where

c1 is a real-valued constant valid for all i

c2 is a real-valued constant valid for all i

i is the index in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 multiplexed stream of the byte containing the final bit of any system_clock_reference field in the stream

CSPS_flag – The CSPS_flag is a 1-bit field. If its value is set to '1' the Program Stream meets the constraints defined in 2.7.9.

system_audio_lock_flag – The system_audio_lock_flag is a 1-bit field indicating that there is a specified, constant rational relationship between the audio sampling rate and the system_clock_frequency in the system target decoder. The system_clock_frequency is defined in 2.5.2.1 and the audio sampling rate is specified in ISO/IEC 13818-3. The system_audio_lock_flag may only be set to '1' if, for all presentation units in all audio elementary streams in the Program Stream, the ratio of system_clock_frequency to the actual audio sampling rate, SCASR, is constant and equal to the value indicated in the following table at the nominal sampling rate indicated in the audio stream.

$$SCASR = \frac{system\_clock\_frequency}{audio\_sample\_rate\_in\_the\_P\text{–}STD} \qquad (2\text{-}24)$$

The notation $\frac{X}{Y}$ denotes real division.

Nominal audio sampling frequency (kHz)	16	32	22,05	44,1	24	48
SCASR	27 000 000 / 16 000	27 000 000 / 32 000	27 000 000 / 22 050	27 000 000 / 44 100	27 000 000 / 24 000	27 000 000 / 48 000

system_video_lock_flag – The system_video_lock_flag is a 1-bit field indicating that there is a specified, constant rational relationship between the video frame rate and the system clock frequency in the system target decoder. Subclause 2.5.2.1 defines system_clock_frequency and the video frame rate is specified in ITU-T Rec. H.262 | ISO/IEC 13818-2. The system_video_lock_flag may only be set to '1' if, for all presentation units in all video elementary streams in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program, the ratio of system_clock_frequency to the actual video frame rate, SCFR, is constant and equal to the value indicated in the following table at the nominal frame rate indicated in the video stream.

$$SCFR = \frac{system\_clock\_frequency}{frame\_rate\_in\_the\_P\text{–}STD} \qquad (2\text{-}25)$$

Nominal frame rate (Hz)	23,976	24	25	29,97	30	50	59,94	60
SCFR	1 126 125	1 125 000	1 080 000	900 900	900 000	540 000	450 450	450 000

The values of the ratio SCFR are exact. The actual frame rate differs slightly from the nominal rate in cases where the nominal rate is 23,976, 29,97, or 59,94 frames per second.

video_bound – The video_bound is a 5-bit integer in the inclusive range from 0 to 16 and is set to a value greater than or equal to the maximum number of ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 11172-2 streams in the Program Stream of which the decoding processes are simultaneously active. For the purpose of this subclause, the decoding process of an ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 11172-2 video stream is active if the P-STD buffer is not empty, or if a Presentation Unit is being presented in the P-STD model, or if the reorder buffer is not empty.

packet_rate_restriction_flag – The packet_rate_restriction_flag is a 1-bit flag. If the CSPS flag is set to '1', the packet_rate_restriction_flag indicates which constraint is applicable to the packet rate, as specified in 2.7.9. If the CSPS flag is set to value of '0', then the meaning of the packet_rate_restriction_flag is undefined.

reserved_bits – This 7-bit field is reserved for future use by ISO/IEC. Until otherwise specified by ITU-T | ISO/IEC it shall have the value '111 1111'.

stream_id – The stream_id is an 8-bit field that indicates the coding and elementary stream number of the stream to which the following P-STD_buffer_bound_scale and P-STD_buffer_size_bound fields refer.

If stream_id equals '1011 1000' the P-STD_buffer_bound_scale and P-STD_buffer_size_bound fields following the stream_id refer to all audio streams in the Program Stream.

If stream_id equals '1011 1001' the P-STD_buffer_bound_scale and P-STD_buffer_size_bound fields following the stream_id refer to all video streams in the Program Stream.

If the stream_id takes on any other value it shall be a byte value greater than or equal to '1011 1100' and shall be interpreted as referring to the stream coding and elementary stream number according to Table 2-18.

Each elementary stream present in the Program Stream shall have its P-STD_buffer_bound_scale and P-STD_buffer_size_bound specified exactly once by this mechanism in each system header.

P-STD_buffer_bound_scale – The P-STD_buffer_bound_scale is a 1-bit field that indicates the scaling factor used to interpret the subsequent P-STD_buffer_size_bound field. If the preceding stream_id indicates an audio stream, P-STD_buffer_bound_scale shall have the value '0'. If the preceding stream_id indicates a video stream, P-STD_buffer_bound_scale shall have the value '1'. For all other stream types, the value of the P-STD_buffer_bound_scale may be either '1' or '0'.

P-STD_buffer_size_bound – The P-STD_buffer_size_bound is a 13-bit unsigned integer defining a value greater than or equal to the maximum P-STD input buffer size, BS_n, over all packets for stream n in the Program Stream. If P-STD_buffer_bound_scale has the value '0', then P-STD_buffer_size_bound measures the buffer size bound in units of 128 bytes. If P-STD_buffer_bound_scale has the value '1', then P-STD_buffer_size_bound measures the buffer size bound in units of 1024 bytes. Thus:

if (P-STD_buffer_bound_scale == 0)

$$BS_n \leq P - STD\_buffer\_size\_bound \times 128$$

else

$$BS_n \leq P - STD\_buffer\_size\_bound \times 1024$$

2.5.3.7 Packet layer of Program Stream

The packet layer of the Program Stream is defined by the PES packet layer in 2.4.3.6.

2.5.4 Program Stream map

The Program Stream Map (PSM) provides a description of the elementary streams in the Program Stream and their relationship to one another. When carried in a Transport Stream this structure shall not be modified. The PSM is present as a PES packet when the stream_id value is 0xBC (refer to Table 2-18).

NOTE – This syntax differs from the PES packet syntax described in 2.4.3.6.

Definition for the descriptor() fields may be found in 2.6.

2.5.4.1 Syntax of Program Stream map

See Table 2.35.

Table 2-35 – Program Stream map

Syntax	No. of bits	Mnemonic
program_stream_map() {		
packet_start_code_prefix	24	bslbf
map_stream_id	8	uimsbf
program_stream_map_length	16	uimsbf
current_next_indicator	1	bslbf
reserved	2	bslbf
program_stream_map_version	5	uimsbf
reserved	7	bslbf
marker_bit	1	bslbf
program_stream_info_length	16	uimsbf
for (i = 0; i < N; i++) {		
descriptor()		
}		
elementary_stream_map_length	16	uimsbf
for (i = 0; i < N1; i++) {		
stream_type	8	uimsbf
elementary_stream_id	8	uimsbf
elementary_stream_info_length	16	uimsbf
for (i = 0; i < N2; i++) {		
descriptor()		
}		
}		
CRC_32	32	rpchof
}		

2.5.4.2 Semantic definition of fields in Program Stream map

packet_start_code_prefix – The packet_start_code_prefix is a 24-bit code. Together with the map_stream_id that follows it constitutes a packet start code that identifies the beginning of a packet. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001 in hexadecimal).

map_stream_id – This is an 8-bit field whose value shall be 0xBC.

program_stream_map_length – The program_stream_map_length is a 16-bit field indicating the total number of bytes in the program_stream_map immediately following this field. The maximum value of this field is 1018 (0x3FA).

current_next_indicator – This is a 1-bit field, when set to '1' indicates that the Program Stream Map sent is currently applicable. When the bit is set to '0', it indicates that the Program Stream Map sent is not yet applicable and shall be the next table to become valid.

program_stream_map_version – This 5-bit field is the version number of the whole Program Stream Map. The version number shall be incremented by 1 modulo 32 whenever the definition of the Program Stream Map changes. When the current_next_indicator is set to '1', then the program_stream_map_version shall be that of the currently applicable Program Stream Map. When the current_next_indicator is set to '0', then the program_stream_map_version shall be that of the next applicable Program Stream Map.

program_stream_info_length – The program_stream_info_length is a 16-bit field indicating the total length of the descriptors immediately following this field.

marker_bit – A marker_bit is a 1-bit field that has the value '1'.

elementary_stream_map_length – This is a 16-bit field specifying the total length, in bytes, of all elementary stream information in this program stream map. It includes the stream_type, elementary_stream_id, and elementary_stream_info_length fields.

stream_type – This 8-bit field specifies the type of the stream according to Table 2-29. The stream_type field shall only identify elementary streams contained in PES packets. A value of 0x05 is prohibited.

elementary_stream_id – The elementary_stream_id is an 8-bit field indicating the value of the stream_id field in the PES packet headers of PES packets in which this elementary stream is stored.

elementary_stream_info_length – The elementary_stream_info_length is a 16-bit field indicating the length in bytes of the descriptors immediately following this field.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire program stream map.

2.5.5 Program Stream directory

The directory for an entire stream is made up of all the directory data carried by Program Stream Directory packets identified with the directory_stream_id. The syntax for program_stream_directory packets is defined in Table 2-36.

NOTE 1 – This syntax differs from the PES packet syntax described in 2.4.3.6.

Directory entries may be required to reference I-pictures in a video stream as defined in ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 11172-2. If an I-picture that is referenced in a directory entry is preceded by a sequence header with no intervening picture headers, the directory entry shall reference the first byte of the sequence header. If an I-picture that is referenced in a directory entry is preceded by a group of pictures header with no intervening picture headers and no immediately preceding sequence header, the directory entry shall reference the first byte of the group of pictures header. Any other picture that a directory entry references shall be referenced by the first byte of the picture header.

NOTE 2 – It is recommended that I-pictures immediately following a sequence header should be referenced in directory structures so that the directory contains an entry at every point where the decoder may be reset completely.

Directory references to audio streams as defined in ISO/IEC 13818-3 and ISO/IEC 11172-3 shall be the syncword of the audio frame.

NOTE 3 – It is recommended that the distance between referenced access units not exceed half a second.

Access units shall be referenced in a program_stream_directory packet in the same order that they appear in the bitstream.

2.5.5.1 Syntax of Program Stream directory packet

See Table 2-36.

2.5.5.2 Semantic definition of fields in Program Stream directory

packet_start_code_prefix – The packet_start_code_prefix is a 24-bit code. Together with the stream_id that follows, it constitutes a packet start code that identifies the beginning of a packet. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001 in hexadecimal).

directory_stream_id – This 8-bit field shall have a value '1111 1111' (0xFF).

PES_packet_length – The PES_packet_length is a 16-bit field indicating the total number of bytes in the program_stream_directory immediately following this field (refer to Table 2-18).

number_of_access_units – This 15-bit field is the number of access_units that are referenced in this Directory PES packet.

prev_directory_offset – This 45-bit unsigned integer gives the byte address offset of the first byte of the packet start code of the previous Program Stream Directory packet. This address offset is relative to the first byte of the start code of the packet which contains this previous_directory_offset field. The value '0' indicates that there is no previous Program Stream Directory packet.

next_directory_offset – This 45-bit unsigned integer gives the byte address offset of the first byte of the packet start code of the next Program Stream Directory packet. This address offset is relative to the first byte of the start code of the packet which contains this next_directory_offset field. The value '0' indicates that there is no next Program Stream Directory packet.

packet_stream_id – This 8-bit field is the stream_id of the elementary stream that contains the access unit referenced by this directory entry.

PES_header_position_offset_sign – This 1-bit field is the arithmetic sign for the PES_header_position_offset described immediately following. A value of '0' indicates that the PES_header_position_offset is a positive offset. A value of '1' indicates that the PES_header_position_offset is a negative offset.

PES_header_position_offset – This 44-bit unsigned integer gives the byte offset address of the first byte of the PES packet containing the access unit referenced. The offset address is relative to the first byte of the start-code of the packet containing this PES_header_position_offset field. The value '0' indicates that no access unit is referenced.

reference_offset – This 16-bit field is an unsigned integer indicating the position of the first byte of the referenced access unit, measured in bytes relative to the first byte of the PES packet containing the first byte of the referenced access unit.

Table 2-36 – Program Stream directory packet

Syntax	No. of bits	Mnemonic
directory_PES_packet(){		
packet_start_code_prefix	24	bslbf
directory_stream_id	8	uimsbf
PES_packet_length	16	uimsbf
number_of_access_units	15	uimsbf
marker_bit	1	bslbf
prev_directory_offset[44..30]	15	uimsbf
marker_bit	1	bslbf
prev_directory_offset[29..15]	15	uimsbf
marker_bit	1	bslbf
prev_directory_offset[14..0]	15	uimsbf
marker_bit	1	bslbf
next_directory_offset[44..30]	15	uimsbf
marker_bit	1	bslbf
next_directory_offset[29..15]	15	uimsbf
marker_bit	1	bslbf
next_directory_offset[14..0]	15	uimsbf
marker_bit	1	bslbf
for (i = 0; i < number_of_access_units; i++) {		
packet_stream_id	8	uimsbf
PES_header_position_offset_sign	1	tcimsbf
PES_header_position_offset[43..30]	14	uimsbf
marker_bit	1	bslbf
PES_header_position_offset[29..15]	15	uimsbf
marker_bit	1	bslbf
PES_header_position_offset[14..0]	15	uimsbf
marker_bit	1	bslbf
reference_offset	16	uimsbf
marker_bit	1	bslbf
reserved	3	bslbf
PTS[32..30]	3	uimsbf
marker_bit	1	bslbf
PTS[29..15]	15	uimsbf
marker_bit	1	bslbf
PTS[14..0]	15	uimsbf
marker_bit	1	bslbf
bytes_to_read[22..8]	15	uimsbf
marker_bit	1	bslbf
bytes_to_read[7..0]	8	uimsbf
marker_bit	1	bslbf
intra_coded_indicator	1	bslbf
coding_parameters_indicator	2	bslbf
reserved	4	bslbf
}		
}		

PTS (presentation_time_stamp) – This 33-bit field is the PTS of the access unit that is referenced. The semantics of the coding of the PTS field are as described in 2.4.3.6.

bytes_to_read – This 23-bit unsigned integer is the number of bytes in the Program Stream after the byte indicated by *reference_offset* that are needed to decode the access unit completely. This value includes any bytes multiplexed at the systems layer including those containing information from other streams.

intra_coded_indicator – This is a 1-bit flag. When set to '1' it indicates that the referenced access unit is not predictively coded. This is independent of other coding parameters that might be needed to decode the access unit. For example, this field shall be coded as '1' for video Intra frames, whereas for 'P' and 'B' frames this bit shall be coded as '0'. For all PES packets containing data which is not from an ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream, this field is undefined (see Table 2-37).

Table 2-37 – Intra_coded indicator

Value	Meaning
0	Not Intra
1	Intra

coding_parameters_indicator – This 2-bit field is used to indicate the location of coding parameters that are needed to decode the access units referenced. For example, this field can be used to determine the location of quantization matrices for video frames.

Table 2-38 – Coding_parameters indicator

Value	Meaning
00	All coding parameters are set to their default values
01	All coding parameters are set in this access unit, at least one of them is not set to a default
10	Some coding parameters are set in this access unit
11	No coding parameters are coded in this access unit

2.6 Program and program element descriptors

Program and program element descriptors are structures which may be used to extend the definitions of programs and program elements. All descriptors have a format which begins with an 8-bit tag value. The tag value is followed by an 8-bit descriptor length and data fields.

2.6.1 Semantic definition of fields in program and program element descriptors

The following semantics apply to the descriptors defined in 2.6.2 through 2.6.34.

descriptor_tag – The descriptor_tag is an 8-bit field which identifies each descriptor.

Table 2-39 provides the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined, ITU-T Rec. H.222.0 | ISO/IEC 13818-1 reserved, and user available descriptor tag values. An 'X' in the TS or PS columns indicates the applicability of the descriptor to either the Transport Stream or Program Stream respectively. Note that the meaning of fields in a descriptor may depend on which stream it is used in. Each case is specified in the descriptor semantics below.

Table 2-39 – Program and program element descriptors

descriptor_tag	TS	PS	Identification
0	n/a	n/a	Reserved
1	n/a	n/a	Reserved
2	X	X	video_stream_descriptor
3	X	X	audio_stream_descriptor
4	X	X	hierarchy_descriptor
5	X	X	registration_descriptor
6	X	X	data_stream_alignment_descriptor
7	X	X	target_background_grid_descriptor
8	X	X	video_window_descriptor
9	X	X	CA_descriptor
10	X	X	ISO_639_language_descriptor
11	X	X	system_clock_descriptor
12	X	X	multiplex_buffer_utilization_descriptor
13	X	X	copyright_descriptor
14	X		maximum bitrate descriptor
15	X	X	private data indicator descriptor
16	X	X	smoothing buffer descriptor
17	X		STD_descriptor
18	X	X	IBP descriptor
19-63	n/a	n/a	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
64-255	n/a	n/a	User Private

descriptor_length – The descriptor_length is an 8-bit field specifying the number of bytes of the descriptor immediately following descriptor_length field.

2.6.2 Video stream descriptor

The video stream descriptor provides basic information which identifies the coding parameters of a video elementary stream as described in ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 (see Table 2-40).

Table 2-40 – Video stream descriptor

Syntax	No. of bits	Mnemonic
video_stream_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
multiple_frame_rate_flag	1	bslbf
frame_rate_code	4	uimsbf
MPEG_1_only_flag	1	bslbf
constrained_parameter_flag	1	bslbf
still_picture_flag	1	bslbf
if (MPEG_1_only_flag == '0'){		
profile_and_level_indication	8	uimsbf
chroma_format	2	uimsbf
frame_rate_extension_flag	1	bslbf
reserved	5	bslbf
}		
}		

2.6.3 Semantic definitions of fields in video stream descriptor

multiple_frame_rate_flag – This 1-bit field when set to '1' indicates that multiple frame rates may be present in the video stream. When set to a value of '0' only a single frame rate is present.

frame_rate_code – This is a 4-bit field as defined in 6.3.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2, except that when the multiple_frame_rate_flag is set to a value of '1' the indication of a particular frame rate also permits certain other frame rates to be present in the video stream, as specified in Table 2-41:

Table 2-41 – Frame rate code

Coded as	Also includes
23,976	
24,0	23,976
25,0	
29,97	23,976
30,0	23,976 24,0 29,97
50,0	25,0
59,94	23,976 29,97
60,0	23,976 24,0 29,97 30,0 59,94

MPEG_1_only_flag – This is a 1-bit field which when set to '1' indicates that the video stream contains only ISO/IEC 11172-2 data. If set to '0' the video stream may contain both ISO/IEC 13818-2 video data and constrained parameter ISO/IEC 11172-2 video data.

constrained_parameter_flag – This is a 1-bit field which when set to '1' indicates that the video stream shall not contain unconstrained ISO/IEC 11172-2 video data. If this field is set to '0' the video stream may contain both constrained parameters and unconstrained ISO/IEC 11172-2 video streams. If the MPEG_1_only_flag is set to '0', the constrained_parameter_flag shall be set to '1'.

still_picture_flag – This is a 1-bit field, which when set to '1' indicates that the video stream contains only still pictures. If the bit is set to '0' then the video stream may contain either moving or still picture data.

profile_and_level_indication – This bit field is the same manner as the profile_and_level_indication fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream. The value of this field indicates a profile and level that is equal to or higher than any profile and level in any sequence in the associated video stream. For the purposes of this subclause, an ISO/IEC 11172-2 constrained parameter stream is considered to a be a Main Profile at Low Level stream (MP @ LL).

chroma_format – This 2-bit field is coded in the same manner as the chroma_format fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream. The value of this field shall be at least equal to or higher than the value of the chroma_format field in any video sequence of the associated video stream. For the purposes of this subclause, an ISO/IEC 11172-2 video stream is considered to have chroma_format field with the value '01', indicating 4:2:0.

frame_rate_extension_flag – This is a 1-bit flag which when set to '1' indicates that either or both the frame_rate_extension_n and the frame_rate_extension_d fields are non-zero in any video sequences of the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream. For the purposes of this subclause, an ISO/IEC 11172-2 video stream is constrained to have both fields set to zero.

2.6.4 Audio stream descriptor

The audio stream descriptor provides basic information which identifies the coding version of an audio elementary stream as described in ISO/IEC 13818-3 or ISO/IEC 11172-3 (see Table 2-42).

Table 2-42 – Audio stream descriptor

Syntax	No. of bits	Mnemonic
audio_stream_descriptor(){		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
free_format_flag	1	bslbf
ID	1	bslbf
layer	2	bslbf
variable_rate_audio_indicator	1	bslbf
reserved	3	bslbf
}		

2.6.5 Semantic definition of fields in audio stream descriptor

free_format_flag – This 1-bit field when set to '1' indicates that the audio stream may contain one or more audio frames with the bitrate_index set to '0000'. If set to '0', then the bitrate_index is not '0000' (refer to 2.4.2.3 of ISO/IEC 13818-3) in any audio frame of the audio stream.

ID – This 1-bit field when set to '1' indicates that the ID field is set to '1' in each audio frame in the audio stream (refer to 2.4.2.3 of ISO/IEC 13818-3).

layer – This 2-bit field is coded in the same manner as the layer field in the ISO/IEC 13818-3 or ISO/IEC 11172-3 audio streams (refer to 2.4.2.3 of ISO/IEC 13818-3). The layer indicated in this field shall be equal to or higher than the highest layer specified in any audio frame of the audio stream.

variable_rate_audio_indicator – This 1-bit flag, when set to '0' indicates that the bit rate of the associated audio stream may vary between consecutive audio frames. Continuously coded variable rate audio should be presented without discontinuities.

2.6.6 Hierarchy descriptor

The hierarchy descriptor provides information to identify the program elements containing components of hierarchically-coded video and audio, and private streams which are multiplexed in multiple streams as described in this Recommendation | International Standard, in ITU-T Rec. H.262 | ISO/IEC 13818-2 and in ISO/IEC 13818-3. (See Table 2-43.)

Table 2-43 – Hierarchy descriptor

Syntax	No. of bits	Mnemonic
hierarchy_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved	4	bslbf
hierarchy_type	4	uimsbf
reserved	2	bslbf
hierarchy_layer_index	6	uimsbf
reserved	2	bslbf
hierarchy_embedded_layer_index	6	uimsbf
reserved	2	bslbf
hierarchy_channel	6	uimsbf
}		

11/13/200

2.6.7 Semantic definition of fields in hierarchy descriptor

hierarchy_type – The hierarchical relation between the associated hierarchy layer and its hierarchy embedded layer is defined in Table 2-44.

