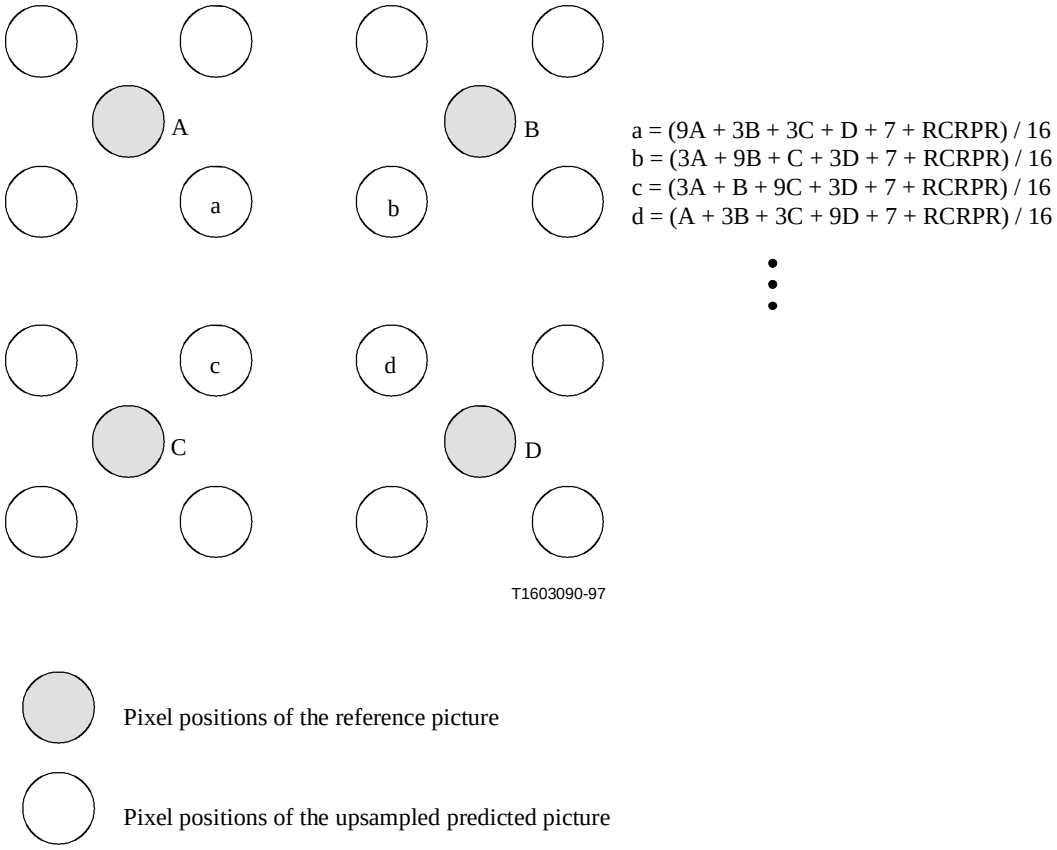


Figure P.4/H.263 – Factor-of-4 upsampling for pixels inside the picture

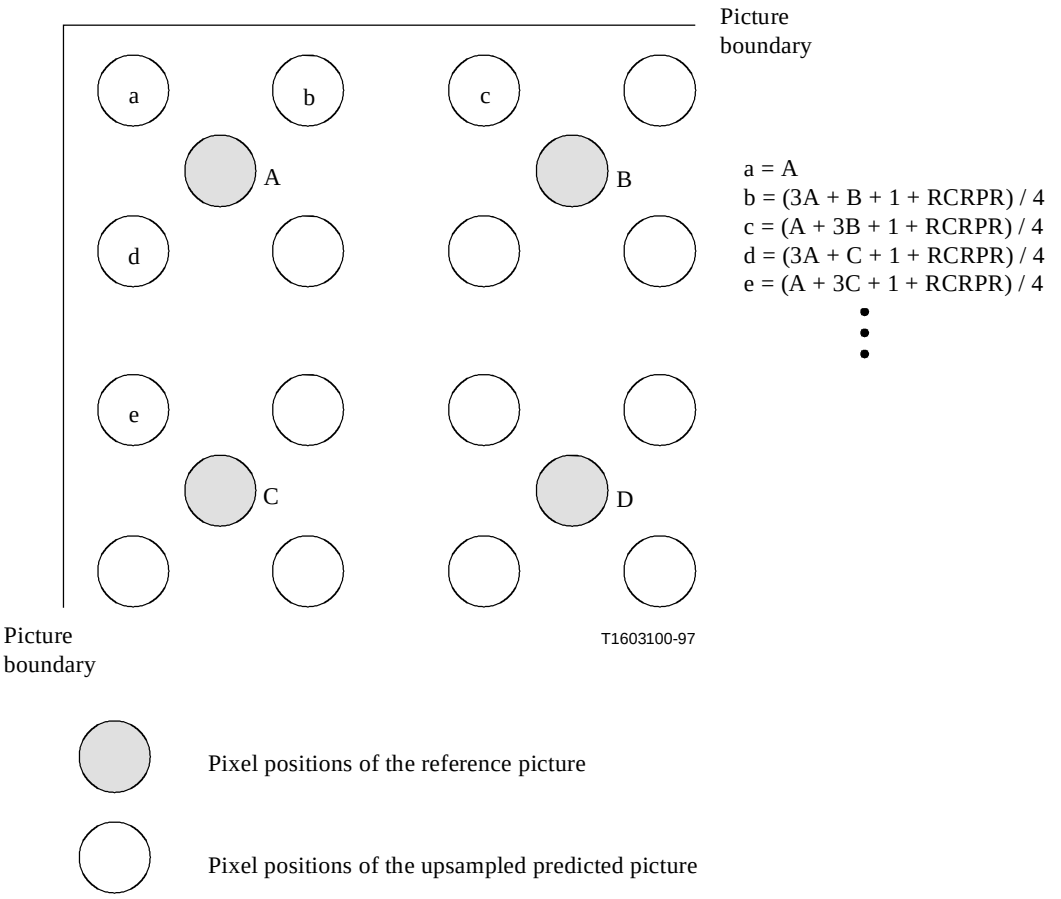


Figure P.5/H.263 – Factor-of-4 upsampling for pixels at the picture boundary (fill mode = *clip*)

P.5.2 Factor-of-4 downsampling

The pixel value interpolation method for factor-of-4 downsampling is shown in Figure P.6. Since x- and y-displacements with $\frac{1}{2}$-pixel accuracy is sufficient for precise factor-of-4 downsampling, both "10" and "11" are allowed as the value of the displacement accuracy (WDA) field (if present) specified in P.2.1.

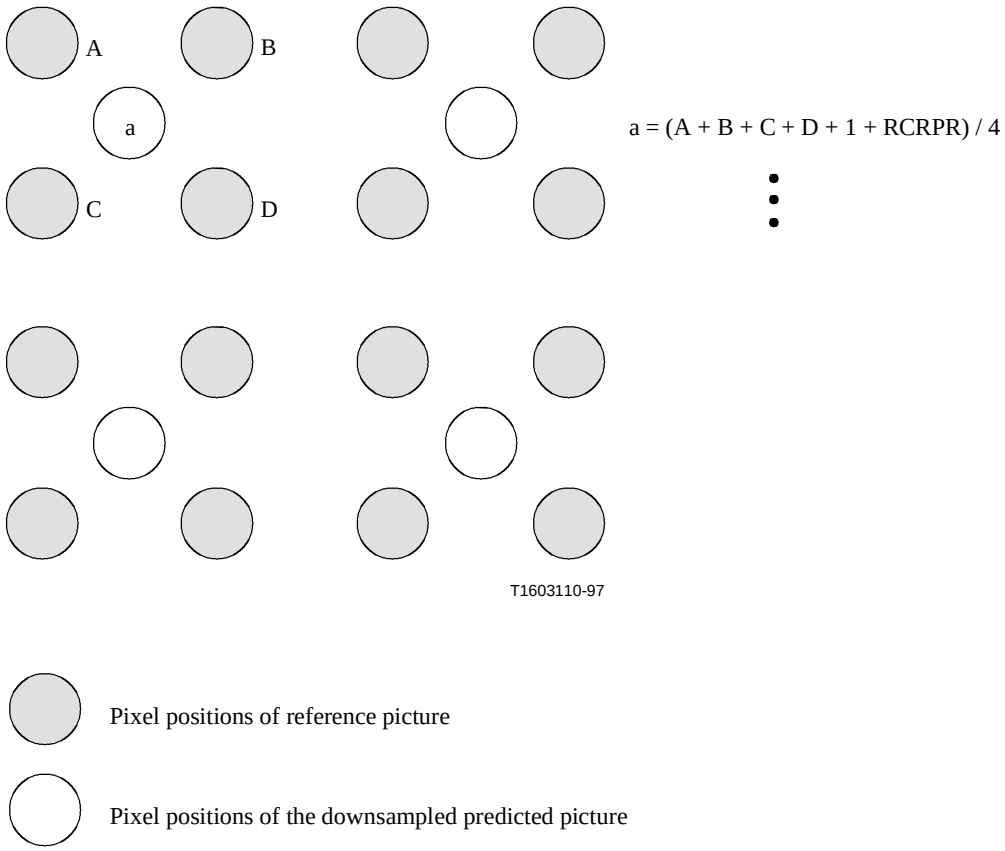


Figure P.6/H.263 – Factor-of-4 downsampling

ANNEX Q

Reduced-Resolution Update mode

Q.1 Introduction

This annex describes an optional Reduced-Resolution Update mode of this Recommendation. The capability of this mode is signaled by external means (for example, Recommendation H.245). The use of this mode is indicated in the PLUSPTYPE field of the picture header.

