ARCHIVED MARCH 21, 2006

SMPTE STANDARD

for Television — MPEG-2 Video Recoding Data Set — Compressed Stream Format

Table of contents

1 Scope

This standard specifies the stream format of the MPEG-2 recoding data set for the representation of compressed ISO/IEC 13818-2 MPEG coding information, as used in applications requiring transport systems of reduced data capacity.

The coding information is derived from an ISO/IEC 13818-2 compliant MPEG bit stream during the decoding process, as described in ISO/IEC 13818-2.

The information based on this stream format may be transported by various means; for example, the elementary stream format defined in SMPTE 328M.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE 327M-2000, Television — MPEG-2 Video Recoding Data Set

ITU-T H.262, ISO/IEC 13818-2:1996, Information Technology — Generic Coding of Moving Pictures and Associated Audio Information: Video

3 General

The main application of this standard is to preserve the quality of the video signal when cascading decoders and coders for editing or transcoding purposes by feeding forward previous coding decisions.

The MPEG-2 recoding data set in combination with an MPEG-2 decoded or partially decoded picture is effective for implementing editing or transcoding of the MPEG-2 bit stream. There are, however, applications in which the transmission of all the recoding data set is not possible. Some current equipment may have restricted capacity for the transmission of the recoding data. This limitation has an impact on subsequent compression stages which can make use of the MPEG-2 recoding process.

In order to decrease the bit rate for the recoding data set, the MPEG-2 recoding data set is converted into an MPEG-like stream, which is called the compressed stream format of the MPEG-2 recoding set. This standard defines this stream format.

The compressed stream format of the MPEG-2 recoding set much resembles an MPEG-2 video stream, except that the compressed stream format of the MPEG-2 recoding set does not have DCT coefficients which occupy the major part of an MPEG video stream.

Copyright © 2000 by THE SOCIETY OF
MOTION PICTURE AND TELEVISION ENGINEERS
595 W. Hartsdale Ave., White Plains, NY 10607
(914) 761-1100

Approved
January 20, 2000

By adopting an MPEG-like form for the compressed stream format of the MPEG-2 recoding set, use is made of an efficient compression for this data set for storage and transmission operations. Moreover, it may be possible to reuse the MPEG decoding devices to decode this compressed stream format of the MPEG-2 recoding set.

The compressed format of the MPEG-2 recoding set syntactical structure is modified by red_bw_flag and red_bw_indicators. The selection of one of the variations of the syntactical structure will be determined by the application by taking into account a balance between bit rate and the number of elements in the recoding data set.

This standard does not describe the transport mechanism, but describes the stream structure of the compressed stream format of the MPEG-2 recoding set. This stream format is independent of application, and all the transport information in the reduced bandwidth recoding data transportation system should be based on this stream.

The transport mechanism depends on application, which should be defined in other standards documents. The first of these is the MPEG-2 elementary stream editing information, SMPTE 328M.

4 Definitions

4.1 bit stream: An ordered series of bits conforming to ISO/IEC 13818-2.

4.2 cascading: This describes the process where video that has once been coded (compressed) is subsequently decoded and coded once more. This cascading step could carry on for any number of generations.

4.3 coding: This is the process by which an uncompressed video sequence is compressed to a bit stream that conforms to the ISO/IEC 13818-2 standard.

4.4 compressed stream format of the MPEG-2 recoding data set: This is the compressed information of the recoding data set which is used in reduced bandwidth recoding data transportation systems.

4.5 decoder: A compressed bit stream decoder that complies with the ISO/IEC 13818-2 standard.

4.6 macroblock: Defined in the ISO/IEC 13818-2 standard as a block of 16×16 luminance pixels.

4.7 MPEG profile/level: As defined in ISO/IEC 13818-2.

4.8 picture: As defined in ISO/IEC 13818-2.

4.9 recoding data set: The set of information derived from an ISO/IEC 13818-2 compliant stream which, when accompanied by decoded or partially decoded video information, assists in the recreation of the original MPEG-2 stream.

4.10 reduced bandwidth recoding data transportation system: This is the system that does not have enough bandwidth capacity for transporting a full set of the recoding data set, but is able to transport the compressed stream format of the MPEG-2 recoding data set.