Table 2-44 – Hierarchy_type field values

Value	Description
0	Reserved
1	ITU-T Rec. H.262 | ISO/IEC 13818-2 Spatial Scalability
2	ITU-T Rec. H.262 | ISO/IEC 13818-2 SNR Scalability
3	ITU-T Rec. H.262 | ISO/IEC 13818-2 Temporal Scalability
4	ITU-T Rec. H.262 | ISO/IEC 13818-2 Data partitioning
5	ISO/IEC 13818-3 Extension bitstream
6	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Private Stream
7-14	Reserved
15	Base layer

hierarchy_layer_index – The hierarchy_layer_index is a 6-bit field that defines a unique index of the associated program element in a table of coding layer hierarchies. Indices shall be unique within a single program definition.

hierarchy_embedded_layer_index – The hierarchy_embedded_layer_index is a 6-bit field that defines the hierarchy table index of the program element that needs to be accessed before decoding of the elementary stream associated with this hierarchy_descriptor. This field is undefined if the hierarchy_type value is 15 (base layer).

hierarchy_channel – The hierarchy_channel is a 6-bit field that indicates the intended channel number for the associated program element in an ordered set of transmission channels. The most robust transmission channel is defined by the lowest value of this field with respect to the overall transmission hierarchy definition.

NOTE – A given hierarchy_channel may at the same time be assigned to several program elements.

2.6.8 Registration descriptor

The registration_descriptor provides a method to uniquely and unambiguously identify formats of private data (see Table 2-45).

Table 2-45 – Registration descriptor

Syntax	No. of bits	Identifier
registration_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
format_identifier	32	uimsbf
for (i = 0; i < N; i++){		
additional_identification_info	8	bslbf
}		
}		

2.6.9 Semantic definition of fields in registration descriptor

format_identifier – The format_identifier is a 32-bit value obtained from a Registration Authority as designated by SC29.

additional_identification_info – The meaning of additional_identification_info bytes, if any, are defined by the assignee of that format_identifier, and once defined they shall not change.

2.6.10 Data stream alignment descriptor

The data stream alignment descriptor describes which type of alignment is present in the associated elementary stream. If the data_alignment_indicator in the PES packet header is set to '1' and the descriptor is present, alignment – as specified in this descriptor – is required (see Table 2-46).

Table 2-46 – Data stream alignment descriptor

Syntax	No. of bits	Mnemonic
data_stream_alignment_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
alignment_type	8	uimsbf
}		

2.6.11 Semantics of fields in data stream alignment descriptor

alignment_type – Table 2-47 describes the video alignment type when the data_alignment_indicator in the PES packet header has a value of '1'. In each case of alignment_type value the first PES_packet_data_byte following the PES header shall be the first byte of a start code of the type indicated in Table 2-47. At the beginning of a video sequence, the alignment shall occur at the start code of the first sequence header.

NOTE – Specifying alignment type '01' from Table 2-47 does not preclude the alignment from beginning at a GOP or SEQ header.

The definition of access unit for video data is given in 2.1.1.

Table 2-47 – Video stream alignment values

Alignment type	Description
00	Reserved
01	Slice, or video access unit
02	Video access unit
03	GOP, or SEQ
04	SEQ
05-FF	Reserved

Table 2-48 describes the audio alignment type when the data_alignment_indicator in the PES packet header has a value of '1'. In this case the first PES_packet_data_byte following the PES header is the first byte of an audio sync word.

Table 2-48 – Audio stream alignment values

Alignment type	Description
00	Reserved
01	Sync word
02-FF	Reserved

2.6.12 Target background grid descriptor

It is possible to have one or more video streams which, when decoded, are not intended to occupy the full display area (e.g. a monitor). The combination of target_background_grid_descriptor and video_window_descriptors allows the display of these video windows in their desired locations. The target_background_grid_descriptor is used to describe a grid of unit pixels projected on to the display area. The video_window_descriptor is then used to describe, for the associated stream, the location on the grid at which the top left pixel of the display window or display rectangle of the video presentation unit should be displayed. This is represented in Figure 2-3.

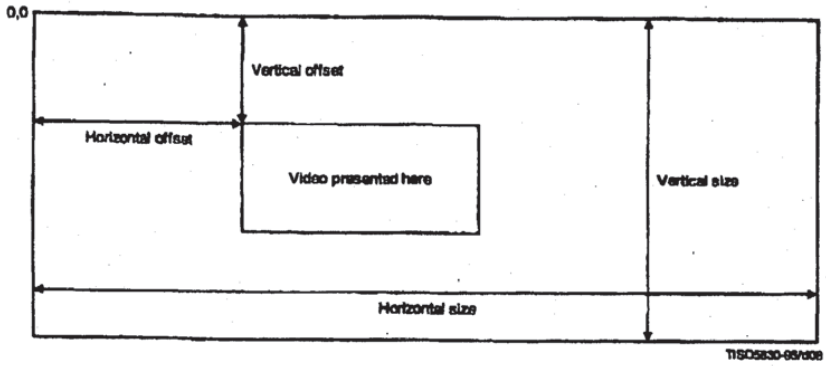

Figure 2-3 – Target background grid descriptor display area

2.6.13 Semantics of fields in target background grid descriptor

horizontal_size – The horizontal size of the target background grid in pixels.

vertical_size – The vertical size of the target background grid in pixels.

aspect_ratio_information – Specifies the sample aspect ratio or display aspect ratio of the target background grid. Aspect_ratio_information is defined in ITU-T Rec. H.262 | ISO/IEC 13818-2 (see Table 2-49).

Table 2-49 – Target background grid descriptor

Syntax	No. of bits	Mnemonic
target_background_grid_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
horizontal_size	14	uimsbf
vertical_size	14	uimsbf
aspect_ratio_information	4	uimsbf
}		

2.6.14 Video window descriptor

The video window descriptor is used to describe the window characteristics of the associated video elementary stream. Its values reference the target background grid descriptor for the same stream. Also see target_background_grid_descriptor in 2.6.12 (see Table 2-50).

Table 2-50 – Video window descriptor

Syntax	No. of bits	Mnemonic
video_window_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
horizontal_offset	14	uimsbf
vertical_offset	14	uimsbf
window_priority	4	uimsbf
}		

11/13/200

2.6.15 Semantic definition of fields in video window descriptor

horizontal_offset – The value indicates the horizontal position of the top left pixel of the current video display window or display rectangle if indicated in the picture display extension on the target background grid for display as defined in the target_background_grid_descriptor. The top left pixel of the video window shall be one of the pixels of the target background grid (refer to Figure 2-3).

vertical_offset – The value indicates the vertical position of the top left pixel of the current video display window or display rectangle if indicated in the picture display extension on the target background grid for display as defined in the target_background_grid_descriptor. The top left pixel of the video window shall be one of the pixels of the target background grid (refer to Figure 2-3).

window_priority – The value indicates how windows overlap. A value of 0 being lowest priority and a value of 15 is the highest priority, i.e. windows with priority 15 are always visible.

2.6.16 Conditional access descriptor

The conditional access descriptor is used to specify both system-wide conditional access management information such as EMMs and elementary stream-specific information such as ECMs. It may be used in both the TS_program_map_section (refer to 2.4.4.8) and the program_stream_map (refer to 2.5.3). If any elementary stream is scrambled, a CA descriptor shall be present for the program containing that elementary stream. If any system-wide conditional access management information exists within a Transport Stream, a CA descriptor shall be present in the conditional access table.

When the CA descriptor is found in the TS_program_map_section (table_id = 0x02), the CA_PID points to packets containing program related access control information, such as ECMs. Its presence as program information indicates applicability to the entire program. In the same case, its presence as extended ES information indicates applicability to the associated program element. Provision is also made for private data.

When the CA descriptor is found in the CA_section (table_id = 0x01), the CA_PID points to packets containing system-wide and/or access control management information, such as EMMs.

The contents of the Transport Stream packets containing conditional access information are privately defined (see Table 2-51).

Table 2-51 – Conditional access descriptor

Syntax	No. of bits	Mnemonic
CA_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
CA_system_ID	16	uimsbf
reserved	3	bslbf
CA_PID	13	uimsbf
for (i = 0; i < N; i++) {		
private_data_byte	8	uimsbf
}		
}		

2.6.17 Semantic definition of fields in conditional access descriptor

CA_system_ID – This is a 16-bit field indicating the type of CA system applicable for either the associated ECM and/or EMM streams. The coding of this is privately defined and is not specified by ITU-T | ISO/IEC.

CA_PID – This is a 13-bit field indicating the PID of the Transport Stream packets which shall contain either ECM or EMM information for the CA systems as specified with the associated CA_system_ID. The contents (ECM or EMM) of the packets indicated by the CA_PID is determined from the context in which the CA_PID is found, i.e. a TS_program_map_section or the CA table in the Transport Stream, or the stream_id field in the Program Stream.

2.6.18 ISO 639 language descriptor

The language descriptor is used to specify the language of the associated program element (see Table 2-52).

Table 2-52 – ISO 639 language descriptor

Syntax	No. of bits	Mnemonic
ISO_639_language_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
for (i = 0; i < N; i++) {		
ISO_639_language_code	24	bslbf
audio_type	8	bslbf
}		
}		

2.6.19 Semantic definition of fields in ISO 639 language descriptor

ISO_639_language_code – Identifies the language or languages used by the associated program element. The ISO_639_language_code contains a 3-character code as specified by ISO 639, Part 2. Each character is coded into 8 bits according to ISO 8859-1 and inserted in order into this 24-bit field. In the case of multilingual audio streams the sequence of ISO_639_language_code fields shall reflect the content of the audio stream.

audio_type – The audio_type is an 8-bit field which specifies the type of stream defined in Table 2-53.

Table 2-53 – Audio type values

Value	Description
0x00	Undefined
0x01	Clean effects
0x02	Hearing impaired
0x03	Visual impaired commentary
0x04 - 0xFF	Reserved

clean effects – This field indicates that the referenced program element has no language.

hearing impaired – This field indicates that the referenced program element is prepared for the hearing impaired.

visual_impaired_commentary – This field indicates that the referenced program element is prepared for the visually impaired viewer.

2.6.20 System clock descriptor

This descriptor conveys information about the system clock that was used to generate the timestamps.

If an external clock reference was used, the external_clock_reference_indicator may be set to '1'. The decoder optionally may use the same external reference if it is available.

If the system clock is more accurate than the 30 ppm accuracy required, then the accuracy of the clock can be communicated by encoding it in the clock_accuracy fields. The clock frequency accuracy is:

$$\text{clock\_accuracy\_integer} \times 10^{-\text{clock\_accuracy\_exponent}} \text{ ppm} \qquad (2\text{-}26)$$

If clock_accuracy_integer is set to 0, then the system clock accuracy is 30 ppm. When the external_clock_reference_indicator is set to '1', the clock accuracy pertains to the external reference clock (see Table 2-54).

Table 2-54 – System clock descriptor

Syntax	No. of bits	Mnemonic
system_clock_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
external_clock_reference_indicator	1	bslbf
reserved	1	bslbf
clock_accuracy_integer	6	uimsbf
clock_accuracy_exponent	3	uimsbf
reserved	5	bslbf
}		

2.6.21 Semantic definition of fields in system clock descriptor

external_clock_reference_indicator – This is a 1-bit indicator. When set to '1', it indicates that the system clock has been derived from an external frequency reference that may be available at the decoder.

clock_accuracy_integer – This is a 6-bit integer. Together with the clock_accuracy_exponent, it gives the fractional frequency accuracy of the system clock in parts per million.

clock_accuracy_exponent – This is a 3-bit integer. Together with the clock_accuracy_integer, it gives the fractional frequency accuracy of the system clock in parts per million.

2.6.22 Multiplex buffer utilization descriptor

The multiplex buffer utilization descriptor provides bounds on the occupancy of the STD multiplex buffer. This information is intended for devices such as remultiplexers, which may use this information to support a desired re-multiplexing strategy (see Table 2-55).

Table 2-55 – Multiplex buffer utilization descriptor

Syntax	No. of bits	Mnemonic
multiplex_buffer_utilization_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
bound_valid_flag	1	bslbf
LTW_offset_lower_bound	15	uimsbf
reserved	1	bslbf
LTW_offset_upper_bound	14	uimsbf
}		

2.6.23 Semantic definition of fields in multiplex buffer utilization descriptor

bound_valid_flag – A value of '1' indicates that the LTW_offset_lower_bound and the LTW_offset_upper_bound fields are valid.

LTW_offset_lower_bound – This 15-bit field is defined only if the bound_valid flag has a value of '1'. When defined, this field has the units of (27 MHz / 300) clock periods, as defined for the LTW_offset (refer to 2.4.3.4). The LTW_offset_lower_bound represents the lowest value that any LTW_offset field would have, if that field were coded in

every packet of the stream or streams referenced by this descriptor. Actual LTW_offset fields may or may not be coded in the bitstream when the multiplex buffer utilization descriptor is present. This bound is valid until the next occurrence of this descriptor.

LTW_offset_upper_bound – This 15-bit field is defined only if the bound_valid has a value of '1'. When defined, this field has the units of (27 MHz / 300) clock periods, as defined for the LTW_offset (refer to 2.4.3.4). The LTW_offset_upper_bound represents the largest value that any LTW_offset field would have, if that field were coded in every packet of the stream or streams referenced by this descriptor. Actual LTW_offset fields may or may not be coded in the bitstream when the multiplex buffer utilization descriptor is present. This bound is valid until the next occurrence of this descriptor.

2.6.24 Copyright descriptor

The copyright_descriptor provides a method to enable audio-visual works identification. This copyright_descriptor applies to programs or program elements within programs (see Table 2-56).

Table 2-56 – Copyright descriptor

Syntax	No. of bits	Identifier
copyright_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
copyright_identifier	32	uimsbf
for (i = 0; i < N; i++){		
additional_copyright_info	8	bslbf
}		
}		

2.6.25 Semantic definition of fields in copyright descriptor

copyright_identifier – This field is a 32-bit value obtained from the Registration Authority.

additional_copyright_info – The meaning of additional_copyright_info bytes, if any, are defined by the assignee of that copyright_identifier, and once defined, they shall not change.

2.6.26 Maximum bitrate descriptor

See Table 2-57.

Table 2-57 – Maximum bitrate descriptor

Syntax	No. of bits	Identifier
maximum_bitrate_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved	2	bslbf
maximum_bitrate	22	uimsbf
}		

2.6.27 Semantic definition of fields in maximum bitrate descriptor

maximum_bitrate – The maximum bitrate is coded as a 22-bit positive integer in this field. The value indicates an upper bound of the bitrate, including transport overhead, that will be encountered in this program element or program. The value of maximum_bitrate is expressed in units of 50 bytes/second. The maximum_bitrate_descriptor is included in the Program Map Table (PMT). Its presence as extended program information indicates applicability to the entire program. Its presence as ES information indicates applicability to the associated program element.

2.6.28 Private data indicator descriptor

See Table 2-58.

Table 2-58 – Private data indicator descriptor

Syntax	No. of bits	Identifier
private_data_indicator_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
private_data_indicator	32	uimsbf
}		

2.6.29 Semantic definition of fields in Private data indicator descriptor

private_data_indicator – The value of the private_data_indicator is private and shall not be defined by ITU-T | ISO/IEC.

2.6.30 Smoothing buffer descriptor

This descriptor is optional and conveys information about the size of a smoothing buffer, SB_n, associated with this descriptor, and the associated leak rate out of that buffer, for the program element(s) that it refers to.

In the case of Transport Streams, bytes of Transport Stream packets of the associated program element(s) present in the Transport Stream are input to a buffer SB_n of size given by sb_size, at the time defined by equation 2-4.

In the case of Program Streams, bytes of all PES packets of the associated elementary streams, are input to a buffer SB_n of size given by sb_size, at the time defined by equation 2-21.

When there is data present in this buffer, bytes are removed from this buffer at a rate defined by sb_leak_rate. The buffer, SB_n shall never overflow. During the continuous existence of a program, the value of the elements of the Smoothing Buffer descriptor of the different program element(s) in the program, shall not change.

The meaning of the smoothing buffer_descriptor is only defined when it is included in the PMT or the Program Stream Map.

If, in the case of a Transport Stream, it is present in the ES info in the Program Map Table, all Transport Stream packets of the PID of that program element enter the smoothing buffer.

If, in the case of a Transport Stream, it is present in the program information, the following Transport Stream packets enter the smoothing buffer:

- all Transport Stream packets of all PIDs listed as elementary_PIDs in the extended program information as well as;
- all Transport Stream packets of the PID which is equal to the PMT_PID of this section;
- all Transport Stream packets of the PCR_PID of the program.

All bytes that enter the associated buffer also exit it.

At any given time there shall be at most one descriptor referring to any individual program element and at most one descriptor referring to the program in its entirety.

Table 2-59 – Smoothing buffer descriptor

Syntax	No. of bits	Mnemonic
smoothing_buffer_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved	2	bslbf
sb_leak_rate	22	uimsbf
reserved	2	bslbf
sb_size	22	uimsbf
}		

2.6.31 Semantic definition of fields in smoothing buffer descriptor

sb_leak_rate – This 22-bit field is coded as a positive integer. Its contents indicate the value of the leak rate out of the SB_n buffer for the associated elementary stream or other data in units of 400 bits/s.

sb_size – This 22-bit field is coded as a positive integer. Its contents indicate the value of the size of the multiplexing buffer smoothing buffer SB_n for the associated elementary stream or other data in units of 1 byte (see Table 2-59).

2.6.32 STD descriptor

This descriptor is optional and applies only to the T-STD model and to video elementary streams, and is used as specified 2.4.2. This descriptor does not apply to Program Streams (see Table 2-60).

Table 2-60 – STD descriptor

Syntax	No. of bits	Mnemonic
STD_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
reserved	7	bslbf
leak_valid_flag	1	bslbf
}		

2.6.33 Semantic definition of fields in STD descriptor

leak_valid_flag – The leak_valid_flag is a 1-bit flag. When set to '1', the transfer of data from the buffer MB_n to the buffer EB_n in the T-STD uses the leak method as defined in 2.4.2.3. If this flag has a value equal to '0', and the vbv_delay fields present in the associated video stream do not have the value 0xFFFF, the transfer of data from the buffer MB_n to the buffer EB_n uses the vbv_delay method as defined in 2.4.2.3.

2.6.34 IBP descriptor

This optional descriptor provides information about some characteristics of the sequence of frame types in the video sequence (see Table 2-61).

Table 2-61 – IBP descriptor

Syntax	No. of bits	Mnemonic
ibp_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
closed_gop_flag	1	uimsbf
identical_gop_flag	1	uimsbf
max_gop-length	14	uimsbf
}		

2.6.35 Semantic definition of fields in IBP descriptor

closed_gop_flag – This 1-bit flag when set to '1' indicates that a group of pictures header is encoded before every I-frame and that the closed_gop flag is set to '1' in all group of pictures headers in the video sequence.

identical_gop_flag – This 1-bit flag when set to '1' indicates that the number of P-frames and B-frames between I-frames, and the picture coding types and sequence of picture types between I-pictures is the same throughout the sequence, except possibly for the pictures up to the second I-picture.

max_gop_length – This 14-bit unsigned integer indicates the maximum number of the coded pictures between any two consecutive I-pictures in the sequence. The value of 0 is forbidden.

2.7 Restrictions on the multiplexed stream semantics

2.7.1 Frequency of coding the system clock reference

The Program Stream shall be constructed such that the time interval between the bytes containing the last bit of system_clock_reference_base fields in successive packs shall be less than or equal to 0,7 s. Thus:

$$| t(i) - t(i') | \leq 0{,}7 \text{ s}$$

for all i and i' where i and i' are the indexes of the bytes containing the last bit of consecutive system_clock_reference_base fields.

2.7.2 Frequency of coding the program clock reference

The Transport Stream shall be constructed such that the time interval between the bytes containing the last bit of program_clock_reference_base fields in successive occurrences of the PCRs in Transport Stream packets of the PCR_PID for each program shall be less than or equal to 0,1 s. Thus:

$$| t(i) - t(i') | \leq 0{,}1 \text{ s}$$

for all i and i' where i and i' are the indexes of the bytes containing the last bit of consecutive program_clock_reference_base fields in the Transport Stream packets of the PCR_PID for each program.

There shall be at least two (2) PCRs, from the specified PCR_PID within a Transport Stream, between consecutive PCR discontinuities (refer to 2.4.3.4) to facilitate phase locking and extrapolation of byte delivery times.

2.7.3 Frequency of coding the elementary stream clock reference

The Program Stream and Transport Stream shall be constructed such that if the elementary stream clock reference field is coded in any PES packets containing data of a given elementary stream the time interval in the PES_STD between the bytes containing the last bit of successive ESCR_base fields shall be less than or equal to 0,7 s. In PES Streams the ESCR encoding is required with the same interval. Thus:

$$| t(i) - t(i') | \leq 0{,}7 \text{ s}$$

for all i and i' where i and i' are the indexes of the bytes containing the last bits of consecutive ESCR_base fields.

NOTE – The coding of elementary stream clock reference fields is optional; they need not be coded. However if they are coded, this constraint applies.

2.7.4 Frequency of presentation timestamp coding

The Program Stream and Transport Stream shall be constructed so that the maximum difference between coded presentation timestamps referring to each elementary video or audio stream is 0,7 s. Thus:

$$| tp_n(k) - tp_n(k'') | \leq 0{,}7 \text{ s}$$

for all n, k, and k'' satisfying:

- $P_n(k)$ and $P_n(k'')$ are presentation units for which presentation timestamps are coded;
- k and k'' are chosen so that there is no presentation unit, $P_n(k')$ with a coded presentation timestamp and with $k < k' < k''$; and
- No decoding discontinuity exists in elementary stream n between $P_n(k)$ and $P_n(k'')$.

In the case of still pictures the 0,7 s constraint does not apply.

2.7.5 Conditional coding of timestamps

For each elementary stream of a Program Stream or Transport Stream, a presentation timestamp (PTS) shall be encoded for the first access unit.