The Reduced-Resolution Update mode is expected to be used when encoding a highly active scene, and provides the opportunity to increase the coding picture rate while maintaining sufficient subjective quality. This mode allows the encoder to send update information for a picture that is encoded at a reduced resolution, while preserving the detail in a higher resolution reference image to create a final image at the higher resolution.

The syntax of the bitstream in this mode is identical to the syntax for coding without the mode, but the semantics, or interpretation of the bitstream is somewhat different. In this mode, the portion of the picture covered by a macroblock is twice as wide and twice as high. Thus, there is approximately one-quarter the number of macroblocks as there would be without this mode. Motion vector data also refers to blocks of twice the normal height and width, or 32 × 32 and 16 × 16 instead of the normal 16 × 16 and 8 × 8. On the other hand, the DCT or texture data should be thought of as describing 8 × 8 blocks on a reduced-resolution version of the picture. To produce the final picture, the texture data is decoded at a reduced resolution and then up-sampled to the full resolution of the picture.

After upsampling, the full resolution texture image is added to the (already full resolution) motion compensated image to create the image for display and further reference.

In this mode, a picture which has the horizontal size H and vertical size V as indicated in the picture header is created as a final image for display.

In this mode, a referenced picture used for prediction and created for further decoding has a horizontal size H_R and a vertical size V_R which are the same as in the default mode as defined in 4.1. That is, the H_R and V_R are:

$$H_R = ((H + 15) / 16) * 16$$

$$V_R = ((V + 15) / 16) * 16$$

where H and V are the horizontal size and the vertical size as indicated in the picture header and "/" is defined as division by truncation.

Then in this annex, the texture is coded at a reduced resolution with height and width of H_C and V_C where:

$$H_C = ((H_R + 31) / 32) * 32$$

$$V_C = ((V_R + 31) / 32) * 32$$

and "/" is defined as division by truncation.

If H_C and H_R, or V_C and V_R are not identical to each other, such as in the QCIF format, an extension of the referenced picture is performed , and the picture is decoded in the same manner as if the width and height are H_C and V_C. Then the resulting picture which is tiled by 32 * 32 macroblocks is cropped at the right and the bottom to the width H_R and height V_R, and this cropped picture is stored as a reference picture for further decoding.

If both H and V are the same as those of a resulting picture having the width H_C and the height V_C, this resulting picture is used for display. Otherwise, this resulting picture is further cropped to the size H * V, and the cropped picture here is used for display purpose only.

If the Temporal, SNR, and Spatial Scalability mode (Annex O) or Reference Picture Resampling mode (Annex P) is also used with this option, the source format of the current picture might be different from that of the reference picture. In this case, the resampling of the reference picture shall be performed before decoding.

NOTE – This mode can be used with the Reference Picture Selection mode (see Annex N) without modification, since the reference picture (after possible resampling by the Reference Picture Resampling mode) has the same size as indicated in the current picture header when this option is used.

Q.2 Decoding procedure

Figure Q.1 shows the block diagram of the block decoding in the Reduced-Resolution Update mode.

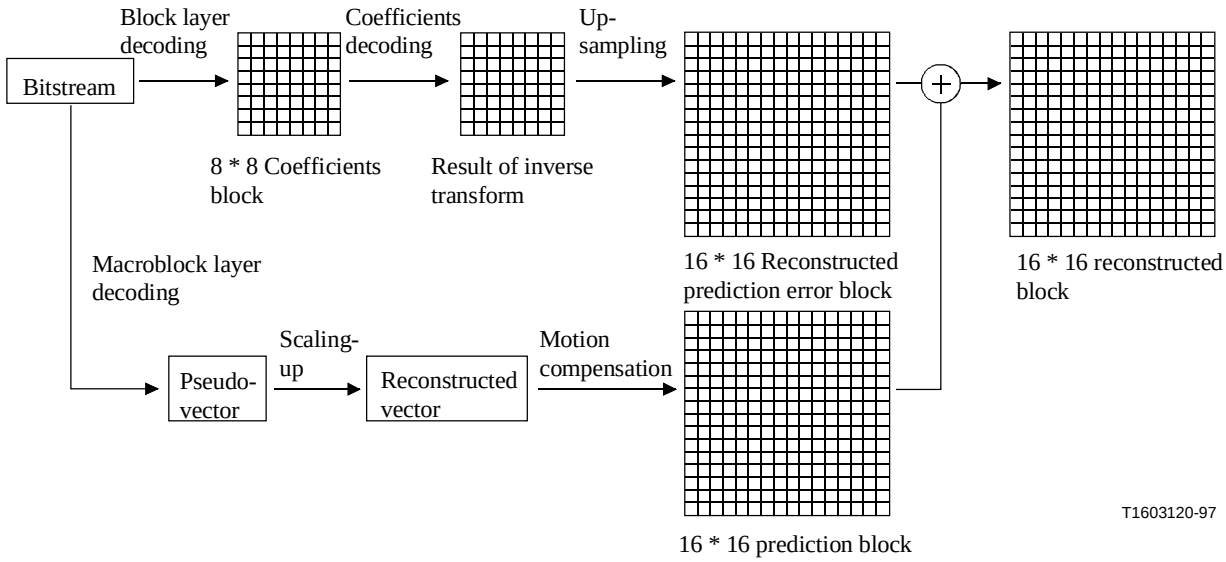


Figure Q.1/H.263 – Block diagram of block decoding in Reduced-Resolution Update mode

The decoding procedure is given in the following subclauses:

Q.2.1 Reference preparation

In some cases, the available reference picture has different size from H_C and V_C. Then, the reference picture shall be converted before decoding procedure according to Q.2.1.1 or Q.2.1.2.

Q.2.1.1 Reference Picture Resampling

If the Temporal, SNR, and Spatial Scalability mode (Annex O) or Reference Picture Resampling mode (Annex P) is also used with this option, the source format of the current picture might be different from that of the reference picture. In this case, the resampling of the reference picture according to each annex shall be performed first.

Q.2.1.2 Extension of Reference Picture

If H_R or V_R is not divisible by 32, such as with the QCIF format, the reference picture is extended. The detailed procedure for this extension is defined in Q.3.

Q.2.2 Macroblock layer decoding

Decoding can be thought of as operating on "enlarged" blocks of size 32 × 32 in luminance and 16 × 16 in chrominance. The texture and motion data for each enlarged block is decoded to create a 32 × 32 motion block and a 32 × 32 texture block, as described in Q.2.2.1 and Q.2.2.2 respectively. These motion and texture blocks are then added as described in Q.2.2.3.

Q.2.2.1 Motion compensation

First, each component of the macroblock motion vector (or four macroblock motion vectors) is formed from MVD (and possibly $MVD_{2\text{-}4}$). If in Improved PB-Frames mode, MV_F and MV_B for B-Picture is also formed from MVDB. The detailed procedure for this motion vector formation is defined in Q.4. If the current picture mode is a B-picture or EP-picture, the motion vector for forward and backward are also obtained according to Q.4.

The motion vector for the two chrominance blocks of the macroblock is obtained from the macroblock motion vector according to 6.1.1. If either the Advanced Prediction mode or the Deblocking Filter mode is in use and thus four motion vectors are defined for the macroblock, the motion vector for both chrominance blocks is obtained from the four motion vectors according to F.2. If in the Improved PB-frames mode, the creation of the chrominance vector is specified in Annex M. If a B-picture or EP-picture is used, the creation of the chrominance vector is specified in Annex O.

Then, a prediction is formed from the motion vector for an INTER macroblock. Four 16 × 16 luminance prediction blocks are obtained from the macroblock motion vector, and two 16 × 16 chrominance prediction blocks are obtained from the chrominance motion vector. For interpolation for subpixel prediction, refer to 6.1.2. If Advanced Prediction mode is also used, an enlarged overlapped motion compensation is performed to obtain four 16 × 16 luminance prediction blocks using enlarged weighting matrices, for which a detailed procedure is defined in Q.5. If the current picture mode is an Improved PB-frame, B-picture, or EP-picture, the prediction is obtained according to the other relevant annexes, except that the size of the predicted blocks is 16 × 16 instead of 8 × 8.

Q.2.2.2 Texture decoding

First, the bitstream of the block layer is decoded according to 5.4. Then, coefficients are decoded and the 8 × 8 reduced-resolution reconstructed prediction error blocks are obtained as the result of inverse transform according to 6.2.