4.11 reduced bandwidth stream: The highest syntactic structure of the reduced MPEG-2 recoding data set. It contains the recoding information for one picture.

4.12 stripe: A horizontal row of macroblocks spanning the full width of the decoded picture (stripes are numbered from top to bottom starting at zero).

4.13 transcoding: A conversion within the MPEG-2 stream domain, such as bit rate changing or changing the group of pictures (GOP) structure.

5 Compressed stream format of the MPEG-2 recoding data set

5.1 Compressed stream format of the MPEG-2 recoding set syntax

The compressed stream format of the MPEG-2 recoding set is a subset of the ISO/IEC 13818-2 stream and is extracted from the original MPEG-2 video stream.

5.1.1. Start codes

Start codes are specific bit patterns that do not otherwise occur in the compressed stream format of the MPEG-2 recoding data set.

Each start code consists of a start code prefix followed by a start code value. The start code prefix is a string of 23 bits with the value zero followed by a single bit with the value one.

The start code value is an eight-bit integer that identifies the type of start code. Most types of start code have just one start code value. However, slice_start_code is represented by many start code values. In this case, the start code value is the slice_vertical_position for the slice.

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

Table 1 defines the start code values for the start codes used in the compressed stream format of MPEG-2 recoding set.

The use of the start codes is defined in the following syntax description with the exception of the sequence_error_code. The sequence_error_code has been allocated for use by a media interface to indicate where uncorrectable errors have been detected (see tables 2 - 22).

5.2 Compressed stream format of the MPEG-2 recoding set semantics

Almost all of the syntax elements in the compressed stream format of the MPEG-2 recoding set have the same meaning as those defined in ISO/IEC 13818-2. Some elements are newly defined and some have a different meaning. Definitions for these elements are given below.

5.2.1 Recoding stream information

Re_coding_stream_info_ID: A 16-bit integer that is used for identification of re_coding_stream_info(). The value is 1001 0001 1110 1100 (0×91 ec).

red_bw_flag: A 1-bit flag. Its definition is given in SMPTE 327M.

red_bw_indicator: A 2-bit integer defined in SMPTE 327M.

num_other_bits: A 7-bit integer. This syntax element is defined in SMPTE 327M.

num_mv_bits: An 8-bit integer defined in SMPTE 327M.

num_coef_bits: A 14-bit integer. This syntax element is defined in SMPTE 327M.

num_coef_bits, num_mv_bits, and num_other_bits exist when red_bw_flag is 0. These three data sets are repeated by the number of macroblocks. The data sets match the macroblocks respectively, which are scanned from left top to right bottom horizontally first, as in MPEG-2.

The value of red_bw_indicator specifies the level of reduced information and it ranges from 0 to 3. As the value is increased, the more information is reduced.

As the compressed stream format of the MPEG-2 recoding data set syntax structure depends on this red_bw_flag and red_bw_indicator, the decoder shall recognize these elements in the current bit stream. Then it shall parse the bit stream accordingly.

The details of each red_bw_indicator value are described as follows:

– Indicator 0: This stream transports all the recoding data set, except that the information such as num_coef_bits, num_mv_bits, and num_other_bits are not transported. All the recoding data set could be derived from this stream. Therefore, the position of slice_start, skipped_mb, and mb_quant should be the same as in the original stream.

– Indicator 1: This stream is the same as indicator 0, except that coded_block_pattern() information is not transported. This stream is not guaranteed to have the values of slice_start, skipped_mb, and mb_quant to be the same as in the original stream.

– Indicator 2: This stream transports only quantizer_scale_code information in slice() and macroblock(). Other information which is carried in slice() and macroblock() is not guaranteed to be the same as in the original stream.

– Indicator 3: The information of slice() and macroblock() is not transported by this stream.

The relation between red_bw_indicator and the elements of the MPEG-2 recoding data set described in SMPTE 327M is shown in table 23.