A decoding discontinuity exists at the start of an access unit $A_n(j)$ in an elementary stream n if the decoding time $td_n(j)$ of that access unit is greater than the largest value permissible given the specified tolerance on the system_clock_frequency. For video, except when trick mode status is true or when low_delay flag is '1', this is allowed only at the start of a video sequence. If a decoding discontinuity exists in any elementary video or audio stream in the Transport Stream or Program Stream, then a PTS shall be encoded referring to the first access unit after each decoding discontinuity except when trick mode status is true.

When low_delay is '1' a PTS shall be encoded for the first access unit after an EB_n or B_n underflow.

A PTS may only be present in a ITU-T Rec. H.222.0 | ISO/IEC 13818-1 video or audio elementary stream PES packet header if the first byte of a picture start code or the first byte of an audio access unit is contained in the PES packet.

A decoding_timestamp (DTS) shall appear in a PES packet header if and only if the following two conditions are met:

- a PTS is present in the PES packet header;
- the decoding time differs from the presentation time.

2.7.6 Timing constraints for scalable coding

If an audio sequence is coded using an ISO/IEC 13818-3 extension bitstream, corresponding decoding/presentation units in the two layers shall have identical PTS values.

If a video sequence is coded as a SNR enhancement of another sequence, as specified in 7.8 of ITU-T Rec. H.262 | ISO/IEC 13818-2, the set of presentation times for both sequences shall be the same.

If a video sequence is coded as two partitions, as specified in 7.10 of ITU-T Rec. H.262 | ISO/IEC 13818-2, the set of presentation times for both partitions shall be the same.

If a video sequence is coded as a spatial scalable enhancement of another sequence, as specified in 7.7 of ITU-T Rec. H.262 | ISO/IEC 13818-2, the following shall apply:

- If both sequences have the same frame rate, the set of presentation times for both sequences shall be the same.

  NOTE – that this does not imply that the picture coding type is the same in both layers.

- If the sequences have different frame rates, the set of presentation times shall be such that as many presentation times as possible shall be common to both sequences.
- The picture from which the spatial prediction is made shall be one of the following:
  - the coincident or most recently decoded lower layer picture;
  - the coincident or most recently decoded lower layer picture that is an I- or P-picture;
  - the second most recently decoded lower layer picture that is an I- or P-picture, and provided that the lower layer does not have low_delay set to '1'.

If a video sequence is coded as a temporally scalable enhancement of another sequence, as specified in 7.9 of ITU-T Rec. H.262 | ISO/IEC 13818-2, the following lower layer pictures may be used as the reference. Times are relative to presentation times:

- the coincident or most recently presented lower layer picture;
- the next lower layer picture to be presented.

2.7.7 Frequency of coding P-STD_buffer_size in PES packet headers

In a Program Stream, the P-STD_buffer_scale and P-STD_buffer_size fields shall occur in the first PES packet of each elementary stream and again whenever the value changes. They may also occur in any other PES packet.

2.7.8 Coding of system header in the Program Stream

In a Program Stream, the system header may be present in any pack, immediately following the pack header. The system header shall be present in the first pack of an Program Stream. The values encoded in all the system headers in the Program Stream shall be identical.

2.7.9 Constrained system parameter Program Stream

A Program Stream is a "Constrained System Parameters Stream" (CSPS) if it conforms to the bounds specified in this subclause. Program Streams are not limited to the bounds specified by the CSPS. A CSPS may be identified by means of the CSPS_flag defined in the system header in 2.5.3.5. The CSPS is a subset of all possible Program Streams.

Packet Rate

In the CSPS, the maximum rate at which packets shall arrive at the input to the P-STD is 300 packets per second if the value encoded in the rate_bound field (refer to 2.5.3.6) is less than or equal to 4 500 000 bits/s if the packet_rate_restriction_flag is set to '1', and less than or equal to 2 000 000 bits/s if the packet_rate_restriction_flag is set to '0'. For higher bit rates the CSPS packet rate is bounded by a linear relation to the value encoded in the rate_bound field.

Specifically, for all packs p in the Program Stream when the packet_rate_restriction_flag (refer to 2.5.3.5) is set to a value of '1',

$$NP \leq (t(i') - t(i)) \times 300 \times \max\left[1, \frac{R_{max}}{4.5 \times 10^6}\right] \quad (2\text{-}27)$$

and if the packet_rate_restriction_flag is set to a value of '0'

$$NP \leq (t(i') - t(i)) \times 300 \times \max\left[1, \frac{R_{max}}{2.0 \times 10^6}\right] \quad (2\text{-}28)$$

where

$$R_{max} = 8 \times 50 \times rate\_bound \quad \text{bits/s} \quad (2\text{-}29)$$

NP is the number of packet_start_code_prefixes and system_header_start_codes between adjacent pack_start_codes or between the last pack_start_code and the MPEG_program_end_code as defined in Table 2-31 and semantics in 2.5.3.2.

t(i) is the time, measured in seconds, encoded in the SCR of pack p.

t(i') is the time, measured in seconds, encoded in the SCR for pack p + 1, immediately following pack p, or in the case of the final pack in the Program Stream, the time of arrival of the byte containing the last bit of the MPEG_program_end_code.

Decoder Buffer Size

In the case of a CSPS the maximum size of each input buffer in the system target decoder is bounded. Different bounds apply for video elementary streams and audio elementary streams.

In the case of a video elementary stream in a CSPS, the following applies:

BS_n has a size which is equal to the sum of the size of the video buffer verifier (vbv) as specified in ITU-T Rec. H.262 | ISO/IEC 13818-2 and an additional amount of buffering BS_{add}. BS_{add} is specified as:

$$BS_{add} \leq MAX\,[6 \times 1024, R_{vmax} \times 0{,}001]\ bytes$$

where R_{vmax} is the maximum video bit rate of the video elementary stream.

In the case of an audio elementary stream in a CSPS, the following applies:

$$BS_n \leq 4096\ bytes$$

2.7.10 Transport Stream

Sample rate locking in Transport Streams

In the Transport Stream there shall be a specified constant rational relationship between the audio sampling rate and the system clock frequency in the system target decoder, and likewise a specified rational relationship between the video

frame rate and the system clock frequency.The system_clock_frequency is defined in 2.4.2. The video frame rate is specified in ITU-T Rec. H.262 | ISO/IEC 13818-2 or in ISO/IEC 11172-2. The audio sampling rate is specified in ISO/IEC 13818-3 or in ISO/IEC 11172-3. For all presentation units in all audio elementary streams in the Transport Stream, the ratio of system_clock_frequency to the actual audio sampling rate, SCASR, is constant and equal to the value indicated in the following table at the nominal sampling rate indicated in the audio stream.

$$SCASR = \frac{system\_clock\_frequency}{audio\_simple\_rate\_in\_the\_T\text{-}STD} \qquad (2\text{-}30)$$

The notation $\frac{X}{Y}$ denotes real division.

Nominal audio sampling frequency (kHz)	16	32	22,05	44,1	24	48
SCASR	$\frac{27\,000\,000}{16\,000}$	$\frac{27\,000\,000}{32\,000}$	$\frac{27\,000\,000}{22\,050}$	$\frac{27\,000\,000}{44\,100}$	$\frac{27\,000\,000}{24\,000}$	$\frac{27\,000\,000}{48\,000}$

For all presentation units in all video elementary streams in the Transport Stream, the ratio of system_clock_frequency to the actual video frame rate, SCFR, is constant and equal to the value indicated in the following table at the nominal frame rate indicated in the video stream.

$$SCFR = \frac{system\_clock\_frequency}{frame\_rate\_in\_the\_T\text{-}STD} \qquad (2\text{-}31)$$

Nominal frame rate (Hz)	23,976	24	25	29,97	30	50	59,94	60
SCFR	1 126 125	1 125 000	1 080 000	900 900	900 000	540 000	450 450	450 000

The values of the SCFR are exact. The actual frame rate differs slightly from the nominal rate in cases where the nominal rate is 23,976, 29,97, or 59,94 frames per second.

2.8 Compatibility with ISO/IEC 11172

The Program Stream of this Recommendation | International Standard is defined to be forward compatible with ISO/IEC 11172-1. Decoders of the Program Stream as defined in this Recommendation | International Standard shall also support decoding of ISO/IEC 11172-1.

Annex A

CRC Decoder Model

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

A.0 CRC decoder model

The 32-bit CRC Decoder Model is specified in Figure A.1.

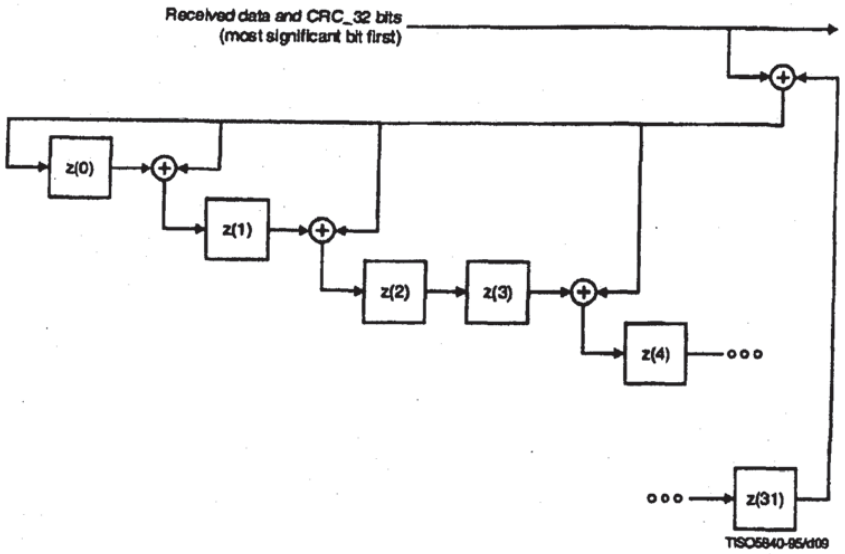

Figure A.1 – 32-bit CRC decoder model

The 32-bit CRC Decoder operates at bit level and consists of 14 adders '+' and 32 delay elements z(i). The input of the CRC decoder is added to the output of z(31), and the result is provided to the input z(0) and to one of the inputs of each remaining adder. The other input of each remaining adder is the output of z(i), while the output of each remaining adder is connected to the input of z(i + 1), with i = 0, 1, 3, 4, 6, 7 , 9, 10, 11, 15, 21, 22, and 25. Refer to Figure A.1 above.

This is the CRC calculated with the polynomial:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \qquad \text{(A-1)}$$

Bytes are received at the input of the CRC decoder. Each byte is shifted into the CRC decoder one bit at a time, with the left most bit (msb) first. For example, if the input is byte 0x01 the seven '0's enter the CRC decoder first, followed by the one '1'. Before the CRC processing of the data of a section the output of each delay element z(i) is set to its initial value '1'. After this initialization, each byte of the section is provided to the input of the CRC decoder, including the four CRC_32 bytes. After shifting the last bit of the last CRC_32 byte into the decoder, i.e. into z(0) after the addition with the output of z(31), the output of all delay elements z(i) is read. In the case where there are no errors, each of the outputs of z(i) shall be zero. At the CRC encoder the CRC_32 field is encoded with a value such that this is ensured.

Annex B

Digital Storage Medium Command and Control (DSM-CC)

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

B.0 Introduction

The DSM CC protocol is a specific application protocol intended to provide the basic control functions and operations specific to managing an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream on digital storage media. This DSM CC is a low-level protocol above network/OS layers and below application layers.

The DSM-CC shall be transparent in the following sense:

- it is independent of the DSM used;
- it is independent of whether the DSM is located at a local or remote site;
- it is independent of the network protocol with which the DSM-CC is interfaced;
- it is independent of the various operating systems on which the DSM is operated.

B.0.1 Purpose

Many applications of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 DSM Control Commands require access to an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream stored on a variety of digital storage media at a local or remote site. Different DSM have their own specific control commands and thus, a user would need to know different sets of specific DSM control commands in order to access ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams from different DSM. This brings many difficulties to the interface design of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 or ISO 11172-1 application system. To overcome this difficulty, a set of common DSM control commands, which is independent of the specific DSM used, is suggested in this annex. This annex is informative only. ISO/IEC 13818-6 defines DSM-CC extension with a broader scope.

B.0.2 Future applications

Beyond the immediate applications supported by the current DSM control commands, future applications based on extensions of DSM command control could include the following:

Video on demand

Video programs are provided as requested by a customer through various communication channels. The customer could select a video program from a list of programs available from a video server. Such applications could be used by hotels, cable TV, educational institutions, hospitals, etc.

Interactive video services

In these applications, the user provides frequent feedback controlling the manipulation of stored video and audio. These services can include video based games, user controlled video tours, electronic shopping, etc.

Video networks

Various applications may wish to exchange stored audio and video data through some type of computer network. Users could route AV information through the video network to their terminals. Electronic publishing and multimedia applications are examples of this kind of application.

B.0.3 Benefits

Specifying the DSM control commands independ of the DSM, end-users can perform ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoding without having to fully understand the detailed operation of the specific DSM used.

The DSM control commands are codes to give end users the assurance that the ISO/IEC 13818-1 bitstreams can be played and stored with the same semantics, independent of the DSM and user interface. They are fundamental commands for the control of DSM operation.

B.0.4 Basic functions

B.0.4.1 Stream selection

The DSM-CC provides the means to select an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream upon which to perform the succeeding operations. Such operations include creation of a new bitstream. Parameters of this function include:

- index of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream (the mapping between this index and a name meaningful to an application is outside the scope of the current DSM-CC)
- mode (retrieval/storage)

B.0.4.2 Retrieval

The DSM CC provides the means to:

- play an identified ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream;
- play from a given presentation time;
- set the playback speed (normal or fast);
- set the playback duration (until a specified presentation time, the end of the bitstream in forward play or the beginning in reverse play or the issuance of a stop command);
- set the direction (forward or reverse);
- pause;
- resume;
- change the access point in the bitstream;
- stop.

B.0.4.3 Storage

The DSM-CC provides the means to:

- cause storage of a valid bitstream for a specified duration;
- cause storage to stop.

DSM-CC provides a useful but limited subset of functionality that may be required in DSM based ITU-T Rec. H.222.0 | ISO/IEC 13818-1 applications. It is fully expected that significant additional capabilities will be added through subsequent extensions.

B.1 General elements

B.1.1 Scope

The scope of this work consists of the development of a Recommendation | International Standard to specify a useful set of commands for control of digital storage media on which an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is stored. The commands can perform remote control of a digital storage media in a general way independently of the specific DSM and apply to any ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream stored on a DSM.

B.1.2 Overview of the DSM-CC application

The current DSM-CC syntax and semantics cover the single user to DSM application. The user's system is capable of retrieving an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream and is also (optionally) capable of generating an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream. The control channel over which the DSM commands and acknowledgments are sent is shown in Figure B.1 as an out of band channel. This can also be accomplished by inserting the DSM-CC commands and acknowledgments into the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams if an out of band channel is not available.

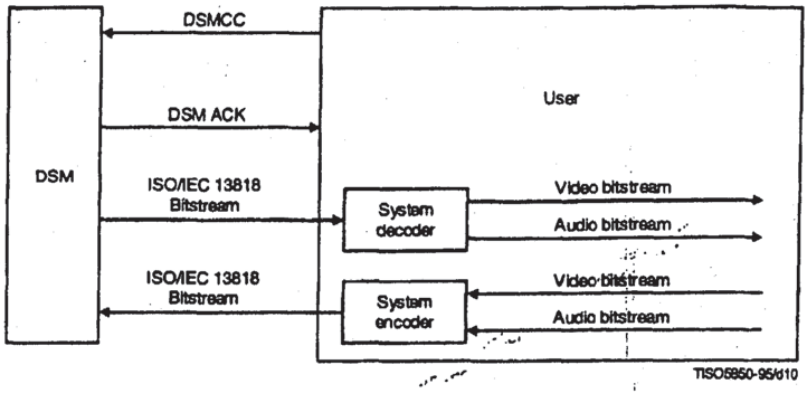

Figure B.1 – Configuration of DSM-CC application

B.1.3 The transmission of DSM-CC commands and acknowledgments

The DSM-CC is encoded into a DSM-CC bitstream according to the syntax and semantics defined in B.2.2 through B.2.9. The DSM-CC bitstream can be transmitted both as a stand alone bitstream and in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems bitstream.

When the DSM-CC bitstream is transmitted in stand alone mode, its relationship to the Systems bitstream and the decoding process is illustrated in Figure B.2. In this case, the DSM-CC bitstream is not embedded in the Systems bitstream. This transmission mode can be used in the applications when the DSM is connected directly with the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoder. It can also be used in the applications where the DSM-CC bitstream could be controlled and transmitted by other types of network multiplexors.

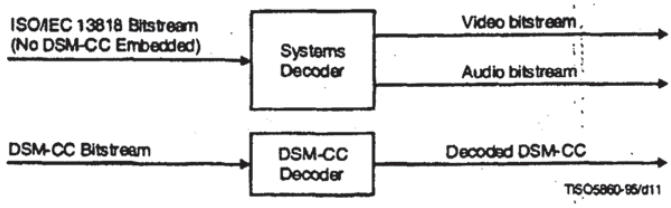

Figure B.2 – DSM-CC bitstream decoded as a standalone bitstream

For some applications, it is desirable to transmit the DSM-CC in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems bitstream so that some features of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems bitstream could be applied to the DSM-CC bitstream as well. In this case, the DSM-CC bitstream is embedded in the systems bitstream by the systems multiplexor.

The DSM-CC bitstream is encoded by the systems encoder in the following process. First, the DSM-CC bitstream is

then multiplexed into either a Program Stream (PS) or a Transport Stream (TS) according to the requirement of the transmission media. The decoding procedures are the inverse of the encoding procedures and are illustrated in the block diagram of the Systems decoder depicted in Figure B.3.

In Figure B.3, the output of the Systems decoder is a video bitstream, audio bitstream and/or DSM-CC bitstream. The DSM-CC bitstream is identified by the stream_id, value '1111 0010' as defined by the stream_id Table 2-18. Once the DSM-CC bitstream is identified, it follows the rules as specified by T-STD or P-STD.

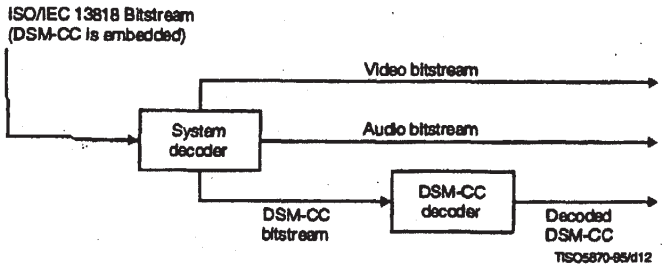

Figure B.3 – DSM-CC bitstream decoded as part of the system bitstream

B.2 Technical elements

B.2.1 Definitions

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

B.2.1.1 DSM-CC: Digital Storage Media Command and Control Commands that are specified by Recommendation H.222.0 | ISO/IEC 13818-1 for the control of digital storage media at a local or remote site containing an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream.

B.2.1.2 DSM ACK: The acknowledgment from the DSM-CC command receiver to the command initiator.

B.2.1.3 MPEG bitstream: An ISO/IEC 11172-1 Systems stream, ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Stream or ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Transport stream.

B.2.1.4 DSM-CC server: A system, either local or remote, used to store and/or retrieve an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream.

B.2.1.5 point of random access: A point in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream with the property that for at least one elementary stream within the bitstream, the next access unit, 'N', completely contained in the bitstream can be decoded without reference to previous access units, and for every elementary stream in the bitstream all access units with the same or later presentation times are completely contained subsequently in the bitstream and can be completely decoded by a system target decoder without access to information prior to the point of random access. The bitstream as stored on the DSM may have certain points of random access; the output of the DSM may include additional points of random access manufactured by the DSM's own manipulation of the stored material (e.g. storing quantization matrices so that a sequence header can be generated whenever necessary). A point of random access has an associated PTS, namely the actual or implied PTS of access unit 'N'.

B.2.1.6 current operational PTS value: The actual or implied PTS associated with the last point of random access preceding the last access unit provided from the DSM from the currently selected ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream. If no access unit has been provided from this ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream, the DSM is incapable of providing random access into the current bitstream, then the current operational PTS value is the first point of random access in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream.

B.2.1.7 DSM-CC bitstream: A sequence of bits satisfying the syntax of B.2.2.

B.2.2 Specification of DSM-CC syntax

- Every DSM control command shall commence with a start_code, as specified in Table B.1.
- Every DSM control command shall have a packet_length to specify the number of byte in a DSM-CC packet.
- When the DSM-CC bitstream is transmitted as a PES packet as defined in 2.4.3.6, the fields up to the packet_length field are identical to those specified in 2.4.3.6. In other words, if the DSM-CC packet is encapsulated in a PES packet, the PES packet start code is the only start code at the beginning of the packet.
- The actual control command or acknowledgment shall follow the last byte of the packet_length field.
- An acknowledgment stream shall be provided by the DSM control bitstream receiver after the requested operation is started or is completed, depending on the command received.
- At all times the DSM is responsible for providing a normative ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. This may include manipulating the trick mode bits defined in 2.4.3.6.

Table B.1 – DSM-CC syntax

Syntax	No. of bits	Mnemonic
DSM_CC() {		
packet_start_code_prefix	24	bslbf
stream_id	8	uimsbf
packet_length	16	uimsbf
command_id	8	uimsbf
If (command_id == '01') {		
control()		
} else if (command_id == '02') {		
ack()		
}		
}		

B.2.3 Semantics of fields in specification of DSM-CC syntax

packet_start_code_prefix – This is a 24-bit code. Together with the stream_id that follows it constitutes a DSM-CC packet start code that identifies the beginning of a DSM-CC packet bitstream. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001).

stream_id – This 8-bit field specifies the bitstream type and shall have a value '1111 0010' for the DSM-CC bitstream. Refer to Table 2-19.

packet_length – This 16-bit field specifies the number of bytes in the DSM-CC packet immediately following the last byte of this field.

command_id – This 8-bit unsigned integer identifies the bitstream is a control command or an acknowledgment stream. The values are defined in Table B.2.