Then the 16 × 16 reconstructed prediction error blocks are obtained by upsampling the 8 × 8 reduced-resolution reconstructed prediction error blocks. For the creation of the edge pixels in each 16 × 16 reconstructed prediction error block, only the pixels which belong to the corresponding block are used. The detailed procedure is defined in Q.6.

Q.2.2.3 Reconstruction of block

For each luminance and chrominance block, a summation of the prediction and prediction error is performed. The procedure is identical to 6.3.1, except that the size of blocks is 16 × 16 instead of 8 × 8. Then the clipping is performed according to the 6.3.2.

Then, a block boundary filter is applied to the boundary pixels of the 16 × 16 reconstructed blocks. The detailed procedure is described in Q.7.

Q.2.3 Picture store

If both H_R and V_R are divisible by 32, such as with the CIF format, the resulting picture reconstructed as described in Q.2.2 is stored as a reference picture as it is for further decoding. Otherwise, such as the QCIF format, the reconstructed picture which is just covered with 32 × 32 macroblocks is cropped at the right and the bottom to the width H_R and height V_R, and this cropped picture is stored as a reference picture for further decoding.

Q.2.4 Display

If both H and V are the same as H_C and V_C, the resulting picture in Q.2.2 is used for display purposes as it is. Otherwise, this resulting picture is further cropped to the size $H \times V$, and the cropped picture is used for display purpose only.

Q.3 Extension of referenced picture

If H_R or V_R is not divisible by 32, such as with the QCIF format, the extension of the referenced picture is performed before decoding macroblock/block layer. The width and the height of the extended reference picture for luminance are the next larger size that would be divisible by 32, and those for chrominance are the next larger size that would be divisible by 16.

NOTE – The width and height of the referenced picture in the default mode are always extended to be divisible by 16 even if the picture format has a width or height that is not divisible by 16, because the picture shall be decoded as if the width or height had the next larger size that would be divisible by 16. See 4.1.

If neither Unrestricted Motion Vector mode, Advanced Prediction mode nor Deblocking Filter mode is used with this option, the extended pixels can be arbitrary values, because the extended pixels will be never used as a reference pixels of the decoded picture to be reconstructed and displayed.

If either Unrestricted Motion Vector mode, Advanced Prediction mode or Deblocking Filter mode is also used with this option, the extension of the referenced picture is performed by duplicating the edge pixel of the referenced picture, in order to ensure the decoding when motion vectors point outside the right and bottom edge of the picture.

For example, if the Reduced-Resolution Update mode is used for a QCIF, the width of the referenced picture is 176 and the height is 144, which are not divisible by 32. In order to cover a QCIF picture with 32 × 32-sized macroblocks, the number of macroblock row should be 6, and the number of macroblock column should be 5. Therefore, the width of the extended referenced picture is 192 and the height is 160.

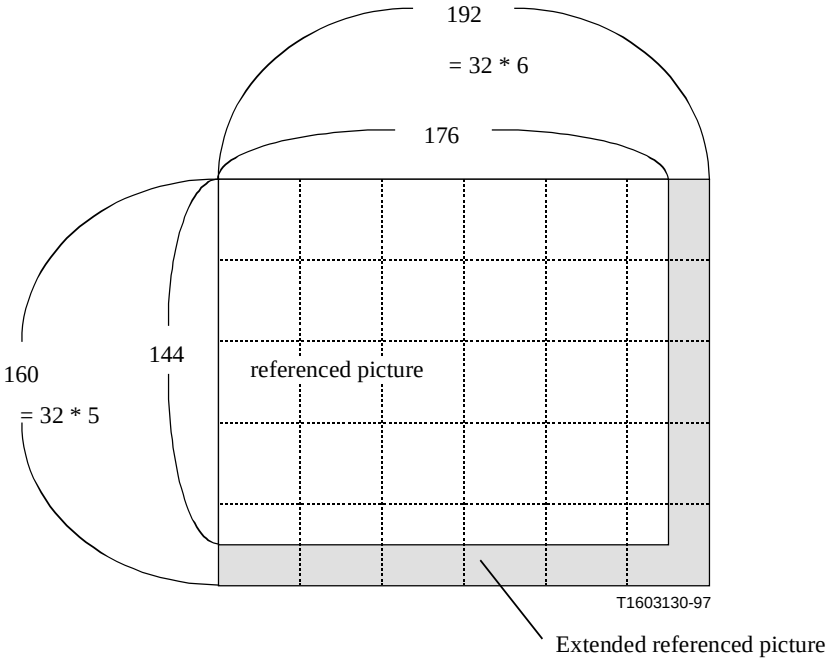


Figure Q.2/H.263 – Extension of referenced picture for QCIF picture size

The extension of the referenced picture in QCIF is illustrated in Figure Q.2. The extended referenced picture for luminance is given by the following formula:

$$R_{RRU}(x, y) = R(x', y')$$

where:

x, y = spatial coordinates of the extended referenced picture in the pixel domain,

x', y = spatial coordinates of the referenced picture in the pixel domain,

$R_{RRU}(x, y)$ = pixel value of the extended referenced picture at (x, y),

R(x´, y´) = pixel value of the referenced picture at (x´, y´),

$$x' = \begin{cases} 175 & \text{if } x > 175 \text{ and } x < 192 \\ x & \text{otherwise,} \end{cases}$$

$$y' = \begin{cases} 143 & \text{if } y > 143 \text{ and } y < 160 \\ y & \text{otherwise.} \end{cases}$$

The referenced pictures for chrominance is also extended in the same manner.

Q.4 Reconstruction of motion vectors

In Reduced-Resolution Update mode, the motion vector range is enlarged to approximately double size in both the horizontal and vertical directions. In order to realize an enlarged range using the VLC for MVD defined in Table 14, each vector component is restricted to be only half-pel or zero value. Therefore the range of each motion vector component is [–31.5, 30.5] in the default Reduced-Resolution Update mode. If the Unrestricted Motion Vector mode is used, the vector range [–limit, limit] as defined in D.2 applies to the pseudo-motion vectors and translates into [–(2 * limit–0.5), 2 * limit–1.5] for the motion vectors. For CIF this means that the pseudo-motion vector range is [–32, 31.5] and the motion vector range is [–63.5, 62.5]. If the UUI field is set to "01", the motion vectors are unlimited. However, the motion vectors (not just the pseudo-motion vectors) are always limited to point not more than 15 pixels outside the coded area as described in D.1.1. Figure Q.3 illustrates the possible positions of macroblock motion vector or four motion vector predictions around the (0, 0) vector value. The dashed lines indicate the integer coordinates.

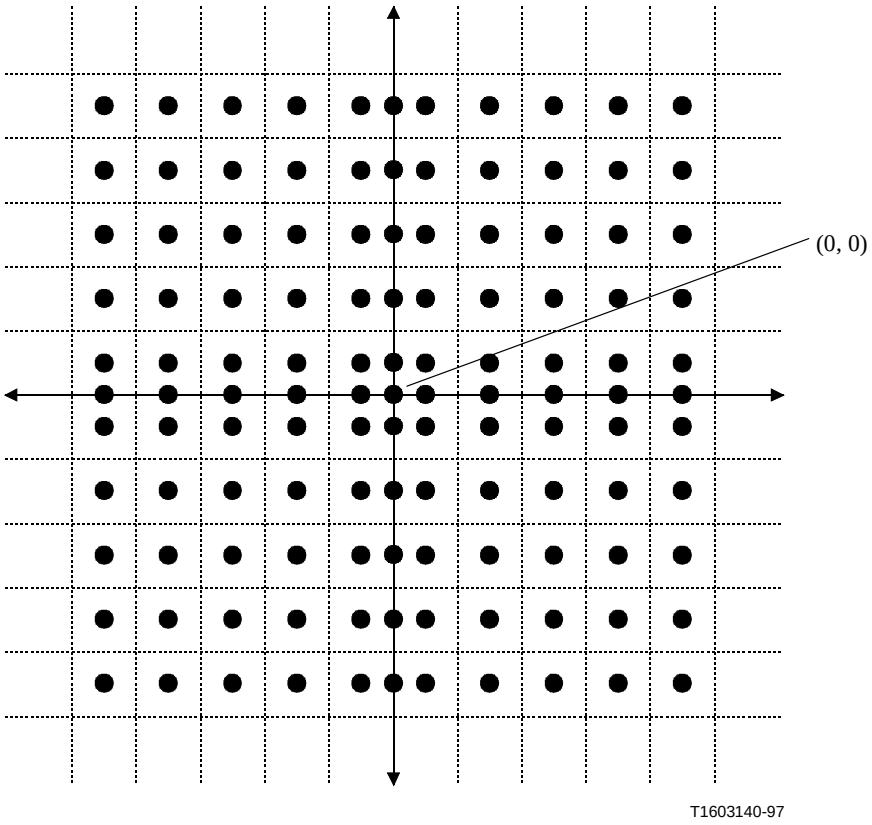


Figure Q.3/H.263 – Reconstruction of motion vector

For the macroblock using differential motion vectors in a B-picture, the motion vector for forward and backward prediction are obtained independently. In the Reduced-Resolution Update mode, the motion vector component MV_c for a luminance block is reconstructed from MVD and $MVD_{2\text{-}4}$ as follows:

1) Pseudo-prediction vector component **pseudo-P_C** is created from the prediction vector component **P_C**.

   pseudo-P_C = 0 if $P_C = 0$

   pseudo-P_C = sign(P_C) * (|P_C| + 0.5) / 2.0 if $P_C \neq 0$

   "/" indicates floating-point division (without loss of accuracy). The prediction vector component P_C is defined as the median value of the vector components MV1, MV2 and MV3 as defined in 6.1.1 and F.2.

