INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO INFORMATION

Source: Video Syntax Sub-Group ISO/IEC JTC1/SC29/WG11
Title: Revised Syntax and Semantics for MPEG-2 Video MPEG93/ ???
2 April 93, Sydney

Modified Syntax

7.2.2 Video Sequence

video_sequence() {	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
if (nextbits() == extension_start_code) {		
sequence_extension()		
do {		
extension_and_user_data(0)		
do {		
if (next_bits() == group_start_code) {		
group_of_pictures_header()		
extension_and_user_data(1)		
}		
picture_header()		
extensions_and_user_data(2)		
picture_data()		
} while ((next_bits() == picture_start_code) ||		
next_bits() == group_start_code))		
if (nextbits() != sequence_end_code) {		
sequence_header()		
sequence_extension()		
}		
} while (nextbits() != sequence_end_code)		
} else {		

do {		
do {		
group_of_pictures_header()		
if (next_bits() == user_data_start_code)		
user_data()		
do {		
picture_header()		
if (next_bits() == user_data_start_code)		
user_data()		
picture_data()		
} while (next_bits() == picture_start_code)		
} while (next_bits() == group_start_code)		
if (nextbits() != sequence_end_code)		
sequence_header()		
} while (nextbits() != sequence_end_code)		
}		
sequence_end_code		
}		

1

2 **7.2.3 Sequence header**

sequence_header() {	No. of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
pel_aspect_ratio	4	uimsbf
frame_rate	4	uimsbf
bit_rate	18	uimsbf
marker_bit	1	"1"
vbv_buffer_size	10	uimsbf
constrained_parameter_flag	1	
load_intra_quantizer_matrix	1	
if (load_intra_quantizer_matrix)		
intra_quantizer_matrix[64]	8*64	uimsbf
load_non_intra_quantizer_matrix	1	
if (load_non_intra_quantizer_matrix)		
non_intra_quantizer_matrix[64]	8*64	uimsbf
next_start_code()		

3

Sequence extension

sequence_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
profile_and_level_indication	8	uimsbf
non_interlaced_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker	1	
vbv_buffer_size_extension	5	uimsbf
frame_rate_extension	8	uimsbf
next_start_code()		
}		

Note: The flags above are usually zero. However if many more are added then attention must be paid to start-code emulation.

Extension and user data

extension_and_user_data(i) {	No. of bits	Mnemonic
while ((nextbits()==extension_start_code) ||		
(nextbits()==user_start_code)) {		
if (nextbits()==extension_start_code)		
extension_data(i)		
if (nextbits()==user_start_code)		
user_data()		
}		
}		
}		

User data

user_data() {	No. of bits	Mnemonic
user_data_start_code	32	bslbf
while(nextbits() != '0000 0000 0000 0000 0000 0001') {		
user_data	8	
}		
next_start_code()		
}		

Sequence display extension

sequence_display_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
video_format	3	uimsbf
colour_description	1	uimsbf
if (colour_description) {		
colour_primaries	8	uimsbf
transfer_characteristics	8	uimsbf
matrix_coefficients	8	uimsbf
}		
display_horizontal_dimension	14	uimsbf
marker_bit	1	"1"
display_vertical_dimension	14	uimsbf
next_start_code()		
}		

Quant matrix

quant_matrix_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
load_intra_quantizer_matrix	1	uimsbf
if (load_intra_quantizer_matrix)		
intra_quantizer_matrix[64]	8 * 64	uimsbf
load_non_intra_quantizer_matrix	1	uimsbf
if (load_non_intra_quantizer_matrix)		
non_intra_quantizer_matrix[64]	8 * 64	uimsbf
load_chroma_intra_quantizer_matrix	1	"0"
load_chroma_non_intra_quantizer_matrix	1	"0"
next_start_code()		
}		

7.2.4 Group of pictures header

group_of_pictures_header() {	No. of bits	Mnemonic
group_start_code	32	bslbf
time_code	25	
closed_gop	1	
broken_link	1	
next_start_code()		
}		

1 **.2.5 Picture header**

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

2

picture_coding_extension() {	No . of bits	Mnemonic
extension_start_code	32	bslbf
extension_id	4	uimsbf
forward_horizontal_f_code	4	uimsbf
forward_vertical_f_code	4	uimsbf
backward_horizontal_f_code	4	uimsbf
backward_vertical_f_code	4	uimsbf
intra_dc_precision	2	uimsbf
picture_structure	2	uimsbf
top_field_first	1	uimsbf
frame_pred_frame_dct	1	uimsbf
concealment motion vectors	1	uimsbf
q_scale_type	1	uimsbf
intra_vlc_format	1	uimsbf
alternate_scan	1	uimsbf
number_of_field_displayed_code	1	uimsbf
chroma_postprocessing_type	1	uimsbf
non_interlaced_frame	1	uimsbf
composite_display_flag	1	uimsbf
if (composite_display_flag) {		
v-axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code()		
}		

frame_pred_frame_dct is 1 indicates that the dct is frame based and the prediction is frames based and the prediction is 16x16 (as in MPEG-1). 0 enables all of the field dct, field pred and dual prime.

Picture pan-scan extension

picture_pan_scan_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
for (i=0; i<number_of_pan_offsets; i++) {		
pan_horizontal_left_upper_offset_integer	12	uimsbf
marker	1	
pan_horizontal_left_upper_offset_sub_pel	3	uimsbf
pan_vertical_left_upper_offset_integer	12	uimsbf
marker	1	
pan_vertical_left_upper_offset_sub_pel	3	uimsbf
}		
next_start_code()		
}		

```
if (non_interlaced_sequence)
    number_of_pan_offsets = 1
else
    number_of_pan_offsets = number_of_fields_displayed
```

Picture Data

picture_data() {	No. of bits	Mnemonic
do {		
slice()		
} while (nextbits() == slice_start_code)		
next_start_code()		
}		