Table B.2 – Command_id assigned values

Value	Command_id
0x00	Forbidden
0x01	Control
0x02	Ack
0x03 - 0xFF	Reserved

B.2.4 Control layer

Constraints on setting flags in DSM-CC control

- At most one of the flags for select, playback and storage shall be set to '1' for each DSM control command. If none of these bits are set, then this command shall be ignored.
- At most one of pause_mode, resume_mode, stop_mode, play_flag, and jump_flag shall be set for each retrieval command. If none of these bits are set, then this command shall be ignored.
- At most one of record_flag and stop_mode shall be selected for each storage command. If none of these bits are set, then this command shall be ignored.

See Table B.3.

Table B.3 – DSM-CC control

Syntax	No. of bits	Mnemonic
control() {		
select_flag	1	bslbf
retrieval_flag	1	bslbf
storage_flag	1	bslbf
reserved	12	bslbf
marker_bit	1	bslbf
If (select_flag == '1') {		
bitstream_id [31..17]	15	bslbf
marker_bit	1	bslbf
bitstream_id [16..2]	15	bslbf
marker_bit	1	bslbf
bitstream_id [1..0]	2	bslbf
select_mode	5	bslbf
marker_bit	1	bslbf
}		
if (retrieve_flag == '1') {		
jump_flag	1	bslbf
play_flag	1	bslbf
pause_mode	1	bslbf
resume_mode	1	bslbf
stop_mode	1	bslbf
reserved	10	bslbf
marker_bit	1	bslbf
if (jump_flag == '1') {		
reserved	7	bslbf
direction_indicator	1	bslbf
time_code()		
}		
if (play_flag == '1'){		
speed_mode	1	bslbf
direction_indicator	1	bslbf
reserved	6	bslbf
time_code()		
}		
}		
if (storage_flag == '1') {		
reserved	6	bslbf
record_flag	1	bslbf
stop_mode	1	bslbf
if (record_flag == '1') {		
time_code()		
}		
}		
}		

B.2.5 Semantics of fields in control layer

marker_bit –This is a 1-bit marker that is always set to '1' to avoid start code emulation.

reserved_bits – This 12-bit field is reserved for future use by this Recommendation I International Standard for DSM control commands. Until otherwise specified by ITU-T I ISO/IEC it shall have the value '0000 0000 0000'.

select_flag – This 1-bit flag when set to '1' specifies a bitstream selection operation. When it is set to '0' no bitstream selection operation shall occur.

retrieval_flag – This 1-bit flag when set to '1' specifies that a specific retrieval (playback) action will occur. The operation starts from the current operational PTS value.

storage_flag – This 1-bit flag when set to '1' specifies that a storage operation is to be executed.

bitstream_ID – This 32-bit field is coded in three parts. The parts are combined to form an unsigned integer specifying which ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is to be selected. It is the DSM server's responsibility to map the names of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams stored on its DSM uniquely to a series of numbers which could be represented by the bitstream_ID.

select_mode – This 5-bit unsigned integer specifies which mode of bitstream operation is requested. Table B.4 specifies the defined modes.

Table B.4 – Select mode assigned values

Code	Mode
0x00	Forbidden
0x01	Storage
0x02	Retrieval
0x03 - 0x1F	Reserved

jump_flag – This 1-bit flag when set to '1' specifies a jump in the playback pointer to a new access unit. The new PTS is specified by a relative time_code with respect to the current operational PTS value. This function is only valid when the current ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is in the "stop" mode.

play_flag – This 1-bit flag when set to '1' specifies to play a bitstream for a certain time period. The speed, direction, and play duration are additional parameters in the bit stream. The play starts from the current operational PTS value.

pause_mode – This is a one-bit code specifying to pause the playback action and keep the playback pointer at the current operational PTS value.

resume_mode – This is a one-bit code specifying to continue the playback action from the current operational PTS value. Resume only has meaning if the current bitstream is in the "pause" state, and the bitstream will be set to the forward play state at normal speed.

stop_mode – This is a one-bit code specifying to stop a bitstream transmission.

direction_indicator – This is a one-bit code to indicate the playback direction. If this bit is set to '1', it stands for a forward play. Otherwise it stands for a backward play.

speed_mode – This is a 1-bit code to specify the speed scale. If this bit is set to '1', it specifies that the speed is normal play. If this bit is set to '0', it specifies that the speed is fast play (i.e. fast forward or fast reverse).

record_flag – This is one-bit flag to specify the request of recording the bitstream from an end user to a DSM for a specified duration or until the reception of a stop command, whichever comes first.

B.2.6 Acknowledgment layer

Constraints on setting flags in DSM-CC control

Only one of the acks bits specified below can be set to '1' for each DSM ack bitstream (see Table B.5).

Table B.5 – DSM-CC Acknowledgment

Syntax	No. of bits	Mnemonic
ack() {		
select_ack	1	bslbf
retrieval_ack	1	bslbf
storage_ack	1	bslbf
error_ack	1	bslbf
reserved	10	bslbf
marker_bit	1	bslbf
cmd_status	1	bslbf
If (cmd_status == '1' &&		
(retrieval_ack == '1' || storage_ack == '1')) {		
time_code()		
}		
}		

B.2.7 Semantics of fields in acknowledgment layer

select_ack – This 1-bit field when it is set to '1' indicates that the ack() command is to acknowledge a select command.

retrieval_ack – This 1-bit field when set to '1' indicates that the ack() command is to acknowledge a retrieval command.

storage_ack – This 1-bit field when set to '1' indicates that the ack() command is to acknowledge a storage command.

error_ack – This 1-bit field when set to '1' indicates a DSM error. The defined errors are EOF (end of file on forward play or start of file on reverse play) on a stream being retrieved and Disk Full on a stream being stored. If this bit is set to '1', cmd_status is undefined. The current bitstream is still selected.

cmd_status – This 1-bit flag set to '1' indicates that the command is accepted. When set to '0' it indicates the command is rejected. The semantics vary according to the command received as follows:

- If select_ack is set and cmd_status is set to '1', it specifies that the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is selected and the server is ready to provide the selected mode of operation. The current operational PTS value is set to the first point of random access of the newly selected ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream. If cmd_status is set to '0', the operation has failed and no bitstream is selected.
- If retrieval_ack is set and cmd_status is set to '1', it specifies that the retrieval operation is initiated for all retrieval commands. The position of the current operational PTS pointer is reported by the succeeding time_code.
- For the play_flag command with infinite_time_flag != '1', a second acknowledgment will be sent. This will acknowledge that the play operation has ended by reaching the duration defined by the play_flag command.
- If the cmd_status is set to '0' in a retrieval acknowledgment, the operation has failed. Possible reasons for this failure include an invalid bitstream_ID, jumping beyond the end of a file, or a function not supported such as reverse play in standard speed.
- If storage_ack is set, it specifies that the storage operation is being started for the record_flag command or is completed by the stop_mode command. The PTS of the last complete access unit stored is reported by the succeeding time_code.
- If the recording operation is ended by reaching the duration defined by the storage_flag command, another acknowledgment shall be sent and the current operational PTS value after the recording shall be reported.
- If the cmd_status is set to '0' in a storage acknowledgment, the operation has failed. Possible reasons for this failure include an invalid bitstream_ID, or the inability of the DSM to store data.

B.2.8 Time code

Constraints on time code

- A forward operation of specified duration given by a time_code terminates after the actual or implied PTS of an access unit is observed such that PTS minus the current operational PTS value at the start of the operation modulo 2^{33} exceeds the duration.

- A backward operation of specified duration given by a time_code terminates after the actual or implied PTS of an access unit is observed such that current operational PTS value at the start of the operation minus that PTS modulo 2^{33} exceeds the duration.
- For all the commands in the control() layer, the time_code is specified as a relative duration with respect to the current operational PTS value.
- For all the commands in the ack() layer, the time_code is specified by the current operational PTS value.

See Table B.6.

Table B.6 – Time code

Syntax	No. of bits	Mnemonic
time_code() {		
reserved	7	bslbf
infinite_time_flag	1	bslbf
if (infinite_time_flag == '0') {		
reserved	4	bslbf
PTS [32..30]	3	bslbf
marker	1	bslbf
PTS [29..15]	15	bslbf
marker_bit	1	bslbf
PTS [14..0]	15	bslbf
marker_bit	1	bslbf
}		
}		

B.2.9 Semantics of fields in time code

infinite_time_flag – This 1-bit flag when set to '1' indicates an infinite time period. This flag is set to '1' in applications where a time period for a specific operation could not be defined in advance.

PTS [32..0] – The presentation timestamp of the access unit of the bitstream. Depending upon the function, this can be an absolute value or a relative time delay in cycles of the 90 kHz system clock.

Annex C

Program Specific Information

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

C.0 Explanation of Program Specific Information in Transport Streams

Subclause 2.4.4 contains the normative syntax, semantics and text concerning Program Specific Information. In all cases, compliance with the constraints of 2.4.4 is required. This annex provides explanatory information on how to use the PSI functions, and considers examples of how it may be used in practice.

C.1 Introduction

This Recommendation | International Standard provides a method for describing the contents of Transport Stream packets for the purpose of the demultiplexing and presentation of programs. The coding specification accommodates this function through the Program Specific Information (PSI). This annex discusses the use of PSI.

The PSI may be thought of as belonging to four tables:

1) Program Association Table (PAT);
2) TS Program Map Table (PMT);
3) Network Information Table (NIT);
4) Conditional Access Table (CAT).

The contents of the PAT, PMT and CAT are specified in this Recommendation | International Standard. The NIT is a private table, but the PID value of the Transport Stream packets which carry it is specified in the PAT. It must however, follow the section structure defined in this Recommendation | International Standard.

C.2 Functional Mechanism

The tables listed above are conceptual in that they need never be regenerated in a specified form within a decoder. While these structures may be thought of as simple tables, they may be partitioned before they are sent in Transport Stream packets: The syntax supports this operation by allowing the tables to be partitioned into sections and by providing a normative mapping method into Transport Stream packet payloads. A method is also provided to carry private data in a similar format. This is advantageous as the same basic processing in the decoder can then be used for both the PSI data and the private data helping to keep cost down. For advice on the optimum placing of PSI in the Transport Stream, see Annex D.

Each section is uniquely identified by the combination of the following elements:

i) **table_id**

The 8-bit table_id identifies to which table the section belongs.

- Sections with table_id 0x00 belong to the Program Association Table.
- Sections with table_id 0x01 belong to the Conditional Access Table.
- Sections with table_id 0x02 belong to the TS Program Map Table.

Other values of the table_id can be allocated by the user for private purposes.

It is possible to set up filters looking at the table_id field to identify whether a new section belongs to a table of interest or not.

ii) **table_id_extension**

This 16-bit field exists in the long version of a section. In the Program Association Table it is used to identify the transport_stream_id of the stream – effectively a user-defined label which allows one Transport Stream to be distinguished from another within a network or across networks. In the Conditional Access Table this field currently has no meaning and is therefore marked as "reserved" meaning that it shall be coded as 0xFFFF, but that a meaning may be defined by ITU-T | ISO/IEC in a subsequent revision of this Recommendation | International Standard. In a TS Program Map section the field contains the program_number, and thereby identifies the program to which the data in the section refers. The table_id_extension can also be used as a filter point in certain cases.

85

iii) **section_number**

The section_number field allows the sections of a particular table to be reassembled in their original order by the decoder. There is no obligation within this Recommendation | International Standard that sections must be transmitted in numerical order, but this is recommended, unless it is desired to transmit some sections of the table more frequently than others, e.g. due to random access considerations.

iv) **version_number**

When the characteristics of the Transport Stream described in the PSI change, (e.g. extra programs added, different composition of elementary streams for a given program), then new PSI data has to be sent with the updated information as the most recently transmitted version of the sections marked as "current" must always be valid. Decoders need to be able to identify whether the most recently received section is identical with the section they have already processed / stored (in which case the section can be discarded), or whether it is different, and may therefore signify a configuration change. This is achieved by sending a section with the same table_id, table_id_extension, and section_number as the previous section containing the relevant data, but with the next value version_number.

v) **current_next_indicator**

It is important to know at what point in the bitstream the PSI is valid. Each section can therefore be numbered as valid "now" (current), or as valid in the immediate future (next). This allows the transmission of a future configuration in advance of the change, giving the decoder the opportunity to prepare for the change. There is however no obligation to transmit the next version of a section in advance, but if it is transmitted, then it shall be the next correct version of that section.

C.3 The Mapping of Sections into Transport Stream Packets

Sections are mapped directly into Transport Stream packets, that is to say without a prior mapping into PES packets. Sections do not have to start at the beginning of Transport Stream packets, (although they may), because the start of the first section in the payload of a Transport Stream packet is pointed to by the pointer_field. The presence of the pointer_field is signaled by the payload_unit_start_indicator being set to a value of '1' in PSI packets. (In non-PSI packets, the indicator signals that a PES packet starts in the Transport Stream packet). The pointer_field points to the start of the first section in the Transport Stream packet. There is never more than one pointer_field in a Transport Stream packet, as the start of any other section can be identified by counting the length of the first and any subsequent sections, since no gaps between sections within a Transport Stream packet are allowed by the syntax.

It is important to note that within Transport Stream packets of any single PID value, one section must be finished before the next one is allowed to be started, or else it is not possible to identify to which section header the data belongs. If a section finishes before the end of a Transport Stream packet, but it is not convenient to open another section, a stuffing mechanism is provided to fill up the space. Stuffing is performed by filling each remaining byte of the packet with the value 0xFF. Consequently the table_id value 0xFF is forbidden, or else this would be confused with stuffing. Once a 0xFF byte has occurred at the end of a section, then the rest of the Transport Stream packet must be stuffed with 0xFF bytes, allowing a decoder to discard the rest of the Transport Stream packet. Stuffing can also be performed using the normal adaptation_field mechanism.

C.4 Repetition Rates and Random Access

In systems where random access is a consideration, it is recommended to re-transmit PSI sections several times, even when changes do not occur in the configuration, as in the general case, a decoder needs the PSI data to identify the contents of the Transport Stream, to be able to start decoding. This Recommendation | International Standard does not place any requirements on the repetition or occurrence rate of PSI sections. Clearly though, repeating sections frequently helps random access applications, whilst causing an increase in the amount of bitrate used by PSI data. If program mappings are static or quasi-static, they may be stored in the decoder to allow faster access to the data than having to wait for it to be re-transmitted. The trade-off between the amount of storage required and the desired impact on channel acquisition time may be made by the decoder manufacturer.

C.5 What is a Program?

The concept of a program has a precise definition within this Recommendation | International Standard [refer to 2.1.42 program (system)]. For a Transport Stream the time base is defined by the PCR. This effectively creates a virtual channel within the Transport Stream.

Note that this is not the same definition as is commonly used in broadcasting, where a "program" is a collection of elementary streams not only with a common timebase, but also with a common start and end time. A series of "broadcaster programs" (referred to in this annex as events) can be transmitted sequentially in a Transport Stream using the same program_number to create a "broadcasting conventional" TV-channel (sometimes called a service).

Event descriptions could be transmitted in private_sections().

A program is denoted by a program_number which has significance only within a Transport Stream. The program_number is a 16-bit unsigned integer and thus permits 65535 unique programs to exist within a Transport Stream (program_number = 0 is reserved for identification of the NIT). Where several Transport Streams are available to the decoder (e.g. in a cable network), in order to successfully demultiplex a program, the decoder must be notified of both the transport_stream_id (to find the right multiplex) and the program_number of the service (to find the right program within the multiplex).

The Transport Stream mapping may be accomplished via the optional Network Information Table. Note that the Network Information Table may be stored in decoder non-volatile memory to reduce channel acquisition time. In this case, it needs to be transmitted only often enough to support timely decoder initialization set-up operations. The contents of the NIT are private, but shall take at least the minimum section structure.

C.6 Allocation of program_number

It may not be convenient in all cases to group together all the program element which share a common clock reference as one program. It is conceivable to have a multi-service Transport Stream with only one set of PCRs, common to all. In general, though, a broadcaster may prefer to logically split up the Transport Stream into several programs, where the PCR_PID (location of the clock reference) is always the same. This method of splitting the program elements into pseudo-independent programs can have several uses. Two examples follow:

i) *multilingual transmissions into separate markets*

One video stream may be accompanied by several audio streams in different languages. It is advisable to include an example of the ISO_639_language_descriptor associated with each audio stream to enable the selection of the correct program and audio. It is reasonable to have several program definitions with different program_numbers, where all the programs reference the same video stream and PCR_PID, but have different audio PIDs. It is, however, also reasonable and possible to list the video stream and all the audio streams as one program, where this does not exceed the section size limit of 1024 bytes.

ii) *Very large program definitions*

There is a maximum limit on the length of a section of 1024 bytes (including section header and CRC_32). This means that no single program definition may exceed this length. For the great majority of cases, even with each program element having several descriptors, this size is adequate. However, one may envisage cases in very high bitrate systems, which could exceed this limit. It is then in general possible to identify methods of splitting the references of the streams, so that they do not all have to be listed together. Some program elements could be referenced under more than one program, and some under only one or the other, but not both.

C.7 Usage of PSI in a Typical System

A communications system, especially in broadcast applications, may consist of many individual Transport Streams. Each one of the four PSI data structures may appear in each and every Transport Stream in a system. There must always be a complete version of the program association table listing all programs within the Transport Stream and a complete TS program map table, containing complete program definitions for all programs within the Transport Stream. If any streams are scrambled, then there must also be a conditional access table present listing the relevant EMM streams (Entitlement Management Messages). The presence of a NIT is fully optional.

The PSI tables are mapped into Transport Stream packets via the section structure described above. Each section has a table_id field in its header, allowing sections from PSI tables and private data in private_sections to be mixed in Transport Stream packets of the same PID value or even in the same Transport Stream packet. Note, however, that within packets of the same PID, a complete section must be transmitted before the next section can be started. This is only possible for packets labeled as containing TS Program Map Table section or NIT packets however, since private sections may not be mapped into PAT or CAT packets.

It is required that all PAT sections be mapped into Transport Stream packets with PID = 0x0000 and all CA sections be mapped into packets with PID = 0x0001. PMT sections may be mapped into packets of user-selected PID value, listed as the PMT_PID for each program in the Program Association Table. Likewise, the PID for the NIT-bearing Transport Stream packets is user-selected, but must be pointed to by the entry "program_number == 0x00" in the PAT, if the NIT exists.

The contents of any CA parameter streams are entirely private, but EMMs and ECMs must also be sent in Transport Stream packets to be compliant with this Recommendation | International Standard.

Private data tables may be sent using the private_section() syntax. Such tables could be used for example in a broadcasting environment to describe a service, an upcoming event, broadcast schedules and related information.

C.8 The Relationships of PSI Structures

Figure C.1 shows an example of the relationship between the four PSI structures and the Transport Stream. Other examples are possible, but the figure shows the primary connections.

In the following subclauses, each PSI table is described.

C.8.1 Program Association Table

Every Transport Stream must contain a complete valid Program Association Table. The Program Association Table gives the correspondence between a program_number and the PID of the Transport Stream packets that carry the definition of that program (the PMT_PID). The PAT may be partitioned into up to 255 sections before it is mapped into Transport Stream packets. Each section carries a part of the overall PAT. This partitioning may be desirable to minimize data loss in error conditions. That is, packet loss or bit errors may be localized to smaller sections of the PAT, thus allowing other sections to still be received and correctly decoded. If all PAT information is put into one section, an error causing a changed bit in the table_id, for example, would cause the loss of the entire PAT. However, this is still permitted as long as the section does not extend beyond the 1024 byte maximum length limit.

Program 0 (zero) is reserved and is used to specify the Network PID. This is a pointer to the Transport Stream packets which carry the Network Information Table.

The Program Association Table is always transmitted without encryption.

C.8.2 Program Map Table

The Program Map Table provides the mapping between a program number and the program elements that comprise it. This table is present in Transport Stream packets having one or more privately-selected PID values. These Transport Stream packets may contain other private structures as defined by the table_id field. It is possible to have TS PMT sections referring to different programs carried in Transport Stream packets having a common PID value.

This Recommendation | International Standard requires a minimum of program identification: program number, PCR PID, stream types and program elements PIDs. Additional information for either programs or elementary streams may be conveyed by use of the descriptor() construct. Refer to C.8.6.

Private data may also be sent in Transport Stream packets denoted as carrying TS program map table sections. This is accomplished by the use of the private_section(). In a private_section() the application decides whether version_number and current_next_indicator represent the values of these fields for a single section or whether they are applicable to many sections as parts of a larger private table.

NOTES

1 Transport stream packets containing the Program Map Table are transmitted unencrypted.

2 It is possible to transmit information on events in private descriptors carried within the TS_program_map_section()s.

C.8.3 Conditional Access Table

The Conditional Access (CA) Table gives the association between one or more CA systems, their EMM streams and any special parameters associated with them.

NOTE – The (private) contents of the Transport Stream packets containing EMM and CA parameters if present will, in general, be encrypted (scrambled).