2) Pseudo-macroblock vector component **pseudo-MV_C** is obtained by adding the motion vector differences MVD (and $MVD_{2\text{-}4}$) from Table 14 to the pseudo-P_C.

   In the default Reduced-Resolution Update mode, the value of **pseudo-MV_C** is restricted to the range [−16, 15.5]. Only one of the pair will yield a pseudo-MV_C falling within the permitted range. The procedure is performed in the similar way as defined in 6.1.1.

   If the Unrestricted Motion Vector mode is also used with the Reduced-Resolution Update mode, **pseudo-MV_C** is obtained by adding the motion vector differences MVD (and $MVD_{2\text{-}4}$) from Table D.3.

   If four motion vectors are present, the procedure is performed in the similar way as defined in F.2.

3) Motion vector component $\mathbf{MV_C}$ is obtained from pseudo-MV_C in the following formula:

$MV_C = 0$ if pseudo-$MV_C = 0$

$MV_C = \text{sign}(\text{pseudo-}MV_C) * (2.0 * |\text{pseudo-}MV_C| - 0.5)$ if pseudo-$MV_C \neq 0$

As a result, each vector component is restricted to have a half-integer or zero value, and the range of each motion vector component is enlarged to approximately twice the pseudo-motion vector range.

4) If the current picture mode is an Improved PB-frame, or when the MBTYPE indicates the direct mode in a B-picture, the motion vector components $\mathbf{MV_F}$ and/or $\mathbf{MV_B}$ for forward and backward prediction are created.

First, pseudo-motion vector components **pseudo-$\mathbf{MV_F}$** and/or **pseudo-$\mathbf{MV_B}$** are calculated based on the rules for the prediction modes defined in Annexes O or M.

In the case of the Bidirectional prediction in the Improved PB-frames mode (see M.2.1) or when the MBTYPE indicates the direct mode in a B-picture (see O.5.2), pseudo-MV_F and pseudo-MV_B are calculated from **pseudo-$\mathbf{MV_D}$** and **pseudo-$\mathbf{MV_C}$** assuming that pseudo-MV_D is zero and pseudo-MV_C is MV, as defined in Annexes G and M.

In the case of forward prediction in the Improved PB-frames mode (see M.2.2), **pseudo-$\mathbf{MV_{DB}}$** is obtained by decoding the variable length code MVDB according to Table 13. Then, pseudo-MV_F is obtained by adding the pseudo-MV_{DB} to the **pseudo-Predictor**. In order to form the pseudo-Predictor, the predictor obtained according to the procedure defined in M.2.2 is converted to the pseudo-Predictor vector according to the formula defined in item 1) of this subclause.

In the case of backward prediction in the Improved PB-frames mode (see M.2.3), pseudo-MV_B is set to zero.

Then, the motion vectors MV_F and/or MV_B for forward and backward prediction are obtained from pseudo-MV_F and/or pseudo-MV_B according to the formula defined in item 3) of this subclause.

Q.5 Enlarged overlapped motion compensation for luminance

If Advanced Prediction mode is also used with Reduced-Resolution Update mode, enlarged matrices of weighting values are used to perform the overlapped motion compensation. Except that the size of each block and weighting matrices is 16 × 16, the procedure of the creation of each prediction block is identical to the description in F.3.

The enlarged matrices of weighting values for the 16 × 16 luminance prediction are given in Figures Q.4, Q.5, and Q.6.

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

Figure Q.4/H.263 – Weighting values, H_0, for prediction with motion vector of current 16 × 16 luminance block

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

Figure Q.5/H.263 – Weighting values, H_1, for prediction with motion vector of current 16 × 16 luminance blocks on top or bottom of current 16 × 16 luminance block

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

Figure Q.6/H.263 – Weighting values, H_2, for prediction with motion vector of 16 × 16 luminance blocks to the left or right of current 16 × 16 luminance block

Q.6 Upsampling of the reduced-resolution reconstructed prediction error

The 16 × 16 reconstructed prediction error block is obtained by upsampling the 8 × 8 reduced-resolution reconstructed prediction error block. In order to realize a simple implementation, filtering is closed within a block which enables to perform an individual upsampling on block basis. Figure Q.7 shows the positioning of samples. The upsampling procedure for the luminance and chrominance pixels which are inside the 16 × 16 reconstructed prediction error blocks is defined in Q.6.1. For the creation of the luminance and chrominance pixels which are at the boundary of 16 × 16 reconstructed prediction error block, the procedure is defined in Q.6.2. Chrominance blocks as well as luminance blocks are up-sampled. The symbol "/" in Figures Q.8 and Q.9 indicates division by truncation.

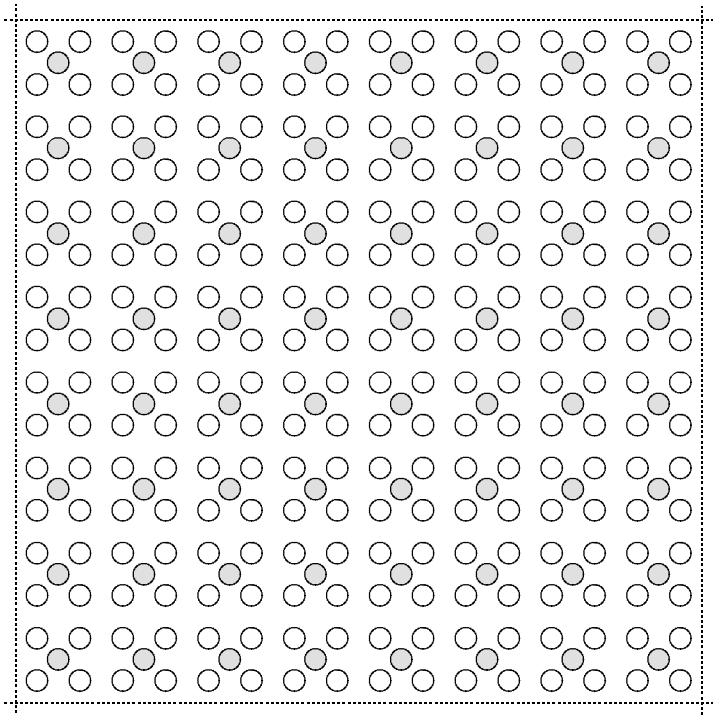


Figure Q.7/H.263 – Positioning of samples in 8 × 8 reduced-resolution reconstructed prediction error block and 16 × 16 reconstructed prediction error block

Q.6.1 Upsampling procedure for the pixels inside a 16 × 16 reconstructed prediction error block

The creation of reconstructed prediction error for pixels inside block is described in Figure Q.8. "/" indicates division by truncation.

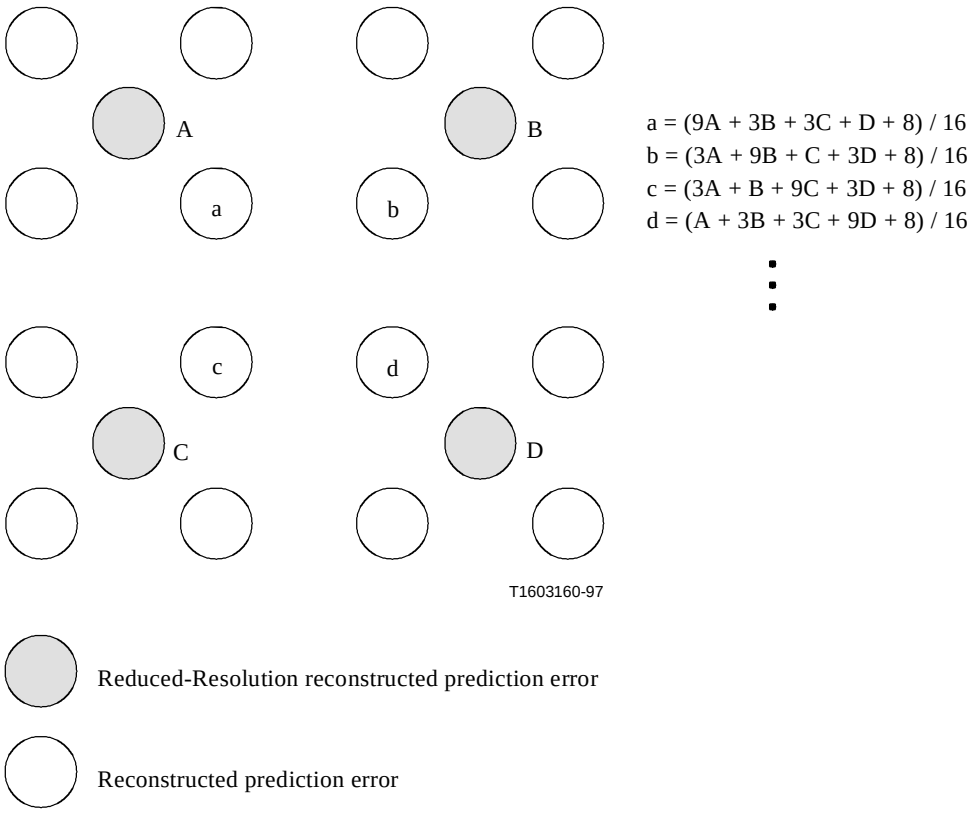


Figure Q.8/H.263 – Creation of reconstructed prediction error for pixels inside block

Q.6.2 Upsampling procdedure for the pixels at the boundary of 16 × 16 reconstructed prediction error block

The creation of reconstructed prediction error for pixels of a 16 × 16 block is shown in Figure Q.9.