7.2.6 Slice layer

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

1 **7.2.7 Macroblock layer**

macroblock() {	No. of bits	Mnemonic
if (<sequence extension was not present>)		
while (nextbits() == '0000 0001 111')		
macroblock_stuffing	11	vlclbf
while (nextbits() == '0000 0001 000')		
macroblock_escape	11	vlclbf
macroblock_address_increment	1-11	vlclbf
macroblock_type	1-8	vlclbf
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((picture_structure == 'frame') &&		
(frame_pred_frame_dct == 0) &&		
(macroblock_intra || macroblock_pattern))		
dct_type	1	uimsbf
if (macroblock_quant)		
quantizer_scale_code	5	uimsbf
if (macroblock_motion_forward ||		
(macroblock_intra && concealment_motion_vectors))		
forward_motion_vectors()	...	...
if (macroblock_motion_backward)		
backward_motion_vectors()	...	...
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	
if (macroblock_pattern)		
coded_block_pattern()	...	...
for (i=0; i<block_count; i++) {		
block(i)		
}		
if (picture_coding_type == 4)		
end_of_macroblock	1	"1"
}		

2

3

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

1

2

motion_vector () {	No. of bits	Mnemonic
motion_horizontal_code	1-13	vlclbf
if ((horizontal_f!=1) && (motion_horizontal_code != 0))		
motion_horizontal_r	1-8	uimsbf
if (dmv == 1)		
dmv_horizontal	1-2	vlcbf
motion_vertical_code	1-13	vlclbf
if ((vertical_f!=1) && (motion_vertical_code != 0))		
motion_vertical_r	1-8	uimsbf
if (dmv == 1)		
dmv_vertical	1-2	vlcbf
}		

3

field_motion_vector () {	No. of bits	Mnemonic
motion_vertical_field_select	1	uimsbf
motion_vector()	...	...
}		

4

coded_block_pattern () {	No. of bits	Mnemonic
coded_block_pattern_420	3-9	vlclbf
if ((chroma_format == 4:4:4) || (chroma_format == 4:2:2))		
extension of coded block pattern		
}		

5

6

block(i) {	No. of bits	Mnemonic
if (pattern_code[i]) {		
if (macroblock_intra) {		
if (i<4) {		
dct_dc_size_luminance	2-9	vlclbf
if(dct_dc_size_luminance != 0)		
dct_dc_differential	1-11	uimsbf
} else {		
dct_dc_size_chrominance	2-10	vlclbf
if(dct_dc_size_chrominance !=0)		
dct_dc_differential	1-11	uimsbf
}		
} else {		
First DCT coefficient	...	
}		
if (picture_coding_type != 4) {		
while (nextbits() != End of block)		
Subsequent DCT coefficients	...	
End of block	...	
}		
}		
}		

Modified Semantics

7.3 Video bit stream semantics

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

7.3.1 Video sequence

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

7.3.2 Sequence header

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

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

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

horizontal_size -- The horizontal_size is a 14 bit unsigned integer, the 12 least significant bits are defined in horizontal_size_value, the 2 most significant bits are defined in horizontal_size_extension. The horizontal_size is the width of the displayable part of each luminance picture in pixels. The width of the encoded luminance picture in macroblocks, mb_width, is (horizontal_size+15)/16. The displayable part of the picture is left-aligned in the encoded picture.

vertical_size -- The vertical_size is a 14 bit unsigned integer, the 12 least significant bits are defined in vertical_size_value, the 2 most significant bits are defined in vertical_size_extension. The vertical_size is the height of the displayable part of each luminance picture in pixels. The height of the encoded

luminance picture in macroblocks, mb_height, is (vertical_size+15)/16. The displayable part of the picture is top-aligned in the encoded picture.

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

Table 7-3--- pel_aspect_ratio

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

frame_rate -- This is a four-bit integer defined in the following Table 7-4. This field is renamed with respect to MPEG-1 where it was called picture_rate. If non_interlaced_sequence is "0", frame_rate specifies the number of frames per second of the intended display sequence. If non_interlaced_sequence is "1" then frame_rate specifies the number of non-interlaced frames per second and in this case also the number of coded pictures per second.

Table 7-4 --- frame_rate

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

bit_rate -- This is a 30 bit integer. The lower 18 bits of the integer are in bit_rate and the upper 12 bits are in bit_rate_extension. The 30 bit integer specifies the bit rate of the bit stream measured in units of 400 bits/second, rounded upwards. The value zero is forbidden. The value 3FFFFFFF identifies variable bit rate operation.

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

vbv_buffer_size -- This is a 15-bit integer. The lower 10 bits of the integer are in vbv_buffer_size and the upper 5 bits are in vbv_buffer_size_extension. The integer definines the size of the VBV (Video Buffering Verifier, see Annex C) buffer needed to decode the sequence. It is defined as:

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

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

constrained_parameters_flag -- *{Resurect original meaning from MPEG-1}*

In MPEG-2 bit-streams constrained_parameters_flag shall be "0"

load_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if intra_quantiser_matrix follows. *{ The following text is removed: If it is set to "0" then the default values defined below are used until the next occurrence of the sequence header.}* If it is set to "0" then there is no change in the values that shall be used. The occurance of a sequence header causes the default values defined below to be used.

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

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

load_non_intra_quantiser_matrix -- This is a one-bit flag which is set to "1" if non_intra_quantiser_matrix follows. *{ The following text is removed: If it is set to "0" then the default values defined below are used until the next occurrence of the sequence header.}* If it is set to "0" then there is no change in the values that shall be used. The occurance of a sequence header causes the default values defined below to be used.

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

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

extension_start_code -- The extension_start_code is the bit string 000001B5 in hexadecimal. It identifies the beginning of extension data.

extension_start_code_identifier — The extension_start_code shall be followed by a four bit integer indicating the type of extension data that follows. These extension identification codes are shown in the follwing table:

Extension ID	Name	Follows Sequence Header (i=0)	Follows GOP Header (i=1)	Follows Picture Header (i=2)
0000	reserved			
0001	Sequence Extension	Always		
0010	Sequence Display	Optional		
0011	Quant Matrix	Optional		Optional
0100	Sequence Frequency	Conditional		
0101	Sequence Spatial	Conditional		
0110	10 bit input ?			
0111	Picture Pan-Scan			Optional
1000	Picture Coding			Always
1001	Picture Spatial			Optional
1010	FF/FR ????			
1011	reserved			
...	...			
1111	reserved			

profile_and_level_indication – This is 8 bit integer is used to signal the profile and level identification.

Bits	Field Size (bits)	Meaning
7	1	shall be 0. (Escape for future use)
6:4	3	Profile Id
3:0	4	Level Id

non_interlaced_sequence – When set to "1" the video sequence contains only non-interlaced frames.

chroma_format - This is a two bit integer indicating the chrominance format as defined in the Table ?-?.