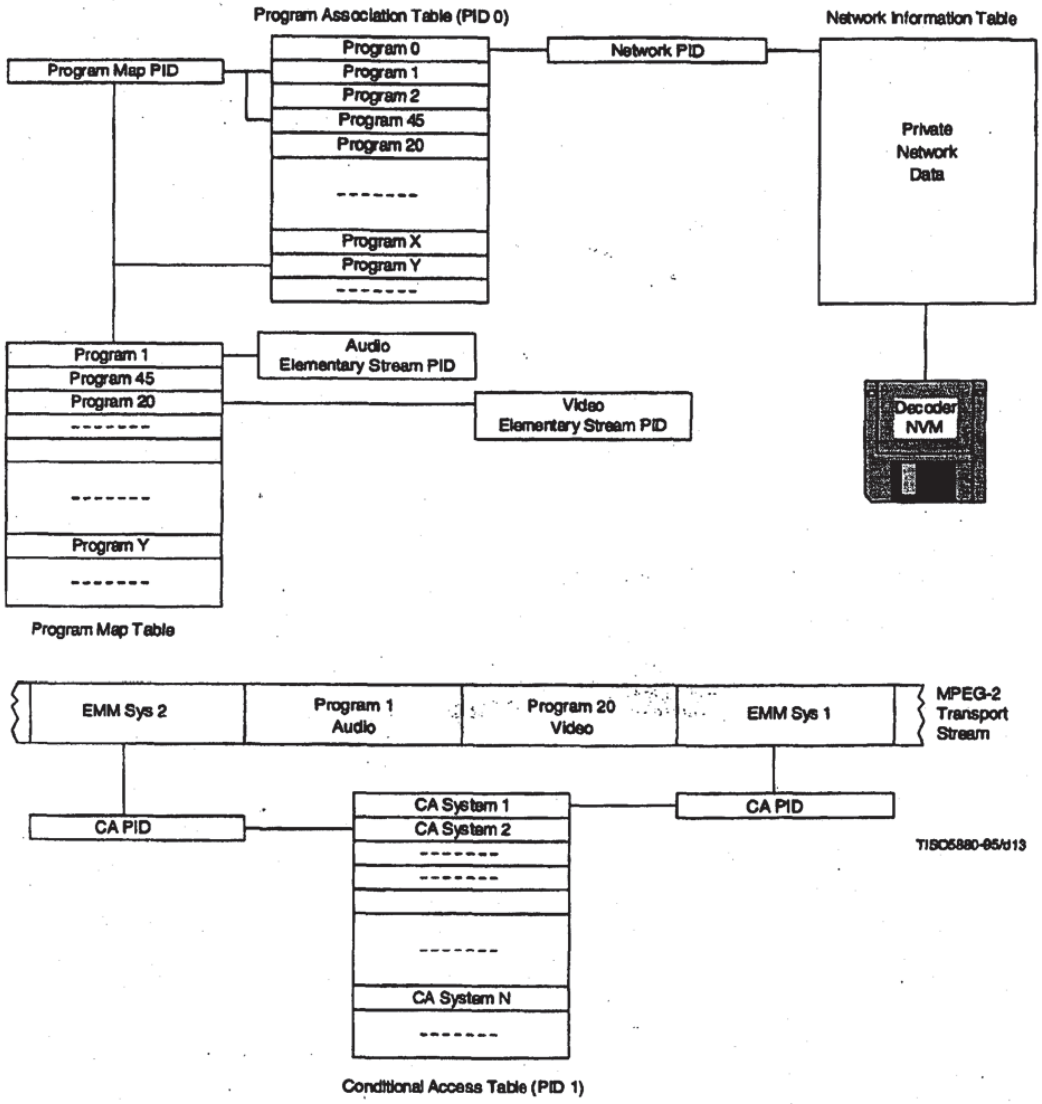

Figure C.1 – Program and network mapping relationships

89

C.8.4 Network Information Table

The contents of the NIT are private and not specified by this Recommendation | International Standard. In general, it will contain mappings of user-selected services with transport_stream_ids, channel frequencies, satellite transponder numbers, modulation characteristics, etc.

C.8.5 Private_section()

Private_sections() can occur in two basic forms, the short version (where only the fields up to and including section_length are included) or the long version (where all the fields up to and including last_section_number are present, and after the private data bytes the CRC_32 field is present).

Private_section()s can occur in PIDs which are labeled as PMT_PIDs or in Transport Stream packets with other PID values which contain exclusively private_sections(), including the PID allocated to the NIT. If the Transport Stream packets of the PID carrying the private_section()s are identified as a PID carrying private_sections (stream_type assignment value 0x05), then only private_sections may occur in Transport Stream packets of that PID value. The sections may be either of the short or long type.

C.8.6 Descriptors

There are several normative descriptors defined in this Recommendation | International Standard. Many more private descriptors may also be defined. All descriptors have a common format: {tag, length, data}. Any privately defined descriptors must adhere to this format. The data portion of these private descriptors are privately defined.

One descriptor (the CA_descriptor()), is used to indicate the location (PID value of transport packets) of ECM data associated with program elements when it is found in a TS PMT section. When found in a CA section it refers to EMMs.

In order to extend the number of private_descriptors available, the following mechanism could be used: A private descriptor_tag could be privately defined to be constructed as a composite descriptor. This entails privately defining a further sub_descriptor as the first field of the private data bytes of the private descriptor. The described structure is as indicated in Tables C.1 and C.2.

Table C.1 – Composite_descriptor

Syntax	No. of bits	Mnemonic
Composite_descriptor(){		
descriptor_tag(privately defined)	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
for (i = 0; i < N; i++){		
sub_descriptor()		
}		
}		

Table C.2 – Sub-descriptor

Syntax	No. of bits	Mnemonic
sub_descriptor() {		
sub_descriptor_tag	**8**	**uimsbf**
sub_descriptor_length	**8**	**uimsbf**
for (i = 0; i < N; i++) {		
private_data_byte	**8**	**uimsbf**
}		
}		

C.9 Bandwidth Utilization and Signal Acquisition Time

Any implementation of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream must make reasonable bandwidth demands for PSI information and, in applications where random access is a consideration, should promote fast signal acquisition. This subclause analyses this issue and gives some broadcast application examples.

The packet-based nature of the Transport Stream allows for the interspersing of PSI information with fine granularity in the multiplexed data. This provides significant flexibility in the construction and transmission of PSI.

Signal acquisition time in a real decoder is dependent on many factors, including: FDM tuning slew time, demultiplexing time, sequence headers, I-frame occurrence rate and scrambling key retrieval and processing.

This subclause examines both the bitrate and signal acquisition time impacts of the PSI syntax subclauses 2.4.4.4 and 2.4.4.9. It is assumed that the Conditional Access Table does not need to be received dynamically at every program change. This assumption is also made of the private EMM streams. This is because these streams do not contain the quickly-varying ECM components used for program element scrambling (encryption).

Also, in the discussion below, the time to acquire and process ECMs has been neglected.

The tables given below provide bandwidth usage values for a range of Transport Stream conditions. One axis of the table is the number of programs contained in a single Transport Stream. The other axis is the frequency with which the PSI information is transmitted in the Transport Stream.

This frequency will be a key determinant of the component of signal acquisition time due to PSI structures.

Both bandwidth usage tables assume that only the minimum program mapping information is provided. This means that the PID values and stream types are provided with no additional descriptors. All programs in the example are composed of two elementary streams. Program associations are 2 bytes long, while the minimal program map is 26 bytes long. There is additional overhead associated with version numbers, section lengths, etc. This will be on the order of 1-3% of the total PSI bitrate usage in sections of moderate to maximum length (a few hundred bytes to 1024 bytes) and will thus be ignored here.

The above assumptions allow forty-six (46) program associations to map into one Program Association Table Transport Stream packet (if no adaptation field is present). Similarly, seven (7) TS_program_map_sections fit into a single Transport Stream packet. It may be noted that to facilitate easy "drop/add" it is possible to transmit only one (1) TS_program_map_section per PMT_PID. This may cause an undesirable increase in PSI bitrate usage, however.

Table C.3 – Program association table bandwidth usage (bit/s)

	Number of Programs Per Transport Stream				
Frequency of PA Table Information (s^{-1})	1	5	10	32	128
1	1504	1504	1504	1504	4512
10	15040	15040	15040	15040	45120
[illegible]					
50	75200	75200	75200	75200	225600
100	150400	150400	150400	150400	451200

NOTE – Since 46 program_association_sections fit into one transport packet, the numbers in the table do not change until the last column.

Table C.4 – Program map table bandwidth usage (bit/s)

	Number of Programs Per Transport Stream				
Frequency of PM Table Information (s^{-1})	1	5	10	32	128
1	1504	1504	3008	7520	28576
10	15040	15040	30080	75200	285760
[illegible]					
50	75200	75200	150400	376000	1428800
100	150400	150400	300800	601600	2857600

Using a frequency of 25 Hz for the two PSI Tables, yields a worst case contribution to the signal acquisition time of approximately 80 ms. This would only occur when the required PAT data was "just missed" and then, once the PAT was acquired and decoded, the required PMT data was also "just missed". This doubling of the worst case acquisition time is one disadvantage of the extra level of indirection introduced by the PAT structure. This effect could be reduced by coordinated transmission of related PAT and PMT packets. Presumably, the advantage that this approach offers for "drop/add" re-multiplexing operations is compensatory.

With the 25 Hz PSI frequency, the following examples may be constructed (all examples leave ample allowance for various datalink, FEC, CA and routing overheads):

6 MHz CATV channel

- five 5.2-Mbit/s programs: 26.5 Mbit/s (includes transport overhead)
- total PSI bandwidth: 5.2 kbit/s
- CA bandwidth: 500 kbit/s

  total ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport bandwidth: 27.1 Mbit/s
- PSI Overhead: 0.28 %

OC-3 fiber channel (155 Mbit/s)

- 32 3.9-Mbit/s programs: 127.5 Mbit/s (includes transport overhead)
- total PSI bandwidth: 225.6 kbit/s
- CA bandwidth: 500 kbit/s

  total ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport bandwidth: 128.2 Mbit/s
- PSI Overhead: 0.18 %

C-band satellite transponder

- 128 256-kbit/s audio programs: 33.5 Mbit/s (includes transport overhead)
- total PSI bandwidth: 826.4 kbit/s
- CA bandwidth: 500 kbit/s

  total ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport bandwidth: 34.7 Mbit/s
- PSI Overhead: 2.4 % (actually would be lower if only one PID used per program)

As expected, the percent overhead increases for lower-rate services since many more services are possible per Transport Stream. However, the overhead is not excessive in all cases. Higher transmission rates (than 25 Hz) for the PSI data may be used to decrease the impact on channel acquisition time with only modest bitrate demand increases.

Annex D

Systems Timing Model and Application Implications of this Recommendation | International Standard

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

D.0 Introduction

The ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems specification includes a specific timing model for the sampling, encoding, encoder buffering, transmission, reception, decoder buffering, decoding, and presentation of digital audio and video in combination. This model is embodied directly in the specification of the syntax and semantic requirements of compliant ITU-T Rec. H.222.0 | ISO/IEC 13818-1 data streams. Given that a decoding system receives a compliant bit stream that is delivered correctly in accordance with the timing model it is straightforward to implement the decoder such that it produces as output high quality audio and video which are properly synchronized. There is no normative requirement, however, that decoders be implemented in such a way as to provide such high quality presentation output. In applications where the data are not delivered to the decoder with correct timing, it may be possible to produce the desired presentation output, however such capabilities are not in general guaranteed. This informative annex describes the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems timing model in detail, and gives some suggestions for implementing decoder systems to suit some typical applications.

D.0.1 Timing Model

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems embodies a timing model in which all digitized pictures and audio samples that enter the encoder are presented exactly once each, after a constant end to end delay, at the output of the decoder. As such, the sample rates, i.e. the video frame rate and the audio sample rate, are precisely the same at the decoder as they are at the encoder. This timing model is diagrammed in Figure D.1:

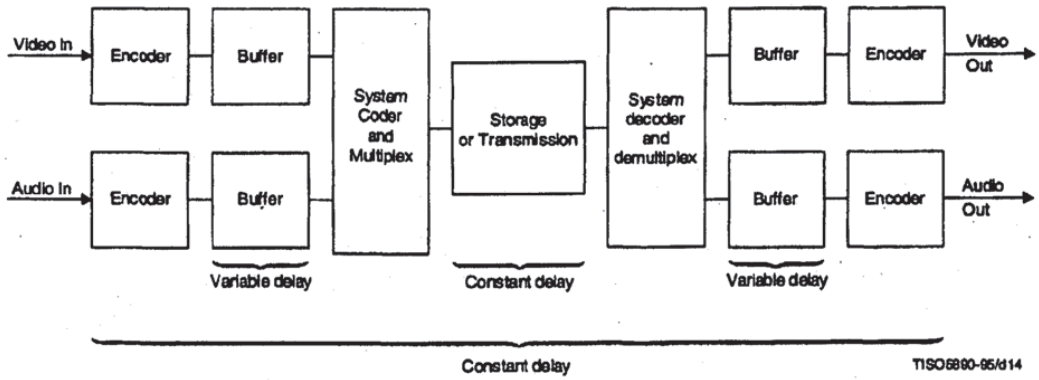

Figure D.1 – Constant delay model

As indicated in Figure D.1, the delay from the input to the encoder to the output or presentation from the decoder is constant in this model[2)], while the delay through each of the encoder and decoder buffers is variable. Not only is the delay through each of these buffers variable within the path of one elementary stream, the individual buffer delays in the video and audio paths differ as well. Therefore the relative location of coded bits representing audio or video in the combined stream does not indicate synchronization information. The relative location of coded audio and video is

2) Constant delay as indicated for the entire system is required for correct synchronizaton, however some deviations are possible. Network delay is discussed as being constant. Slight deviations may be tolerated, and network adaptation may allow greater variations of network delay. Both of these are discussed later.

constrained only by the System Target Decoder (STD) model such that the decoder buffers must behave properly; therefore coded audio and video that represent sound and pictures that are to be presented simultaneously may be separated in time within the coded bit stream by as much as one second, which is the maximum decoder buffer delay that is allowed in the STD model.

The audio and video sample rates at the encoder are significantly different from one another, and may or may not have an exact and fixed relationship to one another, depending on whether the combined stream is a Program Stream or a Transport Stream, and on whether the System_audio_locked and System_video_locked flags are set in the Program Stream. The duration of a block of audio samples (an audio presentation unit) is generally not the same as the duration of a video picture.

There is a single, common system clock in the encoder, and this clock is used to create timestamps that indicate the correct presentation and decoding timing of audio and video, as well as to create timestamps that indicate the instantaneous values of the system clock itself at sampled intervals. The timestamps that indicate the presentation time of audio and video are called Presentation Time Stamps (PTS). Those that indicate the decoding time are called Decoding Timestamps (DTS), and those that indicate the value of the system clock are called the System Clock Reference (SCR) in Program Streams and the Program Clock Reference (PCR) in Transport Streams. It is the presence of this common system clock in the encoder, the timestamps that are created from it, and the recreation of the clock in the decoder and the correct use of the timestamps that provide the facility to synchronize properly the operation of the decoder.

Encoder implementations may not follow this model exactly, however the data stream which results from the actual encoder, storage system, network, and one or more multiplexor must follow the model precisely. (Delivery of the data may deviate somewhat, depending on the application). Therefore in this annex, the term "encoder system clock" is used to mean either the actual common system clock as described in this model or the equivalent function, however it may be implemented.

Since the end-to-end delay through the entire system is constant, the audio and video presentations are precisely synchronized. The construction of System bit streams is constrained such that when they are decoded by a decoder that follows this model with the appropriately sized decoder buffers, those buffers are guaranteed never to overflow nor underflow, with specific exceptions allowing intentional underflow.

In order for the decoder system to incur the precise amount of delay that causes the entire end-to-end delay to be constant, it is necessary for the decoder to have a system clock whose frequency of operation and absolute instantaneous value match those of the encoder. The information necessary to convey the encoder's system clock is encoded in the SCR or PCR; this function is explained below.

Decoders which are implemented in accordance with this timing model such that they present audio samples and video pictures exactly once (with specific intentionally coded exceptions), at a constant rate, and such that decoder buffers behave as in the model, are referred to in this annex as precisely timed decoders, or those that produce precisely timed output. Decoder implementations are not required by this International Standard to present audio and video in accordance with this model; it is possible to construct decoders that do not have constant delay, or equivalently do not present each picture or audio sample exactly once. In such implementations, however, the synchronization between presented audio and video may not be precise, and the behaviour of the decoder buffers may not follow the reference decoder model. It is important to avoid overflow at the decoder buffers, as overflow causes a loss of data that may have significant effects on the resulting decoding process. This annex covers primarily the operation of such precisely timed decoders and some of the options that are available in implementing these decoders.

D.0.2 Audio and Video Presentation Synchronization

Within the coding of this Recommendation | International Standard Systems data are timestamps concerning the presentation and decoding of video pictures and blocks of audio samples. The pictures and blocks are called "Presentation Units", abbreviated PU. The sets of coded bits which represent the PUs and which are included within the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bit stream are called "Access Units", abbreviated AU. An audio access unit is abbreviated AAU, and a video access unit is abbreviated VAU. In ISO/IEC 13818-3 audio the term "audio frame" has the same meaning as AAU or APU depending on the context. A VPU is a picture, and a VAU is a coded picture.

Some, but not necessarily all, AAUs and VAUs have associated with them PTSs. A PTS indicates the time that the PU which results from decoding the AU which is associated with the PTS should be presented to the user. The audio PTSs and video PTSs are both samples from a common time clock, which is referred to as the System Time Clock or STC. With the correct values of audio and video PTSs included in the data stream, and with the presentation of the audio and video PUs occurring at the time indicated by the appropriate PTSs in terms of the common STC, precise synchronization of the presented audio and video is achieved at the decoding system. While the STC is not part of the normative content of this Recommendation | International Standard, and the equivalent information is conveyed in this Recommendation |

International Standard via such terms as the system_clock_frequency, the STC is an important and convenient element for explaining the timing model, and it is generally practical to implement encoders and decoders which include an STC in some form.

PTSs are required for the conveyance of accurate relative timing between audio and video, since the audio and video PUs generally have significantly different and essentially unrelated durations. For example, audio PUs of 1152 samples each at a sample rate of 44 100 samples per second have a duration of approximately 26,12 ms, and video PUs at a frame rate of 29,97 Hz have a duration of approximately 33,76 ms. In general the temporal boundaries of APUs and VPUs rarely, if ever, coincide. Separate PTSs for audio and video provide the information that indicates the precise temporal relation of audio and video PUs without requiring any specific relationship between the duration and interval of audio and video PUs.

The values of the PTS fields are defined in terms of the System Target Decoder or STD, which is a fundamental normative constraint on all System bit streams. The STD is a mathematical model of an idealized decoder which specifies precisely the movement of all bits into and out of the decoder's buffers, and the basic semantic constraint imposed on the bit stream is that the buffers within the STD must never overflow nor underflow, with specific exceptions provided for underflow in special cases. In the STD model the virtual decoder is always exactly synchronized with the data source, and audio and video decoding and presentation are exactly synchronized. While exact and consistent, the STD is somewhat simplified with respect to physical implementations of decoders in order to clarify its specification and to facilitate its broad application to a variety of decoder implementations. In particular, in the STD model each of the operations performed on the bit stream in the decoder is performed instantaneously, with the obvious exception of the time that bits spend in the decoder buffers. In a real decoder system the individual audio and video decoders do not perform instantaneously, and their delays must be taken into account in the design of the implementation. For example, if video pictures are decoded in exactly one picture presentation interval 1/P, where P is the frame rate, and compressed video data are arriving at the decoder at bit rate R, the completion of removing bits associated with each picture is delayed from the time indicated in the PTS and DTS fields by 1/P, and the video decoder buffer must be larger than that specified in the STD model by R/P. The video presentation is likewise delayed with respect to the STD, and the PTS should be handled accordingly. Since the video is delayed, the audio decoding and presentation should be delayed by a similar amount in order to provide correct synchronization. Delaying decoding and presentation of audio and video in a decoder may be implemented for example by adding a constant to the PTS values when they are used within the decoder.

Another difference between the STD and precise practical decoder implementation is that in the STD model the explicit assumption is made that the final audio and video output is presented to the user instantaneously and without further delay. This may not be the case in practice, particularly with cathode-ray tube displays, and this additional delay should also be taken into account in the design. Encoders are required to encode audio and video such that the correct synchronization is achieved when the data is decoded with the STD. Delays in the input and sampling of audio and video, such as video camera optical charge integration, must be taken into account in the encoder.

In the STD model proper synchronization is assumed and the timestamps and buffer behaviour are tested against this assumption as a condition of bit stream validity. Of course in a physical decoder precise synchronization is not automatically the case, particularly upon start-up and in the presence of timing jitter. Precise decoder timing is a goal to be targeted by decoder designs. Inaccuracy in decoder timing affects the behaviour of the decoder buffers. These topics are covered in more detail in later subclauses of this annex.

The STD includes Decoding Time Stamps (DTS) as well as PTS fields. The DTS refers to the time that an AU is to be extracted from the decoder buffer and decoded in the STD model. Since the audio and video elementary stream decoders are instantaneous in the STD, the decoding time and presentation time are identical in most cases; the only exception occurs with video pictures which have undergone re-ordering within the coded bit stream, i.e. I- and P-pictures in the case of non-low-delay video sequences. In cases where re-ordering exists, a temporary delay buffer in the video decoder is used to store the appropriate decoded I- or P-picture until it should be presented. In all cases where the decoding and presentation times are identical in the STD, i.e. all AAUs, B-picture VAUs, and I- and P-picture VAUs within low-delay video sequences, the DTS is not coded, as it would have the same value as the PTS. Where the values differ, both are coded if either is coded. For all AUs where only the PTS is coded, this field may be interpreted as being both the PTS and the DTS.

Since PTS and DTS values are not required for every AAU and VAU, the decoder may choose to interpolate values which are not coded. PTS values are required with intervals not exceeding 700 ms in each elementary audio and video stream. These time intervals are measured in presentation time, that is, in the same context as the values of the fields, not in terms of the times that the fields are transmitted and received. In cases of data streams where the system, video and audio clocks are locked, as defined in the normative part of this Recommendation | International Standard, each AU following one for which a DTS or PTS is explicitly coded has an effective decoding time of the sum of that for the previous AU plus a fixed and specified difference in value of the STC. For example, in video coded at 29,97 Hz each

picture has a difference in time of 3003 cycles of the 90 kHz portion of the STC from the previous picture when the video and system clocks are locked. The same time relationship exists for decoding successive AUs, although re-ordering delay in the decoder affects the relationship between decoder AUs and presented PUs. When the data stream is coded such that the video or audio clock is not locked to the system clock the time difference between decoding successive AUs may be estimated using the same values as indicated above; however these time differences are not exact due to the fact that relationships between the frame rate, audio sample rate, and system clock frequency were not exact at the encoder.