Table 1 – Start code value

Name	Start code value (hexadecimal)
picture_start_code	00
slice_start_code	01 through AF
reserved	B0
reserved	B1
user_data_start_code	B2
sequence_header_code	B3
sequence_error_code	B4
extension_start_code	B5
reserved	B6
sequence_end_code	B7
group_start_code	B8

Table 2 – Compressed stream format of the MPEG-2 recoding set

compressed_stream_format_of_MPEG_2_re_coding_set() {	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
sequence_extension()		
extension_and_user_data(0)		
if (nextbits() == group_start_code) {		
group_of_pictures_header()		
extension_and_user_data(1)		
}		
picture_header()		
picture_coding_extension()		
re_coding_stream_info()		
extensions_and_user_data(2)		
if (! red_bw_flag || (red_bw_indicator <=2))		
picture_data()		
sequence_end_code	32	bslbf
}		

Table 3 – Sequence header

sequence_header() {	No. of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_value	10	uimsbf
constrained_parameters_flag	1	bslbf
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
next_start_code()		
}		

Table 4 – 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
progressive_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_extension	8	uimsbf
low_delay	1	uimsbf
frame_rate_extension_n	2	uimsbf
frame_rate_extension_d	5	uimsbf
next_start_code()		
}		

Table 5 – Extension and user data

extension_and_user_data(i) {	No. of bits	Mnemonic
while ((nextbits() == extension_start_code) ||		
(nextbits() == user_data_start_code)) {		
if ((i ! == 1) && (nextbits() == extension_start_code))		
extension_data(i)		
if (nextbits() == user_data_start_code)		
user_data()		
}		
}		

Table 6 – Extension data

extension_data(i) {	No. of bits	Mnemonic
while (nextbits() == extension_start_code) {		
extension_start_code	32	bslbf
if (i == 0) /* follows sequence_extension() */		
sequence_display_extension()		
/* NOTE – i never takes the value 1 because extension_data()		
never follows a group_of_pictures_header() */		
if (i == 2) { /* follows picture_coding_extension() */		
if (nextbits() == "Quant Matrix Extension ID")		
quant_matrix_extension()		
else if (nextbits() == "Copyright Extension ID")		
copyright_extension()		
else		
picture_display_extension()		
}		
}		
}		

Table 7 – Sequence display extension

sequence_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
video_format	3	uimsbf
color_description	1	uimsbf
if (color_description) {		
color_primaries	8	uimsbf
transfer_characteristics	8	uimsbf
matrix_coefficients	8	uimsbf
}		
display_horizontal_size	14	uimsbf
marker_bit	1	bslbf
display_vertical_size	14	uimsbf
next_start_code()		
}		

Table 8 – Quant matrix extension

quant_matrix_extension() {	No. of bits	Mnemonic
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	uimsbf
if (load_chroma_intra_quantizer_matrix)		
chroma_intra_quantizer_matrix[64]	8*64	uimsbf
load_chroma_non_intra_quantizer_matrix	1	uimsbf
if (load_chroma_non_intra_quantizer_matrix)		
chroma_non_intra_quantizer_matrix[64]	8*64	uimsbf
next_start_code()		
}		

Table 9 – Picture display extension

picture_display_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
for (i = 0; i<number_of_frame_center_offsets; i++) {		
frame_center_horizontal_offset	16	simsbf
marker_bit	1	bslbf
frame_center_vertical_offset	16	simsbf
marker_bit	1	bslbf
}		
next_start_code()		
}		

Table 10 – Copyright extension

copyright_extension() {	No. of bits	Mnemonic
extension_start_code_identifier	4	uimsbf
copyright_flag	1	bslbf
copyright_identifier	8	uimsbf
original_or_copy	1	bslbf
reserved	7	uimsbf
marker_bit	1	bslbf
copyright_number_1	20	uimsbf
marker_bit	1	bslbf
copyright_number_2	22	uimsbf
marker_bit	1	bslbf
copyright_number_3	22	uimsbf
next_start_code()		
}		

Table 11 – 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	uimsbf
}		
next_start_code()		
}		

Table 12 – Group of pictures header

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

Table 13 – 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	bslbf
forward_f_code	3	bslbf
}		
if (picture_coding_type == 3) {		
full_pel_backward_vector	1	bslbf
backward_f_code	3	bslbf
}		
while (nextbits() == 1) {		
extra_bit_picture /* with the value 1 */	1	uimsbf
extra_information_picture	8	uimsbf
}		
extra_bit_picture /* with the value 0 */	1	uimsbf
next_start_code()		
}		