Table ?-?. Meaning of chroma_format

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

horizontal_size_extension -- This word forms the 2 most significant bits from horizontal_size.

vertical_size_extension -- This word forms the 2 most significant bits from vertical_size.

bit_rate_extension -- This word forms the 12 most significant bits from bit_rate.

vbv_buffer_size_extension -- This word forms the 5 most significant bits from vbv_buffer_size.

frame_rate_extension -- This word forms an 8 bit extension to frame_rate. The semantics of this are not yet defiend.

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

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

7.?.? Sequence Display Extension

video_format - This is a three bit integer indicating the representation of the pictures before being coded in accordance with this specification. Its meaning is defined in Table ?-?.

Table ?-?. Meaning of video_format

video_format	Meaning
000	component
001	PAL
010	NTSC
011	SECAM
100	MAC
101	reserved
110	reserved
111	reserved

color_primaries -- This 8-bit integer describes the chromaticity coordinates of the source primaries, and is define in the following table:

Value	Primaries		
0	Invalid code (reserved for start code)		
1	CCIR Recommendation 709 (1990)		
	primary	x	y
	green	0.300	0.600
	blue	0.150	0.060
	red	0.640	0.330
	white D65	0.3127	0.3290
2	Unspecified Video		
	Image characteristics are unknown.		
3	User defined primaries		
4	CCIR Recommendation 624-4 System M		
	primary	x	y
	green	0.21	0.71
	blue	0.14	0.08
	red	0.67	0.33
	white C	0.310	0.316

5	CCIR Recommendation 624-4 System B,G			
	primary		x	y
	green		0.29	0.60
	blue		0.15	0.06
	red		0.64	0.33
	white D65		0.313	0.329
6	SMPTE 170M			
	primary		x	y
	green		0.310	0.595
	blue		0.155	0.070
	red		0.630	0.340
	white D65		0.3127	0.3290
7	SMPTE 240M (1987)			
	primary		x	y
	green		0.310	0.595
	blue		0.155	0.070
	red		0.630	0.340
	white D65		0.3127	0.3291
8 to 255	reserved for future use			

Default value: 1*{Editors Not - why is there a default?}*

transfer_characteristic -- This 8-bit integer describes the opto-electronic transfer characteristic of the source image, and is defined in the following table:

Value	Transfer Characteristic
0	Invalid code (reserved for start code)
1	CCIR Recommendation 709 (1990) $V = 1.099\ L_c^{0.45} - 0.099$ for $1 \geq L_c \geq 0.018$ $V = 4.500\ L_c$ for $0.018 > L_c \geq 0$
2	Unspecified Video Image characteristics are unknown.

3	User defined transfer characteristic
4	CCIR Recommendation 624-4 System M Assumed display gamma 2.2
5	CCIR Recommendation 624-4 System B,G Assumed display gamma 2.8
6	SMPTE 170M $V = 1.099\ L_c^{0.45} - 0.099$ for $1 \geq L_c \geq 0.018$ $V = 4.500\ L_c$ for $0.018 > L_c \geq 0$
7	SMPTE 240M (1987) $V = 1.1115\ L_c^{0.45} - 0.1115$ for $L_c \geq 0.0228$ $V = 4.0\ L_c$ for $0.0228 > L_c$
8 to 255	reserved for future use

Default value: 1*{Editors Not - why is there a default?}*

matrix_coefficients -- This 8-bit integer describes the matrix coefficients used in deriving luminance and color difference signals from the green, blue, and red primaries, and is defined in the following table

Value	Matrix
0	Invalid code (reserved for start code)
1	CCIR Recommendation 709 (1990). $E'_Y = 0.7154\ E'_G + 0.0721\ E'_B + 0.2125\ E'_R$ $E'_{PB} = 0.5389\ (E'_B - E'_Y)$ $E'_{PB} = 0.6349\ (E'_R - E'_Y)$
2	Unspecified Video Image characteristics are unknown.

3	User defined matrix	
4	FCC	$E'_Y = 0.59\,E'_G + 0.11\,E'_B + 0.30\,E'_R$
5	CCIR Recommendation 624-4 System B,G	$E'_Y = 0.587\,E'_G + 0.114\,E'_B + 0.299\,E'_R$ $E'_U = 0.493\,(E'_B - E'_Y)$ $E'_V = 0.877\,(E'_R - E'_Y)$
6	SMPTE 170M	$E'_Y = 0.587\,E'_G + 0.114\,E'_B + 0.299\,E'_R$
7	SMPTE 240M (1987)	$E'_Y = 0.701\,E'_G + 0.087\,E'_B + 0.212\,E'_R$ $E'_{PB} = -0.384\,E'_G + 0.500\,E'_B - 0.116\,E'_R$ $E'_{PR} = -0.445\,E'_G - 0.055\,E'_B + 0.500\,E'_R$
8 to 255	reserved for future use	

Default value: 1 *{Editors Not - why is there a default?}*

display_horizontal_dimension - This is a 14 bit integer indicating the horizontal dimension of the picture to be displayed.

display_vertical_dimension - This is a 14 bit integer indicating the vertical dimension of the picture to be displayed.

7.?.? Quant Matrix Download Extension

load_chroma_intra_quantiser_matrix -- This is a one-bit flag which is set to "0" in the case of 4:2:0 coded data (Main profile).

load_chroma_non_intra_quantiser_matrix -- This is a one-bit flag which is set to "0" in the case of 4:2:0 coded data (Main profile).

7.3.3 Group of pictures layer

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

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

Table 7-5 --- time_code

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

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

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

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

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

extension_start_code -- See Clause 6.5.2.

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

user_data -- See Clause 6.5.2.

7.3.4 Picture header

picture_start_code -- The picture_start_code is a string of 23 bits having the value 00000100 in hexadecimal.

temporal_reference -- The temporal_reference is a 10-bit unsigned integer associated with each input picture. It is incremented by one, modulo 1024, for each input picture. For the earliest picture (in display order) in each group of pictures, the temporal_reference is reset to zero. When a frame is coded as two fields the temporal reference in both fields is the same.

picture_coding_type -- The picture_coding_type identifies whether a picture is an intra-coded picture(I), predictive-coded picture(P), bidirectionally predictive-coded picture(B), or intra-coded with only DC coefficients (D) according to the following Table. D-pictures shall never be included in the same video sequence as the other picture coding types. The meaning of picture_coding_type is defined in Table 7-6.