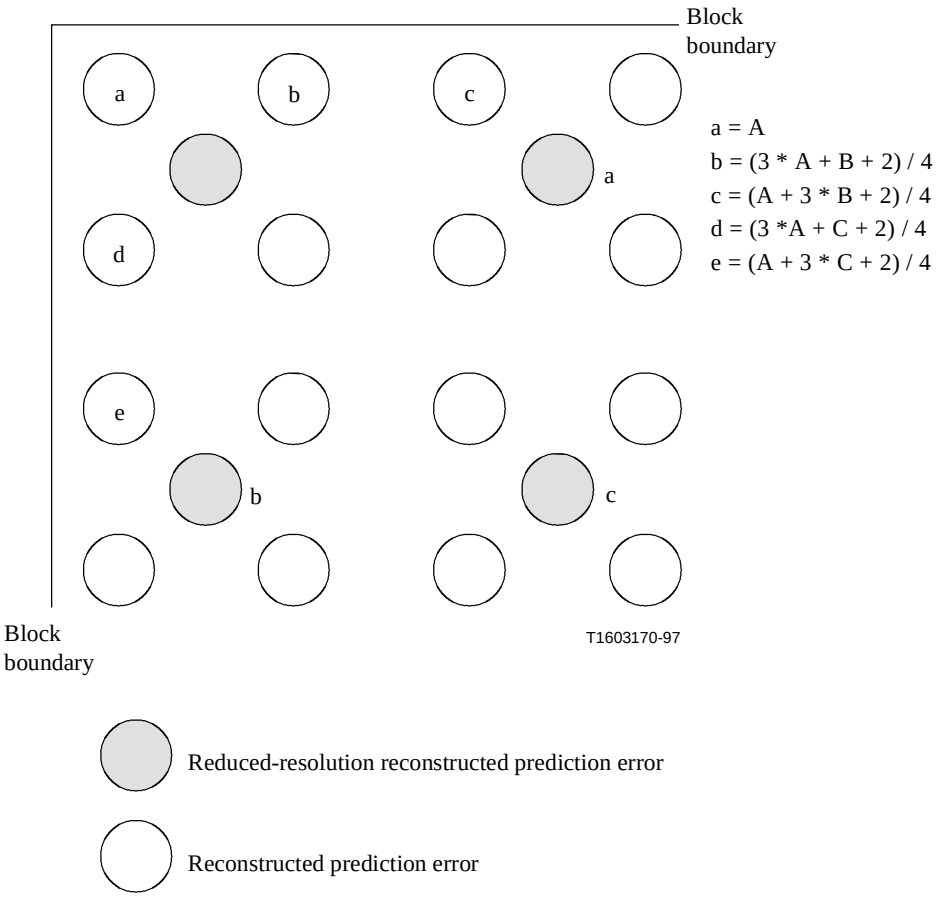


Figure Q.9/H.263 – Creation of reconstructed prediction error for pixels at the block boundary

Q.7 Block boundary filter

The filter operations are performed along the edges of the 16 × 16 reconstructed blocks at the encoder as well as on the decoder side. There are two alternative of filtering, depending on whether Deblocking Filter mode is used or not.

The default filtering in Reduced-Resolution Update mode is performed according to Q.7.1.

If Deblocking Filter mode is also used with Reduced-Resolution Update mode, the filtering is performed according to Q.7.2.

In both cases, filtering is performed on the complete reconstructed image data before storing the data in the picture store for future prediction. No filtering is performed across picture edges, slice edges in Slice Structured mode (see Annex K), or GOB boundaries having GOB headers present in Independent Segment Decoding mode (see Annex R). Chrominance as well as luminance data is filtered.

Q.7.1 Definition of the default block boundary filter

In the Reduced-Resolution Update mode, the default filtering is performed according to this subclause.

If A and B are two pixel values on a line – horizontal or vertical – of the reconstructed picture, and A belongs to one 16 × 16 block called block1 whereas B belongs to a neighboring 16 × 16 block called block2 which is to the right or below of block1. Figure Q.10 shows examples for the position of these pixels.

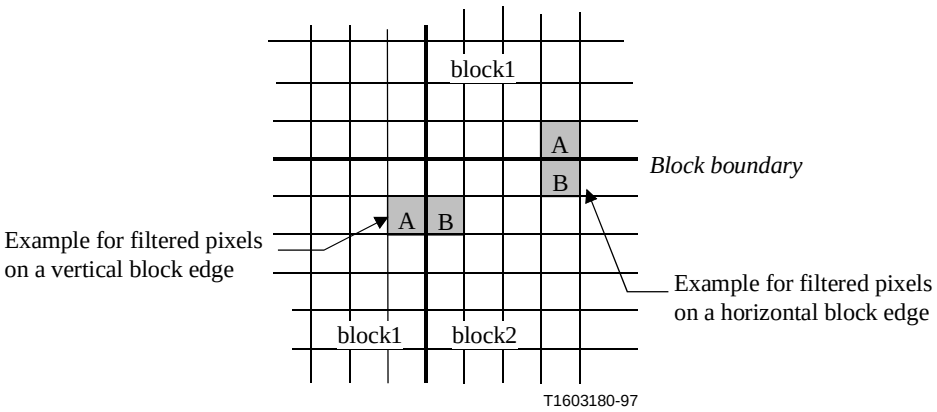


Figure Q.10/H.263 – Default block boundary filter

One of the following conditions must be fulfilled in order to turn the filter on for a particular edge:

- block1 belongs to a coded macroblock (COD==0 || MB-type == INTRA); **or**
- block2 belongs to a coded macroblock (COD==0 || MB-type == INTRA).

A shall be replaced by A1 and B shall be replaced by B1. "/" indicates division by truncation.

$$A1 = (3 * A + B + 2) / 4$$

$$B1 = (A + 3 * B + 2) / 4$$

The order of edges where filtering is performed is identical to the description provided in J.3.

Q.7.2 Definition of the block boundary filter when Deblocking Filter mode is used

If the Deblocking Filter mode (see Annex J) is used with the Reduced-Resolution Update mode, the filtering which is defined in Annex J with one modification is performed on the boundary pixels of 16×16 luminance and chrominance blocks, in place of the filtering described in Q.7.1. The one modification of the filtering in Annex J is that the parameter STRENGTH is given the value of positive infinity. This implies that the function UpDownRamp(x, STRENGTH) defined in J.3 becomes a linear function of x.

As a result, the procedure of the deblocking filter described in J.3 is redefined in the following manner:

B1 = clip(B + d1)

C1 = clip(C – d1)

A1 = A – d2

D1 = D + d2

d1 = (A–4B+4C–D) / 8

d2 = clipd1((A – D) / 4, d1/2)

ANNEX R

Independent Segment Decoding mode

R.1 Introduction

This annex describes the optional Independent Segment Decoding mode of this Recommendation, which allows a picture to be decoded without the presence of any data dependencies across slice boundaries or across GOB boundaries having non-empty GOB headers. The use of this mode is indicated in the PLUSPTYPE field of the picture header. The capability to use this optional mode is negotiated by external means (for example, Recommendation H.245).

When the use of this mode is indicated, the video picture segment boundaries (as defined by the boundaries of the slices or the upper boundaries of the GOBs for which GOB headers are sent, or the boundaries of the picture, whichever bounds a region in the smallest way) are treated as picture boundaries when decoding, including the treatment of motion vectors which cross those boundaries (which result in boundary extrapolation when the Unrestricted Motion Vector mode, the Advanced Prediction mode, the Deblocking Filter mode, or the Temporal, SNR, and Spatial Scalability mode are in use, and which are prohibited when none of those optional modes are in use).

R.2 Mode operation

A video picture segment is defined by the following.

If the Slice Structured mode (see Annex K) is not in use, then one GOB or a plural number of consecutive GOBs forms one video picture segment. The location of the top of each video picture segment is indicated by the presence of a non-empty GOB header for which the border of the video picture segment lies just above the macroblocks in the GOB for which a header is present, or the top of the picture, whichever is lower. The location of the bottom of each video picture segment is defined by the top of the next video picture segment, or the bottom of the picture, whichever is uppermost.