Table 14 – Picture coding extension

picture_coding_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
f_code[0][0] /* forward horizontal */	4	uimsbf
f_code[0][1] /* forward vertical */	4	uimsbf
f_code[1][0] /* backward horizontal */	4	uimsbf
f_code[1][1] /* backward vertical */	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_vic_format	1	uimsbf
alternate_scan	1	uimsbf
repeat_first_field	1	uimsbf
chroma_420_type	1	uimsbf
progressive_frame	1	uimsbf
composite_display_flag	1	uimsbf
if (composite_display_flag) {		
v_axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	uimsbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code()		
}		

Table 15 – Recoding stream information

re_coding_stream_info() {	No. of bits	Mnemonic
user_data_start_code	32	bslbf
re_coding_stream_info_ID	16	bslbf
red_bw_flag	1	uimsbf
if (red_bw_flag)		
red_bw_indicator	2	uimsbf
if (! red_bw_flag) {		
for (i=0; i<number_of_macroblock; i++) {		
marker_bit	1	bslbf
num_other_bits	7	uimsbf
marker_bit	1	bslbf
num_mv_bits	8	uimsbf
marker_bit	1	bslbf
num_coef_bits	14	uimsbf
}		
}		
next_start_code()		
}		

Table 16 – Picture data

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

Table 17 – Slice

slice() {	No. of bits	Mnemonic
slice_start_code	32	bslbf
quantizer_scale_code	5	uimsbf
if (nextbits() == 1) {		
intra_slice_flag	1	bslbf
intra_slice	1	uimsbf
reserved_bits	7	uimsbf
while (nextbits() == 1) {		
extra_bit_slice /* with the value 1 */	1	uimsbf
extra_information_slice	8	uimsbf
}		
}		
extra_bit_slice /* with the value 0 */	1	uimsbf
do {		
macroblock()		
} while (nextbits() ! = 000 0000 0000 0000 0000 0000)		
next_start_code()		
}		

Table 18 – Macroblock

macroblock() {	No. of bits	Mnemonic
while (nextbits() == 0000 0001 000)		
macroblock_escape	11	bslbf
macroblock_address_increment	1-11	vlclbf
macroblock_modes()		
if (macroblock_quant)		
quantizer_scale_code	5	uimsbf
if (! red_bw_flag ||		
(red_bw_flag && (red_bw_indicator <=1))) {		
if (macroblock_motion_forward ||		
(macroblock_intra && concealment_motion_vectors))		
motion_vectors(0)		
if (macroblock_motion_backward)		
motion_vectors(1)		
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	bslbf
}		
if (macroblock_pattern &&		
(! red_bw_flag ||		
(red_bw_flag && (red_bw_indicator == 0))))		
coded_block_pattern()		
}		

Table 19 – Macroblock modes

macroblock_modes() {	No. of bits	Mnemonic
macroblock_type	1-9	vlclbf
if (! red_bw_flag ||		
(red_bw_flag && (red_bw_indicator <=1))) {		
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 picture) &&)		
(frame_pred_frame_dct == 0) &&		
(dct_type_flag == 1) &&		
(! red_bw_flag ||		
(red_bw_flag && (red_bw_indicator <= 1)))) {		
dct_type	1	uimsbf
}		
}		

Table 20 – Motion vectors

motion_vectors (s) {	No. of bits	Mnemonic
if (motion_vector_count == 1) {		
if ((mv_format == field) && (dmv ! = 1))		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0,s)		
} else {		
motion_vertical_field_select[0][s]	1	uimsbf
motion_vector(0,s)		
motion_vertical_field_select[1][s]	1	uimsbf
motion_vector(1,s)		
}		
}		

Table 21 – Motion vector

motion_vector (r,s) {	No. of bits	Mnemonic
motion_code[r][s][0]	1-11	vlclbf
if ((f_code[s][0] ! = 1) && (motion_code[r][s][0] ! = 0))		
motion_residual[r][s][0]	1-8	uimsbf
if (dmv == 1)		
dmvector[0]	1-2	vlclbf
motion_code[r][s][1]	1-11	vlclbf
if ((f_code[s][1] ! = 1) && (motion_code[r][s][1] ! = 0))		
motion_residual[r][s][1]	1-8	uimsbf
if (dmv == 1)		
dmvector[1]	1-2	vlclbf
}		