Note that the PTS and DTS fields do not, by themselves, indicate the correct fullness of the decoder buffers at start up nor at any other time, and equivalently, they do not indicate the amount of time delay that should elapse upon receiving the initial bits of a data stream before decoding should start. This information is retrieved by combining the functions of the PTS and DTS fields and correct clock recovery, which is covered below. In the STD model, and therefore in decoders which are modeled after it, the decoder buffer behaviour is determined completely by the SCR (or PCR) values, the times that they are received, and the PTS and DTS values, assuming that data is delivered in accordance with the timing model. This information specifies the time that coded data spends in the decoder buffers. The amount of data that is in the coded data buffers is not explicitly specified, and this information is not necessary, since the timing is fully specified. Note also that the fullness of the data buffers may vary considerably with time in a fashion that is not predictable by the decoder, except through the proper use of the timestamps.

In order for the audio and video PTSs to refer correctly to a common STC, a correctly timed common clock must be ade available within the decoder system. This is subject of the next subclause.

D.0.3 System Time Clock recovery in the decoder

Within the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems data stream there are, in addition to the PTS and DTS fields, clock reference timestamps. These references are samples of the system time clock, which are applicable both to a decoder and to an encoder. They have a resolution of one part in 27 000 000 per second, and occur at intervals up to 100 ms in Transport Streams, or up to 700 ms in Program Streams. As such, they can be utilized to implement clock reconstruction control loops in decoders with sufficient accuracy for all identified applications.

In the Program Stream, the clock reference field is called the System Clock Reference or SCR. In the Transport Stream, the clock reference field is called the Program Clock Reference or PCR. In general the SCR and PCR definitions may be considered to be equivalent, although there are distinctions. The remainder of this subclause uses the term SCR for clarity; the same statements apply to the PCR except where otherwise noted. The PCR in Transport Streams provides the clock reference for one program, where a program is a set of elementary streams that have a common time base and are intended for synchronized decoding and presentation. There may be multiple programs in one Transport Stream, and each may have an independent time base and a separate set of PCRs.

The SCR field indicates the correct value of the STC when the SCR is received at the decoder. Since the SCR occupies more than one byte of data, and System data streams are defined as streams of bytes, the SCR is defined to arrive at the decoder when the last byte of the system_clock_reference_base field is received at the decoder. Alternatively the SCR can be interpreted as the time that the SCR field should arrive at the decoder, assuming that the STC is already known to be correct. Which interpretation is used depends on the structure of the application system. In applications where the data source can be controlled by the decoder, such as a locally attached DSM, it is possible for the decoder to have an autonomous STC frequency, and so the STC need not be recovered. In many important applications, however, this assumption cannot be made correctly. For example, consider the case where a data stream is delivered simultaneously to multiple decoders. If each decoder has its own autonomous STC with its own independent clock frequency, the SCRs cannot be assured to arrive at the correct time at all decoders; one decoder will in general require the SCRs sooner than the source is delivering them, while another requires them later. This difference cannot be made up with a finite size data buffer over an unbounded length of time of data reception. Therefore the following addresses primarily the case where the STC must slave its timing to the received SCRs (or PCRs).

In a correctly constructed and delivered ITU-T Rec. H.222.0 | ISO/IEC 13818-1 data stream, each SCR arrives at the decoder at precisely the time indicated by the value of that SCR. In this context, "time" means correct value of the STC. In concept, this STC value is the same value that the encoder's STC had when the SCR was stored or transmitted. However, the encoding may have been performed not in real time or the data stream may have been modified since it was originally encoded, and in general the encoder or data source may be implemented in a variety of ways such that the encoder's STC may be a theoretical quantity.

If the decoder's clock frequency matches exactly that of the encoder, then the decoding and presentation of video and audio will automatically have the same rate as those at the encoder, and the end-to-end delay will be constant. With matched encoder and decoder clock frequencies, any correct SCR value can be used to set the instantaneous value of the decoder's STC, and from that time on the decoder's STC will match that of the encoder without the need for further 96

adjustment. This condition remains true until there is a discontinuity of timing, such as the end of a Program Stream or the presence of a discontinuity indicator in a Transport Stream.

In practice a decoder's free-running system clock frequency will not match the encoder's system clock frequency which is sampled and indicated in the SCR values. The decoder's STC can be made to slave its timing to the encoder using the received SCRs. The prototypical method of slaving the decoder's clock to the received data stream is via a phase-locked loop (PLL). Variations of a basic PLL, or other methods, may be appropriate, depending on the specific application requirements.

A straight-forward PLL which recovers the STC in a decoder is diagrammed and described here.

Figure D.2 shows a classic PLL, except that the reference and feedback terms are numbers (STC and SCR or PCR values) instead of signal events such as edges.

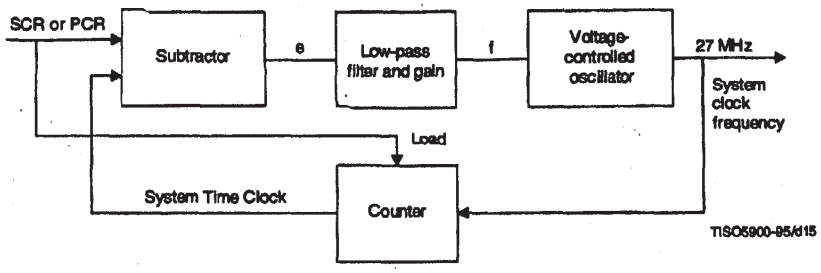

Figure D.2 – STC recovery using PLL

Upon initial acquisition of a new time base, i.e. a new program, the STC is set to the current value encoded in the SCRs. Typically the first SCR is loaded directly into the STC counter, and the PLL is subsequently operated as a closed loop. Variations on this method may be appropriate, i.e. if the values of the SCRs are suspect due to jitter or errors.

The closed-loop action of the PLL is as follows. At the moment that each SCR (or PCR) arrives at the decoder, that value is compared with the current value of the STC. The difference is a number, which has one part in units of 90 kHz and one part in terms of 300 times this frequency, i.e. 27 MHz. The difference value is linearized to be in a single number space, typically units of 27 MHz, and is called "e", the error term in the loop. The sequence of e terms is input to the low-pass filter and gain stage, which are designed according to the requirements of the application. The output of this stage is a control signal "f" which controls the instantaneous frequency of the Voltage Controlled Oscillator (VCO). The output of the VCO is an oscillator signal with a nominal frequency of 27 MHz; this signal is used as the system clock frequency within the decoder. The 27 MHz clock is input to a counter which produces the current STC values, which consist of both a 27 MHz extension, produced by dividing by 300, and a 90 kHz base value which is derived by counting the 90 kHz results in a 33-bit counter. The 33 bit, 90 kHz portion of the STC output is used as needed for comparison with PTS and DTS values. The complete STC is also the feedback input to the subtractor.

The bounded maximum interval between successive SCRs (700 ms) or PCRs (100 ms) allows the design and construction of PLLs which are known to be stable. The bandwidth of the PLLs has an upper bound imposed by this interval. As shown below, in many applications the PLL required has a very low bandwidth, and so this bound typically does not impose a significant limitation on the decoder design and performance.

If the free-running or initial frequency of the VCO is close enough to the correct, encoder's system clock frequency, the decoder may be able to operate satisfactorily as soon as the STC is initialized correctly, before the PLL has reached a defined locked state. For a given decoder STC frequency which differs by a bounded amount from the frequency encoded in the SCRs and which is within the absolute frequency bounds required by the decoder application, the effect of the mis-match between the encoder's and the decoder's STC frequencies if there were not PLL is the gradual and

unavoidable increase or decrease of the fullness of the decoder's buffers, such that overflow or underflow would occur eventually with any finite size of decoder buffers. Therefore the amount of time allowable before the decoder's STC frequency is locked to that of the encoder is determined by the allowable amount of additional decoder buffer size and delay.

If the SCRs are received by the decoder with values and timing that reflect instantaneously correct samples of a constant frequency STC in the encoder, then the error term e converges to an essentially constant value after the loop has reached the locked state. This condition of correct SCR values is synonymous with either constant-delay storage and transmission of the data from the encoder to the decoder, or if this delay is not constant, the effective equivalent of constant delay storage and transmission with the SCR values having been corrected to reflect the variations in delay. With the values of e converging to a constant, variations in the instantaneous VCO frequency become essentially zero after the loop is locked; the VCO is said to have very little jitter or frequency slew. While the loop is in the process of locking, the rate of change of the VCO frequency, the frequency slew rate, can be controlled strictly by the design of the low pass filter and gain stage. In general the VCO slew rate can be designed to meet application requirements, subject to constraints of decoder buffer size and delay.

D.0.4 SCR and PCR Jitter

If a network or a Transport Stream re-multiplexor varies the delay in delivering the data stream from the encoder or storage system to the decoder, such variations tend to cause a difference between the values of the SCRs (or PCRs) and the values that they should have when they are actually received. This is referred to as SCR or PCR jitter. For example, if the delay in delivering one SCR is greater than the delay experienced by other similar fields in the same program, that SCR is late. Similarly, if the delay is less than for other clock reference fields in the program, the field is early.

Timing jitter at the input to a decoder is reflected in the combination of the values of the SCRs and the times when they are received. Assuming a clock recovery structure as illustrated in Figure D.2, any such timing jitter will be reflected in the values of the error term e; and non-zero values of e induce variations in the values of f, resulting in variations in the frequency of the 27 MHz system clock. Variations in the frequency of the recovered clock may or may not be acceptable within decoder systems, depending on the specific application requirements. For example, in precisely timed decoders that produce composite video output, the recovered clock frequency is typically used to generate the composite video sample clock and the chroma sub-carrier; the applicable specifications for sub-carrier frequency stability may permit only very slow adjustment of the system clock frequency. In applications where a significant amount of SCR or PCR jitter is present at the decoder input and there are tight constraints on the frequency slew rate of the STC, the constraints of reasonable additional decoder buffer size and delay may not allow proper operation.

The presence of SCR or PCR jitter may be caused for example by network transmission which incorporates packet or cell multiplexing or variable delay of packets through the network, as may be caused by queuing delays or by variable network access time in shared-media systems.

Multiplexing or re-multiplexing of Transport or Program Streams changes the order and relative temporal location of data packets and therefore also of SCRs or PCRs. The change in temporal location of SCRs causes the value of previously correct SCRs to become incorrect, since in general the time at which they are delivered via a constant delay network is not correctly represented by their values. Similarly, a Program Stream or Transport Stream with correct SCRs or PCRs may be delivered over a network which imposes a variable delay on the data stream, without correcting the SCR or PCR values. The effect is once again SCR or PCR jitter, with attendant effects on the decoder design and performance. The worst case amount of jitter which is imposed by a network on the SCRs or PCRs received at a decoder depends on a number of factors which are beyond the scope of this Recommendation | International Standard, including the depth of queues implemented in each of the network switches and the total number of network switches or re-multiplexing operations which operate in cascade on the data stream.

In the case of a Transport Stream, correction of PCRs is necessary in a re-multiplex operation, creating a new Transport Stream from one or more Transport Streams. This correction is accomplished by adding a correction term to the PCR; this term can be computed as:

$$\Delta PCR = del_{act} - del_{const}$$

where del_{act} is the actual delay experienced by the PCR, and del_{const} is a constant which is used for all PCRs of that program. The value which should be used for del_{const} will depend on the strategy used by the original encoder/multiplexor. This strategy could be, for instance, to schedule packets as early as possible, in order to allow later transmission links to delay them. In Table D.1, three different multiplex strategies are shown together with the appropriate value for del_{const}.

Table D.1 – Re-multiplexing strategy

Strategy	del_{const}
Early	del_{min}
Late	del_{max}
Middle	del_{avg}

When designing a system, private agreements may be needed as to what strategy should be used by the encoder/multiplexors, since this will have an effect on the ability to perform any additional re-multiplexing.

The amount of multiplex jitter allowed is not normatively bounded in this Recommendation | International Standard. However, 4 ms is intended to be the maximum amount of jitter in a well behaved system.

In systems which include re-multiplexors special care might be necessary to ensure that the information in the Transport Stream is consistent. In particular, this applies to PSI and to discontinuity points. Changes in PSI tables might need to be inserted into a Transport Stream in such a way that subsequent re-multiplexor steps never move them so far that information becomes incorrect. For instance, a new version of PMT section in some cases should not be sent within 4 ms of the data affected by the change.

Similarly, it may be necessary for an encoder/mux to avoid inserting PTS or DTS in a ±4 ms window around a discontinuity point.

D.0.5 Clock Recovery in the Presence of Network Jitter

In applications in which there is any significant amount of jitter present in the received clock reference timestamps, there are several choices available for decoder designs; how the decoder is designed depends in large part on the requirements for the decoder's output signal characteristics as well as the characteristics of the input data and jitter.

Decoders in various applications may have differing requirements for the accuracy and stability of the recovered system clock, and the degree of this stability and accuracy that is required may be considered to fall along a single axis. One extreme of this axis may be considered to be those applications where the reconstructed system clock is used directly to synthesize a chroma sub-carrier for use in composite video. This requirement generally exists where the presented video is of the precisely timed type, as described above, such that each coded picture is presented exactly once, and where the output is composite video in compliance with the applicable specifications. In that case the chroma sub-carrier, the pixel clock, and the frame rate all have exactly specified ratios, and all of these have a defined relationship to the system clock. The composite video sub-carrier must have at least sufficient accuracy and stability that any normal television receiver's chroma sub-carrier PLL can lock to the sub-carrier, and the chroma signals which are demodulated using the recovered sub-carrier do not show visible chrominance phase artifacts. The requirement in some applications is to use the system clock to generate a sub-carrier that is in full compliance with the NTSC, PAL, or SECAM specifications, which are typically even more stringent than those imposed by typical television receivers. For example, the SMPTE specification for NTSC requires a sub-carrier accuracy of 3 ppm, with a maximum short term jitter of 1 ns per horizontal line time and a maximum long term drift of 0,1 Hz per second.

In applications where the recovered system clock is not used to generate a chroma sub-carrier, it may still be used to generate a pixel clock for video and it may be used to generate a sample clock for audio. These clocks have their own stability requirements that depend on the assumptions made about the receiving display monitor and on the acceptable amount of audio frequency drift, or "wow and flutter", at the decoder's output.

In applications where each picture and each audio sample are not presented exactly once, i.e. picture and audio sample "slipping" is allowed, the system clock may have relatively loose accuracy and stability requirements. This type of decoder may not have precise audio-video presentation synchronization, and the resulting audio and video presentation may not have the same quality as for precisely timed decoders.

The choice of requirements for the accuracy and stability of the recovered system clock is application dependent. The following focuses on the most stringent requirement which is identified above, i.e. where the system clock is to be used to generate a chroma sub-carrier.

D.0.6 System clock used for chroma sub-carrier generation

The decoder design requirements can be determined from the requirements on the resulting sub-carrier and the maximum amount of network jitter that must be accepted. Similarly, if the system clock performance requirements and the decoder design's capabilities are known, the tolerable maximum network jitter can be determined. While it is beyond the scope of this Recommendation I International Standard to state such requirements, the numbers which are needed to specify the design are identified in order to clarify the statement of the problem and to illustrate a representative design approach.

With a clock recovery PLL circuit as illustrated in Figure D.2, the recovered system clock must meet the requirements of a worst case frequency deviation from the nominal, measured in units of ppm (parts per million), and a worst case frequency slew rate, measured in ppm/s (ppm per second). The peak-to-peak uncorrected network timing jitter has a value that may be specified in milliseconds. In such a PLL the network timing jitter appears as the error term **e** in the diagram, and since the PLL acts as a low-pass filter on jitter at its input, the worst case effect on the 27 MHz output frequency occurs when there is a maximum amplitude step function of PCR timing at the input. The value **e** then has a maximum amplitude equal to the peak-to-peak jitter, which is represented numerically as the jitter times 2**33 in the base portion of the SCR or PCR encoding. The maximum rate of change of the output of the low pass filter (LPF), **f**, with this maximum value of **e** at its input, directly determines the maximum frequency slew rate of the 27 MHz output. For any given maximum value of **e** and maximum rate of change of **f** a LPF can be specified. However, as the gain or cut-off frequency of the LPF is reduced, the time required for the PLL to lock to the frequency represented by the SCRs or PCRs is increased. Implementation of PLLs with very long time constants can be achieved through the use of digital LPF techniques, and possibly analogue filter techniques. With digital LPF implementations, when the frequency term **f** is the input to an analogue VCO, **f** is quantized by a digital to analogue converter, whose step size should be considered when calculating the maximum slew rate of the output frequency.

In order to ensure that **e** converges to a value that approaches zero, the open loop gain of the PLL must be very high, such as might be implemented in an integrator function in the low-pass filter in the PLL.

With a given accuracy requirement, it may be reasonable to construct the PLL such that the initial operating frequency of the PLL meets the accuracy requirement. In this case the initial 27 MHz frequency before the PLL is locked is sufficiently accurate to meet the stated output frequency requirement. If it were not for the fact that the decoder's buffers would eventually overflow or underflow, this initial system clock frequency would be sufficient for long term operation. However, from the time the decoder begins to receive and decode data until the system clock is locked to the time and clock frequency that is represented by the received SCRs or PCRs, data is arriving at the buffers at a different rate than it is being extracted, or equivalently the decoder is extracting access units at times that differ from those of the System Target Decoder (STD) model. The decoder buffers will continue to become more or less full than those of the STD according to the trajectory of recovered system clock frequency with respect to the encoder's clock frequency. Depending on the relative initial VCO frequency and encoder system clock frequency, decoder buffer fullness is either increasing or decreasing. Assuming this relationship is not known, the decoder needs additional data buffering to allow for either case. The decoder should be constructed to delay all decoding operations by an amount of time that is at least equal to the amount of time that is represented by the additional buffering that is allocated for the case of the initial VCO frequency being greater than the encoder's clock frequency, in order to prevent buffer underflow. If the initial VCO frequency is not sufficiently accurate to meet the stated accuracy requirements, then the PLL must reach the locked state before decoding may begin, and there is a different set of considerations regarding the PLL behaviour during this time and the amount of additional buffering and static delay which is appropriate.

A step function in the input timing jitter which produces a step function in the error term **e** of the PLL in Figure D.2 must produce an output frequency term **f** such that when it is multiplied by the VCO gain the maximum rate of change is less than the specified frequency slew rate. The gain of the VCO is stated in terms of the amount of the change in output frequency with respect to a change in control input. An additional constraint on the LPF in the PLL is that the static value of **e** when the loop is locked must be bounded in order to bound the amount of additional buffering and static decoding delay that must be implemented. This term is minimized when the LPF has very high DC gain.

Clock recovery circuits which differ somewhat from that shown in Figure D.2 may be practical. For example, it may be possible to implement a control loop with a Numerically Controlled Oscillator (NCO) instead of a VCO, wherein the NCO uses a fixed frequency oscillator and clock cycles are inserted or deleted from normally periodic events at the output in order to adjust the decoding and presentation timing. There may be some difficulties with this type of approach when used with composite video, as there is a tendency to cause either problematic phase shifts of the sub-carrier or jitter in the horizontal or vertical scan timing. One possible approach is to adjust the period of horizontal scans at the start of vertical blanking, while maintaining the phase of the chroma sub-carrier.

In summary, depending on the values specified for the requirements, it may or may not be practical to construct a decoder which reconstructs the system clock with sufficient accuracy and stability, while maintaining desired decoder buffer sizes and added decoding delay.

D.0.7 Component video and audio reconstruction

If component video is produced at the decoder output, the requirements for timing accuracy and stability are generally less stringent than is the case for composite video. Typically the frequency tolerance is that which the display deflection circuitry can accept, and the stability tolerance is determined by the need to avoid visible image displacement on the display.

The same principles as illustrated above apply, however the specific requirements are generally easier to meet.

Audio sample rate reconstruction again follows the same principles, however the stability requirement is determined by the amount of acceptable long and short term sample rate variation. Using a PLL approach as illustrated in the previous subclause, short term deviation can be made to be very small, and longer term frequency variation is manifested as variation in perceived pitch. Again, once specified bounds on this variation are set specific design requirements can be determined.

D.0.8 Frame Slipping

In some applications where precise decoder timing is not required, the decoder's system time clock may not adjust its operating frequency to match the frequency represented by received SCRs (or PCRs); it may have a free-running 27 MHz clock instead, while still slaving the decoder's STC to the received data. In this case the STC value must be updated as needed to match the received SCRs. Updating the STC upon receipt of SCRs causes discontinuities in the STC value. The magnitude of these discontinuities depends upon the difference between the decoder's 27 MHz frequency and the encoder's 27 MHz, i.e. that which is represented by the received SCRs, and upon the time interval between successive received SCRs or PCRs. Since the decoder's 27 MHz system clock frequency is not locked to that of the received data, it cannot be used to generate the video or audio sample clocks while maintaining the precise timing assumptions of presenting each video and audio presentation unit exactly once and of maintaining the same picture and audio presentation rate at the decoder and the encoder, with precise audio and video synchronization. There are multiple possibilities for implementing decoding and presentation systems using this structure.

In one type of implementation, the pictures and audio samples are decoded at the time indicated by the decoder's STC, while they are presented at slightly different times, according to the locally produced sample clocks. Depending on the relationships of the decoder's sample clocks to the encoder's system clock, pictures and audio samples may on occasion be presented more than one each or not at all; this is referred to as "frame slipping" or "sample slipping", in the case of audio. There may be perceptible artifacts introduced by this mechanism. The audio-video synchronization will in general not be precise, due to the units of time over which pictures, and perhaps audio presentation units, are repeated or deleted. Depending on the specific implementation, additional buffering in the decoder is generally needed for coded data or decoded presentation data. Decoding may be performed immediately before presentation, and not quite at the time indicated in the decoder's STC, or decoded presentation units may be stored for delayed and possibly repeated presentation. If decoding is performed at the time of presentation, a mechanism is required to support deleting the presentation of pictures and audio samples without causing problems in the decoding of predictively coded data.

D.0.9 Smoothing of network jitter

In some applications it may be possible to introduce a mechanism between a network and a decoder in order to reduce the degree of jitter which is introduced by a network. Whether such an approach is feasible depends on the type of streams received and the amount and type of jitter which is expected.

Both the Transport Stream and the Program Stream indicate within their syntax the rate at which the stream is intended to be input to a decoder. These indicated rates are not precise, and cannot be used to reconstruct data stream timing exactly. They may, however, be useful as part of a smoothing mechanism.

For example, a Transport Stream may be received from a network such that the data is delivered in bursts. It is possible to buffer the received data and to transmit data from the buffer to the decoder at an approximately constant rate such that the buffer remains approximately one-half full.