If the Slice Structured mode (see Annex K) is in use, then each slice forms one video picture segment.

In the Independent Segment Decoding mode, each video picture segment is decoded with complete independence from all other video picture segments, and is also independent of all data outside the same video picture segment location in the reference picture(s). This includes:

1) no use of motion vectors outside of the current video picture segment for motion vector prediction (as in 6.1.1);
2) no use of motion vectors outside of the current video picture segment as remote motion vectors for overlapped block motion compensation when the Advanced Prediction mode is in use (see F.3);
3) no deblocking filter operation across video picture segment boundaries (see J.3);
4) no use of motion vectors which reference data outside the current video picture segment unless the Unrestricted Motion Vector mode (see Annex D), the Advanced Prediction mode (see Annex F), the Deblocking Filter mode (see Annex J) or the Temporal, SNR, and Spatial Scalability mode (see Annex O) are in use; in which case the borders of the current video picture segment in the prior picture are extrapolated as described in Annex D to form predictions of the pixels which reference the out-of-bounds region;

5) no bilinear interpolation across the boundaries of the ¼-size or ½-size region corresponding to the current video picture segment for upward prediction in spatial scalability EI- and EP-pictures (as defined in Annex O);

6) when the Reduced-Resolution Update mode (see Annex Q) is in use, no block boundary filter operation across video picture segment boundaries;

7) no use of the Reference Picture Resampling mode with the Independent Segment Decoding mode.

R.3 Constraints on usage

Certain restrictions are placed on the use of other aspects of the video coding syntax when the Independent Segment Decoding mode is in use. These restrictions are made to prevent two pathological cases which otherwise would make operation of the Independent Segment Decoding mode difficult.

R.3.1 Constraint on segment shapes

In the use of the Slice Structured mode (Annex K) without the use of the Rectangular Slice submode (see K.1), there can arise cases in which the shape of a video picture segment may be non-convex (having "inside corners", or even comprising two distinct and separated regions of the picture).

Therefore, the Independent Segment Decoding mode shall not be used with the Slice Structured mode without the simultaneous use of the Rectangular Slice submode of the Slice Structured mode (see Annex K). This constraint is mandated to prevent the need for difficult special-case treatment in order to determine how and when to perform extrapolation of each video picture segment.

R.3.2 Constraint on changes of segment shapes

If the shape of the video picture segments were allowed to change in any way from picture to picture in the bitstream, there could arise cases in which the bitstream would be difficult to decode. This is because in such cases the bitstream content is not sufficient to determine the shape of each video picture segment prior to the possible appearance of motion vectors in the bitstream which require knowledge of the video picture segment shape for proper interpretation.

Therefore, when the Independent Segment Decoding mode is in use, the video picture segmentation for all pictures and frames allowing temporal prediction (i.e. all P-, B-, and EP-pictures and all Improved PB-frames) shall be the same as that used in its temporal reference picture. Also, when the Independent Segment Decoding mode is in use, the video picture segmentation for all EI-pictures shall either be the same or shall differ only by sub-dividing the video picture segmentation used in its reference picture. Also, the Independent Segment Decoding mode shall not be used in any picture or frame which uses reference pictures (all picture types except INTRA) unless the Independent Segment Decoding mode is also used in all of the reference picture(s) for the current picture. As a result of this constraint, the shape of the video picture segments in the Independent Segment Decoding mode shall never change from picture to picture except as changed in I- and EI-pictures (and the manner in which EI-pictures can change segmentation is itself also somewhat constrained).

ANNEX S

Alternative INTER VLC mode

S.1 Introduction

This annex describes an optional Alternative INTER VLC mode of this Recommendation, which improves the efficiency of inter-picture coding when significant changes are evident in the picture. This efficiency improvement is obtained by allowing some VLC codes originally designed for INTRA pictures to be used for some INTER picture coefficients and CBPY data as well. The use of this mode is indicated in the PLUSPTYPE field of the picture header. The capability to use this optional mode is negotiated by external means (for example, Recommendation H.245). The mode contains two syntax alterations, one for the encoding of INTER coefficients and another for the encoding of the INTER CBPY values.

S.2 Alternative INTER VLC for coefficients

The concept behind the design of the INTRA VLC table of Annex I is to use the same codewords as in the original INTER VLC but with a different interpretation of LEVEL and RUN. The INTRA VLC is better suited in cases where there are many and/or large-valued coefficients.

The INTRA VLC is constructed so that codewords have the same value for LAST (0 or 1) in both the INTER and INTRA tables. The INTRA table is therefore produced by "reshuffling" the meaning of the codewords with the same value of LAST. Furthermore, for events with large |LEVEL| the INTRA table uses a codeword which in the INTER table has a large RUN. In INTER blocks having a large number of large-magnitude coefficients, it can sometimes be more efficient to use the INTRA table than the INTER table, and in some such cases the choice of the VLC table can be apparent to the decoder since decoding using the INTER table would result in RUN values so large as to indicate the presence of more than 64 coefficients for a block. Under these circumstances, the INTRA table can be used to improve the efficiency of INTER coding.

S.2.1 Encoder action

The encoder may use the INTRA VLC table for coding an INTER block whenever the decoder can detect its use – in other words, whenever decoding using the INTER VLC table would cause coefficients outside the 64 coefficients of a block to be addressed.

The encoder would normally choose to use the INTRA VLC table for coding an INTER block only when the above condition is satisfied and also when the INTRA VLC usage results in fewer bits than the INTER VLC for the same coefficient values. This will often be the case when there are many large coefficients, due to the way the INTRA VLC was produced (since the ordinary INTER VLC table contains long run-lengths for the same codewords in which the INTRA VLC table contains large coefficient amplitudes).

S.2.2 Decoder action

The decoding process is as follows:

1) The decoder first receives all coefficient codes of a block.
2) The codewords are then interpreted assuming that INTER VLC is used. If the addressing of coefficients stays inside the 64 coefficients of a block, the VLC decoding is finished.
3) If coefficients outside the block are addressed, the codewords shall be interpreted according to the INTRA VLC.

S.3 Alternative INTER VLC for CBPY

The INTER CBPY codewords (Table 13) are designed with the assumption that there are more Y blocks with all zero coefficients than there are with at least one non-zero coefficient. When both C_B and C_R blocks have at least one non-zero coefficient, i.e. $CBPC_5 = CBPC_6 = 1$, this assumption no longer holds. For this reason, when the Alternative INTER VLC mode is in use, the CBPY codewords as defined in Table 13 for INTRA macroblocks shall also be used for INTER macroblocks whenever $CBPC_5 = CBPC_6 = 1$.

ANNEX T

Modified Quantization mode

T.1 Introduction

This annex describes an optional Modified Quantization mode of this Recommendation, which modifies quantizer operation. The use of this mode is indicated in the PLUSPTYPE field of the picture header. The capability to use this optional mode is negotiated by external means (for example, Recommendation H.245).

This mode includes four key features:

1) The bit-rate control ability for encoding is improved by altering the syntax for the DQUANT field.

2) Chrominance fidelity is improved by specifying a smaller step size for chrominance than that for luminance data.

3) The range of representable coefficient values is extended to allow the representation of any possible true coefficient value to within the accuracy allowed by the quantization step size.

4) The range of quantized coefficient levels is restricted to those which can reasonably occur, to improve the detectability of errors and minimize decoding complexity.

T.2 Modified DQUANT Update

This mode modifies the semantics of the DQUANT field. With this mode, it is possible to use DQUANT either to modify QUANT by plus or minus a small amount, or to signal any specific new value for QUANT. The size of the small amount of modification depends on the current value of QUANT. By use of this mode, more flexible control of the quantizer step size can be specified in the DQUANT field.

The codeword for DQUANT in this mode is no longer a two-bit fixed length field. It is a variable length field which can either be two bits or six bits in length. Whether it is two or six bits depends on the first bit of the code. The description below is therefore split into two sections, depending on the first bit.

T.2.1 Small-step QUANT alteration

When the first bit of the DQUANT field is 1, only one additional bit is sent in DQUANT. The single additional bit is used to modify QUANT by a differential value. The change in the value of QUANT is dependent on the second bit of DQUANT and on the prior value of QUANT, as shown in Table T.1.

Example: If the previous value of QUANT is 29 and DQUANT is signalled with the codeword "11", then the differential value is +2, and thus the resulting new QUANT value is 31.

Table T.1/H.263 – Semantics of small-step QUANT alteration

	Change of QUANT	
Prior QUANT	**DQUANT = 10**	**DQUANT = 11**
1	+2	+1
2-10	–1	+1
11-20	–2	+2
21-28	–3	+3
29	–3	+2
30	–3	+1
31	–3	–5

T.2.2 Arbitrary QUANT selection

When the first bit of the DQUANT field is 0, five additional bits are sent in DQUANT. The following five bits represent a new QUANT as defined in 5.1.19.