Table 22 – Coded block pattern

coded_block_pattern() {	No. of bits	Mnemonic
coded_block_pattern_420	3-9	vlclbf
if (chroma_format == 4:2:2)		
coded_block_pattern_1	2	uimsbf
if (chroma_format == 4:4:4)		
coded_block_pattern_2	6	uimsbf
}		

Table 23 – Reduced bandwidth indicators

red_bw_indicator_	num_coef_bits, num_mv_bits, num_other_bits	q_scale_code, q_scale_type	motion_type, mv_vert_field_sel[r][s], mv[r][s][t]	mb_mfwd, mb_mbwd	mb_pattern	coded_block_pattern	mb_intra	slice_start	dct_type	mb_quant	skipped_mb
Indicator 0	0	1	1	1	1	1	1	1	1	1	1
Indicator 1	0	1	1	1	1	0	1	X	1	X	X
Indicator 2	0	1	0	X	X	0	X	X	0	X	X
Indicator 3	0	0	0	0	0	0	0	0	0	0	0

NOTE – 0 = This information is not present.
1 = This information is present and corresponds to that of the original data stream.
X = This information may be present, but is not guaranteed to correspond to that of the original data stream.

5.2.2 Picture data

Picture_data() consists of more than zero slice(). When red_bw_indicator is 3, picture data() is not transported by this stream. It means that this compressed stream format of the MPEG-2 recoding set is intended not to transport all syntax elements below picture_data(), but to transport picture_type, etc.

5.2.3 Slice

When red_bw_flag is 0 or red_bw_indicator is less than or equal to 2, the elements in slice() exist. However, the validity of slice_start_code depends on red_bw_indicator.

When red_bw_flag is 0 or red_bw_indicator is equal to 0, slice start information shall be the same as in the original stream.

5.2.4 Macroblock

The elements in macroblock() are varied by red_bw_indicator, although macroblock_escape, macroblock_address_increment, and macroblock_modes() always exist. However, the validity of macroblock_escape and macroblock_address_increment depend on red_bw_flag and red_bw_indicator. When red_bw_flag is 0 or red_bw_indicator is equal to 0, skipped_mb information derived from macroblock_escape and macro-block_address_increment shall be the same as in the original stream.

When red_bw_indicator is of value equal to or more than 2, motion_vectors() shall not exist.

When red_bw_indicator is equal to or more than 1, coded_block_pattern() shall not exist.

5.2.5 Macroblock modes

The elements in macroblock_modes() are varied by red_bw_flag and red_bw_indicator, although macroblock_type always exists.

When red_bw_indicator is equal to or more than 2, frame_motion_type, field_motion_type, and dct_type shall not exist.

The validity of the parameter derived from macroblock_type depends on red_bw_flag and red_bw_indicator.

macroblock_quant – When red_bw_flag is 0 or red_bw_indicator is 0, macroblock_quant is the same as in the original stream.

When red_bw_flag is set to 1 and red_bw_indicator is 1 or 2, this specifies the existence of quantizer_scale_code in macroblock(). The data may not be the same as in the original stream.

macroblock_motion_forward, macroblock_motion_backward – When red_bw_flag is 0 or red_bw_indicator is 0 or 1, macroblock_motion_forward and macroblock_motion_backward have the same meaning as in the original stream. Otherwise, these are not guaranteed

macroblock_pattern – When red_bw_flag is 0 or red_bw_indicator is 0, macroblock_pattern is the same as in the original stream. When red_bw_indicator is 1, this specifies the existence of dct_type. When red_bw_indicator is 2, this information is not guaranteed.

macroblock_intra – When red_bw_flag is 0 or red_bw_indicator is 0 or 1, macroblock_intra is the same as in the original stream. Otherwise, this information is not guaranteed.

Annex A (informative)
Bibliography

SMPTE 308M-1998, Television — MPEG-2 4:2:2 Profile at High Level

SMPTE 328M-2000, Television — MPEG-2 Video Elementary Stream Editing information

ITU-T H.222.0, ISO/IEC 13818-1:1996, Information Technology — Generic Coding of Moving Pictures and Associated Audio Information: Systems