Table 7-6 --- picture_coding_type

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

vbv_delay -- The vbv_delay is a 16-bit unsigned integer. For constant bitrate operation, the vbv_delay is used to set the initial occupancy of the decoder's buffer at the start of play so that the decoder's

buffer does not overflow or underflow. The vbv_delay measures the time needed to fill the VBV buffer from an initially empty state at the target bit rate, R, to the correct level immediately before the current picture is removed from the buffer.

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

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

where:

$n > 0$

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

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

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

{The description of vbv_delay is likely to be modified by consideration of the VBV arising from low delay and 3/2 pull-down coding. Adrian}

full_pel_forward_vector -- If set to "1", then the motion vector values decoded represent integer pixel offsets (rather than half-pixel units) as reflected in the equations of Clause 6.6.2. If picture_coding_extension is present (MPEG-2) then the value of this flag shall be "0".

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

full_pel_backward_vector -- If set to "1", then the motion vector values decoded represent integer pixel offsets (rather than half pixel units) as reflected in the equations of Clause 6.6.3. If picture_coding_extension is present (MPEG-2) then the value of this flag shall be "0".

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

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

extra_information_picture -- Reserved.

7.?.? Picture Coding Extension

forward_horizontal_f_code -- An unsigned integer taking values 1 through 9. The value zero is forbidden. forward_horizontal_r_size and forward_horizontal_f used in the process of decoding the forward motion vectors are derived from forward_horizontal_f_code as described in Clause 6.6.2. The value in this element is used in place of **forward_f_code** for decoding horizontal motion vectors.

forward_vertical_f_code -- An unsigned integer taking values 1 through 9. The value zero is forbidden. forward_vertical_r_size and forward_vertical_f used in the process of decoding the forward motion vectors are derived from forward_vertical_code as described in Clause 6.6.2. The value in this element s used in place of **forward_f_code** for decoding vertical motion vectors.

backward_horizontal_f_code -- An unsigned integer taking values 1 through 9. The value zero is forbidden. backward_horizontal_r_size and backward_horizontal_f used in the process of decoding the backward motion vectors are derived from backward_horizontal_f_code as described in Clause 6.6.2. The value in this element is used in place of **backward_f_code** for decoding horizontal motion vectors.

backward_vertical_f_code -- An unsigned integer taking values 1 through 9. The value zero is forbidden. backward_vertical_r_size and backward_vertical_f used in the process of decoding the backward motion vectors are derived from backward_vertical_code as described in Clause 6.6.2. The value in this element s used in place of **backward_f_code** for decoding vertical motion vectors.

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

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

Default value: 00

Remark

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

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

picture_structure - This is a 2-bit integer defined in the Table ?-?.

Table ?-? Meaning of picture_structure

picture_structure	Meaning
11	Frame-Picture
01	Top Field
10	Bottom Field
00	reserved

top_field_first — The meaning of this element depends upon picture_structure. In a frame picture top_field_first being set to "1" indicates that the top field of the frame is the earlier field. In a field picture (or when non_interlaced_sequence is set to "1") top_field_first shall have the value "0".

frame_pred_frame_dct - This flag shall be set to "1" to indicate that only frame-DCT shall be used and only 16x16 frame prediction. In a field picture it shall be "0".

concealment_motion_vectors - This flag has the value "1" to indicate that motion vectors are coded for intr macroblocks.

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

0	MPEG1 compatible:linear
1	non linear law

Default value: 0

If **qscale_type** is set to "1" the follwoing table is used:

quantizer_scale_code	quantizer_scale	Binary Representation
00 000	forbidden	
00 001	0.5	00000.1
00 010	1.0	00001.0
00 011	1.5	00001.1
00 100	2.0	00010.0
00 101	2.5	00010.1
00 110	3.0	00011.0
00 111	3.5	00011.1
01 000	4.0	000100.
01 001	5.0	000101.
01 010	6.0	000110.
01 011	7.0	000111.
01 100	8.0	001000.
01 101	9.0	001001.
01 110	10.0	001010.
01 111	11.0	001011.
10 000	12.0	001100.
10 001	14.0	001110.
10 010	16.0	010000.
10 011	18.0	010010.
10 100	20.0	010100.
10 101	22.0	010110.
10 110	24.0	011000.
10 111	26.0	011010.
11 000	28.0	011100.
11 001	32.0	100000.
11 010	36.0	100100.
11 011	40.0	101000.
11 100	44.0	101100.
11 101	48.0	110000.
11 110	52.0	110100.
11 111	56.0	111000.

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

intra_vlc_format	
0	MPEG-1 VLC
1	Alternative Intra VLC

alternate_scan -- If set to "1" the alternate scan is used instead of the default (zig-zag) scan.

number_of_field_displayed_code -- This is a one-bit integer. If it is equal to 1, the frame must displayed during a 3-field duration. If it is equal to 0, the frame must be displayed during a 2-field duration.

It must be zero in field-pictures or if *non_interlaced_sequence* is equal to 1. Frames that are coded as two field-pictures are always displayed during a 2-field-period duration. This syntax element can be used independently of the value of *frame_pred_frame_dct*.

Note : The only way to cause a decoder to automatically perform 3:2 pulldown frame rate conversion with non-MPEG-1 bitstreams is to set the *non_interlaced_sequence* flag to 1 and set the picture_rate to 24 Hz or 23.976 Hz. This automatic conversion always applies to the entire sequence.

picture_extension_data -- Reserved. *{Editors note - not sure if this should be here}*

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

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

Default value: 1

v_axis -- 1 bit integer used in PAL. v-axis is set to 1 on a positive sign, v-axis is set to 0 otherwise.

field_sequence -- 3 bits integer which is defined in the following table:

field sequence	frame	field
000	1	1
001	1	2
010	2	3
011	2	4
100	3	5
101	3	6
110	4	7
111	4	8

sub_carrier -- This is a 1 bit integer. Set to 0 means the sub-carrier/line frequency relationship is correct, set to 1 is not correct.

burst_amplitude -- This is a 7 bits integer defining the burst amplitude (for PAL and NTSC only). The amplitude of the sub-carrier burst is quantized as a CCIR Recommendation 601 luminance signal, with the MSB omitted.

sub_carrier_phase -- This is a 8 bits integer defining the sub-carrier phase (for PAL and NTSC only). Phase of reference sub-carrier at the field-synchronisation datum with respect, to field start as defined in CCIR Recommendation 470, *MSB first*.