However, a variable rate stream should not be delivered at constant rate, and with variable rate streams the smoothing buffer should not always be one-half full. A constant average delay through the buffer requires a buffer fullness that varies with the data rate. The rate that data should be extracted from the buffer and input to the decoder can be approximated using the rate information present in the data stream. In Transport Streams the intended rate is determined by the values of the PCR fields and the number of Transport Stream bytes between them. In Program Streams the

intended rate is explicitly specified as the Program_mux_rate, although as specified in this Recommendation | International Standard the rate may drop to zero at SCR locations, i.e. if the SCR arrives before the time expected when the data is delivered at the indicated rate.

In the case of variable rate streams, the correct fullness of the smoothing buffer varies with time, and may not be determined exactly from the rate information. In an alternative approach, the SCRs or PCRs may be used to measure the time when data enter the buffer and to control the time when data leave the buffer. A control loop can be designed to provide constant average delay through the buffer. It may be observed that such a design is similar to the control loop illustrated in Figure D.2. The performance obtainable from inserting such a smoothing mechanism before a decoder can also be achieved by cascading multiple clock recovery PLLs. The rejection of jitter from the received timing will benefit from the combined low pass filter effect of the cascaded PLLs.

Annex E

Data Transmission Applications

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

E.0 General considerations

- ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport multiplex will be used to transmit data as well as video and audio.
- Data elementary streams are not continuous as may appear video and audio streams in broadcast applications.
- While it is already possible to identify the beginning of a PES packet, it is not always possible to identify the end of a PES packet by the beginning of the next PES packet, because it is possible for one or more Transport packet carrying PES packets to be lost.

E.1 Suggestion

A suitable solution is to transmit the following PES packet just after an associated PES packet. A PES packet without payload may be sent when there are no further PES packets to send.

Table E.1 is an example of such a PES packet.

Table E.1 – PES packet header example

PES packet header fields	Values
packet_start_code_prefix	0x000001
stream_id	assigned
PES_packet_length	0x0003
'10'	'10'
PES_scrambling_control	'00'
PES_priority	'0'
data_alignment_indicator	'0'
copyright	'0'
original_or_copy	'0'
PTS_DTS_flags	'00'
ESCR_flag	'0'
ES_rate_flag	'0'
DSM_trick_mode_flag	'0'
additonal_copy_info_flag	'0'
PES_CRC_flag	'0'
PES_extension_flag	'0'
PES_header_data_length	0x00

Annex F

Graphics of Syntax for this Recommendation | International Standard

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

F.0 Introduction

This annex is an informative annex presenting graphically the Transport Stream and Program Stream syntax. This annex in no way replaces any normative clause(s).

In order to produce clear drawings, not all fields have been fully described or represented. Reserved fields may be omitted or indicated by areas with no detail. Fields length are indicated in bits.

F.0.1 Transport Stream syntax

See Figure F.1.

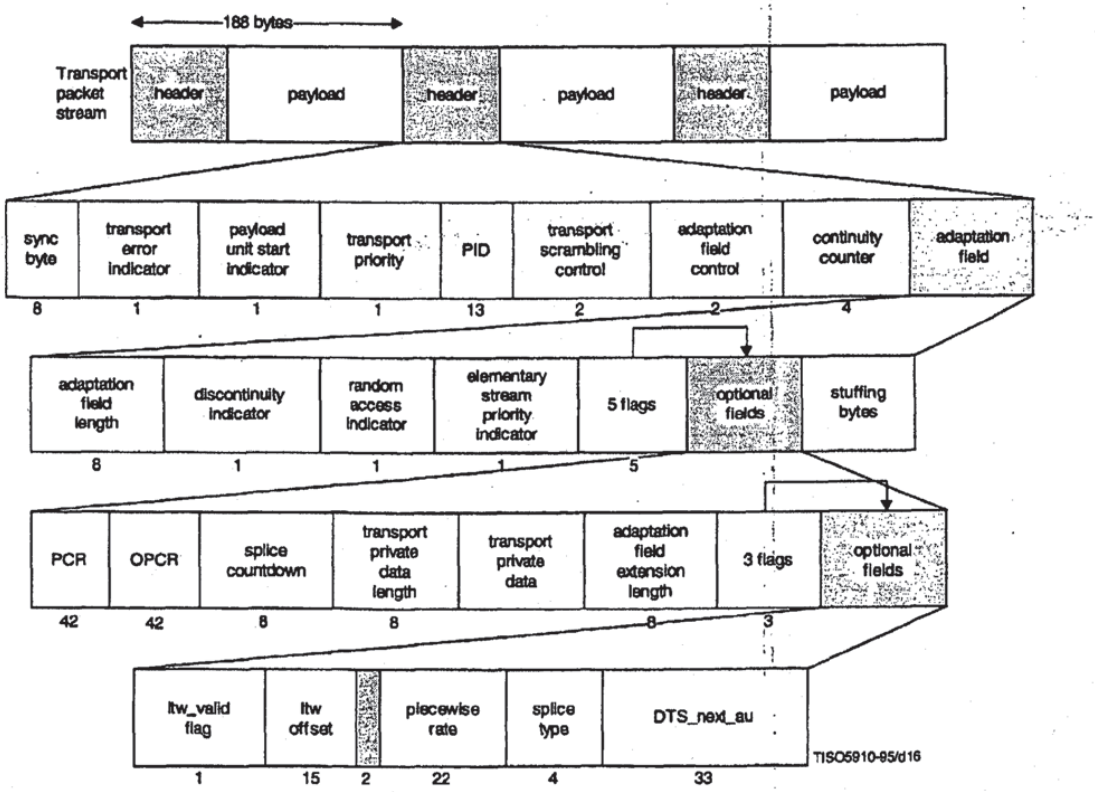

Figure F.1 – Transport Stream syntax diagram

F.0.2 PES packet

See Figure F.2.

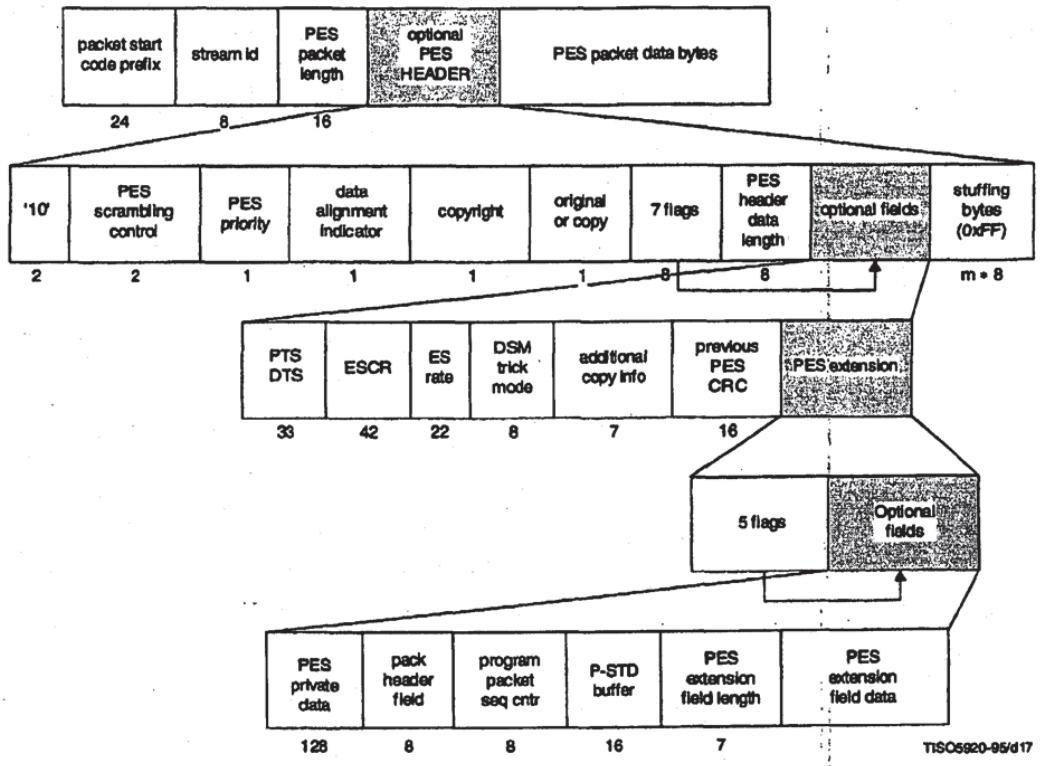

Figure F.2 – PES packet syntax diagram

F.0.3 Program Association Section

See Figure F.3.

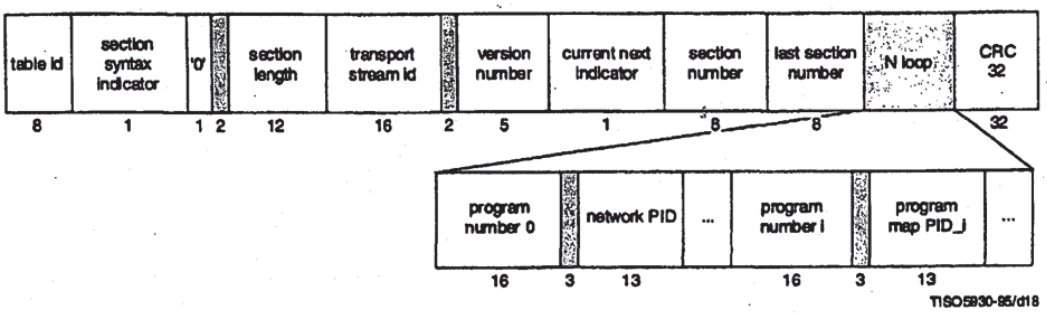

Figure F.3 – Program association section diagram

F.0.4 CA section

See Figure F.4.

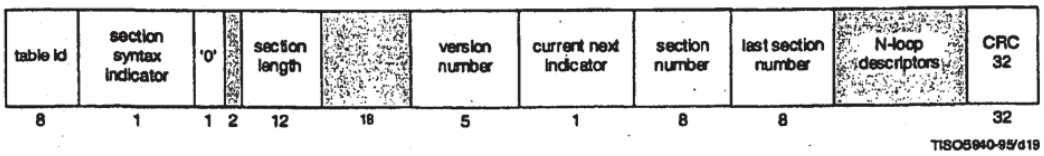

Figure F.4 – Conditional access section diagram

F.0.5 TS program map section

See Figure F.5.

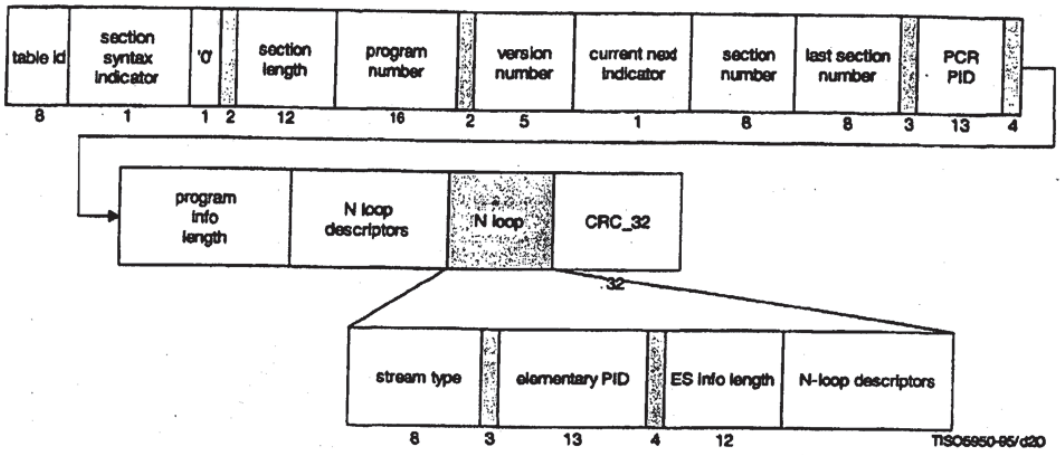

Figure F.5 – TS program map section diagram

F.0.6 Private section

See Figure F.6.

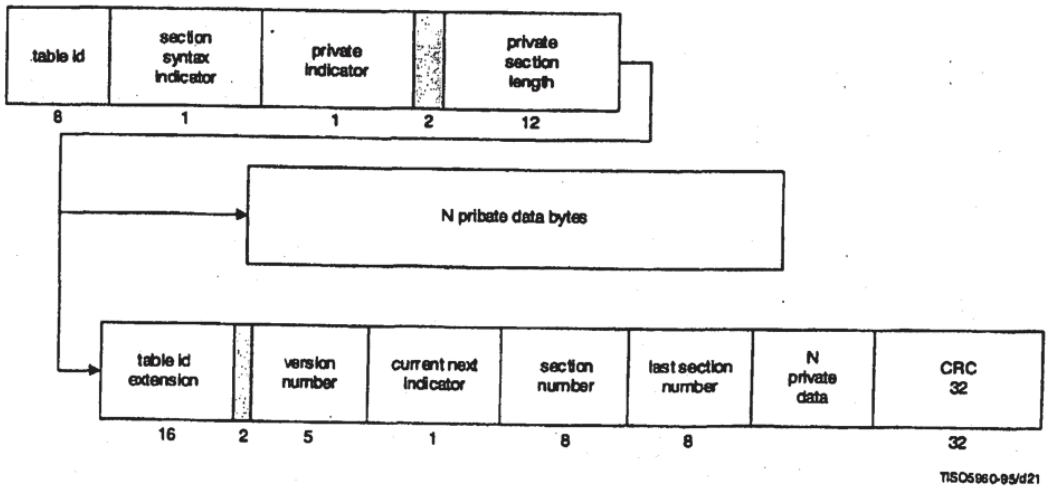

Figure F.6 – Private section diagram

F.0.7 Program Stream

See Figure F.7.

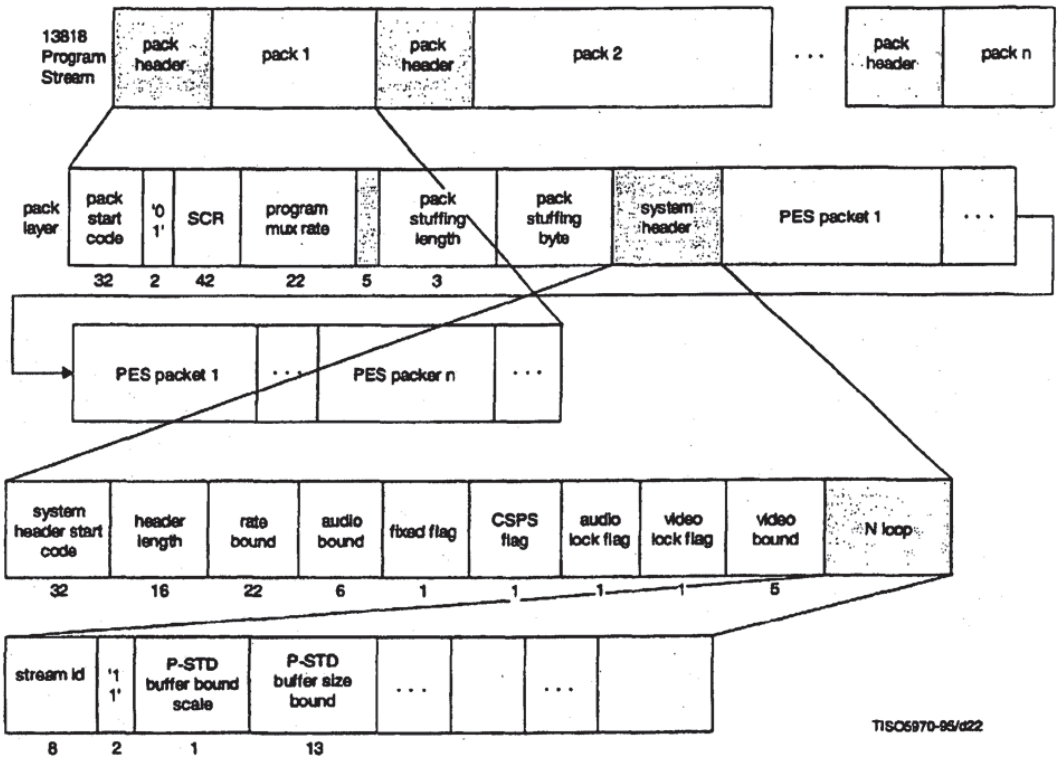

Figure F.7 – Program Stream diagram

F.0.8 Program Stream map

See Figure F.8.

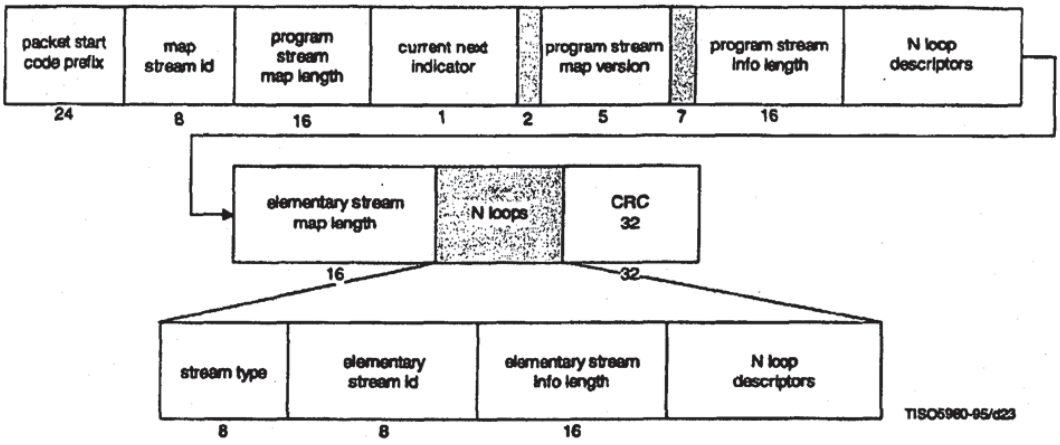

Figure F.8 – Program Stream map diagram

Annex G

General Information

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

G.0 General Information

G.0.1 Sync Byte Emulation

In the choice of PID values it is recommended that the periodic emulation of sync bytes be avoided. Such emulation may potentially occur within the PID field or as a combination of the PID field and adjacent flag settings. It is recommended that emulation of the sync byte be permitted to occur in the same position of the packet header for a maximum of 4-consecutive transport packets.

G.0.2 Skipped picture status and decoding process

Assume that the sequence being displayed contains only I- and P-frames. Denote the next picture to be decoded by picture_next, and the picture currently being displayed by picture_current. Because of the fact that the video encoder may skip pictures, it is possible that not all of the bits of picture_next are present in the STD buffers EB_n or B_n when the time arrives to remove those bits for instantaneous decoding and display. When this case arises, no bits are removed from the buffer and picture_current is displayed again. When the next picture display time arrives, if the remainder of the bits corresponding to picture_next are now in buffer EB_n or B_n, all the bits of picture_next are removed and picture_next is displayed. If all the bits of picture_next are not in the buffer EB_n or B_n, the above process of re-displaying picture_current is repeated. This process is repeated until picture_next can be displayed. Note that if a PTS preceded picture_next in the bitstream, it will be incorrect by some multiple of the picture display interval, which itself may depend on some parameters, and must be ignored.

Whenever the skipped picture situation described above occurs, the encoder is required to insert a PTS before the picture to be decoded after picture_next. This allows the decoder to immediately verify that it has correctly displayed the received picture sequence.

G.0.3 Selection of PID Values

Applications are encouraged to use low numbered PID values (avoiding reserved values as specified in Table 2-4) and group values together as much as possible.

G.0.4 PES start_code emulation

Three consecutive bytes having the value of a packet_start_code_prefix (0x000001), which when concatenated with a fourth byte, may emulate the four bytes of a PES_packet_header at a unintended place in the stream.

Such, so called, start code emulation is not possible in video elementary streams. It is possible in audio and data elementary streams. It is also possible at the boundary of a PES_packet_header and a PES_packet payload, even if the PES_packet payload is video.

Annex H

Private Data

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

H.0 Private Data

Private data is any user data which is not coded according to a standard specified by ITU-T | ISO/IEC and referred to in this Specification. The contents of this data is not and shall not be specified within this Recommendation | International Standard in the future. The STD defined in this Specification does not cover private data other than the demultiplex process. A private party may define each STD for private streams.

Private data may be carried in the following locations within the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 syntax.

1) *Transport Stream packet Table 2-2*

   The data bytes of the transport_packet() syntax may contain private data. Private data carried in this format is referred to as user private within the stream_type Table 2-29. It is permitted for Transport Stream packets containing private data to also include adaptation_field()s.

2) *Transport Stream Adaptation Field Table 2-6*

   The presence of any optional private_data_bytes in the adaptation_field() is signalled by the transport_private_data_flag. The number of the private_data_bytes is inherently restricted by the semantic of the adaptation_field_length field, where the value of the adaptation_field_length shall not exceed 183 bytes.

3) *PES packet Table 2-17*

   There are two possibilities for carrying private data within PES packets. The first possibility is within the PES_packet_header, within the optional 16 bytes of PES_private_data. The presence of this field is signalled by the PES_private_data_flag. The presence of the PES_private_data_flag is signalled by the PES_extension_flag. If present, these bytes, when considered with the adjacent fields, shall not emulate the packet_start_code_prefix.

   The second possibility is within the PES_packet_data_byte field. This may be referred to as private data within PES packets under the stream_type Table 2-29. This category of private data can be split in two: private_stream_1 refers to private data within PES packets which follow the PES_packet() syntax such that all fields up to and including, but not limited to, PES_header_data_length are present. private_stream_2 refers to private data within PES packets where only the first three fields shall be present followed by the PES_packet_data_bytes containing private data.

   Note that PES packets exist within both Program Streams and Transport Streams therefore private_stream_1 and private_stream_2 exist within both Program Streams and Transport Streams.

4) *Descriptors*

   Descriptors exist within Program Streams and Transport Streams. A range of private descriptors may be defined by the user. These descriptors shall commence with descriptor_tag and descriptor_length fields. For private descriptors, the value of descriptor_tag may take the values 64-55 as identified in Table 2-39. These descriptors may be placed within a program_stream_map() Table 2-29, a CA_section() Table 2-27, a TS_program_map_section(), Table 2-28 and in any private section(), Table 2-30.

   Specifically private_data_bytes also appear in the CA_descriptor().

5) *Private Section*

   The private_section Table 2-30 provides a further means to carry private data also in two forms. This type of elementary stream may be identified under stream_type Table 2-29 as private_data in PSI sections. One type of private_section() includes only the first five defined fields, and is followed by private data. For this structure the section_syntax_indicator shall be set to a value of '0'. For the other type, the section_syntax_indicator shall be set to a value of '1' and the full syntax up to and including last_section_number shall be present, followed by private_data_bytes and ending with the CRC_32.

Annex I

Systems conformance and real-time interface

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

I.0 Systems conformance and real-time interface

Conformance for ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Streams and Transport Streams is specified in terms of the normative specifications in this Recommendation | International Standard. These specifications include, among other requirements, a System Target Decoder (T-STD and P-STD) which specifies the behaviour of an idealized decoder when the stream is the input to such a decoder. This model, and the associated verification, do not include information concerning the real-time delivery performance of the stream, except for the accuracy of the system clock frequency which is represented by the Transport Stream and the Program Stream. All Transport Streams and Program Streams must comply with this Recommendation | International Standard.

In addition, there is a real-time interface specification for input of Transport Streams and Program Streams to a decoder. This Recommendation | International Standard allows standardization of the interface between MPEG decoders and adapters to networks, channels, or storage media. The timing effects of channels, and the inability of practical adapters to eliminate completely these effects, causes deviations from the idealized byte delivery schedule to occur. While it is not necessary for all MPEG decoders to implement this interface, implementations which include the interface shall adhere to the specifications. This Recommendation | International Standard covers the real-time delivery behaviour of Transport Streams and Program streams to decoders, such that the coded data buffers in decoders are guaranteed not to overflow nor underflow, and decoders are guaranteed to be able to perform clock recovery with the performance required by their applications.

The MPEG real-time interface specifies the maximum allowable amount of deviation from the idealized byte delivery schedule which is indicated by the Program Clock Reference (PCR) and System Clock Reference (SCR) fields encoded in the stream.

Annex J

Interfacing Jitter-Inducing Networks to MPEG-2 Decoders

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

J.0 Introduction

In this annex the expression system stream will be used to refer to both ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Transport Streams and ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Streams. When the term STD is used, it is understood to mean the P-STD (Program System Target Decoder) for Program Streams and the T-STD (Transport System Target Decoder) for Transport Streams.

The intended byte delivery schedule of a system stream can be deduced by analyzing the stream. A system stream is compliant if it can be decoded by the STD, which is a mathematical model of an idealized decoder. If a compliant system stream is transmitted over a jitter-inducing network, the true byte delivery schedule may differ significantly from the intended byte delivery schedule. In such cases it may not be possible to decode the system stream on such an idealized decoder, because jitter may cause buffer overflows or underflows and may make it difficult to recover the time base. An important example of such a jitter-inducing network is ATM.

The purpose of this annex is to provide guidance and insight to entities concerned with sending system streams over jitter-inducing networks. Network specific compliance models for transporting system streams are likely to be developed for several types of networks, including ATM. The STD plus a real-time interface definition can play an integral role in defining such models. A framework for developing network compliance models is presented in J.2.

Three examples of network encoding to enable the building of jitter-smoothing network adapters are discussed in J.3. In the first example a constant bitrate system stream is assumed and a FIFO is used for jitter smoothing. In the second example the network adaptation layer includes timestamps to facilitate jitter smoothing. In the final example a common network clock is assumed to be available end-to-end, and is exploited to achieve jitter smoothing.

J.4 presents two examples of decoder implementations in which network-induced jitter can be accommodated. In the first example, a jitter-smoothing network adapter is inserted between a network's output and an MPEG-2 decoder. The MPEG-2 decoder is assumed to conform to a real-time MPEG-2 interface specification. This interface requires an MPEG-2 decoder with more jitter tolerance than the idealized decoder of the STD. The network adapter processes the incoming jittered bitstream and outputs a system stream whose true byte delivery schedule conforms to the real-time specification. Example one is discussed in J.4.1. For some applications the network adapter approach will be too costly because it requires two stages of processing. Therefore, in the second example the dejittering and MPEG-2 decoding functions are integrated. The intermediate processing of the jitter-removal device is bypassed, so only a single stage of clock recovery is required. Decoders that perform integrated dejittering and decoding are referred to in this annex as integrated network-specific decoders, or simply integrated decoders. Integrated decoders are discussed in J.4.2.

In order to build either network adapters or integrated decoders a maximum value for the peak-to-peak network jitter must be assumed. In order to promote interoperability, a peak-to-peak jitter bound must be specified for each relevant network type.

J.1 Network compliance models

One way to model the transmission of a system stream across a jitter-inducing network is shown in Figure J.1.

The system stream is input to a network-specific encoding device that converts the system stream into a network specific format. Information to assist in jitter removal at the network output may be part of this format. The network decoder comprises a network-specific decoder and an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoder. The ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoder is assumed to conform to a real-time interface specification, and could have the same architecture as the STD with appropriate buffers made larger to provide more jitter tolerance. The network-specific decoder removes the non- ITU-T Rec. H.222.0 | ISO/IEC 13818-1 data added by the network-specific encoder and dejitters the network's output. The output of the network-specific decoder is a system stream that conforms to the real-time specification.

A network target decoder (NTD) can be defined based on the above architecture. A compliant network bitstream would be one that was able to be decoded by the NTD. A network decoder would be compliant provided it could decode any network bitstream able to be decoded by the NTD. A real network decoder might or might not have the architecture of the NTD.

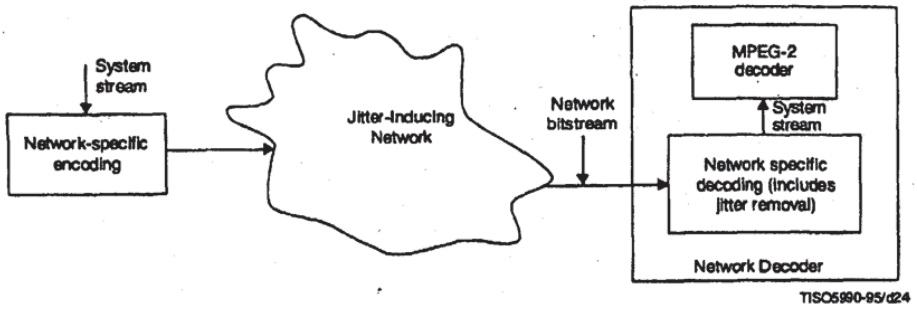

Figure J.1 – Sending system streams over a jitter-inducing network

J.2 Network specification for jitter smoothing

In the case of constant bit rate system streams, jitter smoothing can be accomplished with a FIFO. Additional data that provides specific support for dejittering is not required in the network adaptation layer. After the bytes added by the network encoding are removed, the system stream data is placed in a FIFO. A PLL keeps the buffer approximately half full by adjusting the output rate in response to changes in buffer fullness. In this example the amount of jitter-smoothing achieved will depend on the size of the FIFO and the characteristics of the PLL.

Figure J.2 illustrates a second way to accomplish jitter smoothing. In this example timestamp support from a network adaptation layer is assumed. Using this technique, both constant bit rate and variable bit rate system streams can be dejittered.

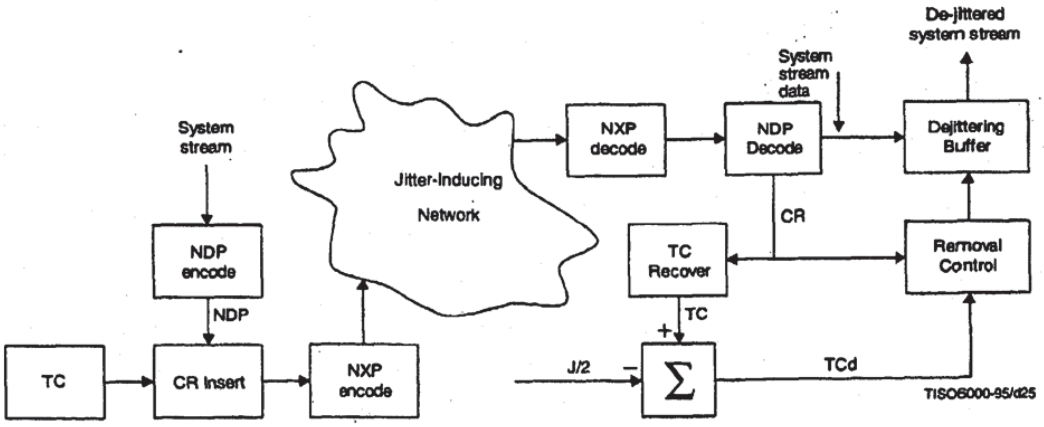

Figure J.2 – Jitter-smoothing using network-layer timestamps

Assume the network adapter is designed to compensate for a peak-to-peak jitter of J seconds. The intended byte delivery schedule is reconstructed using Clock Reference (CRs) samples taken from a Time Clock (TC). The CRs and the TC are analogous to PCRs and the STC. The Network Data Packet (NDP) encode converts each system stream packet into a Network Data Packet (NDP). The network data packets contain a field for carrying CR values, and the current value of the TC is inserted into this field as the NDP leaves the NDP encoder. The Network Transport Packetization (NXP) function encapsulates the NDPs into network transport packets. After transmission across the network, the CRs are extracted by the NDP decoder as the NDPs enter the NDP decoder. The CRs are used to reconstruct the TC, for example by using a PLL. The first MPEG-2 packet is removed from the dejittering buffer when the delayed TC (TCd) is equal to the first MPEG-2 packet's CR. Subsequent MPEG-2 packets are removed when their CR values equal the value of the TCd.

Ignoring implementation details such as the speed of the TC clock recovery loop and the spectral purity of the TC, the size of the dejittering buffer depends only on the maximum peak-to-peak jitter to be smoothed and the largest transport rate that occurs in the system stream. The dejittering buffer size,B_{dj}, is given by

$$B_{dj} = JR_{max}$$

where R_{max} is the maximum data rate of the system stream in bits per second. When packets traversing the network experience the nominal delay, the buffer is half full. When they experience a delay of J/2 seconds, the buffer is empty, and when they experience a delay (advance) of –J/2 seconds the buffer is full.

As a final example, in some cases a common network clock will be available end-to-end, and it may be feasible to lock the system clock frequency to the common clock. The network adapter can smooth jitter with a FIFO. The adapter uses PCRs or SCRs to reconstruct the original byte delivery schedule.

J.3 Example decoder implementations

J.3.1 Network adapter followed by an MPEG-2 decoder

In this implementation a network adapter conforming to the network compliance specification is connected to an MPEG-2 decoder conforming to the real-time interface specification.

J.3.2 Integrated decoder

The example presented in J.4.1 requires two stages of processing. The first stage is necessary to dejitter the network's output. The second stage, recovering the STC by processing PCRs or SCRs, is required for STD decoding. The example presented in this subclause is a decoder that integrates the dejittering and decoding functions in a single system. The STC clock is recovered directly using the jittered PCR or SCR values. For presenting this example, an MPEG-2 transport stream will be assumed.

Figure J.3 illustrates the operation of the integrated decoder. The stream of network packets input to the decoder is assumed to be the same as the one shown in Figure J.2.

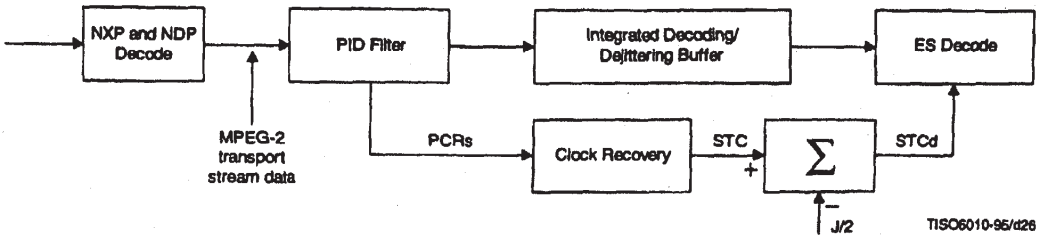

Figure J.3 – Integrated dejittering and MPEG-2 decoding

The incoming network packets are reassembled into MPEG-2 transport stream data by the NXP and NDP decode functions. The jittered ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Transport Stream packets are then filtered to extract packets with the desired PID. For the case illustrated, the PID being decoded is also carrying the PCRs. The PCR values are sent to a PLL to recover the STC. Entire packets for the selected PID are placed in the integrated buffer. A positive value of J/2 s is subtracted from the STC to obtain the delayed STC, STCd. Again, J is the peak-to-peak jitter the network-savvy decoder can accommodate. The delay is introduced to guarantee that all the data required for an access unit has arrived in the buffer when the PTS/DTS of the access unit equals the current value of the STCd.

Ignoring implementation details such as the speed of the STC clock recovery loop and the spectral purity of the STC:

$$\begin{aligned} B_{size} &= B_{dec} + B_{mux} + B_{OH} + 512 + B_J \\ &= B_n + 512 + B_J \end{aligned}$$

where $B_j = R_{max} J$ and R_{max} the maximum rate at which data is input to the PID filter. Depending on the implementation, the integrated memory could be broken into two components as in the transport STD.

Annex K

Splicing Transport Streams

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

K.0 Introduction

For the purposes of this annex, the term 'splicing' refers to the concatenation performed on the Transport level of two different elementary streams, the resulting Transport Stream conforming totally to this Recommendation | International Standard. The two elementary streams may have been generated at different locations and/or at different times, and were not necessarily intended to be spliced together when they were generated. In the following we will call the 'old' stream a continuous elementary stream (video or audio), which has been superseded by another stream (the 'new' one) from a certain point on. This point is called the splice. It is the boundary between data belonging to the 'old' stream and data belonging to the 'new' one.

A splice can be seamless or non-seamless:

- A seamless splice is a splice inducing no decoding discontinuity (refer to 2.7.6). This means that the decoding time of the first access unit of the 'new' stream is consistent with respect to the decoding time of the access unit of the 'old' stream preceding the splice, i.e. it is equal to the one that the next access unit would have had if the 'old' stream had continued. In the following, we will call this decoding time the 'seamless decoding time'.
- A non-seamless splice is a splice which results in a decoding discontinuity, i.e. the decoding time of the first access unit of the 'new' stream is greater than the seamless decoding time.

  NOTE – A decoding time lower than the seamless decoding time is forbidden).

Splicing is allowed to be performed at any transport stream packet boundary, since the resulting stream is legal. But in a general case, if nothing is known about the location of PES packet starts and access unit starts, this constraint imposes that not only the Transport layer is parsed, but also the PES layer and the Elementary Stream layer, and may in some cases, make some processing on the payload of Transport Stream packets necessary. If such complex operations are wished to be avoided, splicing should be performed at locations where the Transport Stream has favourable properties, these properties being indicated by the presence of a splicing point.

The presence of a splicing point is indicated by the splice_flag and splice_countdown fields (refer to 2.4.3.4 for the semantics of these fields). In the following, the Transport Stream packet in which the splice_countdown field value reaches zero will be called 'splicing packet'. The splicing point is located immediately after the last byte of the splicing packet.

K.1 The different types of splicing point

A splicing point can be either an ordinary splicing point or a seamless splicing point.

K.1.1 Ordinary splicing points

If the seamless_splice_flag field is not present, or if its value is zero, the splicing point is ordinary. The presence of an ordinary splicing point only signals alignment properties of the Elementary Stream: the splicing packet ends on the last byte of an Access Unit, and the payload of the next Transport Stream packet of the same PID will start with the header of a PES packet, the payload of which will start with an Elementary Stream Access Point (or with a sequence_end_code() immediately followed by an Elementary Stream Access Point, in the case of video). These properties allow 'Cut and Paste' operations to be performed easily on the Transport level, while respecting syntactical constraints and ensuring bit stream consistency. However, it does not provide any information concerning timing or buffer properties. As a consequence, with such splicing points, seamless splicing can only be done with the help of private arrangements, or by analyzing the payload of the Transport Stream Packets and tracking buffer status and timestamp values.

K.1.2 Seamless splicing points

If the seamless_splice_flag field is present and its value is one, information is given by the splicing point, indicating some properties of the 'old' stream. This information is not aimed at decoders. Its primary goal is to facilitate seamless splicing. Such a splicing point is called a seamless splicing point. The available information is:

- The seamless decoding time, which is encoded as a DTS value in the DTS_next_AU field. This DTS value is expressed in the time base which is valid in the splicing packet.

117

- In the case of a video elementary stream, the constraints that have been applied to the 'old' stream when it was generated, aiming at facilitating seamless splicing. These conditions are given by the value of the splice_type field, in the table corresponding to the profile and level of the video stream.

Note that a seamless splicing point can be used as an ordinary splicing point, by discarding this additional information. This information may also be used if judged helpful to perform non-seamless splicing, or for purposes other than splicing.

K.2 Decoder behaviour on splices

K.2.1 On non-seamless splices

As described above, a non-seamless splice is a splice which results in a decoding discontinuity.

It shall be noted that with such a splice, the constraints related to the decoding discontinuity (see 2.7.6) shall be fulfilled. In particular:

- a PTS shall be encoded for the first access unit of the 'new' stream (except during trick mode operation or when low_delay = '1');
- the decoding time derived from this PTS (or from the associated DTS) shall not be earlier than the seamless decoding time;
- in the case of a video elementary stream, if the splicing packet does not end on a sequence_end_code(), the 'new' stream shall begin with a sequence_end_code() immediately followed by a sequence_header().

In theory, since they introduce decoding discontinuities, such splices result in a non-continuous presentation of presentation units (i.e. a variable length dead time between the display of two consecutive pictures, or between two consecutive audio frames). In practice, the result will depend on how the decoder is implemented, especially in video. With some video decoders, the freezing of one or more pictures may be the preferred solution. See Part 4 of ISO/IEC 13818.

K.2.2 On seamless splices

The aim of having no decoding discontinuity is to allow having no presentation discontinuity. In the case of audio, this can always be ensured. But it has to be noted that in the case of video, presentation continuity is in theory not possible in cases 1) and 2) below:

1) The 'old' stream ends on the end of a low-delay sequence, and the 'new' stream begins with the start of a non-low-delay sequence.
2) The 'new' stream ends on the end of a non-low-delay sequence, and the 'new' stream begins with the start of a low-delay sequence.

The effects induced by such situations is implementation dependent. For instance, in case 1, a picture may have to be presented during two frame periods, and in case 2, a picture may have to be skipped. However, it is technically possible that some implementations support such situations without any undesirable effect.

In addition, referring to 6.1.1.6 of ITU-T Rec. H.262 | ISO/IEC 13818-2, a sequence_end_code() shall be present before the first sequence_header() of the 'new' stream, if at least one sequence parameter (i.e. a parameter defined in the sequence header or in a sequence header extension) has a different value in both streams, with the only exception of those defining the quantization matrix. As an example, if the bit rate field has not the same value in the 'new' stream as in the 'old' one, a sequence_end_code() shall be present. Thus, if the splicing packet does not end on a sequence_end_code, the 'new' stream shall begin with a sequence_end_code followed by a sequence_header.

According to the previous paragraph, a sequence_end_code will be mandatory in most splices, even seamless ones. It has to be noted that ITU-T Rec. H.262 | ISO/IEC 13818-2 specifies the decoding process of video sequences (i.e. data comprised between a sequence_header() and a sequence_end_code()), and nothing is specified about how to handle a sequence change. Thus, for the behaviour of the decoders when such splices are encountered, refer to Part 4 of ISO/IEC 13818.

K.2.3 Buffer Overflow

Even if both elementary streams obey the T-STD model before being spliced, it is not necessarily ensured that the STD buffers do not overflow with the spliced stream in the time interval during which bits of both streams are in these buffers.

In the case of constant bit rate video, if no particular conditions have been applied to the 'old' stream, and if no particular precautions have been taken during splicing, this overflow is possible in the case where the video bit rate of the 'new' stream is greater than the video bit rate of the 'old' one. Indeed, it is certainly true that the buffers MB_n and

EB_n of the T-STD do not overflow if bits are delivered to the T-STD at the 'old' rate. But if the delivery rate is switched to a higher value at the input of TB_n before 'old' bits are completely removed from the T-STD, the fullness of the STD buffers will become higher than if the 'old' stream had continued without splicing, and may cause overflow of EB_n and/or MB_n. In the case of variable bit rate video, the same problem can occur if the delivery rate of the 'new' stream is higher than the one for which provision was made during the creation of the 'old' stream. Such a situation is forbidden.

However, it is possible for the encoder generating the 'old' stream to add conditions in the VBV buffer management in the neighborhood of splicing points, so that provision is made for any 'new' video bit rate lower than a chosen value. For instance, in the case of a seamless splicing point, such additional conditions can be indicated by a 'splice_type' value to which entries correspond in Table 2-7 through Table 2-16 for 'splice_decoding_delay' and 'max_splice_rate'. In that case, if the video bit rate of the 'new' stream is lower than 'max_splice_rate', it is ensured that the spliced stream will not lead to overflow during the time interval during which bits of both streams are in the T-STD buffer.

In the case where no such constraints have been applied, this problem can be avoided by introducing a dead time in the delivery of bits between the 'old' stream and the 'new' one, in order to let the T-STD buffers get sufficiently empty before the bits of the 'new' stream are delivered. If we call t_{in} the time at which the last byte of the last access unit of the 'old' stream enters the STD, and t_{out} the time at which it exits the STD, it is sufficient to ensure that no more bits enter the T-TD the time interval $[t_{in}, t_{out}]$ with the spliced stream than if the 'old' stream had continued without splicing. As an example, in the case where the 'old' stream has a constant bit rate R_{old}, and the 'new' one a constant bit rate R_{new}, it is sufficient to introduce a dead time T_d satisfying the following relations to avoid this risk of overflow:

$$T_d \geq 0 \text{ and } T_d \geq (t_{out} - t_{in}) \times (1 - R_{old}/R_{new})$$

ICS 35.040

Descriptors: data processing, moving pictures, image processing, video recording, sound recording, video data, audio data, data converting, coding (data conversion), system architecture.

Price based on 119 pages

This Page Blank (uspto)