Example: Regardless of the current value of QUANT, if DQUANT is signalled with the code word ’001111’, then the new value of QUANT is 15.

T.3 Altered quantization step size for chrominance coefficients

When the modified quantization mode is in use, the quantization parameter of the chrominance coefficients is different from the quantization parameter of the luminance. The luminance quantization parameter is signaled in the bitstream. It is called QUANT. When this mode is in use, a different quantization parameter termed QUANT_C is used for the inverse quantization of chrominance coefficients. The relation between QUANT and QUANT_C is given in Table T.2. If the Deblocking Filter mode (see Annex J) is in use, QUANT_C shall also be used for the application of the deblocking filter to the chrominance data. Whenever QUANT is discussed herein in any other context, it shall mean the quantization step size of the luminance.

Table T.2/H.263 – Relationship between QUANT and QUANT_C

Range of QUANT	Value of QUANT_C
1-6	QUANT_C = QUANT
7-9	QUANT_C = QUANT – 1
10-11	9
12-13	10
14-15	11
16-18	12
19-21	13
22-26	14
27-31	15

T.4 Modified coefficient range

When the Modified Quantization mode is in use, quantized DCT coefficients having quantization level magnitudes greater than 127 can be represented in the bitstream. This has two advantages:

1) Encoder performance is improved by allowing the true full range of possible coefficient values to be represented.

2) Encoder complexity is reduced by eliminating the need to increase the quantization step size upon encountering certain large coefficient values which would otherwise be unrepresentable.

It is possible that the correct value of a DCT coefficient prior to quantization in the encoder will have a magnitude as high as 2040. Thus, a range of –127 to +127 for LEVEL is insufficient to cover the entire range of possible coefficient values whenever the quantization parameter QUANT or QUANT_C is less than 8. The expanded coefficient range broadens the range of LEVEL to allow any true coefficient value to be more properly encoded.

When the Modified Quantization mode is in use, the meaning of the LEVEL field following an ESCAPE code (0000 011, as per 5.4.2) is altered. In this mode, rather than being forbidden, the bit sequence 1000 0000 is used to represent an EXTENDED-ESCAPE code. An AC coefficient of magnitude greater than 127 is represented by sending an EXTENDED-ESCAPE code, immediately followed by a fixed-length EXTENDED-LEVEL field of 11 bits. An extended coefficient value is encoded into the EXTENDED-LEVEL field by taking the least significant 11 bits of the two's-complement binary representation of LEVEL and cyclically rotating them to the right by 5-bit positions. This rotation is necessary to prevent start code emulation. The cyclic rotation is illustrated in Figure T.1.

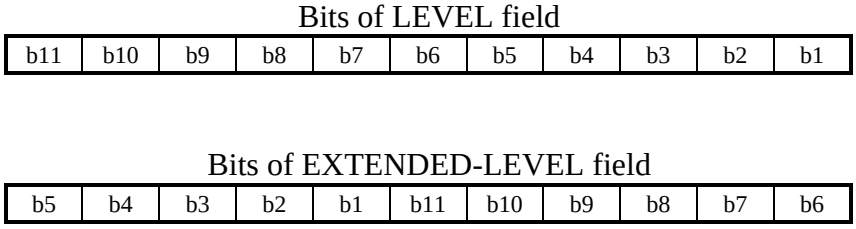


Figure T.1/H.263 – Cyclic rotation of coefficient representation

T.5 Usage restrictions

When the Modified Quantization mode is in use, certain restrictions are placed on the encoded coefficient values. This has several benefits:

1) The detectability of bit errors is improved by prohibiting certain unreasonable coefficient values, thus allowing these values to be recognized as bit errors by the decoder; and

2) Decoder complexity is reduced by reducing the wordlength necessary for inverse quantization prior to clipping.

3) Start-code emulation is prevented for coefficients encoded using the EXTENDED-ESCAPE mechanism described in T.4.

These restrictions are as follows. When the Modified Quantization mode is in use:

1) For any coefficient, the reconstruction level magnitude |REC| produced by the inverse quantization process described in 6.2.1 using the current value of QUANT or QUANT_C as appropriate, and the encoded value of LEVEL, shall be less than 4096. This additional restriction applies to all coefficients, regardless of whether the coefficient is sent using the EXTENDED-ESCAPE mechanism or not.

2) The bitstream shall not use either the normal ESCAPE code or the EXTENDED-ESCAPE code to encode a combination of LAST, RUN, and LEVEL for which there exists a codeword entry in the applicable VLC table, which is either Table 16 (see 5.4.2) or Table I.2 (see I.3).

3) The EXTENDED-ESCAPE code shall be used only when the quantization parameter for the coefficient (QUANT or QUANT_C) is less than eight (8).

4) The EXTENDED-ESCAPE code shall be used only when it is followed by an EXTENDED-LEVEL field representing a value of LEVEL which is outside of the range −127 to +127.

APPENDIX I

Error tracking

I.1 Introduction

This appendix describes a method to recover efficiently after transmission errors if erroneous MBs are reported via a feedback channel to the encoder. The capability of sending and processing feedback information is signalled via external means (for example, by Recommendation H.245). Furthermore, format and content of the feedback message are defined externally (for example, by Recommendation H.245).

I.2 Error tracking

Because INTRA coding stops temporal error propagation, it should be used for macroblocks which are severely affected by transmission errors. This requires that the location and extent of image artefacts can be made available to the encoder. The following algorithm provides an estimated error distribution based on feedback information received by the encoder. It considers spatial error propagation caused by motion-compensated prediction as well as the delay until the reception of the feedback message. The algorithm illustrates one possible approach to evaluate feedback messages for spatio-temporal error tracking. Other algorithms are possible.

Assume N macroblocks within each frame enumerated mb = 1...N from top-left to bottom-right. Let $\{n_{err}, mb_{first}, mb_{last}\}$ be the feedback message to the encoder, where $mb_{first} \leq mb \leq mb_{last}$ indicates a set of erroneous macroblocks in frame n_{err}.

To evaluate the feedback message, the encoder must continuously record information during the encoding of each frame. First, the initial error $E_0(mb, n)$ that would be introduced by the loss of macroblock mb in frame n needs to be stored. Assuming a simple error concealment where erroneous macroblocks are treated as not coded, $E_0(mb, n)$ is computed as the Summed Absolute Difference (SAD) of macroblock mb in frame n and n – 1. Second, the number of pixels transferred from macroblock mb_{source} in frame n – 1 to macroblock mb_{dest} in frame n is stored in dependencies $d(mb_{source}, mb_{dest}, n)$. These dependencies are derived from the motion vectors.

Assume that a feedback message arrives before frame n_{next} is encoded, such that $n_{next} > n_{err}$. Then, the estimated error E(mb, n_{err}) in macroblock mb and frame n_{err} is initialized as:

$$E(mb, n_{err}) = \begin{cases} E_0\ (mb, n_{err}) & \text{for } mb_{first} \le mb \le mb_{last} \\ 0 & \text{else} \end{cases}$$

For subsequent frames n, with $n_{err} < n < n_{next}$, the error may be estimated as:

$$E(mb, n) = \sum_{i=1}^{N} E(i,\ n-1)\ \frac{d(i,\ mb,\ n)}{256}$$

where a uniformly distributed error in each macroblock is assumed after each iteration.

The estimated error E(mb, $n_{next} - 1$) is incorporated into the mode decision of the next frame. For example, macroblock mb is coded in INTRA mode, if E(mb, $n_{next} - 1$) exceeds a threshold.

In practice, error tracking information will only be stored for the latest M frames. Then, if $n_{err} < n_{next} - M$, no error-tracking information is available and the encoder must take special action. For example, the next frame may be coded in INTRA mode. However, other procedures are possible and may be more effective.

APPENDIX II

Recommended Optional Enhancement

II.1 Introduction

With the variety of optional modes available in this Recommendation (H.263 Version 2), it is crucial that several preferred mode combinations for operation be defined, so that option-enhanced terminals will have a high probability of connecting to each other using some syntax better than the "baseline." This appendix contains a list of preferred mode combinations, structured into three "levels" of support. Each level includes support for the levels below it, creating an "onion peel" structure of capabilities. The primary objective of this appendix is to describe the order in which modes should be supported in decoders, rather than to enforce a particular small set of mode combinations upon encoders.

In determining which modes would be placed into this level structure, the primary criteria used were performance-related: improvement in subjective quality, impact on delay, and impact on complexity (this includes computational burden, data dependencies, and ease of implementation). However, because this appendix seeks to address a wide variety of applications and transport layers, features addressing error resilience and ease of packetization were also incorporated into the same level structure. In keeping with these objectives, the levels of support described in this appendix are intended to be *universal*, so that the means of transport will not be an issue and the need for application-specific "profiles" will be minimized. This universality is seen as a key to promoting good interoperability across different types of terminals and networks. To maintain universality, several optional features have not been included in the level structure described herein. The absence of these features is not meant to be a condemnation of their usage; instead, their absence reflects only a collective opinion about the likelihood of widespread adoption of these features across a full spectrum of terminals, networks, and applications.