Scale: 0 = ([360°/256] * 0)

Scale: 1 = ([360°/256] * 1)

... = ...

Scale: 255 = ([360°/256] * 255)

7.?.? Picture Pan Scan Extension

pan_horizontal_left_upper_offset_integer -- this is a 12 bit integer giving the 12 integral part of pan_horizontal_left_upper_offset.

pan_horizontal_left_upper_offset_sub_pel -- this is a 3 bit integer giving the 3 fractional bits of pan_horizontal_left_upper_offset. pan_horizontal_left_upper_offset indicates (to 1/8 pel accuracy) the horizontal postion of the left upper corner of a retangular area which has to displayed.

pan_vertical_left_upper_offset_integer -- this is a 12 bit integer giving the 12 integral part of pan_vertical_left_upper_offset.

pan_vertical_left_upper_offset_sub_pel -- this is a 3 bit integer giving the 3 fractional bits of pan_vertical_left_upper_offset. pan_vertical_left_upper_offset indicates (to 1/8 pel accuracy) the vertical postion of the left upper corner of a retangular area which has to displayed.

7.3.5 Slice header

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

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

quantiser_scale_code -- An unsigned integer in the range 1 to 31 . The decoder shall use this value until another quantiser_scale_code is encountered either at the slice layer or the macroblock layer. The value zero is forbidden. If qscale_type is "0" then quantizer_scale takes the same value as quantiser_scale_code. If qscale_type is "1" then quantizer_scale is derived from qscale_type_code using table ?-? (see where qscale_type is defined).

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

extra_information_slice -- Reserved.

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

7.3.6 Macroblock layer

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

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

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

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

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

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

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

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

mb_row = macroblock_address / mb_width

mb_column = macroblock_address % mb_width

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

NOTE The slice_vertical_position differs from mb_row by one.

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

macroblock_quant -- Derived from macroblock_type.

macroblock_motion_forward -- Derived from macroblock_type.

macroblock_motion_backward -- Derived from macroblock_type.

macroblock_pattern -- Derived from macroblock_type.

macroblock_intra -- Derived from macroblock_type.

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

code	prediction type	motion_vector_count	mv_format	dmv
00	reserved			
01	Field-based prediction	2	field	0
10	Frame-based prediction	1	frame	0
11	Dual-Prime	1	field	1

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

code	prediction type	motion_vector_count	mv_format	dmv
00	reserved			
01	Field-based prediction	1	field	0
10	16x8 MC	2	field	0
11	Dual-Prime	1	field	1

dct_type - This is a one-bit integer indicating whether the macroblock is frame DCT coded or field DCT coded. If this is set to "1", the macroblock is field DCT coded.

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

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

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

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

motion_vectors() — denotes forward_motion_vectors() for forward motion vector decoding and denotes backward_motion_vectors() for backward motion vector decoding.

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

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

dmv_horizontal – is the horizontal coordinate of the differential motion vector expressed in half pel units.

{Editorial Note - This VLC will move to Annex B}

code	value
11	-1
0	0
10	1

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

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

dmv_vertical – is the vertical coordinate of the differential motion vector expressed in half pel units.

{Editorial Note - This VLC will move to Annex B}

code	value
11	-1
0	0
10	1

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

0	prediction comes from the Top Reference Field
1	prediction comes from the Bottom Reference Field

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

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

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

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

Table 7-6 Chroma format

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

7.3.7 Block layer

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

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

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

For luminance blocks:

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

For chrominance blocks:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Block Level

The automatic self-adaptive VLC decoding documented in the working draft is removed.

In the case that intra_vlc_format is zero only the MPEG-1 table from Table B-11 is used.

In the case that intra_vlc_format is one then the MPEG-1 table is used for non-intra macroblocks. In the intra macroblocks the table documented in B-12 is used.

In both tables the escape format defined in Annex B-13 is selected depending on whether or not a sequnce_header_extension is present.

Sequence :

A sequence may contain only P- and B-frames, and no I-frame. However, in this case, the first frame of the sequence shall be a P-frame and shall contain only intra-coded macroblocks.

When a frame is transmitted in the form of two field pictures:

Both fields must be of the same picture_coding_type, except for I-frames, where the first transmitted field must be an I-picture and the second can be an I-field or a P-field. In the latest case, this P-field shall only be predicted from the first I-field of the same frame.

The first transmitted field of a frame may be a top-field or a bottom field, and the next field must be of opposite parity.

The *number_of_field_displayed_code* shall be equal to 0 (2 field duration) for each field picture.

The frame display period is always 2 fields (3:2 pulldown frame-rate conversion cannot be achieved by the mean of *number_of_field_displayed_code* when only field pictures are used)

Modified Decoding Process

Modified Inverse Quantization Process

At the top of page 45, lines 1 to 27. Everywhere the clause "(128*8)" appears this should be replaced by "(dc_reset_value * 8)". The value of dc_reset_value depens on "**intra_dc_precision**".

Page 45, lines 33 to 44. The formulae in these lines are replaced by:

```
if (qscale_type == 0)  {
    if (QAC[i][j]>0)
        dct_recon[i][j]=((2*QAC[i][j]+1)*quantizer_scale
    *wN[i][j])) / 16

    if (QAC[i][j] < 0)
        dct_recon[i][j] = ((2*QAC[i][j] -1) * quantizer_scale
                    *wN[i][j]) / 16
    if (QAC [i][j] == 0)
        dct_recon[i][j] = 0
} else {
    if (QAC [i][j]  > 0)
        dct_recon[i][j] = ((2*QAC[i][j]+1)*(2*quantizer_scale)
                        * wN [i][j])  /32

    if (QAC [i][j]) < 0)
        dct_recon[i][j] = ((2*QAC[i][j]-1)*(2*quantizer_scale)
                        * wN [i][j])  /32

    if (QAC [i][j] == 0)
        dct_recon[i][j] = 0

}
```

IDCT mismatch control

After the above formulae the follwoing needs to be added.

```
sum =  SUM  dct_recon [i] [j]
        i,j
 f  (sum is EVEN)
    toggle the LSB of the sign-magnitude representation of  dct_recon [7][7]
```

On Page 44, the new IDCT mismatch control process is also added to the INTRA case. The formulae on line 30 should be modifed (addition of brackets) as follows:

dct_recon[i][j] = (2*quantizer_scale_value *QAC[i][j] *wI [i][j]) / 16

Line 31 through line 34 are deleted (to remove the old mismatch control mechanism). And the following is added after line 36:

sum = SUM dct_recon [i] [j]
i,j

if (sum is EVEN)
toggle the LSB of the sign-magnitude representation of dct_recon [7][7]

Modified Reconstruction of Motion Vectors

Page 40. In the case of MPEG-2 bit streams there are now separate f_code values for horizontal and vertical motion vectors. The appropriate value for forward_f_code (either forward_horixontal_f_code or forward_vertical_f_code) is used.

Concealment motion vectors : Concealment motion vectors are transmitted with intra-coded macroblocks when *concealment_motion_vectors* is equal to 1. A concealment motion vector shall be a frame motion vector in frame pictures, and a field motion vector in field pictures. It is always followed by the marker_bit "1", to avoid start_code emulation.

Dual-prime:

Dual-prime prediction mode is limited to P-frames. It shall be used only if the preceding frame (in display order) is a P-frame or an I-frame.

The temporal scaling of the vector is done as specified in TM4 section 5.3.3, with the following simplification: 32 must be replaced by 2 and 16 must be replaced by 1.

Reference fields rule:

The two reference fields are always of opposite parity (one is a top field, one is a bottom field).

For forward prediction, the two reference fields are determined as follows:

- If the predicted frame is B-frame in either field or frame structure, or if it is a P-frame in frame structure, the two reference fields are the two field of the most recently transmitted I- or P-frame that is located (temporally) before the predicted frame.

- If the predicted frame is P-frame in field structure, then the reference fields used are not the same for prediction of the two fields of the frame.

To predict the first transmitted field (it may be a top-field or a bottom field), the two reference fields are the two field of the previously transmitted I- or P-frame.

To predict the second transmitted field, one of the reference fields is the first field of the same frame, and the other is the field of the previously transmitted I- or P-frame that has the same parity as the predicted field.

For backward prediction, the two reference fields are the two field of the most recently transmitted I- or P-frame.

Prediction of Motion Vector

Prediction of motion vector is controlled by picture_coding_type, "field_motion_type" and "frame_motion_type". By these field_motion_type and frame_motion_type, motion_vector_count and mv_format is set as Tables.

mv_format = "frame"

[P-frame picture]

--PMV1 is used. PMV2, PMV3 and PMV4 are reset to PMV1.

[B-frame picture]

--PMV1 is used for forward motion vector prediction in a MB.
--PMV2 is reset to PMV1.
--PMV3 is used for backward motion vector prediction in a MB.
--PMV4 is reset to PMV3.

mv_format = "field"

[P-frame or P-field picture]

If prediction_type = "Field based prediction":

--PMV1 is used for vectors used to predict field1 from field1.
--PMV2 is used for vectors used to predict field1 from field2.
--PMV3 is used for vectors used to predict field2 from field2.
--PMV4 is used for vectors used to predict field2 from field1.

PMVs are not reset when they are not used.

If prediction_type = "Dual-prime":

--PMV1 is used for prediction of the Dual-prime motion vector.
--PMV2 is reset to the scaled motion vectors to field1 from field2.
--PMV3 is reset to PMV1. (the transmitted field motion vector).
--PMV4 is reset to the scaled motion vectors to field2 from field1.

[B-frame or B-field picture]

--PMV1 is used for the first transmitted forward motion vector prediction in a MB.
--PMV2 is used for the second transmitted forward motion vector prediction in a MB.
--PMV3 is used for the first transmitted backward motion vector prediction in a MB.
--PMV4 is used for the second transmitted backward motion vector prediction in a MB.

Where the count of transmitted forward(backward) vectors is 1, PMV2 is reset to PMV1, and PMV4 is reset to PMV3. (The motion vector count is derived from "picture_structure" and "prediction_type" by Table.1)

How to use the PMVs in P-pictures

forward_motion_vector(){

```
 1    if(motion_vector_count==1){
 2     if(mv_format==frame){
 3        motion_vector(); /***PMV1.  PMV2 , 3 and 4 are reset to PMV1***/
 4     }else{
 5        forward_field_motion_vector();/***PMV1.  PMV3 is reset to PMV1***/
 6        if(dmv==1){
 7              dmv_horizontal;
 8              dmv_vertical;
 9        }
10     }
11    }else{/***motion_vector_count==2***/
12     forward_field_motion_vector();/***PMV1 or PMV2***/
13     forward_field_motion_vector();/***PMV3 or PMV4***/
14    }
15   }
16
17
18   How to use the PMVs in B-pictures
19
20   forward_motion_vector(){
21    if(motion_vector_count==1){
22     if(mv_format==frame){
23        motion_vector(); /***PMV1.  PMV2 is reset to PMV1***/
24     }else{
25        forward_field_motion_vector();/***PMV1.  PMV2 is reset to PMV1***/
26        if(dmv==1){
27              dmv_horizontal;
28              dmv_vertical;
29        }
30     }
31    }else{/***motion_vector_count==2***/
32     forward_field_motion_vector();/***PMV1***/
33     forward_field_motion_vector();/***PMV2***/
34    }
35   }
36
37   backward_motion_vector(){
38    if(motion_vector_count==1){
39     if(mv_format==frame){
40        motion_vector(); /***PMV3.  PMV4 is reset to PMV3***/
41     }else{
```

```
        backward_field_motion_vector();/***PMV3. PMV4 is reset to PMV3***/
        if(dmv==1){
              dmv_horizontal;
              dmv_vertical;
        }
    }
  }else{/***motion_vector_count==2***/
    backward_field_motion_vector();/***PMV3***/
    backward_field_motion_vector();/***PMV4***/
  }
}
```

Rules for skipped macroblocks.

Definition of skipped macroblocks compatible with MPEG-1. This replaces any contradictory statements in the current WD.

- In all cases, a skipped macroblock is only the result of a prediction, and all DCT coefficients are considered to be zero.
- In I- frame pictures or field pictures, all macroblocks are coded and there is no skipped macroblocks. The syntax element *macroblock_address_increment* is always equal to "1", except for the first macroblock of a slice, in which case it indicates the horizontal position of the slice.
- In P- frame pictures, a skipped macroblock is defined to be a frame-based predicted macroblock with a reconstructed frame motion vector equal to zero.
- In P- field pictures, a skipped macroblock is defined to be a field-based predicted macroblock with a reconstructed field motion vector equal to zero. The reference field for the prediction is the same parity field.
- In B- frame pictures, a skipped macroblock is defined to be a frame-based predicted macroblock with differential frame motion vector(s) equal to zero. The type of prediction (forward, backward or averaged) is the same as the prior macroblock.
- In B- field pictures, a skipped macroblock is defined to be a field-based predicted macroblock with differential field motion vector(s) equal to zero. The type of prediction (forward, backward or averaged) is the same as the prior macroblock. The reference field(s) for the prediction is (are) the same parity field(s).
- In B- frame or field pictures, a skipped macroblock shall not follow an intra-coded macroblock.

Modified Alternate Scan

The zig-zag scan pattern described in 8.1.2 is used as described in that section when "alternate_scan" in the picture header extension is "0" (or when the picture extension is not there - MPEG-1). However in the case that the "alternate_scan" flag in the picture header extension is "1" the following scan shall be used instead:

0	4	6	20	22	36	38	52
1	5	7	21	23	37	39	53
2	8	19	24	34	40	50	54
3	9	18	25	35	41	51	55
10	17	26	30	42	46	56	60

11	16	27	31	43	47	57	61
12	15	28	32	44	48	58	62
13	14	29	33	45	49	59	63

Modifed Transform coefficient VLC

The decoding of coefficients described in 7.2.8 is simplified as follows:

Two tables are provided as described in 7.2.81(sic). However the method of deciding which table to use is simplified.

In all non-intra macroblocks table 0 is used.

When "intra_vlc_format" is "0" table 0 is also used in intra macroblocks.

When "intra_vlc_format" is "1" table 1 is used in intra macroblocks.

Thus 7.2.811(sic) and 7.2.812(sic) are deleted.

7.2.82(sic) First DCT coefficient. When table 0 is used Table B-11 is the codes for the first coefficient in a block the VLC codes are modified as in Table B-11.

Modified Vector Decoding

Vector decoding returns to a scheme like that used in MPEG-1.

Table B-8 is replaced by the table B-8 shown in the lsat section of this document.:

Instructions from Requirements for WD editing.

(by Ken McCann and Jim Williams)

1) Replace subclause 0.3 with clause 6. lines 1 through 28. Of course, ignore the "6" on line 1.

2) Delete remainder of clause 6. You can just leave a null clause 6 if you wish for the current WD version to simplify clause numbering issues.

3) Begin clause 9 as follows:

"9 Profiles and levels

Profiles and levels provide a means of defining subsets of the ISO/IEC 11172-6 syntax and thereby the required decoder capabilities. A profile defines a subset of the tools provided by the11172-6 syntax. A level defines a set of constraints upon the allowed values of parameters within that syntax. Conformance tests will be carried out against defined profiles at defined levels.

In subsequent subclauses the constrained parts of the defined profiles and levels will be described. All syntactical elements and parameter values which are not explicitly constrained may take any of the possible values that are allowed by this Specification. A decoder shall be deemed to be conformant to a given profile at a given level if it is able to properly decode all allowed values of all syntactical elements as specified by that profile at that level. A bitstream shall be deemed to be conformant if it does not exceed the allowed range of allowed values and does not include disallowed syntactical elements."

4) Add the following constraints to the list of constrained elements and values.

- intra_dc_prediction *{Editor - surely intra_dc_precision ?)*
- qscale_type
- leak_factor_code
- dct_type

Use values which will be determined by answers from Video subgroup to Requirements subgroup's questions.

5) Add the following sentence to the end of the second paragraph of subclause 9.1.2.1.

"If the picture rate is greater than 25 pictures per second then the vertical_size_value shall be in the range <0:480].

6) Update the VBV buffer size limitation to 1,835,008 per Video channel

7) The value of profile_and_level_id for Main Level, Main Profile shall be.

0 100 1000

Modified Tables (Annex B)

Table B-1 is modified to re-introduce the entry "macro_block_stuffing". represented by VLC code word "0000 0001 111". This entry shall not be used if Sequence Header Extension is also present in the bit-stream (MPEG-2).

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

Variable length code	motion_code
0000 0011 001	-16
0000 0011 011	-15
0000 0011 101	-14
0000 0011 111	-13
0000 0100 001	-12
0000 0100 011	-11
0000 0100 11	-10
0000 0101 01	-9
0000 0101 11	-8
0000 0111	-7
0000 1001	-6
0000 1011	-5
0000 111	-4
0001 1	-3
0011	-2
011	-1
1	0
010	1
0010	2
0001 0	3
0000 110	4
0000 1010	5
0000 1000	6
0000 0110	7
0000 0101 10	8
0000 0101 00	9
0000 0100 10	10
0000 0100 010	11
0000 0100 000	12
0000 0011 110	13
0000 0011 100	14
0000 0011 010	15
0000 0011 000	16

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

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

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

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

NOTE: Table B-12 (next page) shows a sub-table. The entire "DCT coefficients table-1" is obtained by merging the table into Table B-11. *{As an editorial change the full table will be reproduced in later versions of the working draft rather than the sub-table.}*

Table B-12 DCT coefficients table-1 Sub-Table

variable length code	run	level
0110	EOB	
10s	0	1
010s	1	1
110s	0	2
00101s	2	1
0111s	0	3
00111s	3	1
000110s	4	1
00110s	1	2
000111s	5	1
0000110s	6	1
0000100s	7	1
11100s	0	4
0000111s	2	2
0000101s	8	1
1111000s	9	1
11101s	0	5
000101s	0	6
1111001s	1	3
00100110s	3	2
1111010s	10	1
00100001s	11	1
00100101s	12	1
00100100s	13	1
000100s	0	7
00100111s	1	4
11111100s	2	3
11111101s	4	2
000000100s	5	2
000000101s	14	1
000000111s	15	1
0000001101s	16	1
1111011s	0	8
1111100s	0	9
00100011s	0	10
00100010s	0	11
00100000s	1	5
0000001100s	2	4
11111010s	0	12
11111011s	0	13
11111110s	0	14
11111111s	0	15

On page 67, Table B-13. The existing table is correct and is documents "12-bit Escape" The following is appended and documents "8+8bit Escape" (MPEG-1 method).

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

fixed length code	level
forbidden	-256
1000 0000 0000 0001	-255
1000 0000 0000 0010	-254
...	
1000 0000 0111 1111	-129
1000 0000 1000 0000	-128
1000 0001	-127
1000 0010	-126
...	...
1111 1110	-2
1111 1111	-1
forbidden	0
0000 0001	1
...	...
0111 1111	127
0000 0000 1000 0000	128
0000 0000 1000 0001	129
...	
0000 0000 1111 1111	255

1

2

Annex C

Video buffering verifier

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

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

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

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

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

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

4 *This item applies to cases that all pictures are coded and transmitted.* All of the data for the picture which has been in the buffer longest is instantaneously removed. Then after a period of time, t, specified by the picture_rate in the sequence header and the picture_structure and the number_of_field_displayed_code in the picture header of the last picture decoded, all of the data for the picture which (at that time) has been in the buffer longest is instantaneously removed. The period of time, t, is defined as follows:

$$t = \frac{1}{\text{fields\_per\_picture} * P} * \text{field\_count}$$

Where

fields_per_picture = 2 when picture_structure equals 11 (frame structure).

or fields_per_picture = 1 when picture_structure takes any other value (field structure)

P = Number of pictures per second calculated from the picture_rate field.

field_count is the number of displayed fields calculated from the number_of_field_displayed_code in the picture header of the last picture decoded

Sequence header and group of picture layer data elements which immediately precede a picture are removed at the same time as that picture. The VBV is examined immediately before removing any data (sequence header data, group of picture layer or picture) and immediately after each picture is removed. Each time the VBV is examined its occupancy shall lie between zero bits and B bits where B is the size of the VBV buffer indicated by vbv_buffer_size in the sequence header.

5 *This item applies to cases that some pictures are not coded nor transmitted.* Encoders may wish to realize low buffering delay by allowing pictures to be dropped occasionally. It will regulate its data generation by setting a VBV buffer size B smaller than is normally used. Typically it will be a little larger than the average number of bits per picture period. In this case the VBV operates as follows.

All of the data for the picture which has been in the buffer longest is instantaneously removed. Then after a period of time, t, specified by the picture_rate in the sequence header and the picture_structure and the number_of_field_displayed_code in the picture header of the last picture decoded, all of the data for the picture which (at that time) has been in the buffer longest is instantaneously removed. The period of time, t, is defined as follows:

$$t = \frac{1}{\text{fields\_per\_picture} * P} * \text{field\_count}$$

Where

fields_per_picture = 2 when picture_structure equals 11 (frame structure).

or fields_per_picture = 1 when picture_structure takes any other value (field structure)

P = Number of pictures per second calculated from the picture_rate field.

field_count is the number of displayed fields calculated from the number_of_field_displayed_code in the picture header of the picture about to be decoded

Sequence header, group of picture layer data elements and headers which immediately precede a picture are removed at the same time as that picture.

At some times when a picture is expected to be removed from the VBV buffer there will be not be sufficient data in the buffer to remove a complete picture. In this case the data that is available corresponding to the picture that has been in the buffer longest is removed, and the previously decoded pictures may be displayed again. When a complete picture has been removed it can be decoded.

During this state when a complete picture is not available to be decoded, at the times when the VBV buffer is examined its occupancy shall lie between zero and B bits. At the other times when the VBV is examined its occupancy shall lie between zero bits and B bits where B is the size of the VBV buffer indicated by vbv_buffer_size in the sequence header.

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

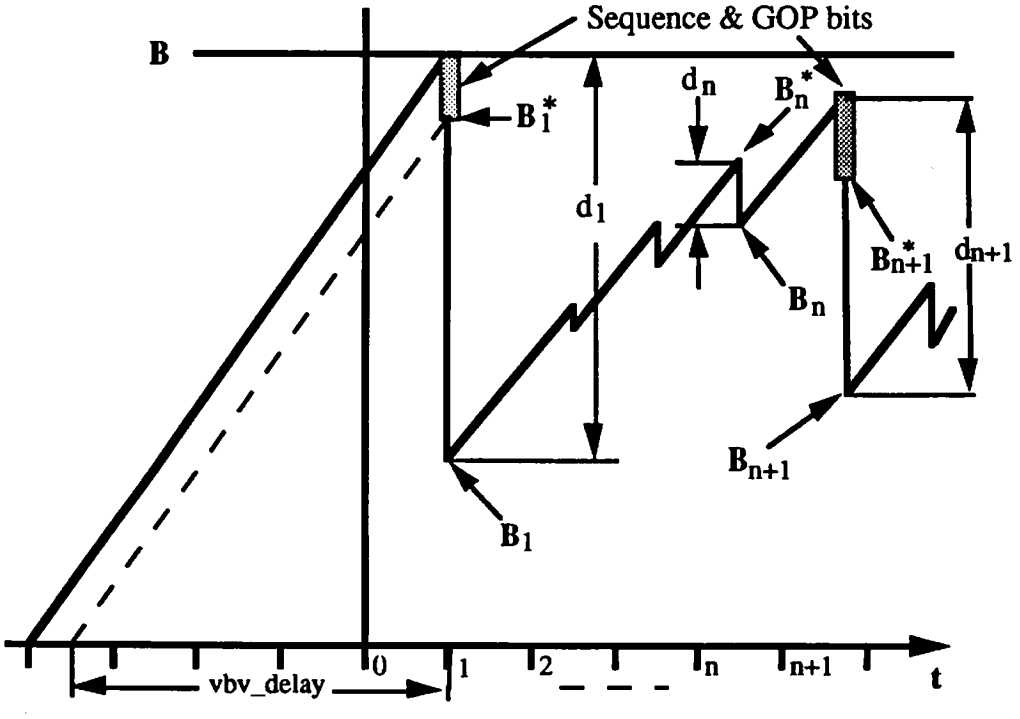

Figure C-1 VBV Buffer Occupancy

1