II.2 Levels of preferred mode support

Each optional mode described in this appendix has been categorized into one of three levels of support. While the ratio of performance improvement to computational complexity was a major criterion used in this categorization, utility for packetization and error resilience were also considered. Decoder support for a given level implies support for all lower levels. Furthermore, decoder support for a given level implies support for all subset combinations of the constituent modes of that level and all lower levels. This last requirement exists so that the limitations placed upon an encoder's choice of mode combinations is minimized. This is in keeping with the primary objective of this appendix, which is to describe the order in which modes should be supported at the decoder, rather than to enforce a particular small set of mode combinations upon the encoder.

Backward compatibility is also an important issue to consider. For this reason, a preferred mode combination including only the Advanced Prediction mode (Annex F) is also recommended (although this Recommendation falls outside the level structure defined below), since Advanced Prediction is regarded as the most beneficial of the optional modes in the original edition (Version 1) of this Recommendation.

II.2.1 Level 1 preferred modes

The first level of support is composed of the following modes:

1) **Advanced INTRA Coding (Annex I)** – Use of this mode improves the coding efficiency for INTRA macroblocks (whether within INTRA pictures or predictively-coded pictures). The additional computational requirements of this mode are minimal at both the encoder and decoder (as low as a maximum of 8 additions/subtractions per 8 × 8 block in the decoding process plus the use of a different but very similar VLC table in order to obtain a significant improvement in coding efficiency). For these reasons, Advanced INTRA Coding is included at this lowest level of support.

2) **Deblocking Filter (Annex J)** – Because of the significant subjective quality improvement that may be realized with a deblocking filter, these filters are already widely in use as a method of post-processing in video communication terminals. Annex J represents the preferred mode of operation for a deblocking filter because it places the filter within the coding loop. This placement eases the implementation of the filter (by reducing the memory requirement) and somewhat improves the coding performance over a post-procesing implementation. As with the Advanced Prediction mode, this mode also includes the four-motion-vector-per-macroblock feature and picture boundary extrapolation for motion compensation, both of which can further improve coding efficiency. The computational requirements of the deblocking filter are several hundred operatons per coded macroblock, but memory accesses and computational dependencies are uncomplicated. This last point is what makes the Deblocking Filter preferable to Advanced Prediction. Also, the benefits of Advanced Prediction are not as substantial when the Deblocking Filter is used as well. Thus, the Deblocking Filter is included at this lowest level of support while Advanced Prediction is deferred to Level 3.

3) **Supplemental Enhancement Information (Full-Frame Freeze Only) (Annex L, see L.4)** – The full-frame freeze is very simple to implement, requiring only that the decoder be able to stop the transfer of data from its output buffer to the video display. This sub-mode is useful for preventing the display of low-fidelity pictures while the encoder is building up a higher fidelity picture.

4) **Modified Quantization (Annex T)** – This mode includes an extended DCT coefficient range, modified DQUANT syntax, and a modified step size for chrominance. The first two features allow for more flexibility at the encoder and may actually decrease the encoder's computational load (by eliminating the need re-encode macroblocks when coefficient level saturation occurs). The third feature noticeably improves chrominance fidelity, typically with little added bit-rate cost and with virtually no increase in computation. At the decoder, the only significant computational burden is the ability to parse several new bitstream symbols.

II.2.2 Level 2 preferred modes

The second level of support is composed of the following modes:

1) **Unrestricted Motion Vectors (With UUI = "1" Sufficient) (Annex D)** – Annex D has two primary features:

   a) picture boundary extrapolation; and

   b) longer motion vector support.

   The first is already supported by the inclusion of Annex J in the first level of support. The longer motion vector support can provide a significant improvement in coding efficiency, especially for large picture sizes, rapid motion, camera movement, and low picture rates. When used with PLUSPTYPE present, this mode also allows for longer motion vector differences, which can significantly simplify encoder operation. The longer motion vectors do present a potential problem for the decoder in terms of memory access, but frame-size-dependent limits on the maximum motion vector size prevent this problem from becoming an appreciable obstacle to implementation.

2) **Slice Structured Mode (Annex K)** – For compatibility with packet-based transport layers, the Slice Structured mode is included here. All submodes of Annex K are to be supported, including the Rectangular Slice submode and the Arbitrary Slice Ordering submode. The additional computational burden imposed by the Slice Structured mode is minimal, limited primarily to bitstream generation and parsing.

3) **Reference Picture Resampling (Implicit Factor-of-4 Mode Only) (Annex P)** – The implicit factor-of-4 mode of Reference Picture Resampling allows for automatic reference picture resampling only when the size of the new frame is changed, as indicated in the picture header. No bitstream overhead is required for this mode of operation. Predictive dynamic resolution changes allow an encoder to make intelligent trade-offs between temporal and spatial resolution. Furthermore, this simplest mode of operation for Annex P (factor of 4 upsampling or downsampling only) adds only a modest amount of computational complexity to both the encoder or decoder, since the factor of 4 case uses a simple fixed FIR filter (requiring roughly 4 operations per pixel, at most).

II.2.3 Level 3 preferred modes

The third and final level of support is composed of the following modes:

1) **Advanced Prediction (Annex F)** – From a coding efficiency standpoint, this mode is the most important of the modes available in the prior version (Version 1) of this Recommendation. It includes overlapped block motion compensation, the four-motion-vector-per-macroblock feature, and it allows for motion vectors to point outside of the picture boundaries. The use of Advanced Prediction results in significant improvements in both subjective and objective performance. It does, however, require an appreciable increase in computation. More importantly, the data dependencies introduced by Advanced Prediction result in a complicated order of processing at the decoder. Coupled with the fact

that the quality improvements are not as obvious when the Deblocking Filter is also in use, these complexity considerations are the reason this mode is part of the third support level, rather than the first. However, since implementations of Recommendation H.263 that were designed prior to the adoption of the other modes in this list might have implemented Advanced Prediction by itself, Advanced Prediction-only operation is also recommended for maximal backward compatibility.

2) **Improved PB-frames (Annex M)** – The Improved PB-frames mode represents a substantial improvement over the basic PB-frames mode (Annex G) and has been demonstrated to significantly improve the coding efficiency for video sequences with a low degree of motion. While the basic version of PB-frames had problems in handling scene cuts and irregular motion, the Improved PB-frames mode may be used effectively at all times. In terms of computational burden, this mode is roughly neutral (especially for macroblock-pipelined architectures). However, because the Improved PB-frames mode can have a negative impact on delay and because it requires an additional frame store to implement (at both the encoder and decoder), it is placed in a higher layer.

3) **Independent Segment Decoding (Annex R) –** While the Independent Segment Decoding mode does introduce a significant amount of complexity (in terms of ease of implementation), it also significantly improves the error robustness of a video decoder, particularly in the presence of packet losses. For this reason, it is likely that many applications in which data can be lost in transmission (whether through packet loss or channel interference) will make use of the Independent Segment Decoding mode, especially in combination with the Slice Structured mode.

4) **Alternate INTER VLC (Annex S)** – This mode has been demonstrated to improve coding efficiency especially for pictures with heavy motion which are coded with high fidelity. This mode uses the same VLC table as the Advanced INTRA Coding mode and is also simple from a computational standpoint (for each block, it requires at most one additional pass over the parsed symbols to decode them into DCT coefficients). However, it is possible that this mode could lead to some difficulty in implementation (especially on highly pipelined architectures), which is why it is not included at the lower levels of support.

II.3 Picture formats and picture clock frequencies

To ensure a high quality level of interoperability, encoders and decoders supporting a large standard picture format (QCIF, CIF, 4CIF, 16CIF) should support all smaller standard picture formats. (As specified elsewhere in this Recommendation, decoders shall support sub-QCIF and QCIF, and encoders shall support sub-QCIF or QCIF.) Decoders should be capable of operation with a smaller picture format at maximum frame rates no lower than the maximum frame rate for which it is capable of operation with a larger standard picture format. For example, a decoder capable of decoding 4CIF pictures at 15 pictures per second [actually 15*(1000/1001) pictures per second] should also be able to decode CIF, QCIF and SQCIF pictures at least at 15 pictures per second.

Encoders and decoders supporting custom picture formats and/or custom picture clock frequencies are recommended to follow the guidelines below:

1) An encoder or decoder which supports a custom picture format should support all standard picture formats equal or smaller in both height and width than those of any supported custom picture format. For example, an encoder supporting a custom picture format of 256 × 256 should support QCIF and SQCIF picture encoding.

2) An encoder or decoder which supports a custom picture clock frequency for a standard picture format should support a higher or equal frame rate (in pictures per second) at the standard picture clock frequency of 30 Hz [actually 30*(1000/1001) Hz] for the same picture format. For example, if a decoder supports QCIF pictures at 12.5 pictures per second with a 25 Hz picture clock, it should at least be able to decode QCIF pictures at 15 pictures per second at 30 Hz. If a custom picture format is used, all supported smaller or equal standard picture formats should be capable of operating at frame rates for the standard picture clock frequency no lower than the maximum frame rate of the custom picture format.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages