INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC JTC1/SC29/WG11 N0801

13 November 1994

Systems

INFORMATION TECHNOLOGY -

GENERIC CODING OF MOVING PICTURES AND ASSOCIATED AUDIO: SYSTEMS
Recommendation H.222.0

ISO/IEC 13818-1

International Standard

Draft of: **1540 Sun 13 Nov 1994**

Contents

Page

Annexes

List of Figures

List of Syntax Tables

List of Equations

Foreword

FOREWORD PROVIDED BY ISO

Introduction

The systems part of this Recommendation | International Standard addresses the combining of one or more elementary streams of video and audio, as well as other data, into single or multiple streams which are suitable for storage or transmission. Systems coding follows the syntactical and semantic rules imposed by this specification and provides information to enable synchronized decoding of decoder buffers over a wide range of retrieval or receipt conditions.

System coding shall be specified in two forms: the **Transport Stream** and the **Program Stream**. Each is optimized for a different set of applications. Both the Transport Stream and Program Stream defined in this Recommendation | International Standard provide coding syntax which is necessary and sufficient to synchronize the decoding and presentation of the video and audio information, while ensuring that data buffers in the decoders do not overflow or underflow. Information is coded in the syntax using time stamps concerning the decoding and presentation of coded audio and visual data and time stamps concerning the delivery of the data stream itself. Both stream definitions are packet-oriented multiplexes.

The basic multiplexing approach for single video and audio elementary streams is illustrated in figure 0-1 on page xi . The video and audio data is encoded as described in ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3. The resulting compressed elementary streams are packetized to produce **PES packets**. Information needed to use PES packets independently of either Transport Streams or Program Streams may be added when PES packets are formed. This information is not needed and need not be added when PES packets are further combined with system level information to form **Transport Streams** or **Program Streams**. This systems standard covers those processes to the right of the vertical dashed line.

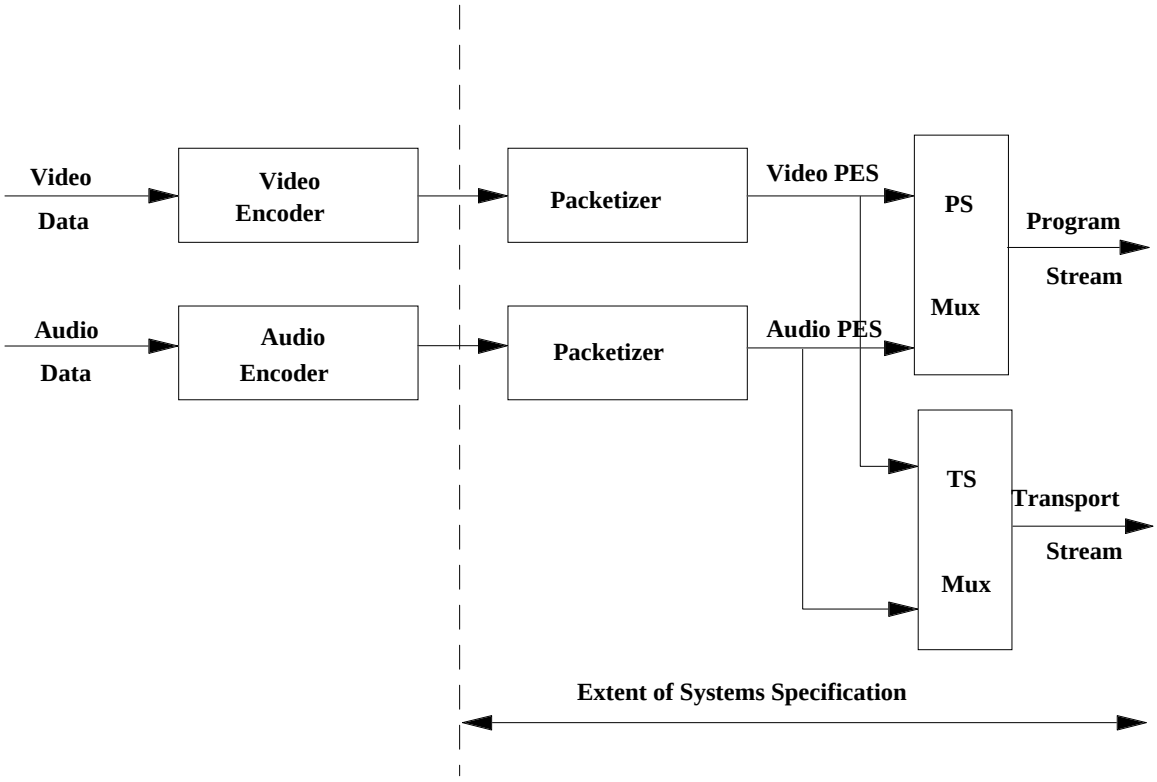

Figure 0-1 -- Simplified overview of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 scope

The **Program Stream** is analogous and similar to ISO/IEC 11172 Systems layer. It results from combining one or more streams of PES packets, which have a common time base, into a single stream.

For applications that require the elementary streams which comprise a single program to be in separate streams which are not multiplexed, the elementary streams can also be encoded as separate Program Streams, one per elementary stream, with a common time base. In this case the values encoded in the SCR fields of the various streams shall be consistent.

Like the single Program Stream, all elementary streams can be decoded with synchronization.

The Program Stream is designed for use in relatively error-free environments and is suitable for applications which may involve software processing of system information such as interactive multi-media applications. Program Stream packets may be of variable and relatively great length.

The **Transport Stream** combines one or more programs with one or more independent time bases into a single stream. PES packets made up of elementary streams that form a program share a common timebase. The Transport Stream is designed for use in environments where errors are likely, such as storage or transmission in lossy or noisy media. Transport Stream packets are 188 bytes in length.

Program and Transport Streams are designed for different applications and their definitions do not strictly follow a layered model. It is possible and reasonable to convert from one to the other; however, one is not a subset or superset of the other. In particular, extracting the contents of a program from a Transport Stream and creating a valid Program Stream is possible and is accomplished through the common interchange format of PES packets, but not all of the fields needed in a Program Stream are contained within the Transport Stream; some

must be derived. The Transport Stream may be used to span a range of layers in a layered model, and is designed for efficiency and ease of implementation in high bandwidth applications.

The scope of syntactical and semantic rules set forth in the systems specification differ: the syntactical rules apply to systems layer coding only, and do not extend to the compression layer coding of the video and audio specifications; by contrast, the semantic rules apply to the combined stream in its entirety.

The systems specification does not specify the architecture or implementation of encoders or decoders, nor those of multiplexors or demultiplexors. However, bit stream properties do impose functional and performance requirements on encoders, decoders, multiplexors and demultiplexors. For instance, encoders must meet minimum clock tolerance requirements. Notwithstanding this and other requirements, a considerable degree of freedom exists in the design and implementation of encoders, decoders, multiplexors, and demultiplexors.

0.1 Transport Stream

The Transport Stream is a stream definition which is tailored for communicating or storing one or more programs of coded data according to ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3 and other data in environments in which significant errors may occur. Such errors may be manifested as bit value errors or loss of packets.

Transport Streams may be either fixed or variable rate. In either case the constituent elementary streams may either be fixed or variable rate. The syntax and semantic constraints on the stream are identical in each of these cases. The Transport Stream rate is defined by the values and locations of Program Clock Reference (PCR) fields, which in general are separate PCR fields for each program.

There are some difficulties with constructing and delivering a Transport Stream containing multiple programs with independent time bases such that the overall bit rate is variable. Refer to 2.4.2.2 on page 14.

The Transport Stream may be constructed by any method that results in a valid stream. It is possible to construct Transport Streams containing one or more programs from elementary coded data streams, from Program Streams, or from other Transport Streams which may themselves contain one or more programs.

The Transport Stream is designed in such a way that several operations on a Transport Stream are possible with minimum effort. Among these are:

1. Retrieve the coded data from one program within the Transport Stream, decode it and present the decoded results as shown in figure 0-2 on page xiii .

2. Extract the Transport Stream packets from one program within the Transport Stream and produce as output a different Transport Stream with only that one program as shown in figure 0-3 on page xiii .

3. Extract the Transport Stream packets of one or more programs from one or more Transport Streams and produce as output a different Transport Stream (not illustrated).

4. Extract the contents of one program from the Transport Stream and produce as output a Program Stream containing that one program as shown in figure 0-4 on page xiv .

5. Take a Program Stream, convert it into a Transport Stream to carry it over a lossy environment, and then recover a valid, and in certain cases, identical Program Stream.

Figure 0-2 on page xiii and figure 0-3 on page xiii illustrate prototypical demultiplexing and decoding systems which take as input a Transport Stream. Figure 0-2 on page xiii illustrates the first case, where a Transport Stream is directly demultiplexed and decoded. Transport Streams are constructed in two layers: a system layer

and a compression layer. The input stream to the Transport Stream decoder has a system layer wrapped about a compression layer. Input streams to the Video and Audio decoders have only the compression layer.

Operations performed by the prototypical decoder which accepts Transport Streams either apply to the entire Transport Stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The Transport Stream system layer is divided into two sub-layers, one for multiplex-wide operations (the Transport Stream packet layer), and one for stream-specific operations (the PES packet layer).

A prototypical decoder for Transport Streams, including audio and video, is also depicted in figure 0-2 to illustrate the function of a decoder. The architecture is not unique -- some system decoder functions, such as decoder timing control, might equally well be distributed among elementary stream decoders and the channel specific decoder -- but this figure is useful for discussion. Likewise, indication of errors detected by the channel specific decoder to the individual audio and video decoders may be performed in various ways and such communication paths are not shown in the diagram. The prototypical decoder design does not imply any normative requirement for the design of a Transport Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

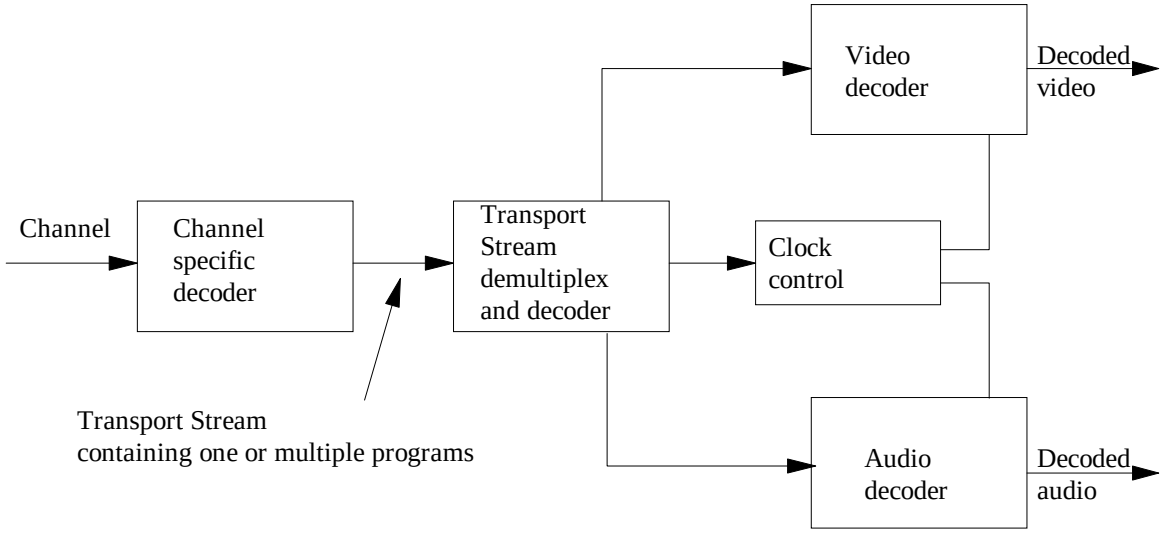

Figure 0-2 -- Prototypical transport demultiplexing and decoding example

Figure 0-3 illustrates the second case, where a Transport Stream containing multiple programs is converted into a Transport Stream containing a single program. In this case the remultiplexing operation may necessitate the correction of Program Clock Reference (PCR) values to account for changes in the PCR locations in the bit stream.

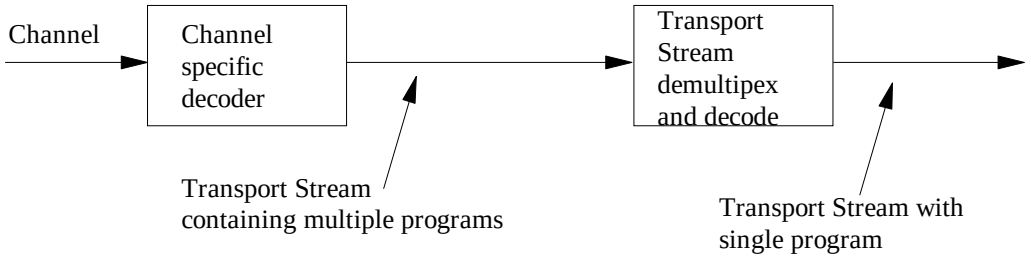

Figure 0-3 -- Prototypical transport multiplexing example

Figure 0-4 on page xiv below illustrates a case in which an multi-program Transport Stream is first demultiplexed and then converted into a Program Stream.

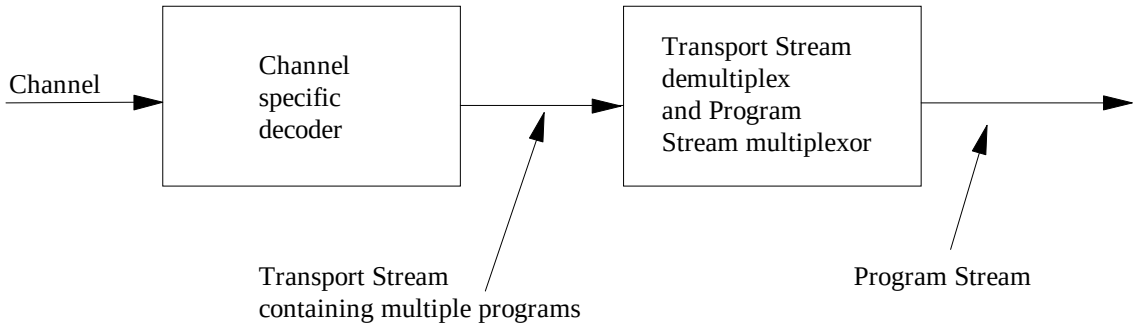

Figure 0-4 -- Prototypical Transport Stream to Program Stream conversion

Figure 0-3 on page xiii and figure 0-4 indicate that it is possible and reasonable to convert between different types and configurations of Transport Streams. There are specific fields defined in the **Transport Stream** and **Program Stream** syntaxes which facilitate the conversions illustrated. There is no requirement that specific implementations of demultiplexors or decoders include all of these functions.

0.2 Program Stream

The Program Stream is a stream definition which is tailored for communicating or storing one program of coded data and other data in environments where errors are very unlikely, and where processing of system coding, e. g. by software, is a major consideration.

Program Streams may be either fixed or variable rate. In either case, the constituent elementary streams may be either fixed or variable rate. The syntax and semantics constraints on the stream are identical in each case. The Program Stream rate is defined by the values and locations of the System Clock Reference (SCR) and mux_rate fields.

A prototypical audio/video Program Stream decoder system is depicted in figure 0-5 on page xv below. The architecture is not unique -- system decoder functions including decoder timing control might equally well be distributed among elementary stream decoders and the channel specific decoder -- but this figure is useful for discussion. The prototypical decoder design does not imply any normative requirement for the design of an Program Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

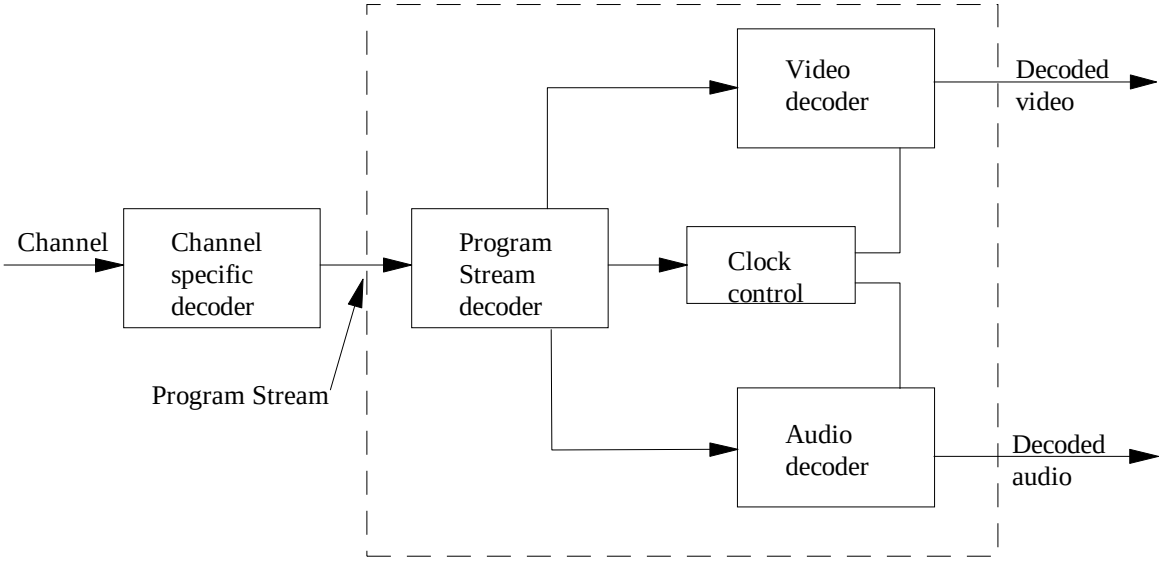

Figure 0-5 -- Prototypical decoder for Program Streams

The prototypical decoder for Program Streams shown in figure 0-5 is composed of System, Video, and Audio decoders conforming to parts 1, 2, and 3, respectively, of this Recommendation | International Standard. In this decoder the multiplexed coded representation of one or more audio and/or video streams is assumed to be stored or communicated on some channel in some channel-specific format. The channel-specific format is not governed by this Recommendation | International Standard, nor is the channel-specific decoding part of the prototypical decoder.

The prototypical decoder accepts as input a Program Stream and relies on a Program Stream Decoder to extract timing information from the stream. The Program Stream Decoder demultiplexes the stream, and the elementary streams so produced serve as inputs to Video and Audio decoders, whose outputs are decoded video and audio signals. Included in the design, but not shown in the figure, is the flow of timing information among the Program Stream decoder, the Video and Audio decoders, and the channel-specific decoder. The Video and Audio decoders are synchronized with each other and with the channel using this timing information.

Program Streams are constructed in two layers: a system layer and a compression layer. The input stream to the Program Stream Decoder has a system layer wrapped about a compression layer. Input streams to the Video and Audio decoders have only the compression layer.

Operations performed by the prototypical decoder either apply to the entire Program Stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The Program Stream system layer is divided into two sub-layers, one for multiplex-wide operations (the pack layer), and one for stream-specific operations (the PES packet layer).

0.3 Conversion between Transport Stream and Program Stream

It may be possible and reasonable to convert between **Transport Streams** and **Program Streams** by means of PES packets. This results from the specification of **Transport Stream** and **Program Stream** as embodied in 2.4.1 on page 10 and 2.5.1 on page 53 of the normative requirements of this Recommendation | International Standard. PES packets may, with some constraints, be mapped directly from the payload of one multiplexed bit stream into the payload of another multiplexed bit stream. It is possible to identify the correct order of PES packets in a program to assist with this if the program_packet_sequence_counter is present in all PES packets.

Certain other information necessary for conversion, e.g. the relationship between elementary streams, is available in tables and headers in both streams. Such data, if available, shall be correct in any stream before and after conversion.

0.4 Packetized Elementary Stream

Transport Streams and **Program Streams** are each logically constructed from PES packets, as indicated in the syntax definitions in 2.4.3.6 on page 33. PES packets shall be used to convert between Transport Streams and Program Streams; in some cases the PES packets need not be modified when performing such conversions. PES packets may be much larger than the size of a Transport Stream packet.

A continuous sequence of PES packets of one elementary stream with one stream ID may be used to construct a PES Stream. When PES packets are used to form a PES stream, they shall include Elementary Stream Clock Reference (ESCR) fields and Elementary Stream Rate (ES_Rate) fields, with constraints as defined in 2.4.3.8 on page 44. The PES stream data shall be contiguous bytes from the elementary stream in their original order. PES streams do not contain some necessary system information which is contained in Program Streams and Transport Streams. Examples include the information in the Pack Header, System Header, Program Stream Map, Program Stream Directory, Program Map Table, and elements of the Transport Stream packet syntax.

The PES Stream is a logical construct that may be useful within implementations of this standard; however it is not defined as a stream for interchange and interoperability. Applications requiring streams containing only one elementary stream can use Program Streams or Transport Streams which each contain only one elementary stream. These streams contain all of the necessary system information. Multiple Program Streams or Transport Streams, each containing a single elementary stream, can be constructed with a common time base and therefore carry a complete program, i.e. with audio and video.

0.5 Timing model

Systems, Video and Audio all have a timing model in which the end-to-end delay from the signal input to an encoder to the signal output from a decoder is a constant. This delay is the sum of encoding, encoder buffering, multiplexing, communication or storage, demultiplexing, decoder buffering, decoding, and presentation delays. As part of this timing model all video pictures and audio samples are presented exactly once, unless specifically coded to the contrary, and the inter-picture interval and audio sample rate are the same at the decoder as at the encoder. The system stream coding contains timing information which can be used to implement systems which

embody constant end-to-end delay. It is possible to implement decoders which do not follow this model exactly; however, in such cases it is the decoder's responsibility to perform in an acceptable manner. The timing is embodied in the normative specifications of this standard, which must be adhered to by all valid bit streams, regardless of the means of creating them.

All timing is defined in terms of a common system clock, referred to as a System Time Clock. In the Program Stream this clock may have an exactly specified ratio to the video or audio sample clocks, or it may have an operating frequency which differs slightly from the exact ratio while still providing precise end-to-end timing and clock recovery.

In the Transport Stream the system clock frequency is constrained to have the exactly specified ratio to the audio and video sample clocks at all times; the effect of this constraint is to simplify sample rate recovery in decoders.

0.6 Conditional access

Encryption and scrambling for conditional access to programs encoded in the Program and Transport Streams is supported by the system data stream definitions. Conditional access mechanisms are not specified here. The stream definitions are designed so that implementation of practical conditional access systems is reasonable, and there are some syntactical elements specified which provide specific support for such systems.

0.7 Multiplex-wide operations

Multiplex-wide operations include the coordination of data retrieval off the channel, the adjustment of clocks, and the management of buffers. The tasks are intimately related. If the rate of data delivery off the channel is controllable, then data delivery may be adjusted so that decoder buffers neither overflow nor underflow; but if the data rate is not controllable, then elementary stream decoders must slave their timing to the data received from the channel to avoid overflow or underflow.

Program Streams are composed of packs whose headers facilitate the above tasks. Pack headers specify intended times at which each byte is to enter the Program Stream Decoder from the channel, and this target arrival schedule serves as a reference for clock correction and buffer management. The schedule need not be followed exactly by decoders, but they must compensate for deviations about it.

Similarly, Transport Streams are composed of Transport Stream packets with headers containing information which specifies the times at which each byte is intended to enter a Transport Stream Decoder from the channel. This schedule provides exactly the same function as that which is specified in the Program Stream.

An additional multiplex-wide operation is a decoder's ability to establish what resources are required to decode a Transport Stream or Program Stream. The first pack of each Program Stream conveys parameters to assist decoders in this task. Included, for example, are the stream's maximum data rate and the highest number of simultaneous video channels. The Transport Stream likewise contains globally useful information.

The Transport Stream and Program Stream each contain information which identifies the pertinent characteristics of, and relationships between, the elementary streams which constitute each program. Such information may include the language spoken in audio channels, as well as the relationship between video streams when multi-layer video coding is implemented.

0.8 Individual stream operations (PES Packet Layer)

The principal stream-specific operations are 1) de-multiplexing, and 2) synchronizing playback of multiple elementary streams.

0.8.1 De-multiplexing

On encoding, Program Streams are formed by multiplexing elementary streams, and Transport Streams are formed by multiplexing elementary streams, Program Streams, or the contents of other Transport Streams. Elementary streams may include private, reserved, and padding streams in addition to audio and video streams. The streams are temporally subdivided into packets, and the packets are serialized. A PES packet contains coded bytes from one and only one elementary stream.

In the Program Stream both fixed and variable packet lengths are allowed subject to constraints as specified in 2.5.1 on page 53 and 2.5.2 on page 53 of this specification. For Transport Streams the packet length is 188 bytes. Both fixed and variable PES packet lengths are allowed, and will be relatively long in most applications.

On decoding, de-multiplexing is required to reconstitute elementary streams from the multiplexed Program Stream or Transport Stream. Stream_id codes in Program Stream packet headers, and Packet ID codes in the Transport Stream make this possible.

0.8.2 Synchronization

Synchronization among multiple elementary streams is accomplished with Presentation Time Stamps (PTS) in the Program Stream and Transport streams. Time stamps are generally in units of 90kHz, but the System Clock Reference (SCR), the Program Clock Reference (PCR) and the optional Elementary Stream Clock Reference (ESCR) have extensions with a resolution of 27MHz. Decoding of N elementary streams is synchronized by adjusting the decoding of streams to a common master time base rather than by adjusting the decoding of one stream to match that of another. The master time base may be one of the N decoders' clocks, the data source’s clock, or it may be some external clock.

Each program in a Transport Stream, which may contain multiple programs, may have its own time base. The time bases of different programs within a Transport Stream may be different.

Because PTSs apply to the decoding of individual elementary streams, they reside in the PES packet layer of both the Transport Streams and Program Streams. End-to-end synchronization occurs when encoders save time stamps at capture time, when the time stamps propagate with associated coded data to decoders, and when decoders use those time stamps to schedule presentations.

Synchronization of a decoding system with a channel is achieved through the use of the SCR in the Program Stream and by its analog, the PCR, in the Transport Stream. The SCR and PCR are time stamps encoding the timing of the bit stream itself, and are derived from the same time base used for the audio and video PTS values from the same program. Since each program may have its own time base, there are separate PCR fields for each program in a Transport Stream containing multiple programs. In some cases it may be possible for programs to share PCR fields. Refer to 2.4.4 on page 44, Program Specific Information (PSI), for the method of identifying which PCR is associated with a program. A program shall have one and only one PCR time base associated with it.

0.8.3 Relation to compression layer

The PES packet layer is independent of the compression layer in some senses, but not in all. It is independent in the sense that PES packet payloads need not start at compression layer start codes, as defined in parts 2 and 3 of this Recommendation | International Standard. For example, video start codes may occur anywhere within the payload of a PES packet, and start codes may be split by a PES packet header. However, time stamps encoded in PES packet headers apply to presentation times of compression layer constructs (namely, presentation units). In addition, when the elementary stream data conforms to ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 13818-3, the PES_packet_data_bytes shall be byte aligned to the bytes of ITU-T Rec. H.222.0 | ISO/IEC 13818-1.

0.9 System reference decoder

Part 1 of ISO/IEC 13818 employs a "System Target Decoder," (STD), one for Transport Streams (refer to 2.4.2 on page 11) referred to as "Transport System Target Decoder"(T-STD) and one for Program Streams (refer to 2.5.2 on page 53) referred to as "Program System Target Decoder"(P-STD), to provide a formalism for timing and buffering relationships. Because the STD is parameterized in terms of ITU-T Rec. H.222.0 | ISO/IEC 13818 fields (for example, buffer sizes) each elementary stream leads to its own parameterization of the STD. Encoders shall produce bit streams that meet the appropriate STD's contraints. Physical decoders may assume that a stream plays properly on its STD; the physical decoder must compensate for ways in which its design differs from that of the STD.

0.10 Applications

The streams defined in this document are intended to be as useful as possible to a wide variety of applications. Application developers should select the most appropriate stream.

Modern data communications networks may be capable of supporting ITU-T Rec. H.222.0 | ISO/IEC 13818 video and ISO/IEC 13818 audio. A real time transport protocol is required. The Program Stream may also be suitable for transmission on such networks.

The Program Stream is also suitable for multimedia applications on CD-ROM. Software processing of the Program Stream may be appropriate.

The Transport Stream may be more suitable for error-prone environments, such as those used for distributing compressed bit-streams over long distance networks and in broadcast systems.

Many applications require storage and retrieval of ITU-T Rec. H.222.0 | ISO/IEC 13818 bitstreams on various digital storage media (DSM). A Digital Storage Media Command and Control (DSM CC) protocol is specified in Annex A and part 6 of this Recommendation | International Standard in order to facilitate the control of such media.

Information technology -- Coding of moving pictures and associated audio

Systems

Section 1: General

1.1 Scope

This part of ITU-T Rec. H.222.0 | ISO/IEC 13818 specifies the system layer of the coding. It was developed principally to support the combination of the video and audio coding methods defined in parts 2 and 3 of this Recommendation | International Standard. The system layer supports five basic functions: 1) the synchronization of multiple compressed streams on decoding, 2) the interleaving of multiple compressed streams into a single stream, 3) the initialization of buffering for decoding start up, 4) continuous buffer management, and 5) time identification.

An ITU-T Rec. H.222.0 | ISO/IEC 13818 multiplexed bit stream is either a **Transport Stream** or a **Program Stream.** Both streams are constructed from **PES packets** and packets containing other necessary information. Both streams support multiplexing of video and audio compressed streams from one program with a common time base. The **Transport Stream** additionally supports the multiplexing of video and audio compressed streams from multiple programs with independent time bases. For almost error-free environments the **Program Stream** is generally more appropriate, supporting software processing of program information. The **Transport Stream** is more suitable for use in environments where errors are likely.

An ITU-T Rec. H.222.0 | ISO/IEC 13818 multiplexed bit stream, whether a Transport Stream or a Program Stream, is constructed in two layers: the outermost layer is the system layer, and the innermost is the compression layer. The system layer provides the functions necessary for using one or more compressed data streams in a system. The video and audio parts of this specification define the compression coding layer for audio and video data. Coding of other types of data is not defined by the specification, but is supported by the system layer provided that the other types of data adhere to the constraints defined in 2.7. on page 80.

1.2 Normative References

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

below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunications Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

1.3 Identical Recommendations | International Standards

ITU-T Rec. H.262 | ISO/IEC 13818-2:1994 *Information technology - Coding of moving pictures and associated audio - Part 2: Video.*

1.4 Additional references

ISO 8859-1:1987, *Information processing- 8 bit single-byte coded graphic character Sets - Part 1: Latin alphabet No. 1.*

ISO/IEC 11172-1:1993 *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 1: Systems.*

ISO/IEC 11172-2:1993 *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 2: Video.*

ISO/IEC 11172-3:1993 *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 3 Audio.*

ISO/IEC 13818-3:1994 *Information technology - Generic coding of moving pictures and associated audio information- Part 3 Audio.*

Recommendation ITU-R BT.601.3 *Encoding parameters of digital television for studios.*

Recommendation ITU-R BT.470-2 *Television systems.*

Recommendation ITU-R BR.648 *Digital recording of audio signals.*

Report ITU-R BO.955.2 *Satellite sound broadcasting of vehicular, portable, and fixed receivers in the range 500 - 3000MHz.*

CCITT Recommendation J.17 *Pre-emphasis used on Sound-Programme Circuits.*

IEEE Standard 1180-1990 *Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform.*

IEC Publication 908:1987, C*D Digital Audio System.*

ISO/CD 13522-1;1993 *Information technology - Coded representation of multimedia and hypermedia information objects - Part 1:Base notation..*

ISO/CD 639-2; 1991, *Terminology - Codes for presentation of names of languages - Part 2: Alpha-3 code.*

Section 2 Technical elements

2.1 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply. If specific to a part, this is parenthetically noted.

2.1.1 access unit [system]: A coded representation of a presentation unit. In the case of audio, an access unit is the coded representation of an audio frame.

In the case of video, an access unit includes all the coded data for a picture, and any stuffing that follows it, up to but not including the start of the next access unit. If a picture is not preceded by a group_start_code or a sequence_header_code, the access unit begins with the picture start code. If a picture is preceded by a group_start_code and/or a sequence_header_code, the access unit begins with the first byte of the first of these start codes. If it is the last picture preceding a sequence_end_code in the bitstream all bytes between the last byte of the coded picture and the sequence_end_code (including the sequence_end_code) belong to the access unit.

2.1.2 bitrate: The rate at which the compressed bit stream is delivered from the channel to the input of a decoder.

2.1.3 byte aligned: A bit in a coded bit stream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.

2.1.4 channel: A digital medium that stores or transports an ITU-T Rec. H.222.0 | ISO/IEC 13818 stream.

2.1.5 coded representation: A data element as represented in its encoded form.

2.1.6 compression: Reduction in the number of bits used to represent an item of data.

2.1.7 constant bitrate: Operation where the bitrate is constant from start to finish of the compressed bit stream.

2.1.8 constrained system parameter stream; CSPS [system]: A Program Stream for which the constraints defined in 2.7.9 on page 83 apply.

2.1.9 CRC: The Cyclic Redundancy Check to verify the correctness of data.

2.1.10 data element: An item of data as represented before encoding and after decoding.

2.1.11 decoded stream: The decoded reconstruction of a compressed bit stream.

2.1.12 decoder: An embodiment of a decoding process.

2.1.13 decoding (process): The process defined in this Recommendation | International Standard that reads an input coded bit stream and outputs decoded pictures or audio samples.

2.1.14 decoding time-stamp; DTS [system]: A field that may be present in a PES packet header that indicates the time that an access unit is decoded in the system target decoder.

2.1.15 digital storage media; DSM: A digital storage or transmission device or system.

2.1.16 DSM-CC; digital storage media command and control.

2.1.17 entitlement control message; ECM: Entitlement Control Messages are private conditional access information which specify control words and possibly other, typically stream-specific, scrambling and/or control parameters.

2.1.18 entitlement management message; EMM: Entitlement Management Messages are private conditional access information which specify the authorization levels or the services of specific decoders. They may be addressed to single decoders or groups of decoders.

2.1.19 editing: The process by which one or more compressed bit streams are manipulated to produce a new compressed bit stream. Conforming edited bit streams must meet the requirements defined in this Recommendation | International Standard.

2.1.20 elementary stream; ES [system]: A generic term for one of the coded video, coded audio or other coded bit streams in PES packets. One elementary stream is carried in a sequence of PES packets with one and only one stream_id.

2.1.21 Elementary Stream Clock Reference; ESCR [system]: A time stamp in the PES Stream from which decoders of PES streams may derive timing.

2.1.22 encoder: An embodiment of an encoding process.

2.1.23 encoding (process): A process, not specified in this Recommendation | International Standard, that reads a stream of input pictures or audio samples and produces a valid coded bit stream as defined in this Recommendation | International Standard.

2.1.24 entropy coding: Variable length lossless coding of the digital representation of a signal to reduce redundancy.

2.1.25 event: An event is defined as a collection of elementary streams with a common time base, an associated start time, and an associated end time.

2.1.26 fast forward playback [video]: The process of displaying a sequence, or parts of a sequence, of pictures in display-order faster than real-time.

2.1.27 forbidden: The term "forbidden", when used in the clauses defining the coded bit stream, indicates that the value shall never be used. This is usually to avoid emulation of start codes.

2.1.28 ITU-T Rec. H.222.0 | ISO/IEC 13818 (multiplexed) stream [system]: A bit stream composed of 0 or more elementary streams combined in the manner defined in this part of ITU-T Rec. H.222.0 | ISO/IEC 13818-1.

2.1.29 layer [video and systems]: One of the levels in the data hierarchy of the video and system specifications defined in parts 1 and 2 of this Recommendation | International Standard.

2.1.30 pack [system]: A pack consists of a pack header followed by zero or more packets. It is a layer in the system coding syntax described in 2.5.3.3 on page 58 of this Recommendation | International Standard.

2.1.31 packet [system]: A packet consists of a header followed by a number of contiguous bytes from an elementary data stream. It is a layer in the system coding syntax described in 2.4.3.6 on page 33 of this Recommendation | International Standard.

2.1.32 packet data [system]: Contiguous bytes of data from an elementary stream present in a packet.

2.1.33 packet identifier; PID [system]: A unique integer value used to associate elementary streams of a program in a single or multi-program Transport Stream as described in 2.4.3 on page 21.

2.1.34 padding [audio]: A method to adjust the average length of an audio frame in time to the duration of the corresponding PCM samples, by conditionally adding a slot to the audio frame.

2.1.35 payload: Payload refers to the bytes which follow the header bytes in a packet. For example, the payload of a Transport Stream packet includes the PES_packet_header and its PES_packet_data_bytes, or pointer_field and PSI sections, or private data; but a PES_packet_payload consists of only PES_packet_data_bytes. The Transport Stream packet header and adaptation fields are not payload.

2.1.36 PES [system]: An abbreviation for Packetized Elementary Stream.

2.1.37 PES packet [system]: The data structure used to carry elementary stream data. It consists of a PES packet header followed by PES packet payload and is described in 2.4.3.6 on page 33.

2.1.38 PES packet header[system]: The leading fields in a PES packet up to and not including the PES_packet_data_byte fields, where the stream is not a padding stream. In the case of a padding stream the PES packet header is similarly defined as the leading fields in a PES packet up to and not including padding_byte fields.

2.1.39 PES Stream [system]: A PES Stream consists of PES packets, all of whose payloads consist of data from a single elementary stream, and all of which have the same stream_id. Specific semantic constraints apply.

2.1.40 presentation time-stamp; PTS [system]: A field that may be present in a PES packet header that indicates the time that a presentation unit is presented in the system target decoder.

2.1.41 presentation unit; PU [system]: A decoded Audio Access Unit or a decoded picture.

2.1.42 program [system]: A program is a collection of program elements. Program elements may be elementary streams. Program elements need not have any defined time base; those that do, have a common time base and are intended for synchronized presentation.

2.1.43 Program Clock Reference; PCR [system]: A time stamp in the Transport Stream from which decoder timing is derived.

2.1.44 program element[system]:A generic term for one of the elementary streams or other data streams that may be included in a program.

2.1.45 Program Specific Information; PSI [system]: PSI consists of normative data which is necessary for the demultiplexing of Transport Streams and the successful regeneration of programs and is described in 2.4.4 on page 44. One case of PSI, the non-mandatory network information table, is privately defined.

2.1.46 random access: The process of beginning to read and decode the coded bit stream at an arbitrary point.

2.1.47 reserved: The term "reserved", when used in the clauses defining the coded bit stream, indicates that the value may be used in the future for ISO defined extensions. Unless otherwise specified within this Recommendation | International Standard, all reserved bits shall be set to '1'.

2.1.48 scrambling[system]: The alteration of the characteristics of a video, audio or coded data stream in order to prevent unauthorized reception of the information in a clear form. This alteration is a specified process under the control of a conditional access system.

2.1.49 source stream: A single non-multiplexed stream of samples before compression coding.

2.1.50 splicing[system]: The concatenation, performed on the system level, of two different elementary streams. The resulting system stream conforms totally to part 1 of this Recommendation | International Standard. The splice may result in discontinuities in timebase, continuity counter, PSI, and decoding.

2.1.51 start codes [system]: 32-bit codes embedded in the coded bit stream that are unique. They are used for several purposes including identifying some of the layers in the coding syntax. Start codes consist of a 24 bit prefix (0x000001) and an 8 bit stream_id as shown in table 2-19 on page 36.

2.1.52 STD input buffer [system]: A first-in first-out buffer at the input of a system target decoder for storage of compressed data from elementary streams before decoding.

2.1.53 still picture: A coded still picture consists of a video sequence containing exactly one coded picture which is intra-coded. This picture has an associated PTS and the presentation time of succeeding pictures, if any, is later than that of the still picture by at least two picture periods.

2.1.54 system header [system]: The system header is a data structure defined in 2.5.3.5 on page 59 of this Recommendation | International Standard that carries information summarizing the system characteristics of the ITU-T Rec. H.222.0 | ISO/IEC 13818 multiplexed stream.

2.1.55 System Clock Reference; SCR [system]: A time stamp in the Program Stream from which decoder timing is derived.

2.1.56 system target decoder; STD [system]: A hypothetical reference model of a decoding process used to describe the semantics of an ITU-T Rec. H.222.0 | ISO/IEC 13818 multiplexed bit stream.

2.1.57 time-stamp [system]: A term that indicates the time of a specific action such as the arrival of a byte or the presentation of a Presentation Unit.

2.1.58 Transport Stream packet header [system]: The leading fields in a Transport Stream packet, up to and including the continuity_counter field.

2.1.59 variable bitrate: Operation where the bitrate varies with time during the decoding of a compressed bit stream.

2.2 Symbols and abbreviations

The mathematical operators used to describe this Recommendation | International Standard are similar to those used in the C programming language. However, integer division with truncation and rounding are specifically defined. The bitwise operators are defined assuming two's-complement representation of integers. Numbering and counting loops generally begin from 0.

2.2.1 Arithmetic operators

+ Addition.

- Subtraction (as a binary operator) or negation (as a unary operator).

++ Increment.

- - Decrement.

* or ×	Multiplication.
^	Power.
/	Integer division with truncation of the result toward 0. For example, 7/4 and -7/-4 are truncated to 1 and -7/4 and 7/-4 are truncated to -1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from 0 unless otherwise specified. For example 3//2 is rounded to 2, and -3//2 is rounded to -2.
DIV	Integer division with truncation of the result towards -∞.
%	Modulus operator. Defined only for positive numbers.
Sign( )	Sign(x) = 1 if x > 0 0 if x == 0 -1 if x < 0
NINT ( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from 0.
sin	Sine.
cos	Cosine.
exp	Exponential.
√	Square root.
$\log_{10}$	Logarithm to base ten.
$\log_e$	Logarithm to base e.

2.2.2 Logical operators

||	Logical OR.
&&	Logical AND.
!	Logical NOT.

2.2.3 Relational operators

>	Greater than.
≥	Greater than or equal to.
<	Less than.
≤	Less than or equal to.
==	Equal to.

!= Not equal to.

max [,...,] the maximum value in the argument list.

min [,...,] the minimum value in the argument list.

2.2.4 Bitwise operators

& AND.

| OR.

>> Shift right with sign extension.

<< Shift left with 0 fill.

2.2.5 Assignment

= Assignment operator.

2.2.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

bslbf Bit string, left bit first, where "left" is the order in which bit strings are written in the Recommendation | International Standard. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.

ch channel.

gr granule of 3 * 32 subband samples in audio Layer II, 18 * 32 sub-band samples in audio Layer III.

main_data The main_data portion of the bit stream contains the scale factors, Huffman encoded data, and ancillary information.

main_data_beg This gives the location in the bit stream of the beginning of the main_data for the frame. The location is equal to the ending location of the previous frame's main_data plus 1 bit. It is calculated from the main_data_end value of the previous frame.

part2_length this value contains the number of main_data bits used for scale factors.

rpchof remainder polynomial coefficients, highest order first.

sb subband.

scfsi scalefactor selector information.

switch_point_l Number of scalefactor band (long block scalefactor band) from which point on window switching is used.

switch_point_s	Number of scalefactor band (short block scalefactor band) from which point on window switching is used.
tcimsbf	two's complement integer, msb (sign) bit first.
uimsbf	Unsigned integer, most significant bit first.
vlclbf	Variable length code, left bit first, where "left" refers to the order in which the variable length codes are written.
window	Number of actual time slot in case of block_type==2, $0 \leq$ window ≤ 2.

The byte order of multi-byte words is most significant byte first.

2.2.7 Constants

π 3.14159265359

e 2.71828182845

2.3 Method of describing bit stream syntax

The bit streams retrieved by the decoder are described in 2.4.1 on page 10 and 2.5.1 on page 53. Each data item in the bit stream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bit stream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their decoding are described in the clauses containing the semantic description of the syntax. The following constructs are used to express the conditions when data elements are present, and are in normal type:

Note this syntax uses the "C"-code convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true.

while (condition) {	If the condition is true, then the group of data elements occurs next in the
data_element	data stream. This repeats until the condition is not true.
. . .	
}	
do {	The data element always occurs at least once. The data element is
data_element	repeated until the condition is not true.
. . . }	
while (condition)	
if (condition) {	If the condition is true, then the first group of data elements occurs next
data_element	in the data stream.
. . .	
}	
else {	If the condition is not true, then the second group of data elements occurs
data_element	next in the data stream.
. . .	
}	

for (i = 0;i<n;i++) {
data_element
. . .
}
	The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to 1 for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} are omitted when only one data element follows.

data_element []	data_element [] is an array of data. The number of data elements is indicated by the context.
data_element [n]	data_element [n] is the n+1th element of an array of data.
data_element [m][n]	data_element [m][n] is the m+1,n+1 th element of a two-dimensional array of data.
data_element [l][m][n]	data_element [l][m][n] is the l+1,m+1,n+1 th element of a three-dimensional array of data.
data_element [m..n]	is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that either figure 2-6 on page 12 or figure 2-7 on page 54 implements a satisfactory decoding procedure. In particular, they define a correct and error-free input bitstream. Actual decoders must include a means to look for start codes and sync bytes (Transport Stream) in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardized.

Definition of bytealigned function

The function bytealigned() returns 1 if the current position is on a byte boundary; that is, the next bit in the bit stream is the first bit in a byte. Otherwise it returns 0.

Definition of nextbits function

The function nextbits() permits comparison of a bit string with the next bits to be decoded in the bit stream.

Definition of next_start_code function

The next_start_code() function removes any zero bit and zero byte stuffing and locates the next start code.

Table 2-1 -- Next start code

Syntax	No. of bits	Mnemonic
next_start_code() {		
while (!bytealigned())		
zero_bit	**1**	**'0'**
while (nextbits() != '0000 0000 0000 0000 0000 0001')		
zero_byte	**8**	**'00000000'**
}		

This function checks whether the current position is byte aligned. If it is not, 0 stuffing bits are present. After that any number of 0 bytes may be present before the start-code. Therefore start-codes are always byte aligned and may be preceded by any number of 0 stuffing bits.

2.4 Transport Stream bitstream requirements

2.4.1 Transport Stream coding structure and parameters

The ITU-T Rec. H.222.0 | ISO/IEC 13818 Transport Stream coding layer allows one or more programs to be combined into a single stream. Data from each elementary stream are multiplexed together with information that allows synchronized presentation of the elementary streams within a program.

A Transport Stream consists of one or more programs. Audio and video elementary streams consist of access units.

Elementary Stream data is carried in PES packets. A PES packet consists of a PES packet header followed by packet data. PES packets are inserted into Transport Stream packets. The first byte of each PES packet header is located at the first available payload location of a Transport Stream packet.

The PES packet header begins with a 32-bit start-code that also identifies the stream or stream type to which the packet data belongs. The PES packet header may contain decoding and presentation time stamps(DTS and PTS). The PES packet header also contains other optional fields. The PES packet data field contains a variable number of contiguous bytes from one elementary stream.

Transport Stream packets begin with a 4 byte prefix, which contains a 13 bit Packet ID (PID), defined in table 2-3 on page 22 . The PID identifies, via the Program Specific Information (PSI) tables, the contents of the data contained in the Transport Stream packet. Transport Stream packets of one PID value carry data of one and only one elementary stream.

The PSI tables are carried in the Transport Stream. There are four PSI tables:

- Program Association Table
- Program Map Table
- Conditional Access Table
- Network Information Table

These tables contain the necessary and sufficient information to demultiplex and present programs. The Program Map Table, in table 2-28 on page 50 , specifies, among other information, which PIDs, and therefore which elementary streams are associated to form each program. This table also indicates the PID of the Transport Stream packets which carry the PCR for each program. The Conditional Access Table shall be present if scrambling is employed. The Network Information Table is optional and its contents are not specified by this Recommendation | International Standard.

Transport Stream packets may be null packets. Null packets are intended for padding of Transport Streams. They may be inserted or deleted by re-multiplexing processes and, therefore, the delivery of the payload of null packets to the decoder cannot be assumed.

This standard does not specify the coded data which may be used as part of conditional access systems. The standard does however provide mechanisms for program service providers to transport and identify this data for decoder processing, and to reference correctly data which are specified by the standard. This type of support is provided both through Transport Stream packet structures and in the conditional access table (refer to table 2-28 on page 49 of the PSI).

2.4.2 Transport Stream system target decoder

The semantics of the Transport Stream specified in 2.4.3 on page 21 and the constraints on these semantics specified in 2.7. on page 80 require exact definitions of byte arrival and decoding events and the times at which these occur. The definitions needed are set out in this Recommendation | International Standard using a hypothetical decoder known as the Transport Stream system target decoder (T-STD). Informative Annex D of this Specification contains a more detailed explanation.

The T-STD is a conceptual model used to define these terms precisely and to model the decoding process during the construction or verification of Transport Streams. The T-STD is defined only for this purpose. There are three types of decoders in the T-STD: video, audio, and systems figure 2-6 on page 12 illustrates an example. Neither the architecture of the T-STD nor the timing described precludes uninterrupted, synchronized play-back of Transport Streams from a variety of decoders with different architectures or timing schedules.

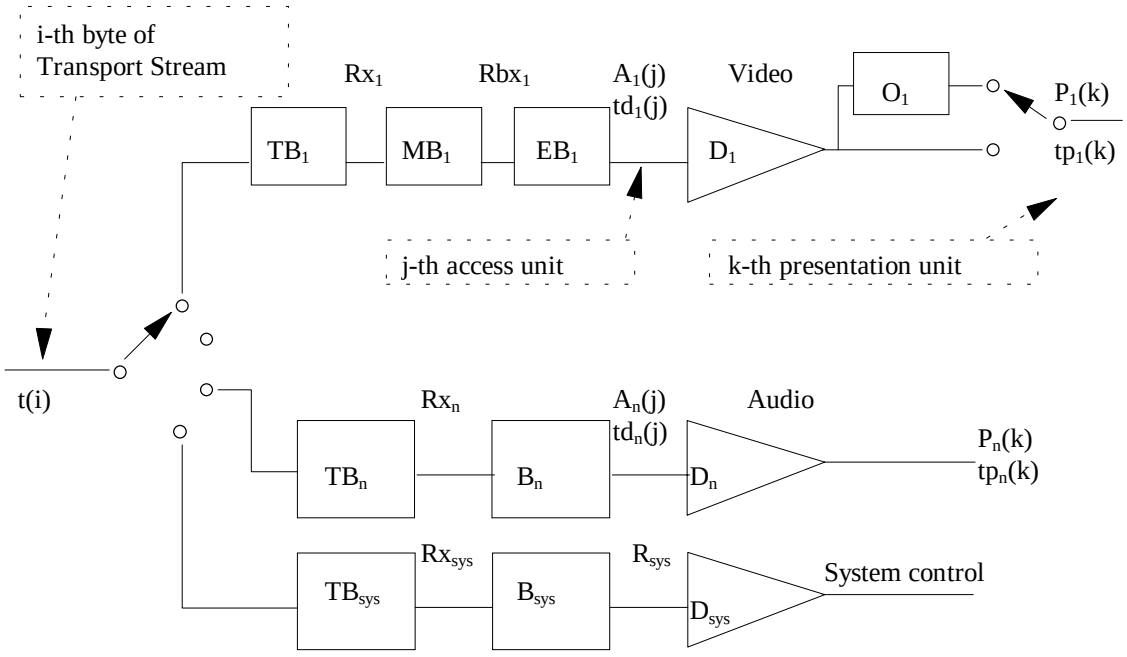

Figure 2-6 -- Transport Stream system target decoder notation

The following notation is used to describe the Transport Stream system target decoder and is partially illustrated in figure 2-6 above.

i, i', i" are indices to bytes in the Transport Stream. The first byte has index 0.

j is an index to access units in the elementary streams.

k, k', k" are indices to presentation units in the elementary streams.

n is an index to the elementary streams.

p is an index to Transport Stream packets in the Transport Stream.

t(i) indicates the time in seconds at which the i^{th} byte of the Transport Stream enters the system target decoder. The value t(0) is an arbitrary constant.

$PCR(i)$ is the time encoded in the PCR field measured in units of the period of the 27 MHz system clock where i is the byte index of the final byte of the program_clock_reference_base field.

$A_n(j)$ is the j^{th} access unit in elementary stream n. $A_n(j)$ is indexed in decoding order.

$td_n(j)$ is the decoding time, measured in seconds, in the system target decoder of the j^{th} access unit in elementary stream n.

$P_n(k)$ is the k^{th} presentation unit in elementary stream n. $P_n(k)$ results from decoding $A_n(j)$. $P_n(k)$ is indexed in presentation order.

$tp_n(k)$ is the presentation time, measured in seconds, in the system target decoder of the k^{th} presentation unit in elementary stream n.

t is time measured in seconds.

$F_n(t)$ is the fullness, measured in bytes, of the system target decoder input buffer for elementary stream n at time t.

B_n is the main buffer for elementary stream n. It is present only for audio elementary streams.

BS_n is the size of buffer, B_n, measured in bytes.

B_{sys} is the main buffer in the system target decoder for system information for the program that is in the process of being decoded.

BS_{sys} is the size of B_{sys}, measured in bytes.

MB_n is the multiplexing buffer, for elementary stream n. It is present only for video elementary streams.

MBS_n is the size of MB_n, measured in bytes.

EB_n is the elementary stream buffer for elementary stream n. It is present only for video elementary streams.

EBS_n is the size of the elementary stream buffer EB_n, measured in bytes.

TB_{sys} is the transport buffer for system information for the program that is in the process of being decoded.

TBS_{sys} is the size of TB_{sys}, measured in bytes.

TB_n is the transport buffer for elementary stream n.

TBS_n is the size of TB_n, measured in bytes.

D_{sys} is the decoder for system information in Program Stream n.

D_n is the decoder for elementary stream n.

O_n is the reorder buffer for video elementary stream n.

R_{sys} is the rate at which data are removed from B_{sys}.

Rx_n is the rate at which data are removed from TB_n.

Rbx_n is the rate at which PES packet payload data are removed from MB_n when the leak method is used. Defined only for video elementary streams.

$Rbx_n(j)$ is the rate at which PES packet payload data are removed from MB_n when the vbv_delay method is used. Defined only for video elementary streams.

Rx_{sys} The rate at which data are removed from TB_{sys}.

R_{es} The video elementary stream rate coded in a sequence header.

2.4.2.1 System clock frequency

Timing information referenced in the T-STD is carried by several data fields defined in this Recommendation | International Standard. Refer to 2.4.3.4 on page 24, and 2.4.3.6 on page 33. In PCR fields this information is coded as the sampled value of a program's system clock. The PCR fields are carried in the adaptation field of the Transport Stream packets with a PID value equal to the PCR_PID defined in the TS_program_map_section of the program being decoded.

Practical decoders may reconstruct this clock from these values and their respective arrival times. The following are minimum constraints which apply to the program's system clock frequency as represented by the values of the PCR fields when they are received by a decoder.

The value of the system clock frequency is measured in Hz and shall meet the following constraints:

$$27\,000\,000 - 810 \leq \text{system\_clock\_frequency} \leq 27\,000\,000 + 810$$

$$\text{rate of change of system\_clock\_frequency with time} \leq 75 \times 10^{-3} \text{ Hz/s}$$

Note - Sources of coded data should follow a tighter tolerance in order to facilitate compliant operation of consumer recorders and playback equipment.

A program's system_clock_frequency may be more accurate than required. Such improved accuracy may be transmitted to the decoder via the System clock descriptor in 2.6.20 on page 76.

Bit rates defined in this Specification are measured in terms of system_clock_frequency. For example, a bit rate of 27 000 000 bits per second in the T-STD would indicate that one byte of data is transferred every eight(8) cycles of the system clock.

The notation "system_clock_frequency" is used in several places in this Recommendation | International Standard to refer to the frequency of a clock meeting these requirements. For notational convenience, equations in which PCR, PTS, or DTS appear lead to values of time which are accurate to some integral multiple of $(300 \times 2^{33}/\text{system\_clock\_frequency})$ seconds. This is due to the encoding of PCR timing information as 33 bits of 1/300 of the system clock frequency plus 9 bits for the remainder, and encoding as 33 bits of the system clock frequency divided by 300 for PTS and DTS.

2.4.2.2 Input to the Transport Stream system target decoder

Input to the Transport Stream system target decoder (T-STD) is a Transport Stream. A Transport Stream may contain multiple programs with independent time bases. However, the T-STD decodes only one program at a time. In the T-STD model all timing indications refer to the time base of that program.

Data from the Transport Stream enters the T-STD at a piecewise constant rate. The i^{th} byte enters at time t(i). The time at which this byte enters the T-STD can be recovered from the input stream by decoding the input program clock reference (PCR) fields, encoded in the Transport Stream packet adaptation field of the program

to be decoded and by counting the bytes in the complete Transport Stream between successive PCRs for the program to be decoded. The PCR is encoded in equation 2-3 in two parts; one, in units of the period of 1/300 times the system clock frequency (yielding 90 kHz), called program_clock_reference_base (equation 2-1), and one, called program_clock_reference_ext (equation 2-2) in units of the period of the system clock frequency. The value encoded in the PCR field indicates the time t(i), where i refers to the byte containing the last bit of the program_clock_reference_base field.

Specifically:

$$PCR\_base(i) = \left(\left(system\_clock\_frequency \times t(i) \right) DIV\ 300 \right) \% \ 2^{33} \tag{2-1}$$

$$PCR\_ext(i) = \left(\left(system\_clock\_frequency \times t(i) \right) DIV\ 1 \right) \% \ 300 \tag{2-2}$$

$$PCR(i) = PCR\_base(i) \times 300 + PCR\_ext(i) \tag{2-3}$$

For all other bytes the input arrival time, t(i) shown in equation 2-4 on page 15 below, is computed from PCR(i") and the transport rate at which data arrive, where the transport rate is determined as the number of bytes in the Transport Stream between the bytes containing the last bit of two successive program_clock_reference fields of the same program divided by the difference between the time values encoded in these same two PCR fields.

$$t(i) = \frac{PCR(i")}{system\_clock\_frequency} + \frac{i - i"}{transport\_rate(i)} \tag{2-4}$$

Where:

i	is the index of any byte in the Transport Stream for i"< i < i'.
i"	is the index of the byte containing the last bit of the most recent program_clock_reference base field applicable to the program being decoded.
PCR(i")	is the time encoded in the program clock reference base and extension fields in units of the system clock.

The transport rate is given by

$$transport\_rate(i) = \frac{\left((i' - i") \times system\_clock\_frequency \right)}{PCR(i') - PCR(i")} \tag{2-5}$$

where

i'	is the index of the byte containing the last bit of the immediately following program_clock_reference_base applicable to the program being decoded.
Note:	i" < i ≤ i'

In the case of a timebase discontinuity, indicated by the discontinuity_indicator in the transport packet adaptation field, the definition given in equation 2-4 and equation 2-5 for the time of arrival of bytes at the input to the T-STD is not applicable between the last PCR of the old timebase and the first PCR of the new timebase. In this case the time of arrival of these bytes is determined according to equation 2-4 with the modification that the transport rate used is that applicable between the last and next to last PCR of the old timebase.

A tolerance is specified for the PCR values. The PCR tolerance is defined as the maximum inaccuracy allowed in received PCRs. This inaccuracy may be due to imprecision in the PCR values or to PCR modification during

remultiplexing. It does not include errors in packet arrival time due to network jitter or other causes. The PCR tolerance is ±500 ns.

In the T-STD model, the inaccuracy will be reflected as an inaccuracy in the calculated transport rate using equation 2-5.

Transport Streams with multiple programs and variable rate.

Note - Transport Streams may contain multiple programs which have independent time bases. Separate sets of PCRs, as indicated by the respective PCR_PID values, are required for each such independent program, and therefore the PCRs cannot be co-located. The Transport Stream rate is piecewise constant for the program entering the T-STD. Therefore, if the Transport Stream rate is variable it can only vary at the PCRs of the program under consideration. Since the PCRs, and therefore the points in the transport Stream where the rate varies, are not co-located, the rate at which the Transport Stream enters the T-STD would have to differ depending on which program is entering the T-STD. Therefore, it is not possible to construct a consistent T-STD delivery schedule for an entire Transport Stream when that Transport Stream contains multiple programs with independent time bases and the rate of the Transport Stream is variable. It is straightforward, however, to construct constant bit rate Transport Streams with multiple variable rate programs.

2.4.2.3 Buffering

Complete Transport Stream packets containing data from elementary stream n, as indicated by its PID, are passed to the transport buffer for stream n, TB_n. This includes duplicate Transport Stream packets and packets with no payload. Transfer of the ith byte from the system target decoder input to TB_n is instantaneous, so that the ith byte enters the buffer for stream n, of size TBS_n, at time t(i).

All bytes that enter the buffer TB_n are removed at the rate Rx_n specified below. Bytes which are part of the PES packet or its contents are delivered to the main buffer B_n for audio elementary streams and system data, and to the multiplexing buffer MB_n for video elementary streams. Other bytes are not, and may be used to control the system. Duplicate Transport Stream packets are not delivered to B_n, MB_n, or B_{sys}.

The buffer TB_n is emptied as follows: when there is no data in TB_n , Rx_n is equal to zero. Otherwise

for video $Rx_n = 1.2 \times R_{max}$ [profile,level] where

R_{max}[profile,level] is specified according to the profile and level which can be found in table 8.12 of part 2 of this Specification. Table 8.12 specifies the upper bound of the rate of each elementary video stream within a specific profile and level.

Rx_n is equal to 1.2 * R_{max} for ISO/IEC 11172-2 video streams, where R_{max} refers to the maximum bitrate for a Constrained Parameters bitstream in ISO/IEC 11172-2.

for audio $Rx_n = 2 \times 10^6$ bits per second

for systems data $Rx_n = 1 \times 10^6$ bits per second

Rx_n is measured with respect to the system clock frequency.

Complete Transport Stream packets containing system information, for the program selected for decoding, enter the system transport buffer, TB_{sys}, at the Transport Stream rate. These include Transport Stream packets whose PID values are 0 or 1, and all Transport Stream packets identified via the Program Association Table (table 2-26 on page 47) as having the program_map_PID value for the selected program. Network Information Table (NIT) data as specified by the NIT PID is not transferred to TB_{sys}.

Bytes are removed from TB_{sys} at the rate Rx_{sys} and delivered to Bsys. Each byte is transferred instantaneously.

Duplicate Transport Stream packets are not delivered to B_{sys}.

Transport packets which do not enter any TB_n or TB_{sys} are discarded.

The transport buffer size is fixed at 512 bytes.

The elementary stream buffer sizes EBS_1 through EBS_n are defined for video as equal to the vbv_buffer_size as it is carried in the sequence header. Refer to ISO/IEC 11172-2 and ITU-T Rec. H.262 | ISO/IEC 13818-2.

The multiplexing buffer size MBS_1 through MBS_n are defined for video as follows:

For Low and Main level

$$MBS_n = BS_{mux} + BS_{oh} + VBV_{max}[profile,level] - vbv\_buffer\_size,$$

where BS_{oh}, PES packet overhead buffering is defined as:

$$BS_{oh} = (1/750)\text{seconds} \times R_{max}[profile,level]$$

and $BS_{mux,}$, additional multiplex buffering is defined as:

$$BS_{mux} = 0.004 \text{ seconds} * R_{max}[profile,level]$$

and where $VBV_{max}[profile,level]$ is defined in table 8-12 and $R_{max}[profile,level]$ is defined in table 8-13 of ITU-T Rec. H.262 | ISO/IEC 13818-2, and vbv buffer size is carried in the sequence header (refer to ITU-T Rec. H.262 | ISO/IEC 13818-2).

For High 1440 and High level

$$MBS_n = BS_{mux} + BS_{oh}$$

where BS_{oh} is defined as:

$$BS_{oh} = (1/750)\text{seconds} \times R_{max}[profile,level]$$

and BS_{mux} is defined as:

$$BS_{mux} = 0.004 \text{ seconds} * R_{max}[profile,level]$$

and where $R_{max}[profile,level]$ is defined in table 8-13 of ITU-T Rec. H.262 | ISO/IEC 13818-2.

For Constrained Parameters ISO/IEC 11172-2 bitstreams

$$MBS_n = BS_{mux} + BS_{oh} + vbv\_max - vbv\_buffer\_size$$

where BS_{oh} is defined as:

$$BS_{oh} = (1/750) \text{ seconds} \times R_{max}$$

and BS_{mux} is defined as:

$$BS_{mux} = 0.004 \text{ seconds} * R_{max}$$

and where R_{max} and vbv_max refer to the maximum bitrate and the maximum vbv_buffer_size for a Constrained Parameters bitstream in ISO/IEC 11172-2 respectively.

A portion BS_{mux} = 4ms x R_{max}[profile,level] of the MBS_n is allocated for buffering to allow multiplexing. The remainder is available for BS_{oh} and may also be available for initial multiplexing.

> Note - Buffer occupancy by PES packet overhead is directly bounded in PES streams by the P-STD which is defined in 2.5.2.4 on page 57. It is possible, but not necessary, to utilize PES streams to construct Transport Streams.

The main buffer sizes BS_1 through BS_n are defined as follows:

for audio $BS_n = BS_{mux} + BS_{dec} + BS_{oh} = 3\,584$ bytes

The size of the access unit decoding buffer BS_{dec}, and the PES packet overhead buffer BS_{oh} are constrained by

$$BS_{dec} + BS_{oh} \leq 2\,848 \text{ bytes}$$

A portion (736 bytes) of the 3 584 byte buffer is allocated for buffering to allow multiplexing. The rest, 2 848 bytes, are shared for access unit buffering BS_{dec}, BS_{oh} and additional multiplexing.

The main buffer B_{sys} for system data is of size BS_{sys} = 1536 bytes.

The following applies only to video elementary streams.

For video elementary streams, data is transferred from MB_n to EB_n using one of two methods: the leak method or the VBV delay method.

Leak method

The leak method transfers data from MB_n to EB_n using a leak rate R_{bx}. The leak method is used whenever any of the following is true:

> The STD descriptor for the elementary stream is not present in the Transport Stream,
> the STD descriptor is present and the leak_valid flag has a value of '1',
> the STD descriptor is present, the leak_valid has a value of '0', and the vbv_delay fields coded in the video stream have the value 0xFFFF, or
> trick mode status is true (refer to 2.4.3.6 on page 33).

For Low and Main level

$Rbx_n = R_{max}$(profile,level)

For High-1440 and High level

Rbx_n = Min {1.05 x R_{es}, R_{max}(profile,level)}

For Constrained Parameters bitstream in ISO/IEC 11172-2

> Rbx_n = 1.2 x R_{max} where R_{max} is the maximum bit rate for a Constrained Parameters bitstream in ISO/IEC 11172-2.

If there is PES packet payload data in MB_n, and buffer EB_n is not full, PES packet payload is transferred from MB_n to EB_n at a rate equal to R_{bx}. If EB_n is full, data are not removed from MB_n. When a byte of data is transferred from MB_n to EB_n, all PES packet header bytes that are in MB_n and immediately precede that byte are instantaneously removed and discarded. When there is no PES packet payload data present in MB_n, no data is removed from MB_n. All data that enters MB_n leaves it. All PES packet payload data bytes enter EB_n instantaneously upon leaving MB_n.

Vbv_delay method

The vbv delay method specifies precisely the time at which each byte of coded video data is transferred from MB_n to EB_n, using the vbv_delay values coded in the video elementary stream. The vbv_delay method is used whenever the STD descriptor for this elementary stream is present in the Transport Stream, the leak_valid flag in the descriptor has the value '0', and vbv_delay fields coded in the video stream are not equal to 0xFFFF. Note that if any vbv_delay values in a video sequence are not equal to 0xFFFF, none of the vbv_delay fields in that sequence shall be equal to 0xFFFF (refer to ISO/IEC 11172-2 and ITU-T Rec. H.262 | ISO/IEC 13818-2).

When the vbv_delay method is used, the final byte of the video picture start code for picture j is transferred from MB_n to the EB_n at the time $td_n(j)$ - vbv_delay(j), where $td_n(j)$ is the decoding time of picture j, as defined above, and vbv_delay(j) is the delay time, in seconds, indicated by the vbv_delay field of picture j. The transfer of bytes between the final bytes of successive picture start codes (including the final byte of the second start code), into the buffer EBn, is at a piecewise constant rate, $R_{bx}(j)$, which is specified for each picture j. Specifically, the rate, $R_{bx}(j)$, of transfer into this buffer is given by:

$$R_{bx}(j) = NB(j) / (vbv\_delay(j) - vbv\_delay(j+1) + td_n(j+1) - td_n(j))$$

Note - vbv_delay(j+1) and $td_n(j+1)$ may have values that differ from those normally expected for periodic video display if the low_delay flag in the video sequence extension is set to '1'. It may not be possible to determine the correct values by examination of the bit stream.

where NB(j) is the number of bytes between the final bytes of the picture start codes (including the final byte of the second start code) of pictures j and j+1, excluding PES packet header bytes.

The $R_{bx}(j)$ derived from 2-10 shall be less than or equal to R_{max}(profile,level) for elementary streams of stream type ‘0x02’(refer to table 2-36 on page 64), where R_{max}(profile,level) is defined in ITU-T Rec. H.262 | ISO/IEC 13818-2, and shall be less than or equal to the maximum bit rate allowed in the constrained parameters set for elementary streams of stream type ‘0x01’, refer to ISO/IEC 11172-2.

When a byte of data is transferred from MB_n to EB_n, all PES packet header bytes that are in MB_n and immediately precede that byte are instantaneously removed and discarded. All data that enters MB_n leaves it. All PES packet payload data bytes enter EB_n instantaneously upon leaving MB_n.

Removal of access units

For each elementary stream buffer EB_n and main buffer B_n all data for the access unit that has been in the buffer longest, $A_n(j)$, and any stuffing bytes that immediately precede it that are present in the buffer at the time $td_n(j)$ are removed instantaneously at time $td_n(j)$. The decoding time $td_n(j)$ is specified in the DTS or PTS fields (refer to 2.4.3.6 on page 33). Decoding times $td_n(j+1)$, $td_n(j+2)$, ... of access units without encoded DTS or PTS fields which directly follow access unit j may be derived from information in the elementary stream. Refer to Annex C of ITU-T Rec. H.262 | ISO/IEC 13818-2, ISO/IEC 13818-3, or ISO/IEC 11172. Also refer to 2.7.5 on page 81. In the case of audio all PES packet headers that are stored immediately before the access unit or that are embedded within the data of the access unit are removed simultaneously with the removal of the access unit. As the access unit is removed it is instantaneously decoded to a presentation unit.

System data

In the case of system data, data is removed from the main buffer B_{sys} at a rate of R_{sys} whenever there is at least 1 byte available in buffer B_{sys}.

$$R_{sys} = \max\left[80\ 000 bits/s, transport\_rate(i) \times 8\ bits/byte \div 500\right] \quad (2\text{-}6)$$

Note - The intention of increasing R_{sys} in the case of high transport rates is to allow an increased data rate for the Program Specific Information.

Low delay

When the low_delay flag in the video sequence extension is set to '1' (6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2) the EB_n buffer may underflow. In this case when the T-STD elementary stream buffer EB_n is examined at the time specified by $td_n(j)$, the complete data for the access unit may not be present in the buffer EB_n. When this case arises, the buffer shall be re-examined at intervals of two field-periods until the data for the complete access unit is present in the buffer. At this time the entire access unit shall be removed from buffer EB_n instantaneously. Overflow of buffer EB_n shall not occur.

When in the low_delay_mode, the flag is set to '1', EB_n underflow is allowed to occur continuously without limit. The T-STD decoder shall remove access unit data from buffer EB_n at the earliest time consistent with the paragraph above and any DTS or PTS values encoded in the bit stream. Note that the decoder may be unable to re-establish correct decoding and display times as indicated by DTS and PTS until the EB_n buffer underflow situation ceases and a PTS or DTS is found in the bit stream.

Trick mode

When the DSM_trick_mode flag (2.4.3.6 on page 33) is set to '1' in the PES Packet header of a packet containing the start of a B-type video access unit and the trick_mode_control field is set to '001' (slow motion) or '010' (freeze frame), or '100' (slow reverse) the B picture access unit is not removed from the video data buffer EB_n until the last time of possibly multiple times that any field of the picture is decoded and presented. Repetition of the presentation of fields and pictures is defined in 2.4.3.8 on page 44 under slow motion, slow reverse, and field_id_cntrl. The access unit is removed instantaneously from EB_n at the indicated time, which is dependent on the value of rep_cntrl.

When the DSM_trick_mode flag is set to '1' in the PES packet header of a packet containing the first byte of a picture start code, trick_mode status becomes true when that picture start code in the PES packet is removed from the buffer. Trick mode status remains true until a PES packet header is received by the T-STD in which the DSM_trick_mode flag is set to '0' and the first byte of the picture start code after that PES packet header is removed from the buffer. When trick mode status is true, the buffer EB_n may underflow. All other constraints from normal streams are retained when trick mode status is true.

2.4.2.4 Decoding

Elementary streams buffered in B_1 through B_n and EB_1 through EB_n are decoded instantaneously by decoders D_1 through D_n and may be delayed in reorder buffers O_1 through O_n before being presented to the viewer at the output of the T-STD. Reorder buffers are used only in the case of a video elementary stream when some access units are not carried in presentation order. These access units will need to be reordered before presentation. In particular, if $P_n(k)$ is an I-picture or a P-picture carried before one or more B-pictures, then it must be delayed in the reorder buffer, O_n, of the T-STD before being presented. Any picture previously stored in O_n is presented before the current picture can be stored. $P_n(k)$ should be delayed until the next I-picture or P-picture is decoded. While it is stored in the reorder buffer, the subsequent B-pictures are decoded and presented.

The time at which a presentation unit $P_n(k)$ is presented to the viewer is $tp_n(k)$. For presentation units that do not require reordering delay, $tp_n(k)$ is equal to $td_n(j)$ since the access units are decoded instantaneously; this is the case, for example, for B-frames. For presentation units that are delayed, $tp_n(k)$ and $td_n(j)$ differ by the time that $P_n(k)$ is delayed in the reorder buffer, which is a multiple of the nominal picture period. Care should be taken to use adequate re-ordering delay from the beginning of video elementary streams to meet the requirements of the entire stream. For example, a stream which initially has only I- and P-pictures but later includes B-pictures should include re-ordering delay starting at the beginning of the stream.

Part 2 of this Recommendation | International Standard explains reordering of video pictures in greater detail.

2.4.2.5 Presentation

The function of a decoding system is to reconstruct presentation units from compressed data and to present them in a synchronized sequence at the correct presentation times. Although real audio and visual presentation devices generally have finite and different delays and may have additional delays imposed by post-processing or output functions, the system target decoder models these delays as zero.

In the T-STD in figure 2-6 on page 12 the display of a video presentation unit (a picture) occurs instantaneously at its presentation time, $tp_n(k)$.

In the T-STD the output of an audio presentation unit starts at its presentation time, $tp_n(k)$, when the decoder instantaneously presents the first sample. Subsequent samples in the presentation unit are presented in sequence at the audio sampling rate.

2.4.2.6 Buffer management

Transport Streams shall be constructed so that conditions defined in this section are satisfied. This section makes use of the notation defined for the System Target Decoder.

TB_n and TB_{sys} shall not overflow. TB_n and TB_{sys} shall empty at least once every second. B_n shall not overflow nor underflow.

EB_n may not underflow except when the low delay flag in the video sequence extension is set to '1' (6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2) or trick_mode status is true.

When the leak method for specifying transfers is in effect, MB_n shall not overflow, and shall become empty at least once every second.

When the vbv_delay method for specifying transfers is in effect, MB_n shall not overflow nor underflow, and EB_n shall not overflow.

The delay of any data through the System Target Decoders buffers shall be less than or equal to one second except for still picture video data. Specifically: $td_n(j)-t(i) <= 1$ second for all j, and all bytes i in access unit $A_n(j)$.

For still picture video data, the delay is constrained by $td_n(j)-t(i) <= 60$ second for all j, and all bytes I in access unit $A_n(j)$.

$$0 \le F_n(t) \le BS_n \qquad \text{for all t and n}$$

and $F_n(t) = 0$ instantaneously before t=t(0).

$F_n(t)$ is the instantaneous fullness of T-STD buffer B_n.

2.4.3 Specification of the Transport Stream syntax and semantics

The following syntax describes a stream of bytes. Transport Stream packets shall be 188 bytes long.

2.4.3.1 Transport Stream

Table 2-2 -- Transport Stream

Syntax	No. of bits	Mnemonic
MPEG_transport_stream() {		
do {		
transport_packet()		
} while (nextbits() == sync_byte)		
}		

2.4.3.2 Transport Stream packet layer

Table 2-3 -- ITU-T Rec. H.222.0 | ISO/IEC 13818 transport packet

Syntax	No. of bits	Mnemonic
transport_packet(){		
sync_byte	**8**	**bslbf**
transport_error_indicator	**1**	**bslbf**
payload_unit_start_indicator	**1**	**bslbf**
transport_priority	**1**	**bslbf**
PID	**13**	**uimsbf**
transport_scrambling_control	**2**	**bslbf**
adaptation_field_control	**2**	**bslbf**
continuity_counter	**4**	**uimsbf**
if(adaptation_field_control=='10' || adaptation_field_control=='11'){		
adaptation_field()		
}		
if(adaptation_field_control=='01' || adaptation_field_control=='11') {		
for (i=0;i<N;i++){		
data_byte	**8**	**bslbf**
}		
}		
}		

2.4.3.3 Semantic definition of fields in Transport Stream packet layer

sync_byte -- The sync_byte is a fixed 8 bit field whose value is '0100 0111' (0x47). Sync_byte emulation in the choice of values for other regularly occuring fields, such as PID, should be avoided.

transport_error_indicator -- The transport_error_indicator is a 1 bit flag. When set to '1' it indicates that at least 1 uncorrectable bit error exists in the associated Transport Stream packet. This bit may be set to '1' by entities external to the transport layer. When set to '1' this bit shall not be reset to '0' unless the bit value(s) in error have been corrected.

payload_unit_start_indicator -- The payload_unit_start_indicator is a 1 bit flag which has normative meaning for Transport Stream packets that carry PES packets or PSI data.

When the payload of the Transport Stream packet contains PES packet data, the payload_unit_start_indicator has the following significance: A '1' indicates that the payload of this Transport Stream packet will commence

with the first byte of a PES packet and a '0' indicates no PES packet shall start in this Transport Stream packet. If the payload_unit_start_indicator is set to '1' then one and only one PES packet starts in this Transport Stream Packet. This also applies to private streams of stream_type 6 (refer to table 2-36 on page 64).

When the payload of the Transport Stream packet contains PSI data, the payload_unit_start_indicator has the following significance: If the Transport Stream packet carries the first byte of a PSI section, the payload_unit_start_indicator value shall be '1', indicating that the first byte of the payload of this Transport Stream packet carries the pointer_field. If the Transport Stream packet does not carry the first byte of a PSI section, the payload_unit_start_indicator value shall be '0', indicating that there is no pointer_field in the payload. Refer to 2.4.4.1 and 2.4.4.2 on page 46. This also applies to private streams of stream_type 5 (refer to table 2-36 on page 64).

For null packets the payload_unit_start_indicator shall be set to '0'.

The meaning of this bit for Transport Stream packets carrying only private data is not defined in this Recommendation | International Standard.

transport_priority -- The transport_priority is a 1 bit indicator. When set to '1' it indicates that the associated packet is of greater priority than other packets having the same PID which do not have the bit set to '1'. The transport mechanism can use this to prioritize its data within an elementary stream. Depending on the application the transport_priority field may be coded regardless of the PID or within one PID only. This field may be changed by channel specific encoders or decoders.

PID -- The PID is a 13 bit field, indicating the type of the data stored in the packet payload. PID value 0x0000 is reserved for the Program Association Table (table 2-26 on page 47). PID value 0x0001 is reserved for the Conditional Access Table (table 2-28 on page 49). PID values 0x0002-0x000F are reserved. PID value 0x1FFF is reserved for null packets.

Table 2-4 -- PID table

value	description
0x0000	Program Association Table
0x0001	Conditional Access Table
0x0002-0x000F	reserved
0x00010 … 0x1FFE	may be assigned as network_PID, Program_map_PID, elementary_PID, or for other purposes.
0x1FFF	Null packet

Note - The transport packets with PID values 0x0000, 0x0001, and 0x0010-0x1FFE are allowed to carry a PCR.

transport_scrambling_control -- This 2 bit field indicates the scrambling mode of the Transport Stream packet payload. The Transport Stream packet header, and the adaptation field when present, shall not be scrambled. In the case of a null packet the value of the transport_scrambling_control field shall be set to '00'.

Table 2-5 -- Scrambling control values

value	**description**
00	not scrambled
01	user defined
10	user defined

11	user defined

adaptation_field_control -- This 2 bit field indicates whether this Transport Stream packet header is followed by an adaptation field and/or payload.

Table 2-6 -- Adaptation field control values

value	description
00	reserved for future use by ISO/IEC
01	no adaptation_field, payload only
10	adaptation_field only, no payload
11	adaptation_field followed by payload

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoders shall discard Transport Stream packets with the adaptation_field_control field set to a value of '00'. In the case of a null packet the value of the adaptation_field_control shall be set to '01'.

continuity_counter -- The continuity_counter is a 4 bit field incrementing with each Transport Stream packet with the same PID. The continuity_counter wraps around to 0 after its maximum value. The continuity_counter shall not be incremented when the adaptation_field_control of the packet equals '00' or '10'.

In Transport Streams, duplicate packets may be sent as two, and only two, consecutive Transport Stream packets of the same PID. The duplicate packets shall have the same continuity_counter value as the original packet and the adaptation_field_control field shall be equal to '01' or '11'. In duplicate packets each byte of the original packet shall be duplicated, with the exception that in the program clock reference fields, if present, a valid value shall be encoded.

The continuity_counter in a particular Transport Stream packet is continuous when it differs by one increment from the continuity_counter value in the previous Transport Stream packet of the same PID, or when either of the non-incrementing conditions (adaptation_field_control set to '00' or '10', or duplicate packets as described above) are met. The continuity counter may be discontinuous when the discontinuity_indicator is set to '1' (refer to 2.4.3.4). In the case of a null packet the value of the continuity_counter is undefined.

data_byte -- Data bytes shall be contiguous bytes of data from the PES packets(refer to 2.4.3.6 on page 33), PSI sections(refer to 2.4.4 on page 44), packet stuffing bytes after PSI sections, or private data not in these structures as indicated by the PID. In the case of null packets with PID value 0x1FFF, data_bytes may be assigned any value. The number of data_bytes, N, is specified by 184 minus the number of bytes in the adaptation_field(), as described in 2.4.3.4 below.

2.4.3.4 Adaptation field

Table 2-7 -- Transport Stream adaptation field

Syntax	No. of Bits	Mnemonic
adaptation_field() {		
adaptation_field_length	**8**	**uimsbf**
if(adaptation_field_length >0) {		
discontinuity_indicator	**1**	**bslbf**
random_access_indicator	**1**	**bslbf**
elementary_stream_priority_indicator	**1**	**bslbf**
PCR_flag	**1**	**bslbf**
OPCR_flag	**1**	**bslbf**
splicing_point_flag	**1**	**bslbf**

Syntax	No. of Bits	Mnemonic
transport_private_data_flag	**1**	**bslbf**
adaptation_field_extension_flag	**1**	**bslbf**
if(PCR_flag == '1') {		
program_clock_reference_base	**33**	**uimsbf**
reserved	**6**	**bslbf**
program_clock_reference_extension	**9**	**uimsbf**
}		
if(OPCR_flag == '1') {		
original_program_clock_reference_base	**33**	**uimsbf**
reserved	**6**	**bslbf**
original_program_clock_reference_extension	**9**	**uimsbf**
}		
if (splicing_point_flag == '1') {		
splice_countdown	**8**	**tcimsbf**
}		
if(transport_private_data_flag == '1') {		
transport_private_data_length	**8**	**uimsbf**
for (i=0; i<transport_private_data_length;i++){		
private_data_byte	**8**	**bslbf**
}		
}		
if (adaptation_field_extension_flag == '1') {		
adaptation_field_extension_length	**8**	**uimsbf**
ltw_flag	**1**	**bslbf**
piecewise_rate_flag	**1**	**bslbf**
seamless_splice_flag	**1**	**bslbf**
reserved	**5**	**bslbf**
if (ltw_flag == '1') {		
ltw_valid_flag	**1**	**bslbf**
ltw_offset	**15**	**uimsbf**
}		
if (piecewise_rate_flag == '1') {		
reserved	**2**	**bslbf**
piecewise_rate	**22**	**uimsbf**
}		
if (seamless_splice_flag == '1'){		
splice_type	**4**	**bslbf**
DTS_next_AU[32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
DTS_next_AU[29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
DTS_next_AU[14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
for (i=0;i<N;i++) {		
reserved	**8**	**bslbf**
}		
}		
for (i=0;i<N;i++){		
stuffing_byte	**8**	**bslbf**
}		
}		
}		

2.4.3.5 Semantic definition of fields in adaptation field

adaptation_field_length -- The adaptation_field_length is an 8 bit field specifying the number of bytes in the adaptation_field immediately following the adaptation_field_length. The value 0 is for inserting a single stuffing byte in a Transport Stream packet. When the adaptation_field_control value is '11', the value of the adaptation_field_length shall be in the range 0 to 182. When the adaptation_field_control value is '10', the value of the adaptation_field_length shall be 183. For Transport Stream packets carrying PES packets, stuffing is needed when there is insufficient PES packet data to completely fill the Transport Stream packet payload bytes. Stuffing is accomplished by defining an adaptation field longer than the sum of the lengths of the data elements in it, so that the payload bytes remaining after the adaptation field exactly accommodates the available PES packet data. The extra space in the adaptation field is filled with stuffing bytes.

This is the only method of stuffing allowed for Transport Stream packets carrying PES packets. For Transport Stream packets carrying PSI, an alternative stuffing method is described in 2.4.4 on page 44.

discontinuity_indicator -- This is a 1 bit field which when set to '1' indicates that the discontinuity state is true for the current Transport Stream packet. When the discontinuity_indicator is set to '0' or is not present, the discontinuity state is false. The discontinuity indicator is used to indicate two types of discontinuities, system time-base discontinuities and continuity_counter discontinuities.

A system time-base discontinuity is indicated by the use of the discontinuity_indicator in Transport Stream packets of a PID designated as a PCR_PID (refer to 2.4.4.9 on page 50). When the discontinuity state is true for a Transport Stream packet of a PID designated as a PCR_PID, the next PCR in a Transport Stream packet with that same PID represents a sample of a new system time clock for the associated program. The system time-base discontinuity point is defined to be the instant in time when the first byte of a packet containing a PCR of a new system time-base arrives at the input of the T-STD. The discontinuity_indicator shall be set to '1' in the packet in which the system time-base discontinuity occurs. The discontinuity_indicator bit may also be set to '1' in Transport Stream packets of the same PCR_PID prior to the packet which contains the new system time-base PCR. In this case, once the discontinuity_indicator has been set to '1', it shall continue to be set to '1' in all Transport Stream packets of the same PCR_PID up to and including the Transport Stream packet which contains the first PCR of the new system time-base. After the occurrence of a system time-base discontinuity, no fewer than two PCRs for the new system time-base shall be received before another system time-base discontinuity can occur. Further, except when trick mode status is true, data from no more than two system time-bases shall be present in the set of T-STD buffers for one program at any time.

Prior to the occurrence of a system time-base discontinuity, the first byte of a Transport Stream packet which contains a PTS or DTS which refers to the new system time-base shall not arrive at the input of the T-STD. After the occurrence of a system time-base discontinuity, the first byte of a Transport Stream packet which contains a PTS or DTS which refers to the previous system time-base shall not arrive at the input of the T-STD.

A continuity_counter discontinuity is indicated by the use of the discontinuity_indicator in any Transport Stream packet. When the discontinuity state is true in any Transport Stream packet of a PID not designated as a PCR_PID, the continuity_counter in that packet may be discontinuous with respect to the previous Transport Stream packet of the same PID. When the discontinuity state is true in a Transport Stream packet of a PID that is designated as a PCR_PID, the continuity_counter may only be discontinuous in the packet in which a system time-base discontinuity occurs. A continuity counter discontinuity point occurs when the discontinuity state is true in a Transport Stream packet and the continuity_counter in the same packet is discontinuous with respect to the previous Transport Stream packet of the same PID. A continuity counter discontinuity point shall occur at most one time from the initiation of the discontinuity state until the conclusion of the discontinuity state. Furthermore, for all PIDs that are not designated as PCR_PIDs, when the discontinuity_indicator is set to '1' in a packet of a specific PID, the discontinuity_indicator may be set to '1' in the next Transport Stream packet of that same PID, but shall not be set to '1' in three consecutive Transport Stream packet of that same PID.

For the purpose of this clause, an elementary stream access point is defined as follows:

video: the first byte of a video sequence header
audio: the first byte of an audio frame.

After a continuity counter discontinuity in a Transport packet which is designated as containing elementary stream data, the first byte of elementary stream data in a Transport Stream packet of the same PID shall be the first byte of an elementary stream access point or in the case of video, the first byte of an elementary stream access point or a sequence_end_code followed by an access point. Each Transport Stream packet which contains elementary stream data with a PID not designated as a PCR_PID, and in which a continuity counter discontinuity point occurs, and in which a PTS or DTS occurs, shall arrive at the input of the T-STD after the system time-base discontinuity for the associated program occurs.
In the case where the discontinuity state is true, if two consecutive Transport Stream packets of the same PID occur which have the same continuity_counter value and have adaptation_field_control values set to '01' or '11', the second packet may be discarded. A Transport Stream shall not be constructed in such a way that discarding such a packet will cause the loss of PES packet payload data or PSI data.

After the occurrence of a discontinuity_indicator set to '1' in a Transport Stream packet which contains PSI information, a single discontinuity in the version_number of PSI sections may occur. At the occurrence of such a discontinuity, a version of the TS_program_map_sections of the appropriate program shall be sent with section_length == 13 and the current_next_indicator == 1, such that there are no program_descriptors and no elementary streams described. This shall then be followed by a version of the TS_program_map_section for each affected program with the version_number incremented by one and the current_next_indicator == 1, containing a complete program definition. This indicates a version change in PSI data.

random_access_indicator -- The random_access_indicator is a 1 bit field that indicates the current and subsequent transport packets with the same PID contain some information to aid random access at this point. Specifically, when the bit is set to '1', the next PES packet to start in the payload of transport packets (refer to table 2-36 on page 64) with the current PID shall contain the first byte of a video sequence header if the PES stream type is 1 or 2(video) or shall contain the first byte of an audio frame if the PES stream type is 3 or 4(audio). In addition, in these cases a presentation timestamp shall be present in this or a subsequent PES packet for the first picture following the sequence header or for the audio frame. When the current PID is equal to PCR_PID, the PCR_flag shall be set to '1'.

elementary_stream_priority_indicator -- The elementary_stream_priority_indicator is a one bit field. It indicates, among packets with the same PID, the priority of the elementary stream data carried within the payload of this Transport Stream packet. A '1' indicates that the payload has a higher priority than the payloads of other Transport Stream packets. In case of video this field may be set to '1' only if the payload contains one or more bytes from an intra-coded slice. A value of '0' indicates that the payload has the same priority as all other packets which do not have this bit set to '1'.

PCR_flag -- The PCR_flag is a 1 bit flag. A value of '1' indicates that the adaptation_field contains a PCR field coded in two parts. A value of '0' indicates that the adaptation field does not contain any PCR field.

OPCR_flag -- The OPCR_flag is a 1 bit flag. A value of '1' indicates that the adaptation_field contains an OPCR field coded in 2 parts. A value of '0' indicates that the adaptation field does not contain any OPCR fields.

splicing_point_flag -- The splicing_point_flag is a 1 bit flag. When set to '1', it indicates that a splice_countdown field shall be present in the associated adaptation field, specifying the occurence of a splicing point. A value of '0' indicates that a splice_countdown field is not present in the adaptation field.

transport_private_data_flag -- The transport_private_data_flag is a 1 bit flag. A value of '1' indicates that the adaptation field contains one or more private_data bytes. A value of '0' indicates the adaptation field does not contain any private_data bytes.

adaptation_field_extension_flag -- The adaptation_field_extension_flag is a 1 bit field which when set to '1' indicates the presence of an adaptation field extension. A value of '0' indicates that an adaptation field extension is not present in the adaptation field.

program_clock_reference_base; program_clock_reference_extension -- The program_clock_reference (PCR) is a 42 bit field coded in two parts. The first part, program_clock_reference_base, is a 33 bit field whose value is given by PCR_base(i), as given in equation 2-1 on page 14 . The second part, program_clock_reference_extension, is a 9 bit field whose value is given by PCR_ext(i), as given in equation 2-2 on page 14 . The PCR indicates the intended time of arrival of the byte containing the last bit of the program_clock_reference_base at the input of the system target decoder.

For Transport Stream packets containing video or audio elementary streams, If a PCR field is present in a Transport Strea, packet containing data from and audio or video elementary stream, it shall be valid fo that elementary stream. Clause the adaptation field, that PCR field must be valid for the elementary stream contained in its Transport Stream packet. Refer to 2.7.2 on page 81 for frequency of coding requirements.

original_program_clock_reference_base; original_program_clock_reference_extension -- The optional original program reference (OPCR) is a 42-bit field coded in two parts. These two parts, the base and the extension, are coded identically to the two corresponding parts of the PCR field. The presence of the OPCR is indicated by the OPCR_flag. The OPCR field shall be coded only in Transport Stream packets in which the PCR field is present. OPCRs are permitted in both single program and multiple program Transport Streams.

OPCR assists in the reconstruction of a single program Transport Stream from another Transport Stream. When reconstructing the original single program Transport Stream, the OPCR may be copied to the PCR field. The resulting PCR value is valid only if the original single program Transport Stream is reconstructed exactly in its entirety. This would include at least any PSI and private data packets which were present in the original Transport Stream and would possibly require other private arrangements. It also means that the OPCR must be an identical copy of its associated PCR in the original single program Transport Stream.

The OPCR is expressed as follows:

$$OPCR\_base(i) = \left(\left(system\_clock\_frequency \times t(i) \right) DIV\ 300 \right) \% \ 2^{33} \tag{2-7}$$

$$OPCR\_ext(i) = \left(\left(system\_clock\_frequency \times t(i) \right) DIV\ 1 \right) \% \ 300 \tag{2-8}$$

$$OPCR(i) = OPCR\_base(i) \times 300 + OPCR\_ext(i) \tag{2-9}$$

The OPCR field is ignored by the decoder. The OPCR field shall not be modified by any multiplexor or decoder.

splice_countdown -- The splice_countdown is an 8 bit field, representing a value which may be positive or negative. A positive value specifies the remaining number of Transport Stream packets, of the same PID, following the associated Transport Stream packet until a splicing point is reached. Duplicate Transport Stream packets and Transport Stream packets which only contain adaptation fields are excluded. The splicing point occurs immediately after the transfer from buffer TB_n in the STD model of the Transport Stream packet in which the splice_countdown reaches the value zero. In the Transport Stream packet where the splice_countdown reaches zero, the last byte of the Transport Stream packet payload shall be the last byte of a coded audio frame or a coded picture. In the case of video, the corresponding access unit may or may not be terminated by a sequence_end_code. Transport Stream packets with the same PID, which follow, may contain data from a different elementary stream of the same type.

The payload of the next Transport Stream packet of the same PID (duplicate packets and packets without payload being excluded) shall commence with the first byte of a PES packet. In the case of audio, the PES packet payload shall commence with an access point. In the case of video, the PES packet payload shall commence with an access point, or with a sequence_end_code, followed by an access point. Thus, the previous coded audio frame or coded picture aligns with the packet boundary, or is padded to make this so. Subsequent to the splicing point, the countdown field may also be present. When the splice_countdown is a negative number whose value is minus n (-n), it indicates that the associated Transport Stream packet is the nth packet following the splicing point (duplicate packets and packets without payload being excluded).

For the purposes of this clause, an access point is defined as follows:

video: the first byte of a video_sequence_header
audio: the first byte of an audio frame

transport_private_data_length -- The transport_private_data_length is an 8 bit field specifying the number of private_data bytes immediately following the transport private_data_length field. The number of private_data bytes shall not be such that private data extends beyond the adaptation field.

private_data_byte -- The private_data_byte is an 8 bit field that shall not be specified by ISO/IEC.

adaptation_field_extension_length -- The adaptation_field_extension_length is an 8 bit field. It indicates the number of bytes of the extended adaptation field data following the end of this field, including reserved bytes if present.

ltw_flag (legal time window_flag) -- This is a 1 bit field which when set to '1' indicates the presence of the ltw_offset field.

piecewise_rate_flag -- This is a 1 bit field which when set to '1' indicates the presence of the piecewise_rate field.

seamless_splice_flag -- This is a one bit flag which when set to '1' indicates that the splice_type and DTS_next_AU fields are present. A value of '0' indicates that neither splice_type nor DTS_next_AU fields are present. This field shall not be set to '1' in Transport Stream packets in which the splicing_point_flag is not set to '1'. Once it is set to '1' in a Transport Stream packet in which the splice_countdown is positive, it shall be set to '1' in all the subsequent Transport Stream packets of the same PID that have the splicing_point_flag set to ‘1’, until the packet in which the splice_countdown reaches zero (including this packet). When this flag is set, if the elementary stream carried in this PID is an audio stream, the splice_type field shall be set to 0000. If the elementary stream carried in this PID is a video stream, it shall fulfill the constraints indicated by the splice_type value.

ltw_valid_flag(legal time window_valid_flag) -- This is a 1 bit field which when set '1' indicates that the value of the ltw_offset shall be valid. A value of '0' indicates that the value in the ltw_offset field is undefined.

ltw_offset (legal time window_offset) -- This is a 15 bit field, the value of which is defined only if the ltw_valid flag has a value of ‘1’. When defined, the legal_time_window_offset is in units of $(300/f_s)$ seconds, where f_s is the system clock frequency of the program that this PID belongs to, and fulfills:

$$t_1(i) - t(i) = offset,$$

where i is the index of the first byte of this Transport Stream packet, offset is the value encoded in this field, t(i) is the arrival time of byte i in the T-STD, and $t_1(i)$ is the upper bound in time of a time period called the Legal Time Window which is associated with this Transport Stream packet.

The Legal Time Window has the property that if this Transport Stream is delivered to a T-STD starting at time $t_1(i)$, i.e. at the end of its Legal Time Window, and all other Transport Stream packets of the same program are delivered at the end of their Legal Time Windows, then

for video: the MB_n buffer for this PID in the T-STD will contain less than 184 bytes of elementary stream data at the time the first byte of the payload of this Transport Stream packet enters it, and no buffer violations in the T-STD will occur.

for audio: the B_n buffer for this PID in the T-STD will contain less than BS_{dec} + 1 bytes of elementary stream data at the time the first byte of this Transport Stream packet enters it, and no buffer violations in the T-STD will occur.

Depending on factors including the size of the buffer MB_n and the rate of data transfer between MB_n and EB_n, it is possible to determine another time $t_0(i)$, such that if this packet is delivered anywhere in the interval $[t_0(i), t_1(i)]$, no T-STD buffer violations will occur. This time period is called the Legal Time Window. The value of t_0 is not defined in this Recommendation | International Standard.

The information in this field is intended for devices such as remultiplexers which may need this information in order to reconstruct the state of the buffers MB_n.

piecewise_rate -- The meaning of this 22 bit field is only defined when both the ltw_flag and the ltw_valid_flag are set to '1'. When defined, it is a positive integer specifying a hypothetical bitrate R which is used to define the end times of the Legal Time Windows of Transport Stream packets of the same PID that follow this packet but do not include the legal_time_window_offset field.

Assume that the first byte of this Transport Stream packet and the N following Transport Stream packets of the same PID have indices A_i, A_{i+1}, ..., A_{i+N}, respectively, and that the N latter packets do not have a value encoded in the field legal_time_window_offset. Then the values $t_1(A_{i+j})$ shall be determined by

$$t_1(A_{i+j}) = t_1(A_i) + j*188*8 \text{ bits/byte } /R,$$

where j goes from 1 to N.

All packets between this packet and the next packet of the same PID to include a legal_time_window_offset field shall be treated as if they had the value

$$\text{offset} = t_1(A_i) - t(A_i),$$

corresponding to the value $t_1(.)$ as computed by the formula above encoded in the legal_time_window_offset field. Note: t(j) is the arrival time of byte j in the T-STD.

The meaning of this field is not defined when it is present in a Transport Stream packet with no legal_time_window_offset field.

splice_type -- This is a 4 bit field. From the first occurrence of this field onwards, it shall have the same value in all the subsequent Transport Stream packets of the same PID in which it is present, until the packet in which the splice_countdown reaches zero (including this packet). If the elementary stream carried in that PID is an audio stream, this field shall have the value 0000. If the elementary stream carried in that PID is a video stream, this field indicates the conditions that shall be respected by this elementary stream for splicing purposes. These conditions are defined as a function of profile, level and splice_type in table 2-8 through table 2-17 on page 32 below.

In these tables, a value for 'splice_decoding_delay' and 'max_splice_rate' means that the following conditions shall be satisfied by the video elementary stream:

1. The last byte of the coded picture ending in the Transport Stream packet in which the splice_countdown reaches zero shall remain in the VBV buffer of the VBV model for an amount of time equal to (splice_decoding_delay - $Dp_n(k)$), where for the purpose of this clause:

- $DP_n(k)$ is the duration for which $P_n(k)$ is presented.
- K is such that $tp_n(k) = td_n(j)$, where j is such that $A_n(j)$ contains the coded picture ending in the Transport Stream packet in which the splice_countdown reaches zero, i.e. the coded picture referred to above.

2. The VBV buffer of the VBV model shall not overflow if its input is switched at the splicing point to a stream of a constant rate equal to 'max_splice_rate' for an amount of time equal to 'splice_decoding_delay'.

Table 2-8 -- Splice parameters table 1

Simple Profile Main Level, Main Profile Main Level, SNR Profile Main Level (both layers), Spatial Profile High-1440 Level (base layer), High Profile Main Level (middle + base layers)

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 15.0 x 10^6 bit/s
0001	splice_decoding_delay = 150 ms ; max_splice_rate = 12.0 x 10^6 bit/s
0010	splice_decoding_delay = 225 ms ; max_splice_rate = 8.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 7.2 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-9 -- Splice parameters table 2

Main Profile Low Level, SNR Profile Low Level (both layers), High Profile MainLevel (base layer) Video

splice_type	conditions
0000	splice_decoding_delay = 115 ms ; max_splice_rate = 4.0 x 10^6 bit/s
0001	splice_decoding_delay = 155 ms ; max_splice_rate = 3.0 x 10^6 bit/s
0010	splice_decoding_delay = 230 ms ; max_splice_rate = 2.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 1.8 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-10 -- Splice parameters table 3

Main Profile High-1440 Level, Spatial Profile High-1440 Level (all layers), High Profile High-1440 Level (middle + base layers) Video

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 60.0 x 10^6 bit/s
0001	splice_decoding_delay = 160 ms ; max_splice_rate = 45.0 x 10^6 bit/s
0010	splice_decoding_delay = 240 ms ; max_splice_rate = 30.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 28.5 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-11 --Splice parameters table 4

Main Profile High Level, High Profile High-1440 Level (all layers), High Profile High Level (middle + base layers) Video

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 80.0 x 10^6 bit/s
0001	splice_decoding_delay = 160 ms ; max_splice_rate = 60.0 x 10^6 bit/s
0010	splice_decoding_delay = 240 ms ; max_splice_rate = 40.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 38.0 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-12 -- Splice parameters table 5

SNR Profile Low Level (base layer) Video

splice_type	conditions
0000	splice_decoding_delay = 115 ms ; max_splice_rate = 3.0 x 10^6 bit/s
0001	splice_decoding_delay = 175 ms ; max_splice_rate = 2.0 x 10^6 bit/s
0010	splice_decoding_delay = 250 ms ; max_splice_rate = 1.4 x 10^6 bit/s
0011-1011	reserved
1100-1111	user-defined

Table 2-13 -- Splice parameters table 6

SNR Profile Main Level (base layer) Video

splice_type	conditions
0000	splice_decoding_delay = 115 ms ; max_splice_rate = 10.0 x 10^6 bit/s
0001	splice_decoding_delay = 145 ms ; max_splice_rate = 8.0 x 10^6 bit/s
0010	splice_decoding_delay = 235 ms ; max_splice_rate = 5.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 4.7 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-14 -- Splice parameters table 7

Spatial Profile High-1440 Level (middle + base layers) Video

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 40.0 x 10^6 bit/s
0001	splice_decoding_delay = 160 ms ; max_splice_rate = 30.0 x 10^6 bit/s
0010	splice_decoding_delay = 240 ms ; max_splice_rate = 20.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 19.0 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-15 -- Splice parameters table 8

High Profile Main Level (all layers), High Profile High-1440 Level (base layer) Video

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 20.0 x 10^6 bit/s
0001	splice_decoding_delay = 160 ms ; max_splice_rate = 15.0 x 10^6 bit/s
0010	splice_decoding_delay = 240 ms ; max_splice_rate = 10.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 9.5 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

Table 2-16 -- Splice parameters table 9

High Profile High Level (base layer) Video

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 25.0 x 10^6 bit/s
0001	splice_decoding_delay = 165 ms ; max_splice_rate = 18.0 x 10^6 bit/s
0010	splice_decoding_delay = 250 ms ; max_splice_rate = 12.0 x 10^6 bit/s
0011-1011	reserved
1100-1111	user-defined

Table 2-17 -- Splice parameters table 10

High Profile High Level (all layers) Video

splice_type	conditions
0000	splice_decoding_delay = 120 ms ; max_splice_rate = 100.0 x 10^6 bit/s
0001	splice_decoding_delay = 160 ms ; max_splice_rate = 75.0 x 10^6 bit/s
0010	splice_decoding_delay = 240 ms ; max_splice_rate = 50.0 x 10^6 bit/s
0011	splice_decoding_delay = 250 ms ; max_splice_rate = 48.0 x 10^6 bit/s
0100-1011	reserved
1100-1111	user-defined

DTS_next_AU (decoding time stamp next access unit) **--** This is a 33 bit field, coded in three parts. It indicates the decoding time of the first access unit following the splicing point, in the case where no splicing (or seamless splicing) has been performed on this splicing point. This decoding time is expressed in the time base which is valid in the Transport Stream Packet in which the splice_countdown reaches zero. From the first occurrence of this field onwards, it shall have the same value in all the subsequent Transport Stream packets of the same PID in which it is present, until the packet in which the splice_countdown reaches zero (including this packet).

stuffing_byte -- This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder. It is discarded by the decoder.

2.4.3.6 PES packet

Table 2-18 -- PES packet

Syntax	No. of Bits	Mnemonic
PES_packet() {		
packet_start_code_prefix	**24**	**bslbf**
stream_id	**8**	**uimsbf**
PES_packet_length	**16**	**uimsbf**

Syntax	No. of Bits	Mnemonic
if(stream_id != program_stream_map		
&& stream_id != padding_stream		
&& stream_id != private_stream_2		
&& stream_id != ECM		
&& stream_id != EMM		
&& stream_id != program_stream_directory		
&& stream_id != DSMCC_stream		
&& stream_id != ITU-T Rec. H.222.1 type E_stream) {		
'10'	**2**	**bslbf**
PES_scrambling_control	**2**	**bslbf**
PES_priority	**1**	**bslbf**
data_alignment_indicator	**1**	**bslbf**
copyright	**1**	**bslbf**
original_or_copy	**1**	**bslbf**
PTS_DTS_flags	**2**	**bslbf**
ESCR_flag	**1**	**bslbf**
ES_rate_flag	**1**	**bslbf**
DSM_trick_mode_flag	**1**	**bslbf**
additional_copy_info_flag	**1**	**bslbf**
PES_CRC_flag	**1**	**bslbf**
PES_extension_flag	**1**	**bslbf**
PES_header_data_length	**8**	**uimsbf**
if (PTS_DTS_flags =='10') {		
'0010'	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
if (PTS_DTS_flags ==‘11’) {		
'0011'	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
'0001'	**4**	**bslbf**
DTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
DTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
DTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
if (ESCR_flag=='1') {		
reserved	**2**	**bslbf**
ESCR_base[32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
ESCR_base[29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
ESCR_base[14..0]	**15**	**bslbf**

Syntax	No. of Bits	Mnemonic
marker_bit	**1**	**bslbf**
ESCR_extension	**9**	**uimsbf**
marker_bit	**1**	**bslbf**
}		
if (ES_rate_flag == '1') {		
marker_bit	**1**	**bslbf**
ES_rate	**22**	**uimsbf**
marker_bit	**1**	**bslbf**
}		
if (DSM_trick_mode_flag == '1') {		
trick_mode_control	**3**	**uimsbf**
if (trick_mode_control == fast_forward) {		
field_id	**2**	**bslbf**
intra_slice_refresh	**1**	**bslbf**
frequency_truncation	**2**	**bslbf**
}		
else if (trick_mode_control == slow_motion) {		
rep_cntrl	**5**	**uimsbf**
}		
else if (trick_mode_control == freeze_frame) {		
field_id	**2**	**uimsbf**
reserved	**3**	**bslbf**
}		
else if (trick_mode_control == fast_reverse') {		
field_id	**2**	**bslbf**
intra_slice_refresh	**1**	**bslbf**
frequency_truncation	**2**	**bslbf**
else if (trick_mode_control == slow_reverse) {		
rep_cntrl	**5**	**uimsbf**
}		
else		
reserved	**5**	**bslbf**
}		
if (additional_copy_info_flag == '1') {		
marker_bit	**1**	**bslbf**
additional_copy_info	**7**	**bslbf**
}		
if (PES_CRC_flag == '1') {		
previous_PES_packet_CRC	**16**	**bslbf**
}		
if (PES_extension_flag == '1') {		
PES_private_data_flag	**1**	**bslbf**
pack_header_field_flag	**1**	**bslbf**
program_packet_sequence_counter_flag	**1**	**bslbf**
P-STD_buffer_flag	**1**	**bslbf**
reserved	**3**	**bslbf**
PES_extension_flag_2	**1**	**bslbf**
if (PES_private_data_flag == '1') {		
PES_private_data	**128**	**bslbf**
}		
if (pack_header_field_flag == '1') {		
pack_field_length	**8**	**uimsbf**
pack_header()		
}		

Syntax	No. of Bits	Mnemonic
if(program_packet_sequence_counter_flag== '1'){		
marker_bit	**1**	**bslbf**
program_packet_sequence_counter	**7**	**uimsbf**
marker_bit	**1**	**bslbf**
MPEG1_MPEG2_identifier	**1**	**bslbf**
original_stuff_length	**6**	**uimsbf**
}		
if (P-STD_buffer_flag == '1') {		
'01'	**2**	**bslbf**
P-STD_buffer_scale	**1**	**bslbf**
P-STD_buffer_size	**13**	**uimsbf**
}		
if (PES_extension_flag_2 == '1'){		
marker_bit	**1**	**bslbf**
PES_extension_field_length	**7**	**uimsbf**
for(i=0;i<PES_extension_field_length;i++) {		
reserved	**8**	**bslbf**
}		
}		
}		
for (i=0;i<N1;i++) {		
stuffing_byte	**8**	**bslbf**
}		
for (i=0;i<N2;i++) {		
PES_packet_data_byte	**8**	**bslbf**
}		
}		
else if (stream_id == program_stream_map		
|| stream_id == private_stream_2		
|| stream_id == ECM		
|| stream_id == EMM		
|| stream_id == program_stream_directory		
|| stream_id == DSMCC_stream)		
|| stream_id == ITU-T Rec. H.222.1 type E stream {		
for (i=0;i<PES_packet_length;i++) {		
PES_packet_data_byte	**8**	**bslbf**
}		
}		
else if (stream_id == padding_stream) {		
for (i=0;i<PES_packet_length;i++) {		
padding_byte	**8**	**bslbf**
}		
}		
}		

2.4.3.7 Semantic definition of fields in PES packet

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).

stream_id -- In Program Streams, the stream_id specifies the type and number of the elementary stream as defined by the stream_id table 2-19. In Transport Streams, the stream_id may be set to any valid value which

correctly describes the elementary stream type as defined in table 2-19. In Transport Streams, the elementary stream type is specified in the Program Specific Information as specified in 2.4.4 on page 44.

Table 2-19 -- Stream_id assignments

stream_id	Note	stream coding
1011 1100	1	program_stream_map
1011 1101	2	private_stream_1
1011 1110		padding_stream
1011 1111	3	private_stream_2
110x xxxx		ISO/IEC 13818-3 or ISO/IEC 11172-3 audio stream number x xxxx
1110 xxxx		ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 video stream number xxxx
1111 0000	3	ECM_stream
1111 0001	3	EMM_stream
1111 0010	5	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A or ISO/IEC 13818-6_DSMCC_stream
1111 0011	2	ISO/IEC_13522_stream
1111 0100	6	ITU-T Rec. H.222.1 type A
1111 0101	6	ITU-T Rec. H.222.1 type B
1111 0110	6	ITU-T Rec. H.222.1 type C
1111 0111	6	ITU-T Rec. H.222.1 type D
1111 1000	6	ITU-T Rec. H.222.1 type E
1111 1001	7	ancillary_stream
1111 1010 … 1111 1110		reserved data stream
1111 1111	4	program_stream_directory

The notation x means that the value '0' or '1' are both permitted and results in the same stream type. The stream number is given by the values taken by the x's.

Note 1: PES packets of type program_stream_directory have unique syntax specified in 2.5.4.1 on page 63.

Note 2: PES packets of type private_stream_1 and ISO/IEC_13552_stream follow the same PES packet syntax as those for ITU-T Rec. H.222.0 | ISO/IEC 13818-2 video and ISO/IEC 13818-3 audio streams.

Note 3: PES packets of type private_stream_2, ECM_stream and EMM_stream are similar to private_stream_1 except no syntax is specified after PES_packet_length field.

Note 4: PES packets of type program_stream_directory have a unique syntax specified in.2.5.5.1 on page 65.

Note 5: PES packets of type DSM-CC_stream have a unique syntax specified in ISO/IEC 13818- 6, which is a compatible extension of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A.

Note 6: This stream_id is associated with stream_type 0x09 in table 2-36 on page 64.

Note 7. This stream_id is only used in PES packets, which carry data from a Program Stream or an ISO/IEC 11172-1 System Stream, in a Transport Stream.

PES_packet_length -- A 16 bit field specifying the number of bytes in the PES packet following the last byte of the field. A value of 0 indicates that the PES packet length is neither specified nor bounded and is allowed only in PES packets whose payload is a video elementary stream contained in Transport Stream packets.

PES_scrambling_control -- The 2 bit PES_scrambling_control field indicates the scrambling mode of the PES packet payload. When scrambling is performed at the PES level, the PES packet header, including the optional fields when present, shall not be scrambled.

Table 2-20 -- PES scrambling control values

Value	Description
00	not scrambled
01	user defined
10	user defined
11	user defined

PES_priority -- This field is a 1 bit field indicating the priority of the payload in this PES packet. A '1' indicates a higher priority of the payload of the PES packet payload than a PES packet payload with this field set to '0'. A multiplexor can use the PES_priority bit to prioritize its data within an elementary stream. This field shall not be changed by the transport mechanism.

data_alignment_indicator -- This is a 1 bit flag. When set to a value of '1' it indicates that the PES packet header is immediately followed by the video start code or audio syncword indicated in the data_stream_alignment_descriptor in table 2-47 on page 72. When set to a value of '0' it is not defined whether any such alignment occurs or not. If the descriptor is not present, alignment as indicated in alignment_type '01' in table 2-48 on page 72 and table 2-49 on page 73 is required.

copyright -- This is a 1 bit field. When set to '1' it indicates that the material of the associated PES packet payload is protected by copyright. When set to '0' it is not defined whether the material is protected by copyright. A copyright descriptor (refer to 2.6.24 on page 77) is associated with the elementary stream which contains this PES packet and the copyright flag is set to '1' the descriptor applies to the material contained in this PES packet.

original_or_copy -- If this bit is set to '1', the contents of the associated PES packet payload is an original. When set to '0' it indicates that it is a copy.

PTS_DTS_flags -- This is a 2 bit field. If the PTS_DTS_flags field equals '10', the PTS fields shall be present in the PES packet header. If the PTS_DTS_flags field equals '11', both the PTS fields and DTS fields shall be present in the PES packet header. If the PTS_DTS_flags field equals '00' no PTS or DTS fields shall be present in the PES packet header. The value '01' is forbidden.

ESCR_flag -- A 1 bit flag, which when set to '1' indicates that ESCR base and extension fields are present in the PES packet header. When set to a value of '0' it indicates that no ESCR fields are present.

ES_rate_flag -- A 1 bit flag, which when set to '1' indicates that the ES_rate field is present in the PES packet header. When set to a value of '0' it indicates that no ES_rate field is present.

DSM_trick_mode_flag -- A 1 bit flag, which when set to '1' it indicates the presence of an 8 bit trick mode field. When set to a value of '0' it indicates that this field is not present.

additional_copy_info_flag -- A 1 bit flag, which when set to '1' indicates the presence of the additional_copy_info field. When set to a value of '0' it indicates that this field is not present.

PES_CRC_flag -- A 1 bit flag, which when set to '1' indicates that a CRC field is present in the PES packet. When set to a value of '0' it indicates that this field is not present.

PES_extension_flag -- A 1 bit flag, which when set to '1' indicates that an extension field exists in this PES packet header. When set to a value of '0' it indicates that this field is not present.

PES_header_data_length -- An 8 bit field specifying the total number of bytes occupied by the optional fields and any stuffing bytes contained in this PES packet header. The presence of optional fields is indicated in the byte that precedes the PES_header_data_length field.

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

PTS (presentation time stamp) -- Presentation times shall be related to decoding times as follows:
The PTS is a 33 bit number coded in three separate fields. It indicates the time of presentation, $tp_n(k)$, in the system target decoder of a presentation unit k of elementary stream n. The value of PTS is specified in units of the period of the system clock frequency divided by 300 (yielding 90 kHz). The presentation time is derived from the PTS according to equation 2-10 below. Refer to 2.7.4 on page 81 for constraints on the frequency of coding presentation timestamps.

$$PTS(k) = \left(\left(system\_clock\_frequency \times tp_n(k) \right) DIV\ 300 \right) \% 2^{33} \qquad (2\text{-}10)$$

where

$tp_n(k)$ is the presentation time of presentation unit $P_n(k)$.

In the case of audio, if a PTS is present in PES packet header it shall refer to the first access unit commencing in the PES packet. An audio access unit commences in a PES packet if the first byte of the audio access unit is present in the PES packet

In the case of video, if a PTS is present in a PES packet header it shall refer to the access unit containing the first picture start code that commences in this PES packet. A picture start code commences in PES packet if the first byte of the picture start code is present in the PES packet.

For audio presentation units (PUs), video PUs in low_delay sequences, and B-pictures, the presentation time $tp_n(k)$ shall be equal to the decoding time $td_n(k)$.

For I and P pictures in non-low_delay sequences and in the case when there is no decoding discontinuity between access units (AUs) k and k', the presentation time $tp_n(k)$ shall be equal to the decoding time $td_n(k')$ of the next transmitted I or P picture (refer to 2.7.5 on page 81). If there is a decoding discontinuity, or the stream ends, the difference between $tp_n(k)$ and $td_n(k)$ shall be the same as if the original stream had continued without a discontinuity and without ending.

> **Note -** A low_delay sequence is a video sequence in which the low_delay flag is set (refer to 6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2).

If there is filtering in audio, it is assumed by the system model that filtering introduces no delay, hence the sample referred to by PTS at encoding is the same sample referred to by PTS at decoding. In the case of scalable coding refer to 2.7.6 on page 82.

DTS (decoding time stamp) -- The DTS is a 33 bit number coded in three separate fields. It indicates the decoding time, $td_n(j)$, in the system target decoder of an access unit j of elementary stream n. The value of DTS is specified in units of the period of the system clock frequency divided by 300 (yielding 90 kHz). The decoding time derived from the DTS according to equation 2-11 below.

$$DTS(j) = \left(\left(system\_clock\_frequency \times td_n(j) \right) DIV\ 300 \right) \% 2^{33} \qquad (2\text{-}11)$$

where

$td_n(j)$ is the decoding time of access unit $A_n(j)$.

In the case of video, if a DTS is present in a PES packet header it shall refer to the access unit containing the first picture start code that commences in this PES packet. A picture start code commences in PES packet if the first byte of the picture start code is present in the PES packet.

In the case of scalable coding refer to 2.7.6 on page 82.

ESCR_base; ESCR_extension --The elementary stream clock reference is a 42 bit field coded in two parts. The first part, ESCR_base, is a 33 bit field whose value is given by ESCR_base(i), as given in equation 2-12. The second part, ESCR_ext, is a 9 bit field whose value is given by ESCR_ext(i), as given in equation 2-13 . The ESCR field indicates the intended time of arrival of the byte containing the last bit of the ESCR_base at the input of the PES-STD for PES streams (refer to 2.5.2.4 on page 57)..

Specifically:

$$ESCR\_base(i) = \left(\left(system\_clock\_frequency * t(i) \right) DIV\ 300 \right) \% 2^{33} \tag{2-12}$$

$$ESCR\_ext(i) = \left(\left(system\_clock\_frequency * t(i) \right) DIV\ 1 \right) \%\ 300 \tag{2-13}$$

$$ESCR(i) = ESCR\_base(i) \times 300 + ESCR\_ext(i) \tag{2-14}$$

The ESCR and ES_rate field (refer to semantics immediately following) contain timing information relating to the sequence of PES streams. These fields shall satisfy the contraints
defined in 2.7.3 on page 81.

ES_rate (elementary stream rate) -- The ES_rate field is a 22 bit unsigned integer specifying the rate at which the system target decoder receives bytes of the PES packet in the case of a PES stream. The ES_rate is valid in the PES packet in which it is included and in subsequent PES packets of the same PES stream until a new ES_rate field is encountered. The value of the ES_rate is measured in units of 50 bytes/second. The value 0 is forbidden. The value of the ES_rate is used to define the time of arrival of bytes at the input of a P-STD for PES streams defined in clause 2.5.2.4 on page 57 of this Recommendation | International Standard. The value encoded in the ES_rate field may vary from PES_packet to PES_packet.

trick_mode_control -- A 3 bit field that indicates which trick mode is applied to the associated video stream. In cases of other types of elementary streams, the meanings of this field and those defined by the following five bits are undefined. For the definition of trick_mode status, refer to the trick mode section of 2.4.2.3 on page 15

When trick_mode status is false, the number of times, N, a picture is output by the decoding process for progressive sequences is specified for each picture by the repeat_first_field and top_field_first fields in the case of ISO/IEC 13818-2 Video, and is specified through the sequence header in the case of ISO/IEC 11172-2 Video.

For interlaced sequences, when trick_mode status is false, the number of times, N, a picture is output by the decoding process for is specified for each picture by the repeat_first_field and progressive_frame fields in the case of ISO/IEC 13818-2 Video.

When trick mode status is true, the number of times that a picture should be displayed is dependent on N.

Note that when the value of this field changes or trick mode operations cease any combination of the following may occur:

- discontinuity in the time base
- decoding discontinuity
- continuity counter discontinuity

Table 2-21 -- Trick mode control values

value	description
'000'	fast forward
'001'	slow motion
'010'	freeze frame
'011'	fast reverse
'100'	slow reverse
'101'-'111'	reserved

In the context of trick mode, the non-normal speed of decoding and presentation may cause the values of certain fields defined in video elementary stream data to be incorrect. Likewise, the semantic constraint on the slice structure may be invalid. The video syntax elements to which this exception applies are:

- bit_rate
- vbv_delay
- repeat_first_field
- v_axis_positive
- field_sequence
- subcarrier
- burst_amplitude
- subcarrier_phase

A decoder cannot rely on the values encoded in these fields when in trick mode.

Decoders are not normatively required to decode the trick_mode_control field. However, the following normative requirements shall apply to decoders that do decode the trick_mode_control field.

fast forward -- the value '000', in the trick_mode_control field. When this value is present it indicates a fast forward video stream and defines the meaning of the following five bits in the PES packet header. The intra_slice_refresh bit may be set to '1' indicating that there may be missing macroblocks which the decoder may replace with co-sited macroblocks of previously decoded pictures. The field_id field, defined in table 2-22 on page 41, indicates which field or fields should be displayed. The frequency_truncation field indicates that a restricted set of coefficients may be included. The meaning of the values of this field are shown in table 2-23 on page 42.

slow motion -- the value '001', in the trick_mode_control field. When this value is present it indicates a slow motiion video stream and defines the meaning of the following five bits in the PES packet header. In the case of progressive sequences, the picture should be displayed N x rep_cntrl times, where N is defined above.

In the case of ISO/IEC 11172-2 Video and ITU-T Rec. H.262 | ISO/IEC 13818-2 Video progressive sequences, the picture should be displayed for N x rep_cntrl picture duration.
In the case of ITU-T Rec. H.262 | ISO/IEC 13818-2 interlaced sequences, the picture should be displayed for N x rep_cntrl field duration. If the picture is a frame picture, the first field to be displayed is the top field if top_field_first is 1, and the bottom field if top_field_first is ‘0’ (refer to ITU-T Rec.

H.262 | ISO/IEC 13818-2). This field is displayed for N x rep_cntrl /2 field duration. The other field of the picture is then displayed for N - N x rep_cntrl / 2 field duration.

freeze frame -- the value '010', in the trick_mode_control field. When this value is present it indicates a freeze frame video stream and defines the meaning of the following five bits in the PES packet header. The field_id field is defined in table 2-22 , "Field ID field control values". The field_id field refers to the first video access unit that commences in the PES packet which contains the field_id field, unless the PES packet contains zero payload bytes. In this case the field_id field refers to the most recent previous video access unit.

fast reverse -- the value '011', in the trick_mode_control field. When this value is present it indicates a fast reverse video stream and defines the meaning of the following five bits in the PES packet header. The intra_slice_refresh bit may be set to '1' indicating that there may be missing macroblocks which the decoder may replace with co-sited macroblocks of previously decoded pictures. The field_id field, defined in table 2-22 , indicates which field or fields should be displayed. The frequency_truncation field indicates that a restricted set of coefficients may be included. The meaning of the values of this field are shown in table 2-23, on page 42 "Coefficient selection values".

slow_reverse -- the value '100', in the trick_mode_control field. When this value is present it indicates a slow reverse video stream and defines the meaning of the following five bits in the PES packet header. In the case of ISO/IEC 11172-2 Video and ITU-T Rec. H.262 | ISO/IEC 13818-2 Video progressive sequences, the picture should be displayed for N x rep_cntrl picture duration, where N is defined above.
In the case of ITU-T Rec. H.262 | ISO/IEC 13818-2 interlaced sequences, the picture should be displayed for N x rep_cntrl field duration. If the picture is a frame picture, the first field to be displayed is the bottom field if top_field_first is 1, and the top field if top_field_first is ‘0’ (refer to ITU-T Rec. H.262 | ISO/IEC 13818-2). This field is displayed for N x rep_cntrl /2 field duration. The other field of the picture is then displayed for N - N x rep_cntrl / 2 field duration.

field_id -- A 2 bit field that indicates which field(s) should be displayed. It is coded according to table 2-22

Table 2-22 -- Field_id field control values

value	description
'00'	display from top field only
'01'	display from bottom field only
'10'	display complete frame
'11'	reserved

intra_slice_refresh -- A 1 bit flag., which when set to '1', indicates that there may be missing macroblocks between coded slices of video data in this PES packet. When set to '0' this may not occur. For more information see part 2 of this Recommendation | International Standard. The decoder may replace missing macroblocks with co-sited macroblocks of previously decoded pictures.

frequency_truncation -- A 2 bit field which indicates that a restricted set of coefficients may have been used in coding the video data in this PES packet. The values are defined in table 2-23.

Table 2-23 -- Coefficient selection values

value	description
'00'	Only DC coefficients are non-zero
'01'	typically only the first three coefficients are non-zero
'10'	typically only the first six coefficients are non-zero
'11'	all coefficients may be non-zero

rep_cntrl -- A 5 bit field that indicates the number of times each field in an interlaced picture should be displayed, or the number of times that a progressive picture should be displayed. It is a function of the trick_mode_control field and the top_field_first bit in the video sequence header whether the top field or the bottom field should be displayed first in the case of interlaced pictures. The value '0' is forbidden.

additional_copy_info -- This 7 bit field contains private data relating to copyright information.

previous_PES_packet_CRC -- The previous_PES_packet_CRC is a 16 bit field that contains the CRC value that yields a zero output of the 16 registers in the decoder similar to the one defined in Annex B, but with the polynomial

$$x^{16}+x^{12}+x^{5}+1$$

after processing the data bytes of the previous PES packet, exclusive of the PES packet header.

Note: This CRC is intended for use in network maintenance such as isolating the source of intermittent errors. It is not intended for use by elementary stream decoders. It is calculated only over the data bytes because PES packet header data can be modified during transport.

PES_private_data_flag -- A 1 bit flag which when set to '1' indicates that the PES packet header contains private data. When set to a value of '0' it indicates that private data is not present in the PES header.

pack_header_field_flag -- A 1 bit flag which when set to '1' indicates that an ISO/IEC 11172 pack header or a Program Stream pack header is stored in this PES packet header. If this field is in a PES packet that is contained in a Program Stream, then this field shall be set to '0'. In a Transport Stream, when set to the value '0' it indicates that no pack header is present in the PES header.

program_packet_sequence_counter_flag -- A 1 bit flag which when set to '1' indicates that the program_packet_sequence_counter, MPEG1_MPEG2_identifier, and original_stuff_length fields are present in this PES packet. When set to a value of '0' it indicates that these fields are not present in the PES header.

P-STD_buffer_flag -- A 1 bit flag which when set to '1' indicates that the P-STD_buffer_scale and P-STD_buffer_size are present in the PES packet header. When set to a value of '0' it indicates that these fields are not present in the PES header.

PES_extension_flag_2 -- A 1 bit field which when set to '1' indicates the presence of the PES_extension_field length field and associated fields. When set to a value of '0' this indicates that the PES_extension_field_length field and any associated fields are not present.

PES_private_data -- This is a 16 byte field which contains private data. This data, combined with the fields before and after, shall not emulate the packet_start_code_prefix (0x000001).

pack_field_length -- This is an 8 bit field which indicates the length, in bytes, of the pack_header_field().

program_packet_sequence_counter -- The program_packet_sequence_counter field is an 7 bit field. It is an optional counter that increments with each successive PES packet from a Program Stream or from an ISO/IEC 11172 System Stream or the PES packets associated with a single program definition in a Transport Stream,

providing functionality similar to a continuity counter. This allows an application to retrieve the original PES packet sequence of a Program Stream or the original packet sequence of the original ISO/IEC 11172 system stream. The counter will wrap around to 0 after its maximum value. Repetition of PES packets shall not occur. Consequently, no two consecutive PES packets in the program multiplex shall have identical program_packet_sequence_counter values.

MPEG1_MPEG2_identifier -- A 1 bit flag which when set to '1' indicates that this PES packet carries information from an ISO/IEC 11172 system stream. When set to '0' it indicates that this PES packet carries information from a Program Stream.

original_stuff_length -- This 6 bit field specifies the number of stuffing bytes used in the original ITU-T Rec. H.222.0 | ISO/IEC 13818-1 PES packet header or in the original ISO/IEC 11172-1 packet header.

P-STD_buffer_scale -- The P-STD_buffer_scale is a 1 bit field, the meaning of which is only defined if this PES packet is contained in a Program Stream. It indicates the scaling factor used to interpret the subsequent P-STD_buffer_size field. If the preceding stream_id indicates an audio stream, P-STD_buffer_scale shall have the value '0'. If the preceding stream_id indicates a video stream, P-STD_buffer_scale shall have the value '1'. For all other stream types, the value may be either '1' or '0'.

P-STD_buffer_size --The P-STD_buffer_size is a 13-bit unsigned integer, the meaning of which is only defined if this PES packet is contained in a Program Stream. It defines the size of the input buffer, BS_n, in the P-STD. If P-STD_buffer_scale has the value '0' then the P-STD_buffer_size measures the buffer size in units of 128 bytes. If P-STD_buffer_scale has the value '1' then the P-STD_buffer_size measures the buffer size in units of 1024 bytes. Thus:

if (P-STD_buffer_scale == 0)

$$BS_n = \textit{P-STD\_buffer\_size} \times 128; \quad (2\text{-}15)$$

else

$$BS_n = \textit{P-STD\_buffer\_size} \times 1024; \quad (2\text{-}16)$$

The encoded value of the P-STD buffer size takes effect immediately when the P-STD_buffer_size field is received by the ITU-T Rec. H.222.0 | ISO/IEC 13818 System Target Decoder. Refer to 2.7.7 on page 83.

PES_extension_field_length -- This is a 7 bit field which specifies the length, in bytes, of the data following this field in the PES extension field up to and including any reserved bytes.

stuffing_byte -- This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder, for example to meet the requirements of the channel. It is discarded by the decoder. No more than 32 stuffing bytes shall be present in one PES packet header.

PES_packet_data_byte -- PES_packet_data_bytes shall be contiguous bytes of data from the elementary stream indicated by the packet's stream_id or PID. When the elementary stream data confirms to ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 13818-3, the PES_packet_data_bytes shall be byte aligned to the bytes of ITU-T Rec. H.222.0 | ISO/IEC 13818-1. The byte-order of the elementary stream shall be preserved. The number of PES_packet_data_bytes, N, is specified by the PES_packet_length field. N shall be equal to the value indicated in the PES_packet_length minus the number of bytes between the last byte of the PES_packet_length field and the first PES_packet_data_byte.

In the case of a private_stream_1 or private_stream_2, the contents of the PES_packet_data_byte field is user definable and will not be specified by ISO in the future.

padding_byte -- This is a fixed 8 bit value equal to '1111 1111'. It is discarded by the decoder.

2.4.3.8 Carriage of Program Streams and ISO/IEC 11172-1 Systems streams in the Transport Stream

The Transport Stream contains fields to support the carriage of Program Streams and ISO/IEC 11172-1 Systems streams, in a way that allows simple reconstruction of the respective stream at the decoder.

When placing a Program Stream into a Transport Stream, Program Stream PES packets with stream_id values of private_stream_1, ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 video, and ISO/IEC 13818-3 or ISO/IEC 11172-3 audio, are carried in Transport Stream packets.

For these PES packets, when reconstructing the Program Stream at the Transport Stream decoder, the PES packet data is copied to the Program Stream being reconstructed.

For Program Streams PES packets with stream_id values of program_stream_map, padding_stream, private_stream_2, ECM, EMM, DSM_CC_stream, or program_stream_directory, all the bytes of the Program Stream PES packet, except for the packet_start_code_prefix, are placed into the data_bytes fields of a new PES packet. The stream_id of this new PES packet has the value of ancillary_stream. This new PES packet is then carried in Transport Stream packets.

When reconstructing the Program Stream at the Transport Stream decoder, for PES packets with a stream_id value of ancillary_stream_id, packet_start_code_prefix is written to the Program Stream being reconstructed, followed by the data_byte fields from these Transport Stream PES packets.

ISO/IEC 11172-1 streams are carried within Transport Streams by first replacing ISO/IEC 11172-1 packet headers with ITU-T Rec. H.262 | ISO/IEC 13818-1 PES packet headers. ISO/IEC 11172-1 packet header field values are copied to the equivalent ITU-T Rec. H.262 | ISO/IEC 13818-1 PES packet header fields.

The program_packet_sequence_counter field is included within the header of each PES packet carrying data from a Program Stream, or an ISO/IEC 11172-1 System stream. This allows the order of PES packets in the original Program Stream, or packets in the original ISO/IEC 11172-1 System stream, to be reproduced at the decoder.

The pack_header() field of a Program Stream, or an ISO/IEC 11172-1 System stream, is carried in the Transport Stream in the header of the immediately following PES packet.

2.4.4 Program specific information

Program Specific Information (PSI) includes both ITU-T Rec. H.222.0 | ISO/IEC 13818-1 normative data and private data that enable de-multiplexing of programs by decoders. Programs are composed of one or more elementary streams, each labeled with a PID. Programs, elementary streams or parts thereof may be scrambled for conditional access. However, Program Specific Information shall not be scrambled.

In Transport Streams, Program Specific Information is classified into four table structures as shown in the table below. While these structures may be thought of as simple tables, they shall be segmented into sections and inserted in Transport Stream packets.

Table 2-24 -- Program specific information

Structure Name	Stream Type	Reserved PID #	Description
Program Association Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x00	Associates Program Number and Program Map Table PID
Program Map Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	Assigned in thePAT	Specifies PID values for components of one or more programs
Network Information Table	Private	Assigned in the PAT	Physical network parameters such as FDM frequencies, Transponder Numbers, etc.
Conditional Access Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x01	Associates one or more (private) EMM streams each with a unique PID value

ITU-T Rec. H.222.0 | ISO/IEC 13818-1-defined PSI tables shall be segmented into one or more sections that are carried within transports packets. A section is a syntactic structure that shall be used for mapping all ITU-T Rec. H.222.0 | ISO/IEC 13818-1-defined PSI tables into Transport Stream packets.

Along with ITU-T Rec. H.222.0 | ISO/IEC 13818-1-defined PSI tables, it is possible to carry private data tables. The means by which private information is carried within Transport Stream packets is not defined by this Specification. It may be structured in the same manner used for carrying of ITU-T Rec. H.222.0 | ISO/IEC 13818-1-defined PSI tables, such that the syntax for mapping this private data is identical to that used for the mapping of ITU-T Rec. H.222.0 | ISO/IEC 13818-1-defined PSI tables. For this purpose, a private section is defined. If the private data is carried in Transport Stream packets with the same PID value as Transport Stream packets carrying Program Map Tables, (as identified in the Program Association Table), then the private_section syntax and semantics shall be used. The data carried in the private_data_bytes may be scrambled. However, no other fields of the private_section shall be scrambled. This private_section allows data to be transmitted with a minimum of structure. When this structure is not used, the mapping of private data within Transport Stream packets is not defined by ITU-T Rec. H.222.0 | ISO/IEC 13818-1.

Sections may be variable in length. The beginning of a section is indicated by a pointer_field in the Transport Stream packet payload. The syntax of this field is specified in table 2-25 on page 46.

Adaptation fields may occur in Transport Stream packets carrying sections.

Within a Transport Stream, packet stuffing bytes of value 0xFF may be found after the last byte of a section, in which case all following bytes until the end of the Transport Stream packet shall also be stuffing bytes of value 0xFF. These bytes may be discarded by a decoder. In such a case, the payload of the next Transport Stream packet with the same PID value shall begin with a pointer_field of value 0x00 indicating that the next section starts immediately thereafter.

Each Transport Stream shall contain one or more Transport Stream packets with PID value 0x0000. These Transport Stream packets together shall contain a complete list of all programs within the Transport Stream. The most recently transmitted version of the table with the current_next_indicator set to a value of '1' shall always apply to the current data in the Transport Stream. Any changes in the programs carried within the Transport Stream shall be described in an updated version of the Program Association Table carried in Transport Stream packets with PID value 0x0000. These sections shall all use table_id value 0x00. Only sections with this value of table_id are permitted within Transport Stream packets with PID value of 0x0000. For a new version of the PAT to become valid, all sections (as indicated in the last_section_number) with a new version_number and with the current_next_indicator set to '1' must exit B_{sys} defined in the T-STD (refer to 2.4.2 on page 11). The PAT becomes valid when the last byte of the section needed to complete the table exits B_{sys}.

Whenever one or more elementary streams within a Transport Stream are scrambled, Transport Stream packets with a PID value 0x0001 shall be transmitted containing CA_sections containing CA_descriptors appropriate to the scrambled streams. The transmitted Transport Stream packets shall together form one complete version of the conditional access table. The most recently transmitted version of the table with the current_next_indicator set to a value of '1' shall always apply to the current data in the Transport Stream. Any changes in scrambling making the existing table invalid or incomplete shall be described in an updated version of the conditional access table. These sections shall all use table_id value 0x01. Only sections with this table_id value are permitted within Transport Stream packets with a PID value of 0x0001. For a new version of the CAT to become valid, all sections (as indicated in the last_section_number) with a new version_number and with the current_next_indicator set to '1' must exit B_{sys}. The CAT becomes valid when the last byte of the section needed to complete the table exits B_{sys}.

Each Transport Stream shall contain one or more Transport Stream packets with PID values which are labeled under the program association table as Transport Stream packets containing TS program map sections. Each program listed in the Program Association Table shall be described in a unique TS_program_map_section. Every program shall be fully defined within the Transport Stream itself. Private data which has an associated elementary_PID field in the appropriate Program Map Table section is part of the program. Other private data may exist in the Transport Stream without being listed in the Program Map Table section. The most recently transmitted version of the TS_program_map_section with the current_next_indicator set to a value of '1' shall always apply to the current data within the Transport Stream. Any changes in the definition of any of the programs carried within the Transport Stream shall be described in an updated version of the corresponding section of the program map table carried in Transport Stream packets with the PID value identified as the program_map_PID for that specific program. All Transport Stream packets which carry a given TS_program_map_section shall have the same PID value. During the continuous existence of a program, including all of its associated events, the program_map_PID shall not change. A program definition shall not span more than one TS_program_map_section. A new version of a TS_program_map_section becomes valid when the last byte of that section with a new version_number and with the current_next_indicator set to '1' exits $B_{sys.}$

Sections with a table_id value of 0x02 shall contain Program Map Table information. Such sections may be carried in Transport Stream packets with different PID values.

The Network Information Table is optional and its contents are private. If present it is carried within Transport Stream packets that shall have the same PID value, called the network_PID. The network_PID value is defined by the user and, when present, shall be found in the Program Association Table under the reserved program_number 0x0000. If the network information table exists, it shall take the form of one or more private_sections.

The maximum number of bytes in a section of a ITU-T Rec. H.222.0 | ISO/IEC 13818-1-defined PSI table is 1024 bytes. The maximum number of bytes in a private_section is 4096 bytes.

There are no restrictions on the occurrence of start codes, sync bytes or other bit patterns in PSI data, whether ITU-T Rec. H.222.0 | ISO/IEC 13818-1 or private.

2.4.4.1 Pointer

The pointer_field syntax is defined in table 2-25 below.

Table 2-25 -- Program specific information pointer

Syntax	No. of bits	Mnemonic
pointer_field	**8**	**uimsbf**

2.4.4.2 Semantics definition of fields in pointer syntax

pointer_field -- This is an 8-bit field whose value shall be the number of bytes, immediately following the pointer_field until the first byte of the first section that is present in the payload of the Transport Stream packet (so a value of 0x00 in the pointer_field indicates that the section starts immediately after the pointer_field). When at least one section begins in a given Transport Stream packet, then the payload_unit_start_indicator shall be set to 1 and the first byte of the payload of that Transport Stream packet shall contain the pointer. When no section begins in a given Transport Stream packet, then the payload_unit_start_indicator shall be set to 0 and no pointer shall be sent in the payload of that packet.

2.4.4.3 Program association table

The Program Association Table provides the correspondence between a program_number and the PID value of the Transport Stream packets which carry the program definition. The program_number is the numeric label associated with a program.

The overall table is to be split into one or more sections with the following syntax.

Program number 0x0000 is reserved to specify the network PID. This identifies the Transport Stream packets

Table 2-26 -- Program association section

Syntax	No. of bits	Mnemonic
program_association_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
transport_stream_id	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i=0; i<N;i++) {		
program_number	**16**	**uimsbf**
reserved	**3**	**bslbf**
if(program_number == '0') {		
network_PID	**13**	**uimsbf**
}		
else {		
program_map_PID	**13**	**uimsbf**
}		
}		
CRC_32	**32**	**rpchof**
}		

which carry the Network Information Table.

2.4.4.4 Table_id assignments

The table_id field identifies the content of a Transport Stream PSI section as shown in table 2-27 below.

Table 2-27 -- table_id assignment values

value	description
0x00	program_association_section
0x01	conditional_access_section(CA_section)
0x02	TS_program_map_section
0x03-0x3F	ITU-T Rec. H.222.0 | ISO/IEC 13818 reserved
0x40-0xFE	User private
0xFF	forbidden

2.4.4.5 Semantic definition of fields in program association section

table_id -- This is an 8 bit field, which shall be set to 0x00 as shown in table 2-27 on page 47 above.

section_syntax_indicator -- The section_syntax_indicator is a 1 bit field which shall be set to '1'.

section_length -- This is a twelve bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the section, starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021

transport_stream_id -- This is a 16 bit field which serves as a label to identify this Transport Stream from any other multiplex within a network. Its value is defined by the user.

version_number -- This 5 bit field is the version number of the whole Program Association Table. The version number shall be incremented by 1 whenever the definition of the Program Association Table changes. Upon reaching the value 31, it wraps around to 0. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable Program Association Table. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable Program Association Table.

current_next_indicator -- A 1 bit indicator, which when set to '1' indicates that the Program Association Table sent is currently applicable. When the bit is set to '0', it indicates that the table sent is not yet applicable and shall be the next table to become valid.

section_number -- This 8 bit field gives the number of this section. The section_number of the first section in the Program Association Table shall be 0x00. It shall be incremented by 1 with each additional section in the Program Association Table.

last_section_number -- This 8 bit field specifies the number of the last section (that is, the section with the highest section_number) of the complete Program Association Table.

program_number -- Program_number is a 16 bit field. It specifies the program to which the program_map_PID is applicable. If this is set to 0x0000 then the following PID reference shall be the network PID. For all other cases the value of this field is user defined. This field shall not take any single value more than once within one version of the program association table. The program_number may be used as a designation for a broadcast channel, for example.

network_PID -- network_PID is a 13 bit field specifying the PID of the Transport Stream packets which shall

contain the Network Information Table. The value of the network_PID field is defined by the user, but shall only take values as specified in table 2-4 on page 23. The presence of the network_PID is optional.

program_map_PID -- program_map_PID is a 13 bit field specifying the PID of the Transport Stream packets which shall contain the program_map_section applicable for the program as specified by the program_number. No program_number shall have more than one program_map_PID assignment. The value of the program_map_PID is defined by the user, but shall only take values as specified in table 2-4 on page 23.

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 B after processing the entire program association section.

2.4.4.6 Conditional access table

The Conditional Access (CA) Table provides the association between one or more CA systems, their EMM streams and any special parameters associated with them. Refer to 2.6.16 on page 74 for a definition of the descriptor() field below.

The table may be segmented into one or more sections, before insertion into Transport Stream packets, with the following syntax.

Table 2-28 -- Conditional access section

Syntax	No. of bits	Mnemonic
CA_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
reserved	**18**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i=0; i<N;i++) {		
descriptor()		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.7 Semantic definition of fields in conditional access section

table_id -- This is an 8 bit field, which shall be always set to 0x01 as shown in table 2-27 on page 47 above.

section_syntax_indicator -- The section_syntax_indicator is a 1 bit field which shall be set to '1'.

section_length -- This is a 12 bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the section starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021.

version_number -- This 5 bit field is the version number of the whole conditional access table. The version number shall be incremented by 1 modulo 32 when a change in the information carried within the CA table occurs. When the current_next_indicator is set to '1', then the version_number shall be that of the currently

applicable Conditional Access Table. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable Conditional Access Table.

current_next_indicator -- A 1 bit indicator, which when set to '1' indicates that the Conditional Access Table sent is currently applicable. When the bit is set to '0', it indicates that the Conditional Access Table sent is not yet applicable and shall be the next Conditional Access Table to become valid.

section_number -- This 8 bit field gives the number of this section. The section_number of the first section in the Conditional Access Table shall be 0x00. It shall be incremented by 1 modulo 256 with each additional section in the Conditional Access Table.

last_section_number -- This 8 bit field specifies the number of the last section (that is, the section with the highest section_number) of the Conditional Access Table.

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 B after processing the entire conditional access section.

2.4.4.8 Program Map Table

The Program Map Table provides the mappings between program numbers and the program elements that comprise them. A single instance of such a mapping is referred to as a "program definition." The program map table is the complete collection of all program definitions for a Transport Stream. This table shall be transmitted in packets, the PID values of which are selected by the encoder. More than one PID value may be used, if desired. The table may be segmented into one or more sections, before insertion into Transport Stream packets, with the following syntax. In each section the section number field shall be set to zero. Sections are identified by the program_number field.

Definition for the descriptor() fields may be found in 2.6. on page 67 of this Recommendation | International Standard.

Table 2-29 -- Transport Stream program map section

Syntax	No. of bits	Mnemonic
TS_program_map_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
program_number	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
reserved	**3**	**bslbf**
PCR_PID	**13**	**uimsbf**
reserved	**4**	**bslbf**
program_info_length	**12**	**uimsbf**
for (i=0; i<N; i++) {		
descriptor()		
}		
for (i=0;i<N1;i++) {		
stream_type	**8**	**uimsbf**
reserved	**3**	**bslbf**
elementary_PID	**13**	**uimsnf**
reserved	**4**	**bslbf**
ES_info_length	**12**	**uimsbf**
for (i=0; i<N2; i++) {		
descriptor()		
}		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.9 Semantic definition of fields in Transport Stream program map section

table_id -- This is an 8 bit field, which in the case of a TS_program_map_section shall be always set to 0x02 as shown in table 2-27 on page 47 above.

section_syntax_indicator -- The section_syntax_indicator is a 1 bit field which shall be set to '1'.

section_length -- This is a 12 bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the section starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021.

program_number -- program_number is a 16 bit field. It specifies the program to which the program_map_PID is applicable. One program definition shall be carried within only one TS_program_map_section. This implies that a program definition is never longer than 1016 bytes. See Informative Annex C for ways to deal with the cases when that length is not sufficient. The program_number may be used as a designation for a broadcast channel, for example. By describing the different program elements belonging to a program, data from different sources (e.g. sequential events) can be concatenated together to form a continuous set of streams using a program_number. For examples of applications refer to Annex C.

version_number -- This 5 bit field is the version number of the TS_program_map_section. The version number shall be incremented by 1 modulo 32 when a change in the information carried within the section occurs. Version number refers to the definition of a single program, and therefore to a single section. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable TS_program_map_section. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable TS_program_map_section.

current_next_indicator -- A 1 bit field, which when set to '1' indicates that the TS_program_map_section sent is currently applicable. When the bit is set to '0', it indicates that the TS_program_map_section sent is not yet applicable and shall be the next TS_program_map_section to become valid.

section_number -- The value of this 8 bit field shall be always 0x00.

last_section_number -- The value of this 8 bit field shall be always 0x00.

PCR_PID -- This is a 13 bit field indicating the PID of the Transport Stream packets which shall contain the PCR fields valid for the program specified by program_number. If no PCR is associated with a program definition for private streams then this field shall take the value of 0x1FFF. Refer to the semantic definition of PCR in 2.4.3.5 on page 25 for restrictions on the choice of PCR_PID value.

program_info_length -- This is a 12 bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the descriptors immediately following the program_info_length field.

stream_type -- This is an 8 bit field specifying the type of program element carried within the packets with the PID whose value is specified by the elementary_PID. The values of stream_type are specified in table 2-36 on page 64.

elementary_PID -- This is a 13 bit field specifying the PID of the Transport Stream packets which carry the associated program element.

ES_info_length -- This is a 12 bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the descriptors of the associated program element immediately following the ES_info_length 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 B after processing the entire Transport Stream program map section.

2.4.4.10 Syntax of the Private section

The use of the private_section is mandatory when private data is sent in Transport Stream packets with a PID value designated as a Program Map Table PID in the Program Association Table. This private_section allows data to be transmitted with the absolute minimum of structure while enabling a decoder to parse the stream. The sections may be used in two ways: if the section_syntax_indicator is set to '1', then the whole structure common to all tables shall be used; if the indicator is set to '0', then only the fields 'table_id' through 'private_section_length' shall follow the common structure syntax and semantics and the rest of the private_section may take any form the user determines. Examples of extended use of this syntax are found in Informative Annex C.

A private table may be made of several private_sections, all with the same table_id.

Table 2-30 -- Private section

Syntax	No. of bits	Mnemonic
private_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
private_indicator	**1**	**bslbf**
reserved	**2**	**bslbf**
private_section_length	**12**	**uimsbf**
if (section_syntax_indicator == '0') {		
for (i=0;i<N;i++) {		
private_data_byte	**8**	**bslbf**
}		
}		
else {		
table_id_extension	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i=0;i<private_section_length-9;i++) {		
private_data_byte	**8**	**bslbf**
}		
CRC_32	**32**	**rpchof**
}		
}		

2.4.4.11 Semantic definition of fields in private section

table_id -- This an 8-bit field, the value of which identifies the Private Table this section belongs to. Some values are reserved (table 2-27 on page 47).

section_syntax_indicator -- This is a 1 bit indicator. When set to '1', it indicates that the private section follows the generic section syntax beyond the private_section_length field. When set to '0', it indicates that the private_data_bytes immediately follow the private_section_length field.

private_indicator -- This is a 1 bit user definable flag that shall not be specified by ITU-T | ISO/IEC in the future.

private_section_length -- A 12-bit field. It specifies the number of remaining bytes in the private section immediately following the private_section_length field up to the end of the private_section. The value in this field shall not exceed 4093.

private_data_byte -- The private_data_byte field is user definable and shall not be specified by ITU-T | ISO/IEC in the future.

table_id_extension -- This is a 16-bit field. Its use and value are defined by the user.

version_number -- This 5-bit field is the version number of the private_section. The version_number shall be incremented by 1 modulo 32 when a change in the information carried within the private_section occurs. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable private_section with the same table_id and section_number.

current_next_indicator -- A 1 bit field, which when set to '1' indicates that the private_section sent is currently applicable. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable private_section. When the bit is set to '0', it indicates that the private_section sent is not yet applicable and shall be the next private_section with the same section_number and table_id to become valid.

section_number -- This 8-bit field gives the number of the private_section. The section_number of the first section in a private table shall be 0x00. The section_number shall be incremented by 1 modulo 256 with each additional section in this private table.

last_section_number -- This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the private table of which this section is a part.

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 B after processing the entire private section.

2.5 Program Stream bitstream requirements

2.5.1 Program Stream coding structure and parameters

The Program Stream coding layer allows one or more elementary streams to be combined into a single stream. Data from each elementary stream is multiplexed and encoded together with information that allows elementary streams to be replayed in synchronism.

A Program Stream consists of one or more elementary streams from one program multiplexed together. Data from elementary streams is stored in PES packets. A PES packet consists of a PES packet header followed by packet data.

The PES packet header begins with a 32-bit start-code that also identifies the stream (refer to table 2-19 on page 36) to which the packet data belongs. The PES packet header may contain just a presentation timestamp or both a presentation timestamp and a decoding timestamp. The PES packet header also contains a number of flags opening a range of optional fields. The packet data contains a variable number of contiguous bytes from one elementary stream.

In a Program Stream, PES packets are organized in packs. A pack commences with a pack header and is followed by zero or more PES packets. The pack header begins with a 32-bit start-code. The pack header is used to store timing and bitrate information.

The Program Stream begins with a system header that optionally may be repeated. The system header carries a summary of the system parameters defined in the stream.

This standard does not specify the coded data which may be used as part of conditional access systems. The standard does however provide mechanisms for program service providers to transport and identify this data for decoder processing, and to reference correctly data which are specified by the standard.

2.5.2 Program Stream system target decoder

The semantics of the Program Stream specified in 2.5.4 on page 62 and the constraints on these semantics specified in 2.5.4 on page 62 require exact definitions of decoding events and the times at which these events occur. The definitions needed are set out in this Recommendation | International Standard using a hypothetical decoder known as the Program Stream system target decoder (P-STD).

The P-STD is a conceptual model used to define these terms precisely and to model the decoding process during the construction of Program Streams. The P-STD is defined only for this purpose. Neither the architecture of the P-STD nor the timing described precludes uninterrupted, synchronized play-back of Program Streams from a variety of decoders with different architectures or timing schedules.

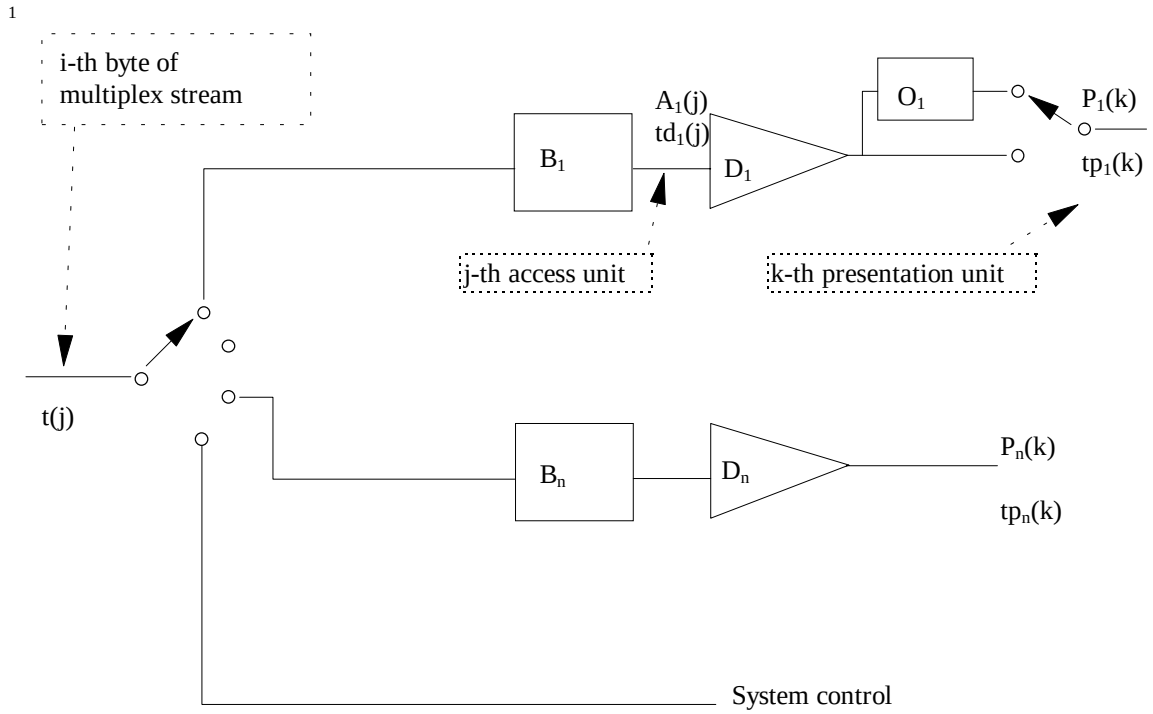

Figure 2-7 -- Program Stream system target decoder notation

The following notation is used to describe the Program Stream system target decoder and is partially illustrated in figure 2-7 above.

i, i' are indices to bytes in the Program Stream. The first byte has index 0.

j is an index to access units in the elementary streams.

k, k',k" are indices to presentation units in the elementary streams.

n is an index to the elementary streams.

t(i) indicates the time in seconds at which the i^{th} byte of the Program Stream enters the system target decoder. The value t(0) is an arbitrary constant.

SCR(i) is the time encoded in the SCR field measured in units of the 27 MHz system clock where i is the byte index of the final byte of the system_clock_reference_base field.

$A_n(j)$ is the j^{th} access unit in elementary stream n. $A_n(j)$ is indexed in decoding order.

$td_n(j)$ is the decoding time, measured in seconds, in the system target decoder of the j^{th} access unit in elementary stream n.

$P_n(k)$ is the k^{th} presentation unit in elementary stream n. $P_n(k)$ is indexed in presentation order.

$tp_n(k)$ is the presentation time, measured in seconds, in the system target decoder of the k^{th} presentation unit in elementary stream n.

t is time measured in seconds.

$F_n(t)$ is the fullness, measured in bytes, of the system target decoder input buffer for elementary stream n at time t.

B_n the input buffer in the system target decoder for elementary stream n.

BS_n is the size of the system target decoder input buffer, measured in bytes, for elementary stream n.

D_n is the decoder for elementary stream n.

O_n is the reorder buffer for video elementary stream n.

2.5.2.1 System clock frequency

Timing information referenced in P-STD is carried by several data fields defined in this Recommendation | International Standard. The fields are defined in 2.5.3.3 on page 58, and 2.4.3.6 on page 33. This information is coded as the sampled value of a system clock.

The value of the system clock frequency is measured in Hz and shall meet the following constraints:

27 000 000 - 810 <= system_clock_frequency <= 27 000 000 + 810

rate of change of system_clock_frequency with time <= 75×10^{-3} Hz/s

The notation "system_clock_frequency" is used in several places in this proposed standard to refer to the frequency of a clock meeting these requirements. For notational convenience, equations in which SCR, PTS, or DTS appear, lead to values of time which are accurate to some integral multiple of (300×2^{33}/system_clock_frequency) seconds. This is due to the encoding of SCR timing information as 33 bits of 1/300 of the system clock frequency plus 9 bits for the remainder, and encoding as 33 bits of thesystem clock frequency divided by 300 for PTS and DTS.

2.5.2.2 Input to the Program Stream system target decoder

Data from the Program Stream enters the system target decoder. The i^{th} byte enters at time t(i). The time at which this byte enters the system target decoder can be recovered from the input stream by decoding the input system clock reference (SCR) fields and the program_mux_rate field encoded in the pack header. The SCR, as

defined in equation 2-19 , is coded in two parts; one, in units the period of 1/300 × the system clock frequency, called system_clock_reference_base (equation 2-17), and one, called system_clock_reference_ext equation (equation 2-18), in units of the period of the system clock frequency. In the following the values encoded in these fields are denoted by SCR_base(i) and SCR_ext(i). The value encoded in the SCR field indicates time t(i), where i refers to the byte containing the last bit of the system_clock_reference_base field.

Specifically:

$$SCR\_base(i) = \left(\left(system\_clock\_ frequency \times t(i) \right) DIV\ 300 \right) \% 2^{33} \tag{2-17}$$

$$SCR\_ext(i) = \left(\left(system\_clock\_ frequency \times t(i) \right) DIV\ 1 \right) \% 300 \tag{2-18}$$

$$SCR(i) = SCR\_base(i) \times 300 + SCR\_ext(i) \tag{2-19}$$

The input arrival time, t(i), as given in equation 2-20, for all other bytes shall be constructed from SCR(i) and the rate at which data arrives, where the arrival rate within each pack is the value represented in the program_mux_rate field in that pack's header.

$$t(i) = \frac{SCR(i')}{system\_clock\_ frequency} + \frac{i - i'}{program\_mux\_rate \times 50} \tag{2-20}$$

Where:

i'	is the index of the byte containing the last bit of the system_clock_reference_base field in the pack header.
i	is the index of any byte in the pack, including the pack header.
SCR(i')	is the time encoded in the system clock reference base and extension fields in units of the system clock.
program_mux_rate	is a field defined in 2.5.3.3 on page 58.

After delivery of the last byte of a pack there may be a time interval during which no bytes are delivered to the input of the P-STD.

2.5.2.3 Buffering

The PES packet data from elementary stream n is passed to the input buffer for stream n, B_n. Transfer of byte i from the system target decoder input to B_n is instantaneous, so that byte i enters the buffer for stream n, of size BS_n, at time t(i).

Bytes present in the pack header, system headers, or PES packet headers of the Program Stream such as SCR, DTS, PTS, and packet_length fields, are not delivered to any of the buffers, but may be used to control the system.

The input buffer sizes BS_1 through BS_n are given by the P-STD buffer size parameter in the syntax in equation 2-15 and equation 2-16 on page 43.

At the decoding time, $td_n(j)$, all data for the access unit that has been in the buffer longest , $A_n(j)$, and any stuffing bytes that immediately preceed it that are present in the buffer at the time $td_n(j)$,are removed instantaneously at time $td_n(j)$. The decoding time $td_n(j)$ is specified in the DTS or PTS fields. Decoding times $td_n(j+1)$, $td_n(j+2)$, ... of access units without encoded DTS or PTS fields which directly follow access unit j may be derived from information in the elementary stream. Refer to Annex C of ITU-T Rec. H.262 | ISO/IEC 13818-2, ISO/IEC 13818-3, or ISO/IEC 11172. Also refer to 2.7.5 on page 81. As the access unit is removed from the buffer it is instantaneously decoded to a presentation unit.

The Program Stream in the notation described in 2.5.2 on page 53, and the PES packets defined in 2.4.3.6 on page 33 shall be constructed and t(i) shall be chosen so that the input buffers of size BS_1 through BS_n neither overflow nor underflow in the program system target decoder. That is:

$$0 \le F_n(t) \le BS_n \qquad \text{for all t and n}$$

and $F_n(t) = 0$ instantaneously before t=t(0).

$F_n(t)$ is the instantaneous fullness of P-STD buffer B_n.

An exception to this condition is that the P-STD buffer B_n may underflow when the low_delay flag in the video sequence header is set to '1' (refer to 2.4.2.3 on page 15) or during trick mode operation (refer to 2.4.3.8 on page 44).

For all Program Streams the delay caused by system target decoder input buffering shall be less than or equal to 1 second except for still picture video data. The input buffering delay is the difference in time between a byte entering the input buffer and when it is decoded.

Specifically: in the case of no still picture video data then the delay is constrained by

$$td_n(j) - t(i) \le 1 \text{ sec}$$

else in the case of still picture video data the delay is constrained by:

$$td_n(j) - t(i) \le 60 \text{ sec}$$

for all bytes contained in access unit j.

For Program Streams, all bytes of each pack shall enter the P-STD before any byte of a subsequent pack.

When the **low_delay** flag in the video sequence extension is set to '1' (refer to 6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2) the VBV buffer may underflow. In this case when the P-STD elementary stream buffer B_n is examined at the time specified by $td_n(j)$, the complete data for the access unit may not be present in the buffer B_n. When this case arises, the buffer shall be re-examined at intervals of two field-periods until the data for the complete access unit is present in the buffer. At this time the entire access unit and associated system data if any shall be removed from buffer B_n instantaneously.

VBV buffer underflow is allowed to occur continuously without limit. The P-STD decoder shall remove access unit data from buffer B_n at the earliest time consistent with the paragraph above and any DTS or PTS values encoded in the bitstream. Note that the decoder may be unable to re-establish correct decoding and display times as indicated by DTS and PTS until the VBV buffer underflow situation ceases and a PTS or DTS is found in the bitstream.

2.5.2.4 PES streams

It is possible to construct a stream of data as a contiguous stream of PES packets each containing data of the same elementary stream and with the same stream_id. Such a stream is called a PES stream. The PES-STD model for a PES stream is identical to that for the Program Stream, with the exception that the Elementary System Clock Reference (ESCR) is used in place of the SCR, and ES_rate in place of program_mux_rate. The demultiplexor sends data to only one elementary stream buffer.

Buffer sizes BS_n in the PES-STD model are defined as follows:

For video, $BS_n = VBV_{max}[profile,level] + BS_{oh}$

BS_{oh} = (1/750) seconds * $R_{max}[profile,level]$, where $VBV_{max}[profile,level]$ and $R_{max}[profile,level]$ are the maximum VBV size and bit rate per profile, level, and layer as defined in tables 8.12 and 8.13 of part 2 of this Recommendation | International Standard. BS_{oh} is allocated for PES packet header overhead.

For audio, BS_n = 2 848 bytes.

2.5.2.5 Decoding and presentation

Decoding and presentation in the Program Stream system target decoder are the same as defined for the Transport Stream system target decoder, in 2.4.2.4 on page 20, and 2.4.2.5 on page 20 respectively.

2.5.3 Specification of the Program Stream syntax and semantics

The following syntax describes a stream of bytes.

2.5.3.1 Program Stream

Table 2-31 -- Program Stream

Syntax	No. of bits	Mnemonic
MPEG2_program_stream() {		
do {		
pack()		
} while (nextbits() == pack_start_code)		
MPEG_program_end_code	**32**	**bslbf**
}		

2.5.3.2 Semantic definition of fields in Program Stream

MPEG_program_end_code -- The MPEG_program_end_code is the bit string '0000 0000 0000 0000 0000 0001 1011 1001' (0x000001B9). It terminates the Program Stream.

2.5.3.3 Pack layer of Program Stream

Table 2-32 -- Program Stream pack

Syntax	No. of bits	Mnemonic
pack() {		
pack_header()		
while (nextbits() == packet_start_code_prefix) {		
PES_packet()		
}		
}		

Table 2-33 -- Program Stream pack header

Syntax	No. of bits	Mnemonic
pack_header() {		
pack_start_code	**32**	**bslbf**
'01'	**2**	**bslbf**
system_clock_reference_base [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
system_clock_reference_base [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
system_clock_reference_base [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
system_clock_reference_extension	**9**	**uimsbf**
marker_bit	**1**	**bslbf**
program_mux_rate	**22**	**uimsbf**
marker_bit	**1**	**bslbf**
marker_bit	**1**	**bslbf**
reserved	**5**	**bslbf**
pack_stuffing_length	**3**	**uimsbf**
for (i=0;i<pack_stuffing_length;i++) {		
stuffing_byte	**8**	**bslbf**
}		
if (nextbits() == system_header_start_code) {		
system_header ()		
}		
}		

2.5.3.4 Semantic definition of fields in program stream pack

pack_start_code -- The pack_start_code is the bit string '0000 0000 0000 0000 0000 0001 1011 1010' (0x000001BA). It identifies the beginning of a pack.

system_clock_reference_base; system_clock_reference_extension -- The system_clock_reference (SCR) is a 42 bit field coded in two parts. The first part, system_clock_reference_base, is a 33 bit field whose value is given by SCR_base(i) as given in equation 2-17 on page 55. The second part, system_clock_reference_extension, is a 9 bit field whose value is given by SCR_ext(i), as given in equation 2-18 on page 55. The SCR indicates the intended time of arrival of the byte containing the last bit of the system_clock_reference_base at the input of the program target decoder.

The frequency of coding requirements for the SCR field are given in 2.7.1. on page 80

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

program_mux_rate -- This is a 22 bit integer specifying the rate at which the P-STD receives the Program Stream during the pack in which it is included. The value of program_mux_rate is measured in units of 50 bytes/second. The value 0 is forbidden. The value represented in program_mux_rate is used to define the time of arrival of bytes at the input to the P-STD in 2.5.2 on page 53 of this Recommendation | International Standard. The value encoded in the program_mux_rate field may vary from pack to pack in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program multiplexed stream.

pack_stuffing_length -- A 3 bit integer specifying the number of stuffing bytes which follow this field.

stuffing_byte -- This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder, for example to meet the requirements of the channel. It is discarded by the decoder. In each pack header no more than 7 stuffing bytes shall be present.

2.5.3.5 System header

Table 2-34 -- Program Stream system header

Syntax	No. of Bits	Mnemonic
system_header () {		
system_header_start_code	**32**	**bslbf**
header_length	**16**	**uimsbf**
marker_bit	**1**	**bslbf**
rate_bound	**22**	**uimsbf**
marker_bit	**1**	**bslbf**
audio_bound	**6**	**uimsbf**
fixed_flag	**1**	**bslbf**
CSPS_flag	**1**	**bslbf**
system_audio_lock_flag	**1**	**bslbf**
system_video_lock_flag	**1**	**bslbf**
marker_bit	**1**	**bslbf**
video_bound	**5**	**uimsbf**
packet_rate_restriction_flag	**1**	**bslbf**
reserved_byte	**7**	**bslbf**
while (nextbits () == '1') {		
stream_id	**8**	**uimsbf**
'11'	**2**	**bslbf**
P-STD_buffer_bound_scale	**1**	**bslbf**
P-STD_buffer_size_bound	**13**	**uimsbf**
}		
}		

2.5.3.6 Semantic definition of fields in system header

system_header_start_code -- The system_header_start_code is the bit string '0000 0000 0000 0000 0000 0001 1011 1011' (0x000001BB). It identifies the beginning of a system header.

header_length -- This 16 bit field indicates the length in bytes of the system header following the header_length field. Note that future extensions of this Recommendation | International Standard may extend the system header.

rate_bound -- A 22 bit field. The rate_bound is an integer value greater than or equal to the maximum value of the program_mux_rate field coded in any pack of the Program Stream. It may be used by a decoder to assess whether it is capable of decoding the entire stream.

audio_bound -- A 6 bit field. The audio_bound is an integer in the inclusive range from 0 to 32 greater than or equal to the maximum number of ISO/IEC 13818-3 and IOS/IEC 11172-3 audio streams in the Program Stream for which the decoding processes are simultaneously active. For the purpose of this Clause, the decoding process of an ISO/IEC 13818-3 or ISO/IEC 11172-3 audio stream is active if the STD buffer is not empty or if a Presentation Unit is being presented in the P-STD model.

fixed_flag -- The fixed_flag is a 1 bit flag. If its value is set to '1' fixed bitrate operation is indicated. If its value is set to '0' variable bitrate operation is indicated. During fixed bitrate operation, the value encoded in all

system_clock_reference fields in the multiplexed ITU-T Rec. H222.0 | ISO/IEC 13818 stream shall adhere to the following linear equation:

$$SCR\_base(i) = ((c1 * i + c2) \text{ DIV } 300) \% 2^{33} \quad (2\text{-}21)$$

$$SCR\_ext(i) = ((c1 * i + c2) \text{ DIV } 1) \% 300 \quad (2\text{-}22)$$

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 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 on page 83 of this Recommendation | International Standard.

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. 2.5.2 on page 53 defines system_clock_frequency 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.

$$SCASR = \frac{system\_clock\_frequency}{audio\_sample\_rate\_in\_the\_P-STD} \quad (2\text{-}23)$$

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 picture rate and the system clock frequency in the system target decoder. 2.5.2 on page 53 defines system_clock_frequency and the video picture rate is specified in part 2 of this Recommendation | International Standard. 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 picture rate, SCPR, is constant and equal to the value indicated in the following table at the nominal picture rate indicated in the video stream.

$$SCPR = \frac{system\_clock\_frequency}{picture\_rate\_in\_the\_P-STD} \quad (2\text{-}24)$$

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

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

video_bound -- The video_bound is a 5 bit integer in the inclusive range from 0 to 16 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 Clause, 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 on page 83. If the CSPS flag is set to value of '0', then the meaning of the packet_rate_restriction_flag is undefined.

reserved_byte -- This byte is reserved for future use by ISO/IEC. Until otherwise specified by ITU-T | ISO/IEC it shall have the value '1111 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 2-19 on page 36.

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)

$$BS_n \le P - STD\_buffer\_size\_bound \times 128;$$

else

$$BS_n \le P - STD\_buffer\_size\_bound \times 1024;$$

2.5.3.7 Packet layer of Program Stream

The packet layer of the Program Stream is defined by the PES packet layer 2.4.3.6 on page 33.

2.5.4 Program Stream map

The Program Stream Map 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.

Definition for the descriptor() fields may be found in 2.6. on page 67 of this Recommendation | International Standard.

2.5.4.1 Syntax of Program Stream map

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 is always 0xBC in hexadecimal.

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 the field is 1018 (bytes).

current_next_indicator -- A 1 bit field, which 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 one 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.

stream_type -- stream_type is a 8 bit field specifying the type of the elementary stream according to the following table.

Table 2-36 -- Stream type assignments

Value	Description
0x00	ITU-T | ISO/IEC Reserved
0x01	ISO/IEC 11172 Video
0x02	ITU-T Rec. H.262 | ISO/IEC 13818-2 Video or ISO/IEC 11172-2 constrained parameter video stream
0x03	ISO/IEC 11172 Audio
0x04	ISO/IEC 13818-3 Audio
0x05	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 private_sections
0x06	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 PES packets containing private data
0x07	ISO/IEC 13522 MHEG
0x08	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A DSM CC
0x09	ITU-T Rec. H.222.1
0x0A	ISO/IEC 13818-6 type A
0x0B	ISO/IEC 13818-6 type B
0x0C	ISO/IEC 13818-6 type C
0x0D	ISO/IEC 13818-6 type D
0x0E	ISO/IEC 13818-1 auxiliary
0x0F-0x7F	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
0x80-0xFF	User Private

Note - An ISO/IEC 13818-1 auxiliary stream is available for data types defined by this Specification, other than audio, video, and DSM CC, such as Program Stream Directory and Program Stream Map.

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 B 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 PES packets identified with the directory_stream_id. The syntax for program_stream_directory PES packets is defined table 2-37 on page 65.

Note - This syntax differs from the PES packet syntax described 2.4.3.6 on page 33.

Directory entries may be required to reference I-pictures in a video stream as defined in part 2 of this Recommendation | International Standard and ISO/IEC 11172. 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 1 - 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 shall be the syncword of the audio frame.

Note 2 - It is recommended that the distance between referenced access units not exceed half a second.

Access units shall be included in the directory in the same order that they appear in the bitstream.

2.5.5.1 Syntax of the PES packet for the Program Stream directory

Table 2-37 -- PES packet syntax for Program Stream directory

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**

Syntax	No. of Bits	Mnemonic
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**
}		
}		

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 is an 8 bit field whose value is always 0xFF in hexadecimal.

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.

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 1st byte of the start code of the previous PES packet that contains Program Stream Directory information. This address offset is relative to the 1st byte of the start code of the PES packet which contains this previous_directory_offset field. The value '0' is reserved to indicate that there is no previous Program Stream Directory information.

next_directory_offset - This 45 bit unsigned integer gives the byte address offset of the first byte of the start code of the next PES packet that contains Program Stream Directory information. This address offset is relative to the 1st byte of the start code of the PES packet which contains this next_directory_offset field. The value '0' is reserved to indicate that there is no next Program Stream Directory information.

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 1st byte of the PES packet containing the access unit referenced. The offset address is relative to the 1st byte of the start code of the PES packet containing this PES_header_position_offset field. The value '0' is reserved to indicate 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.

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 on page 33.

bytes_to_read - This is a 23 bit unsigned integer giving 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 number 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, for video Intra frames this field shall be coded as '1', 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, the value of this field is reserved.

Table 2-38 -- 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 2-39 -- 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 2.6.29 on page 78.

descriptor_tag -- The descriptor_tag is an 8 bit field which identifies each descriptor. Its meaning is given in table 2-40. Some values have normative meaning, some are reserved for future ITU-T | ISO/IEC use and the remaining values may be user defined.

Table 2-40 below 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. This is specified in the descriptor semantics below.

Table 2-40 -- 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.

Table 2-41 -- 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 == 1){		
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 is a 1 bit field which 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 ITU-T Rec. H.262 | ISO/IEC 13818-2 6.3.3, 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 below:

Table 2-42 -- 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 mat 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 is an 8 bit field which is set to the same value as the profile_and_level_indication fields in the video stream.

chroma_format -- This is a 2 bit field which is set to the same value as the chroma_format fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream.

frame_rate_extension_flag -- This is a 1 bit flag which when set to '1' indicates that either or both of the frame_rate_extension_n and frame_rate_extension_d fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream are non-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.

Table 2-43 -- 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 is a 1 bit field which when set to '1' indicates that the audio stream data has the bitrate_index set to '0000'. If set to '0' then the bitrate_index is not '0000'.

ID -- This is a 1 bit field which is set to the same value as the ID fields in the audio stream.

layer -- This is a 2 bit field which is set to the same value as the layer fields in the audio stream.

variable_rate_audio_indicator -- This is a 1 bit flag, which when set to '1' indicates that the associated audio stream may contain audio frames in which the bit rate changes. In all cases the audio stream is intended to be presented without any decoding discontinuity

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 is multiplexed in multiple streams as described in ITU-T Rec. H.222.0 | ISO/IEC 13818-1, ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3.

Table 2-44 -- 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**
}		

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 by the following table 2-45 below.

Table 2-45 -- 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.

Table 2-46 -- 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, 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.

Table 2-47 -- 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 2-48 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 2-48. The definition of access unit for video data is given in 2.1.1 on page 3. At the beginning of a video sequence, the alignment shall occur at the start code of the first sequence header.

Table 2-48 -- 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 2-49 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 frst byte of an audio syncword.

Table 2-49 -- Audio stream alignment values

alignment type	Description
00	reserved
01	syncword
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 the diagram below.

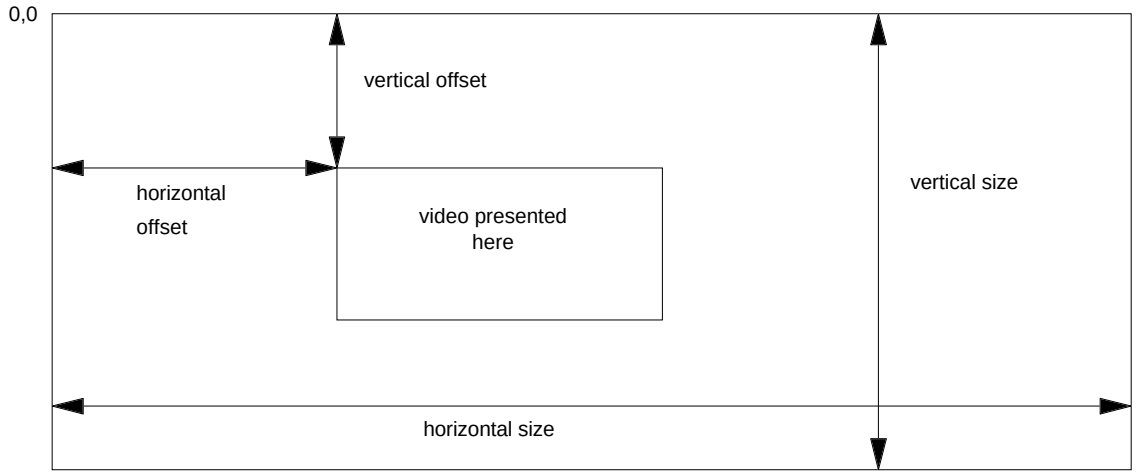

Figure 2-8 -- Target background grid descriptor display area

Table 2-50 -- 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**
}		

2.6.13 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 part 2 of this Recommendation | International Standard.

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 on page 73.

Table 2-51 -- 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**
}		

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.

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.

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 and the program_stream_map. 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.

Table 2-52 -- 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 an 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 an 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.

Table 2-53 -- 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 the 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 by the table 2-54.

Table 2-54 -- 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 should be set. 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:

$$\text{clock\_accuracy\_integer} \times 10^{-\text{clock\_accuracy\_exponent}} \text{ ppm} \qquad (2\text{-}25)$$

If clock_accuracy_integer = 0, then the system clock accuracy is 30 ppm.
When both parts of the descriptor are used, the clock accuracy pertains to the external reference clock.

Table 2-55 -- 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 remultiplexing strategy.

Table 2-56 -- 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 has a value of '1'. When defined, this field has the units of (27 MHz / 300) clock periods, as defined for the LTW_offset. 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. 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.

Table 2-57 -- 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 SC29 which identifies the copyright 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, shall not change.

2.6.26 Maximum bitrate descriptor

Table 2-58 -- 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 the field maximum_bitrate. This integer indicates an upper bound of the maximum 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

Table 2-59 -- 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 will not be defined by ITU-T | ISO/IEC.

2.6.30 Smoothing buffer descriptor

This optional descriptor 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 on page 15.

In the case of Program Streams, bytes of all PES packets of the associated elemetary streams, are input to a buffer SB_n of size given by sb_size, at the time defined by equation 2-20 on page 56.

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 -

> 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, enter the smoothing buffer.

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 2-60 -- 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 -- is coded as a 22 bit positive integer in the sb_leak_rate field. This integer gives 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/second.

sb_size -- is coded as a 22 bit positive integer in the sb_size field. This integer gives 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.

2.6.32 STD descriptor

This descriptor does not apply to Program Streams.

This optional descriptor applies only to the T-STD model and to video elementary streams, and is used as specified 2.4.2 on page 11.

Table 2-61 -- 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 on page 15. 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 on page 15.

This optional descriptor provides information about some charactersistics of the sequence of frame types in the video sequence.

2.6.34 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**
}		bslbf

2.6.35 Semantic definition of fields in IBP_descriptor

closed_gop_flag - This is a one bit flag which 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 is a one bit flag which 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 is fourteen bit unsigned integer which indicates the maximum number of the coded pictures between any two consecutive I-pictures in the sequence. The value zero shall not be used.

2.7 Restrictions on the multiplexed stream semantics

2.7.1 Frequency of coding the system clock reference

The Program Stream shall be constructed so 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 seconds. Thus:

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

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 so 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 seconds. Thus:

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

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

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

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 time stamp coding

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

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

for all n, k, and k" satisfying:

1. $P_n(k)$ and $P_n(k'')$ are presentation units for which presentation time stamps are coded;

2. k and k" are chosen so that there is no presentation unit, $P_{n(}k')$ with a coded presentation time stamp and with $k < k' < k''$; and

3. No decoding discontinuity exists in elementary stream n between $P_n(k)$ and $P_n(k'')$.

This 0,7 second constraint does not apply in the case of still pictures.

2.7.5 Conditional coding of time stamps

For each elementary stream of a Program Stream or Transport Stream, the presentation time stamp (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_time_stamp (DTS) shall appear in a PES packet header if and only if the following two conditions are met:

a) A PTS is present in the PES packet header
b) 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. Note that 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 clause. 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 on page 59. 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 on page 60)is less than or equal to 4 500 000 bits/second if the packet_rate_restriction_flag is set to '1', and less than or equal to 2 000 000 bits / second 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 is set to a value of '1',

$$NP \le \left(tm(i') - tm(i)\right) \times 300 \times \max\left[1, \frac{R_{\max}}{4.5 \times 10^6}\right] \qquad (2\text{-}26)$$

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

$$NP \le \left(tm(i') - tm(i)\right) \times 300 \times \max\left[1, \frac{R_{\max}}{2.0 \times 10^6}\right] \qquad (2\text{-}27)$$

where

$$R_{\max} = 8 \times 50 \times rate\_bound \quad \text{bits/sec} \tag{2-28}$$

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 on page 58 and semantics in 2.5.3.2 on page 58.

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

$t_m(i')$ 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 buffering verifier as specified in part 2 of this Recommendation | International Standard and an additional amount of buffering BS_{add}. BS_{add} is specified as

$$BS_{add} \le MAX\left[6 \times 1024, R_{v\,max} \times 0,001\right] bytes$$

where R_{vmax} is the peak video bit rate

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

$$BS_n \le 4096\, bytes.$$

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 picture rate and the system clock frequency. 2.4.2 on page 11 defines system_clock_frequency. The video picture rate is specified in part 2 of this Recommendation | International Standard or in part 2 of ISO 11172. 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.

$$SCASR = \frac{system\_clock\_frequency}{audio\_sample\_rate\_in\_the\_T-STD} \tag{2-29}$$

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

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

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

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

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

2.8 Compatibility with ISO/IEC 11172

The Program Stream of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 is defined to be forward compatible with ISO/IEC 11172-1. Decoders of the Program Stream as defined in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 shall also support decoding of ISO/IEC 11172-1.

Annex A
(Informative)

Digital Storage Medium Command and Control [DSM CC]

A.0 Introduction

The DSM CC protocol is a specific application protocol intended to provide the basic control functions and operations specific to managing a 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:

1. it is independent of the DSM used,
2. it is independent of whether the DSM is located at a local or remote site,
3. 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.

A.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 needs 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.

A.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.

A.0.3 Benefits

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

The DSM control commands are codes to give end users the guarantee 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.

A.0.4 Basic functions

A.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 on which to perform the succeeding operations. Such operations include creation of a new bitstream. Parameters of this function include:

1. 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)
2. mode (retrieval/storage)

A.0.4.2 Retrieval

The DSM CC provides the means to:

1. play an identified ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream,
2. play from a given presentation time,
3. set the playback speed (normal or fast),
4. 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),
5. set the direction (forward or reverse),
6. pause,
7. resume,
8. change the access point in the bitstream,
9. stop.

A.0.4.3 Storage

The DSM CC provides the means to:

1. cause storage of a valid bitstream for a specified duration,
2. 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.

A.1 General elements

A.1.1 Scope

The scope of this work consists of the development of an 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.

A.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 a ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream and is also [optionally] capable of generating a 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 A-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.

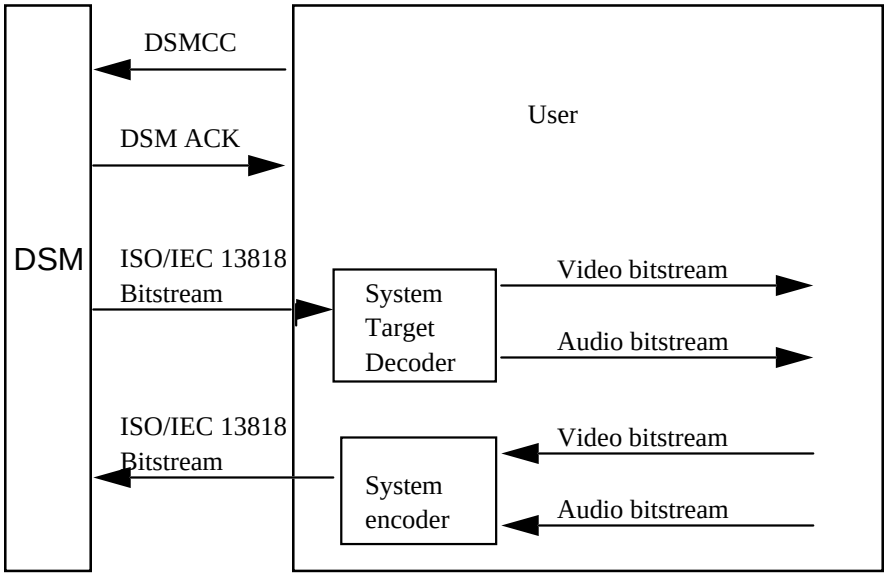

Figure A-1 -- Configuration of DSM CC application

A.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 ~~subclauses~~ A.2.2 through A.2.9. The DSM CC bitstream can be transmitted both as a stand alone bitstream and in a 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 A-2 on page 89. 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.

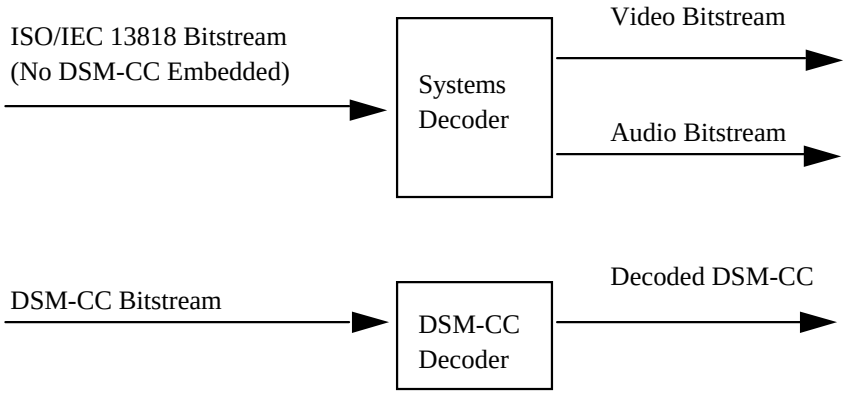

Figure A-2 -- DSM CC bitstream decoded as a standalone bitstream

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 packetized into an packetized element stream (PES) according to the syntax described in 2.4.3.6 of this Recommendation | International Standard. Then the PES is 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 A-3.

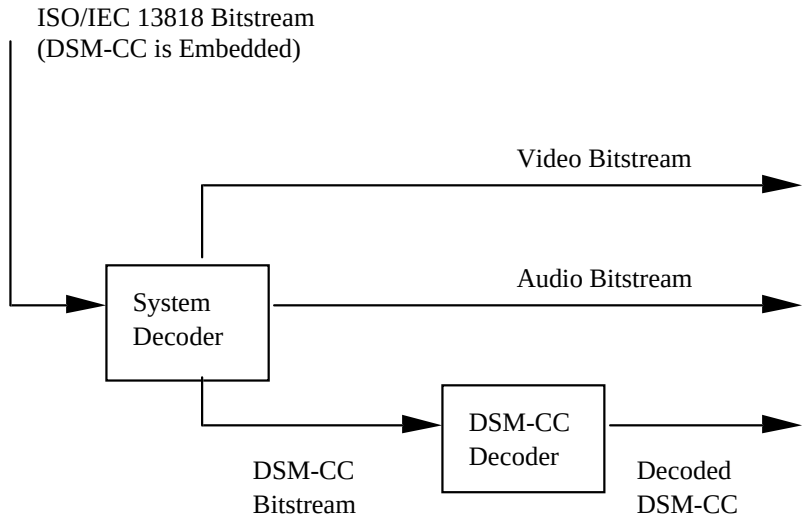

Figure A-3 -- DSM CC bitstream decoded as part of the system bitstream

In figure A-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, which is equal to the binary code '1111 0010' as defined by the stream_id table **2-19** on page 36 of the System part of the IS. Once the DSM CC bitstream is identified, the succeeding bitstream shall be directed to the DSM CC decoder until the next non-DSM CC stream_id is detected.

A.2 Technical elements

A.2.1 Definitions

A.2.1.1 DSM CC -- Digital Storage Media Control Commands that are specified by this International Standard 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.

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

A.2.1.3 MPEG bitstream -- An ISO/IEC 11172-1 Systems stream, Program Stream or Transport stream.

A.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.

A.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'.

A.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.

A.2.1.7 DSM CC Bitstream -- A sequence of bits satisfying the syntax of section A.2.2.

A.2.2 Specification of DSM CC syntax

- Every DSM control command shall start with a start code, which is a 32-bit string derived from the packet_start_code_prefix and the stream_id in the Table A-1 shown below;
- 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 Section 2.4.3.6 of ITU-T Rec. H.222.0 | ISO/IEC 13818-1, the packet_start_code_prefix is identical field as the packet_start_code_prefix and the packet_length is identical field as the PES_packet_length as specified in Section 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 immediately follow the packet_length;
- An acknowledgment stream shall be provided by the DSM control bitstream receiver after the requested operation is started or is done, depending on the specific command received;
- The DSM is responsible at all times for providing a legal ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems bitstream. This may include manipulating the trick mode bits defined in the system part of the IS.

Table A-1 -- ITU-T Rec. H.222.0 | ISO/IEC 13818-1 DSM CC

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()		
}		
}		

A.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 is a 8-bit field specifying the bitstream identification that takes the value '1111 0010' for the DSM CC bitstream.

packet_length -- A 16-bit field specifying the number of byte in the DSM CC packet following the last byte of the field.

command_id -- This is an 8-bit unsigned integer. It identifies whether a bitstream is a control command or an acknowledgment. The value is defined in table A-2 below.

Table A-2 -- Command_id assigned values

value	command_id
0x00	forbidden
0x01	control
0x02	ack
0x03-0xFF	reserved

A.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 is 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 is 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 is ignored.

Table A-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()		
}		
}		
}		

A.2.5 Semantics of fields in control layer

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

reserved_bits -- This is a specific number of bits that is reserved for future extension of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 DSM control commands. These bits shall be set to '0' in all the syntax of DSM CC.

select_flag -- This one-bit flag specifies a bitstream selection operation when it is set to '1'.

retrieval_flag -- This is a one-bit flag specifying that a specific retrieval [playback] action will occur when the flag is set to '1'. The operation starts from the current operational PTS value.

storage_flag -- This is a one-bit flag specifying that a storage operation is to be executed when the flag is set to '1'.

bitstream_ID -- This is a 32 bit string. They are combined from three fields as defined in the syntax to form an unsigned integer to specify 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 is a 5 bit unsigned integer to specify which mode of bitstream operation is requested. The table below specifies the defined modes.

Table A-4 -- Select mode assigned values

code	mode
0	forbidden
1	storage
2	retrieval
3-31	reserved

storage -- The following syntax elements are a bitstream storage command.

retrieval -- The following syntax elements are a bitstream retrieval command.

jump_flag -- This is a one-bit flag specifying to jump 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 activity 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 is a one-bit flag specifying to play a bitstream for a certain time period. The speed, direction, and play duration are additional parameters in the bit stream when the play flag is set to '1'. 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.

A.2.6 Acknowledgement layer

Constraints on setting flags in DSM CC control:

Only one of the acks for select, retrieval, storage, and error shall be selected (set to “1”) for each DSM ack bitstream.

Table A-5 -- DSM CC Acknowledgement

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()		
}		
}		

A.2.7 Semantics of fields in acknowledgement layer

select_ack -- This is a 1-bit code. When it is set to '1', it specifies that the ack() command is to acknowledge a select command.

retrieval_ack -- This is a 1-bit code. When it is set to '1', it specifies that the ack() command is to acknowledge a retrieval command.

storage_ack -- This is a 1-bit code. When it is set to '1', it specifies that the ack() command is to acknowledge a storage command.

error_ack -- This 1-bit code identifies a DSM error when set to '1'. Currently 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, cmd_status is undefined. In either case, the current bitstream is still selected.

cmd_status -- This is a 1-bit flag. It provides to a DSM control commands initiator the response of the DSM command receiver. If it is set to '1', it specifies that the command is accepted, otherwise, the command is rejected. Depending the type of command received, there are the following semantics:

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.

A.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.

Table A-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**
}		
}		

A.2.9 Semantics of fields in time code

infinite_time_flag -- This is a one-bit flag to specify an infinite time period when this flag is set to '1'. This flag is set to '1' in such applications when 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 B
(Normative)

CRC Decoder Model

B.0 CRC decoder model

The 32 bit CRC Decoder Model is specified in figure B-1 below.

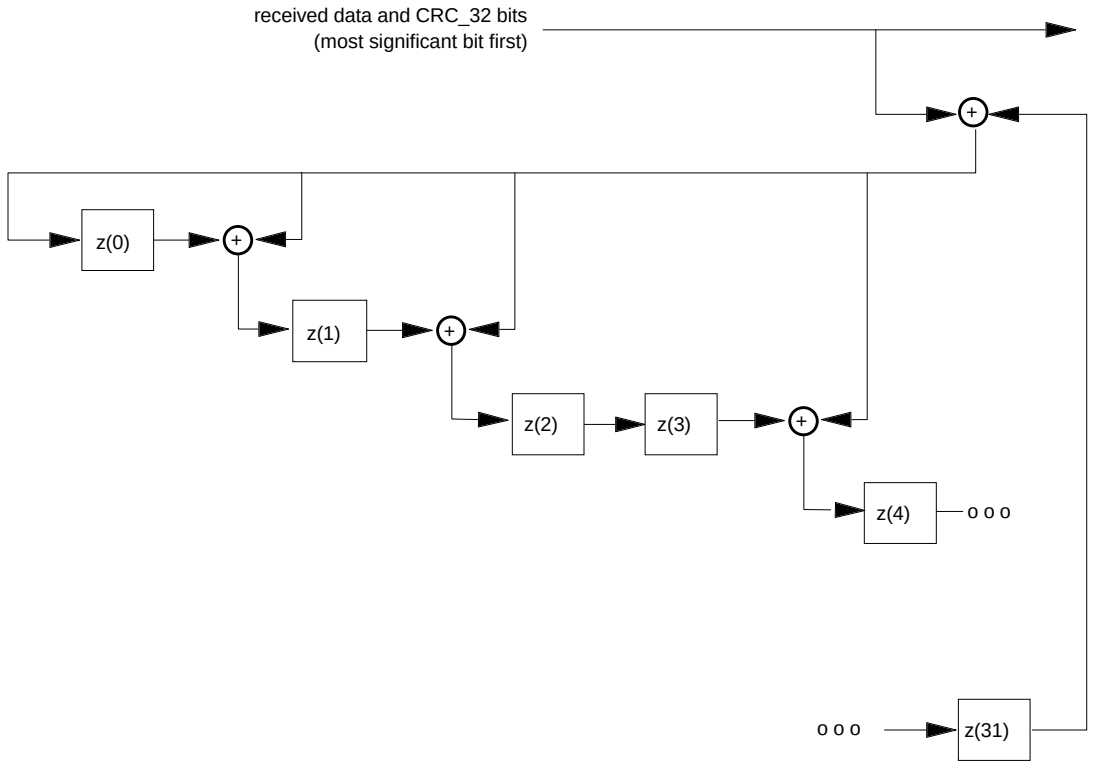

Figure B-1 -- 32 bit CRC decoder model

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. See figure above.

This is the CRC calculated with the polynomial:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$$

At the input of the CRC decoder bytes are received. Each byte is shifted into the CRC decoder one bit at a time, with the most significant bit (msb) first, i. e. from byte 0x01 (the last byte of the startcode prefix), first the seven '0's enter the CRC decoder, 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 case of no errors, each of the outputs of z(i) has to be zero. At the CRC encoder the CRC_32 field is encoded with such value that this is ensured.

Annex C
(Informative.)

Program Specific Information

C.0 Explanation of Program Specific Information in Transport Streams

Section 2.4.4 contains the normative syntax, semantics and text concerning Program Specific Information. In all cases, compliance with the constraints of section 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

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 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 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 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 ISO in a subsequent revision of the 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.

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 the 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. ITU-T Rec. H.222.0 | ISO/IEC 13818-1 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 standard~~.~~ (~~See~~ refer to **Error! Reference source not found.**~~section 2.1.37)~~ ~~"~~**program [system]**~~): A program is a collection of elementary streams within a common timebase.".~~ For a Transport Stream the time base is defined by the PCR~~In concept this means that a program is defined as consisting of all the elementary streams which refer to a common PCR clock~~. 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 setup 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 ~~elementary streams~~ 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 ~~elementary streams~~ 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 elements ~~elementary stream~~ 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 ~~elementary streams~~ 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 the 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 sections, each PSI table is described.

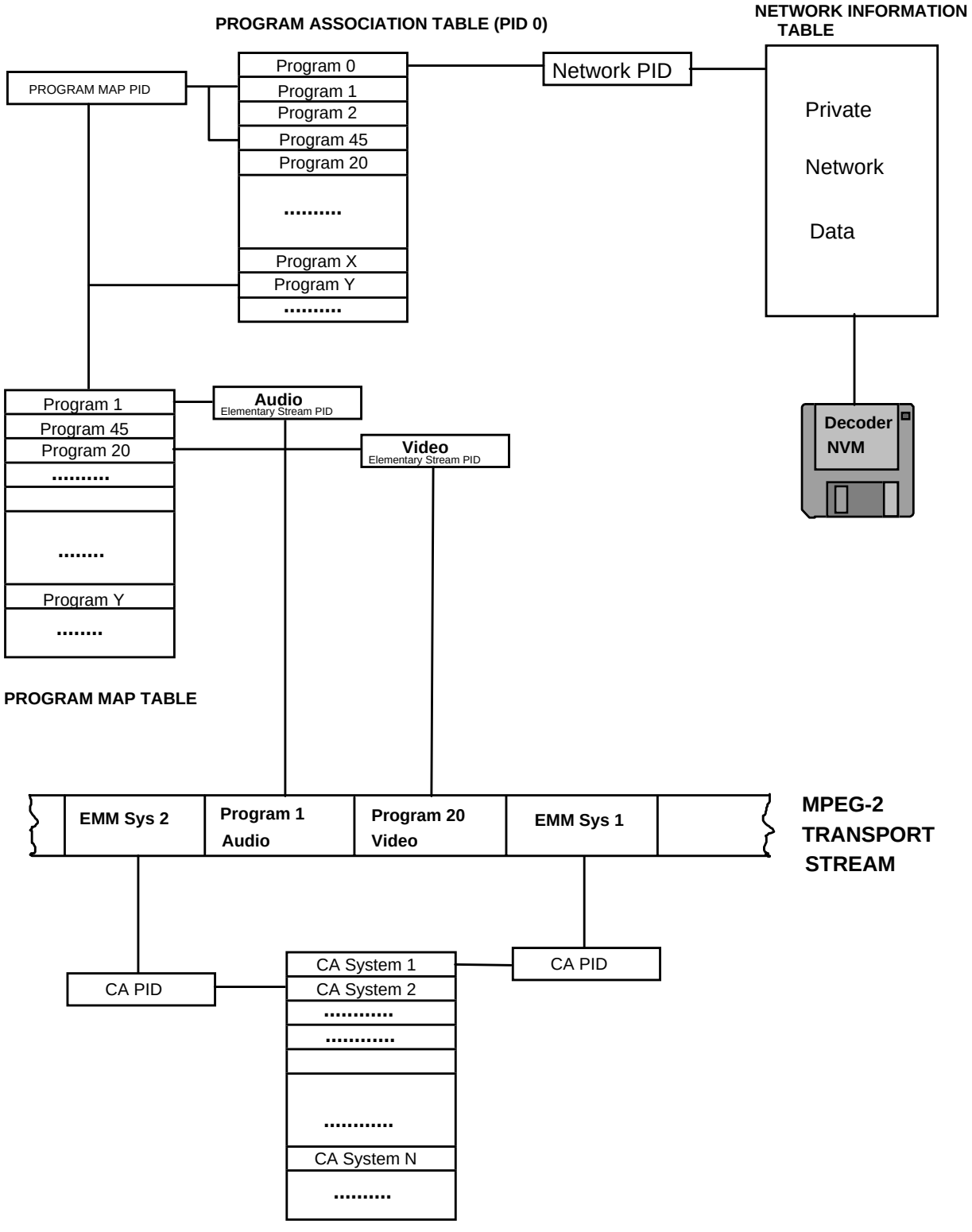

Figure C-1 -- Program and network mapping relationships

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 1 024 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 ~~elementary streams~~ 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.

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 requires a minimum of program identification: program number, PCR PID, ~~elementary~~ stream types and program elements ~~elementary stream~~ PIDs. Additional information for either programs or elementary streams may be conveyed by use of the descriptor() construct. ~~See~~ Refer to section C.~~8~~9.6.

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.

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

Note 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).

C.8.4 Network Information Table

The contents of the NIT are private and not specified by this 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 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 ~~elementary streams~~ 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 follows:

Table C-1 -- Composite_descriptor

Syntax	**No. of bits**	**Mnemonic~~Identifier~~**
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~~Identifier~~**
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 section 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 section examines both the bitrate and signal acquisition time impacts of the PSI syntax sections 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 ~~elementary stream~~ 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 ~~t~~Transport Stream packet (if no adaptation field is present). Similarly, seven (7) TS_program_map_sections fit into a single ~~t~~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 (bps)

		Number of Programs Per Transport Stream				
		1	**5**	**10**	**32**	**128**
Frequency of PA Table Information (sec^{-1})	**1**	1504	1504	1504	1504	4512
	10	15040	15040	15040	15040	45120
	25	37600	37600	37600	37600	112800
	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 (bps)

		Number of Programs Per Transport Stream				
		1	**5**	**10**	**32**	**128**
Frequency of PM Table Information (sec^{-1})	**1**	1504	1504	3008	7520	28576
	10	15040	15040	30080	75200	285760
	25	37600	37600	75200	188000	714400
	50	75200	75200	150400	376000	1428800
	100	150400	150400	300800	601600	2857600

Using a frequency of 25Hz 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 25Hz 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-Mbps programs: 26.5 Mbps (includes transport overhead)
total PSI bandwidth: 75.2 kbps~~Kbps~~
CA bandwidth: 500 kbps~~Kbps~~

total ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport bandwidth: 27.1 Mbps

PSI Overhead: **0.28 %**

OC-3 fiber channel (155 Mbps):

32 3.9-Mbps programs: 127.5 Mbps (includes transport overhead)
total PSI bandwidth: 225.6 kbps~~Kbps~~
CA bandwidth: 500 kbps~~Kbps~~

total ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport bandwidth: 128.2 Mbps

PSI Overhead: **0.18%**

C-band satellite transponder:

128 256-kbps~~Kbps~~ audio programs: 33.5 Mbps (includes transport overhead)
total PSI bandwidth: 826.4 kbps~~Kbps~~
CA bandwidth: 500 kbps~~Kbps~~

total ITU-T Rec. H.222.0 | ISO/IEC 13818-1 transport bandwidth: 34.7 Mbps

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 25Hz)

for the PSI data may be used to decrease the impact on channel acquisition time with only modest bitrate demand increases.

Annex D
(Informative.)

ITU-T H.222.0 | ISO/IEC 13818-1 Systems Timing Model and Application Implications

D.0 Introduction

The ITU-T Rec. H.262 | ISO/IEC 13818-2 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 picture 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 the following figure:

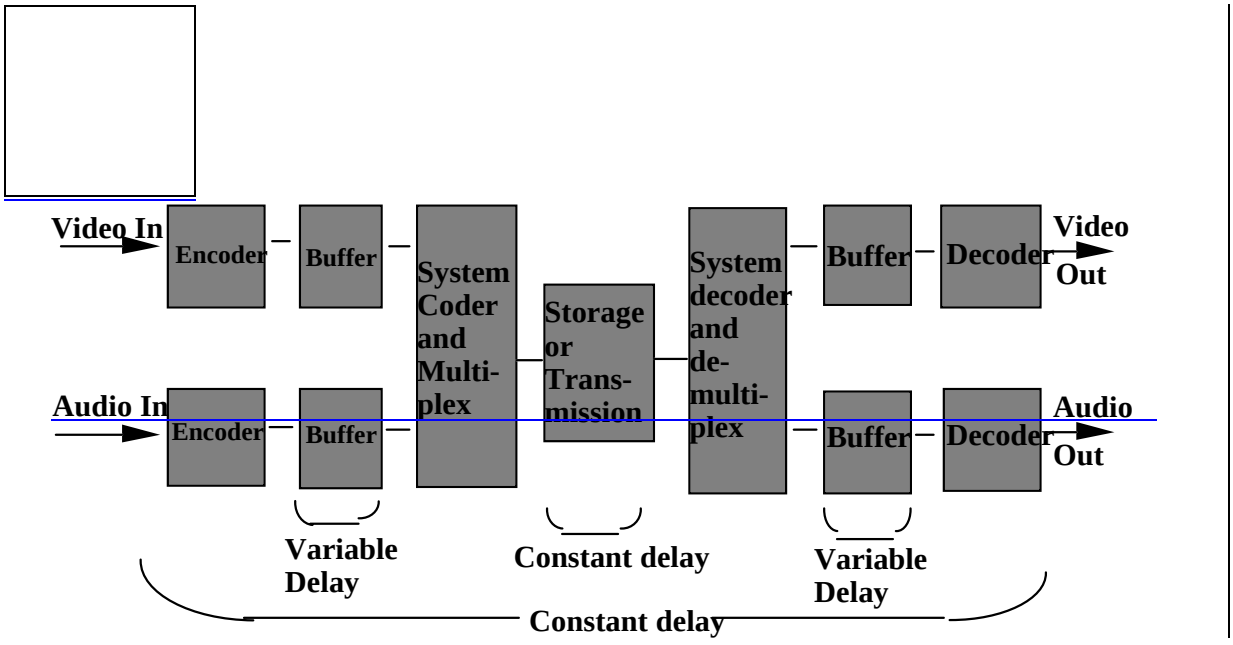

Figure D-1 -- Constant delay model

As indicated in the figure, the delay from the input to the encoder to the output or presentation from the decoder is constant in this model[1], 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 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 time stamps that indicate the correct presentation and decoding timing of audio and video, as well as to create time stamps that indicate the instantaneous values of the system clock itself at sampled intervals. The time stamps that indicate the presentation time of audio and video are called Presentation Time Stamps (PTS); those that indicate the decoding time are called Decoding Time Stamps (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 time stamps that are created from it, and the recreation of the clock in the decoder and the correct use of the time stamps 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 behavior 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

[1]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.

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 ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems data are time stamps 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 International Standard, and the equivalent information is conveyed in the 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 26,12ms, and video PUs at a frame rate of 29,97 Hz have a duration of approximately 33,76ms. 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 picture 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 time stamps and buffer behavior 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 behavior of the decoder buffers. These topics are covered in more detail in later sections 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 reordering 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 700ms 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 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 90kHz 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 picture 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 behavior 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 time stamps.

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

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 time stamps. 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 100ms in Transport Streams, or up to 700ms 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 sub-section 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 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.

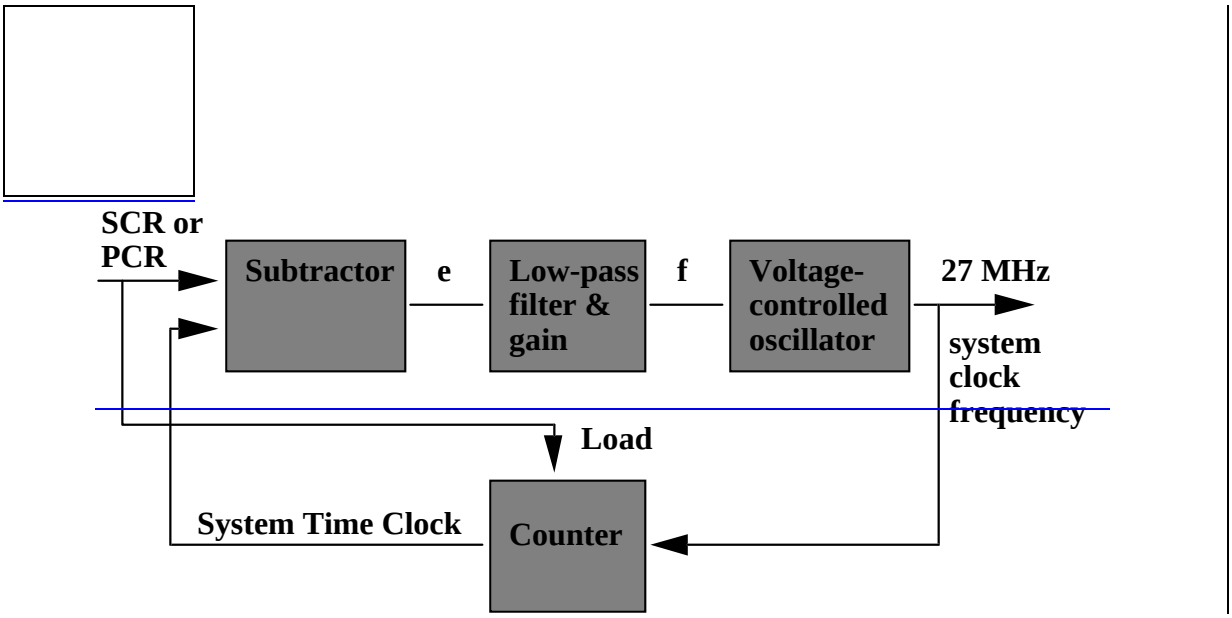

Figure D-2 -- STC recovery using PLL

The diagram 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.

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 90kHz 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 27MHz, 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 27MHz; 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 90kHz base value which is derived by counting the 90kHz results in a 33 bit counter. The 33 bit, 90kHz 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 (700ms) or PCRs (100ms) 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 on page 113, 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 27MHz 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 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 remultiplex 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. Below, three different multiplex strategies are shown together with the appropriate value for del_{const}.

Table D-1 -- Remultiplexing 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 remultiplexing.

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

In systems which include remultiplexors 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 remultiplexor 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 time stamps, 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 picture 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 3ppm, with a maximum short term jitter of 1 ns per horizontal line time and a maximum long term drift of 0,1Hz 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 is 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 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 on page 113, 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 27MHz 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 analog filter techniques. With digital LPF implementations, when the frequency term **f** is the input to an analog VCO, **f** is quantized by a digital to analog 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 27MHz 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 behavior 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 on page 113 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 on page 113 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 section, 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 27MHz 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 27MHz frequency and the encoder's 27MHz, 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 27MHz 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 the 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 on page 113. 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
(Informative.)

Data Transmission Applications

E.0 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 data PES packet, it is not always possible to identify the end of a data PES packet by the beginning of the next data PES packet, as one (or more) Transport packet carrying data PES packets may be lost.

E.1 Suggestion

A suitable solution is to transmit, just after an associated PES packet, the following PES packet. When there is no PES packet to send, a PES packet without payload may be send instead.

Following 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
(Informative.)

Graphics of Syntax for ITU-T H.222.0 | ISO/IEC 13818-1

F.0 Introduction

This part of the standard is an informative annex presenting graphically the Transport Stream and Program Stream syntax . This part in no way replaces the any normative section.

In order to produce clear drawings, not all fields have been fully described or represented. Reserved fields may be omitted or signaled by shaded areas. Fields length are indicated in bits.

F.0.1 Transport Stream syntax

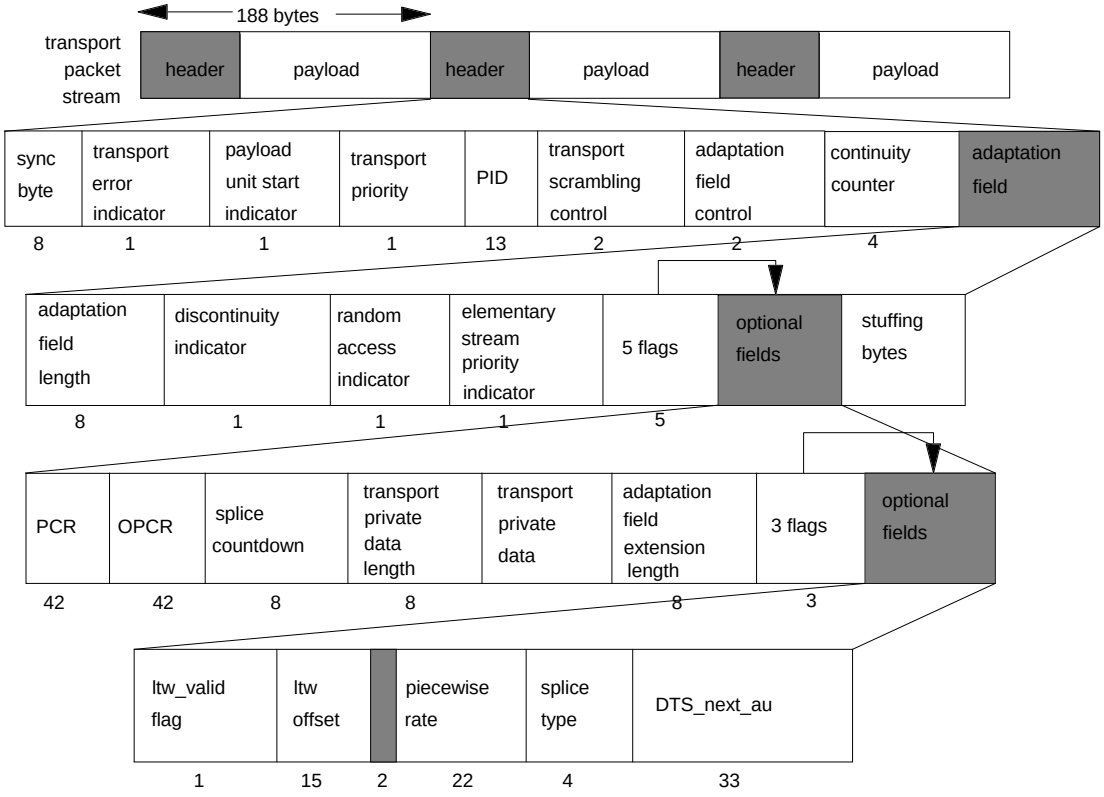

Figure F-1 -- Transport Stream syntax diagram

F.0.2 PES packet

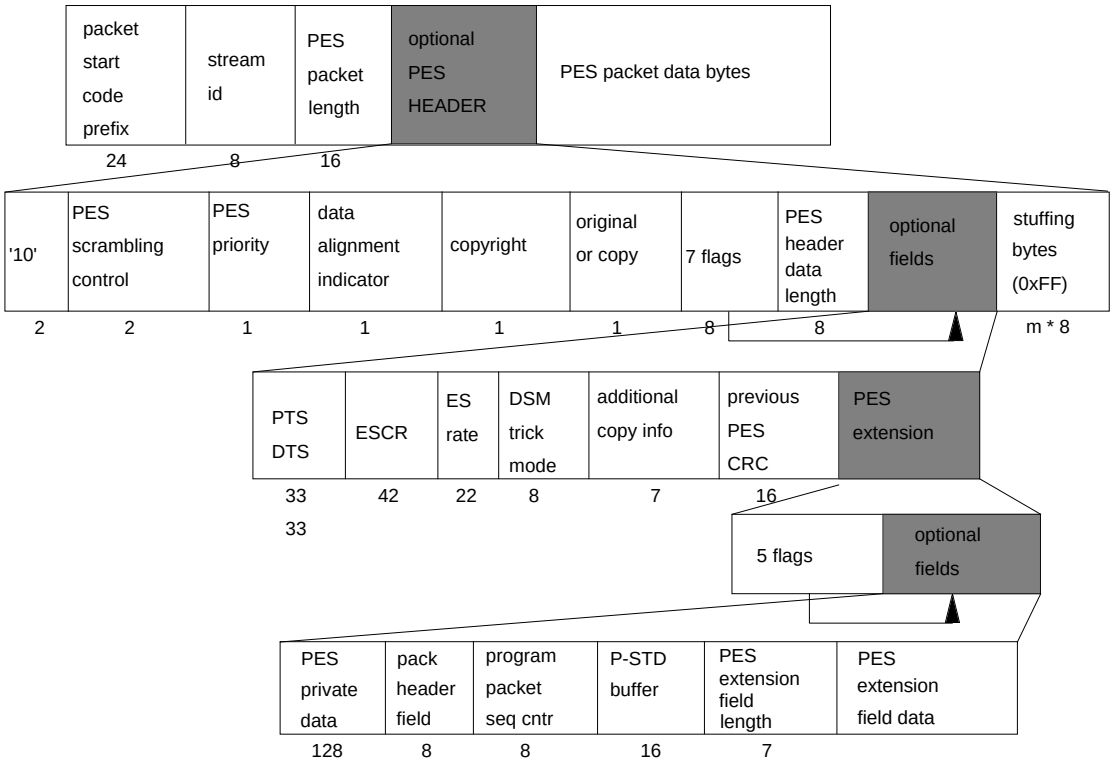

Figure F-2 -- PES packet syntax diagram

F.0.3 Program Association Section

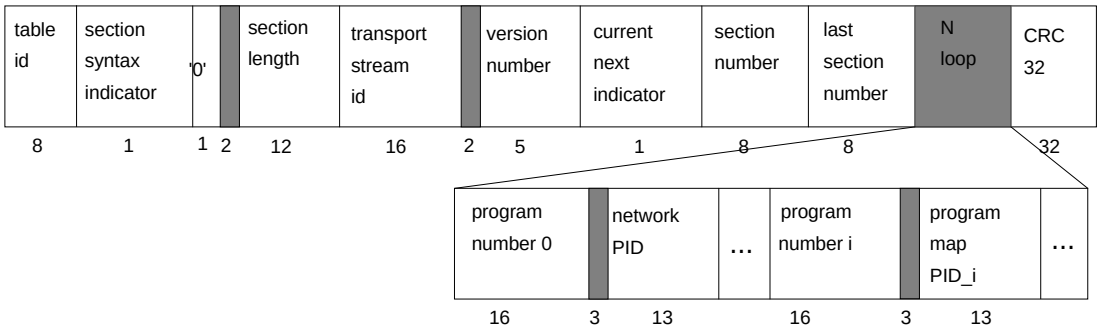

Figure F-3 -- Program association section diagram

F.0.4 CA section

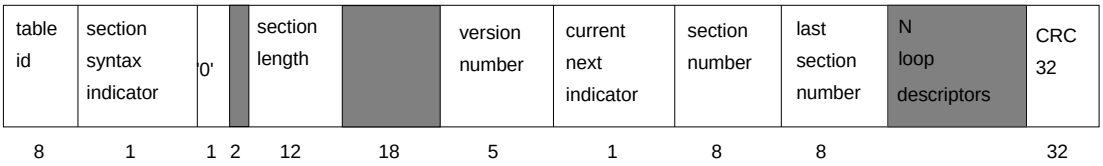

Figure F-4 -- Conditional access section diagram

F.0.5 TS program map section

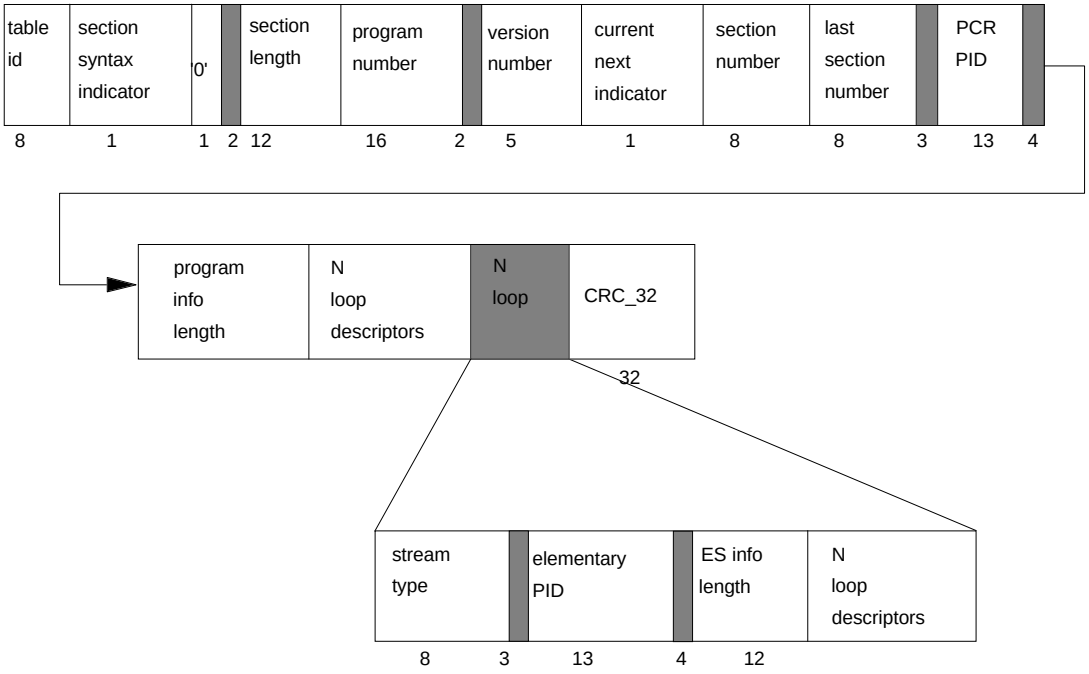

Figure F-5 -- TS program map section diagram

F.0.6 Private section

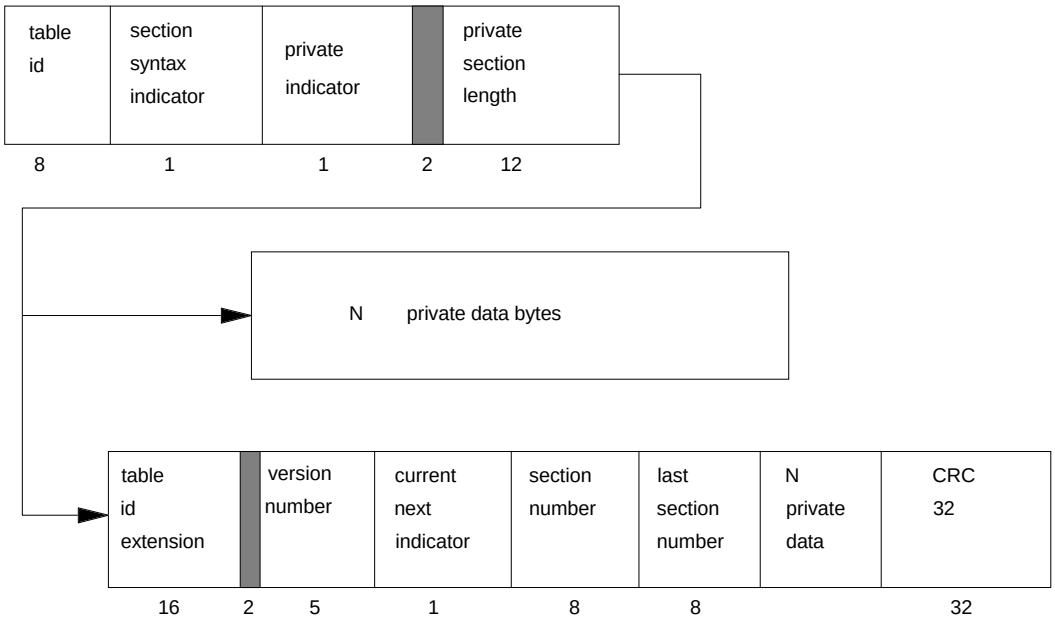

Figure F-6 -- Private section diagram

F.0.7 Program Stream

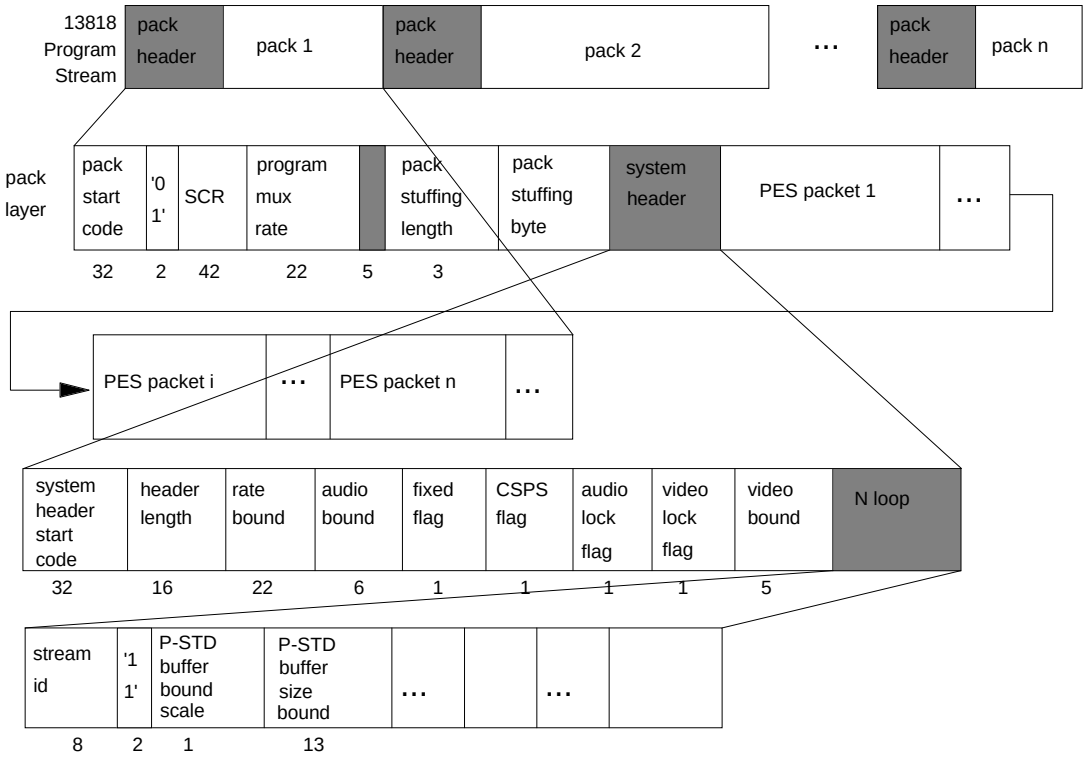

Figure F-7 -- Program Stream diagram

F.0.8 Program Stream map

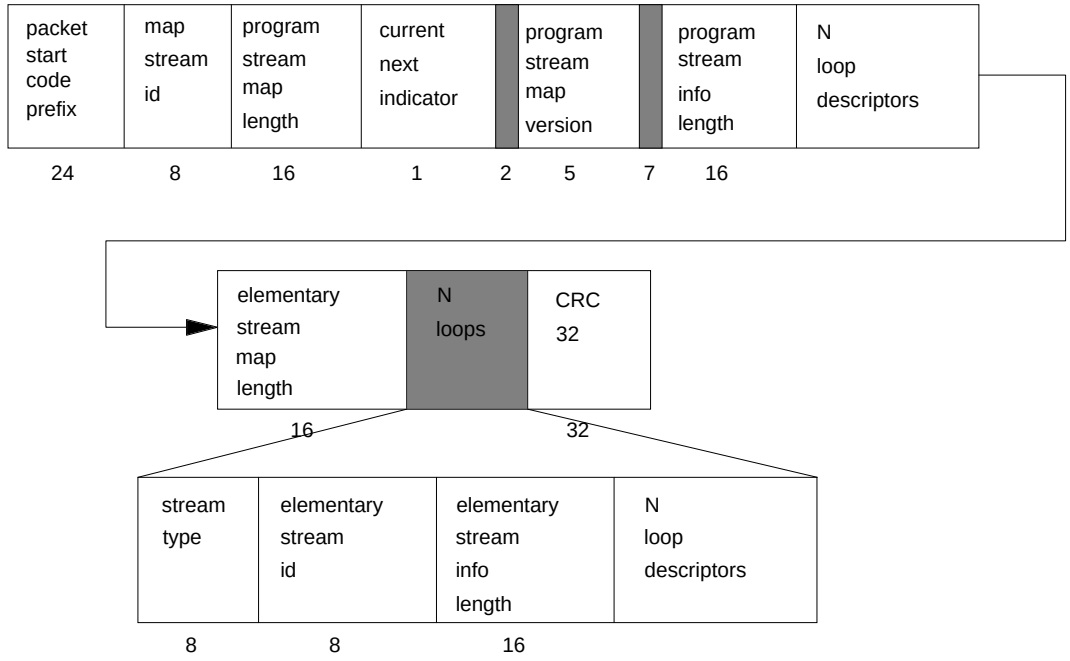

Figure F-8 -- Program Stream map diagram

Annex G
(Informative.)

General Information

G.0 General Information

G.0.1 Sync Byte Emulation

In the choice of PID values it is recommended that the regular emulation of sync bytes should 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 is 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 ~~all the bits for picture_next that are in the STD are instantaneously removed, 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 ~~the STD buffer~~, all the ~~these~~ bits of picture_next are removed and picture_next is displayed. If all the bits of picture_next are not in the ~~STD~~ buffer EB_n or B_n, the above process of ~~removing those bits that are present and~~ redisplaying 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 wrong 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 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
(Informative.)

Private Data

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 ITU-T Rec. H.222.0 | ISO/IEC 13818-1 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 2-3 on page 22.

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 2-36 on page 64. It is permitted for Transport Stream packets containing private data to also include adaptation_field()s.

2. Transport Stream Adaptation Field table 2-7 on page 24.

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 2-18 on page 33

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 2-36 on page 64. 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 - 255 as identified in table 2-40 on page 68. These descriptors may be placed within a program_stream_map() table 2-36 on page 64, a CA_section() table 2-28 on page 49, a TS_program_map_section(), table 2-29 on page 50 and in any private section(), table 2-30 on page 52.

Specifically private_data_bytes also appear in the CA_descriptor().

5. Private Section

The private_section table 2-30 on page 52 provides a further means to carry private data also in two forms. This type of elementary stream may be identified under stream_type table 2-36 on page 64 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
(Informative.)

List of companies having provided patent statements for ITU-T Rec. H.222.0 | ISO/IEC 13818

I.0 Companies having provided patent statements for ITU-T Rec. H.222.0 | ISO/IEC 13818-1

The user's attention is called to the possibility that - for some of the processes specified in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 conformance with this Recommendation | International Standard may require use of an invention covered by patent rights.

By publication of this part of ITU-T Rec. H.222.0 | ISO/IEC 13818-1, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. However, each company listed in this annex has undertaken to file with the Information Technology Task Force (ITTF) a statement of willingness to grant a license under such rights that they hold on reasonable and non-discriminatory terms and conditions to applicants desiring to obtain such a license. Information regarding such patents can be obtained from the following organizations.

The table summarizes the formal patent statements received and indicates the parts of the standard to which the statement applies. The list includes all organizations that have submitted informal patent statements. However, if no "X" is present, no formal patent statement has yet been received from that organization.

Table I-1 -- List of companies supplying patent statements

Company	V	A	S
AT&T	X	X	X
BBC Research Department			
Bellcore	X		
Belgian Science Policy Office	X	X	X
BOSCH	X	X	X
CCETT			
CSELT	X		
David Sarnoff Research Center	X	X	X
Deutsche Thomson-Brandt GmbH	X	X	X
France Telecom CNET			
Fraunhofer Gesellschaft		X	X
GC Technology Corporation	X	X	X
General Instruments			
Goldstar			
Hitachi, Ltd.			
International Business Machines Corporation	X	X	X
IRT		X	
KDD	X		
Massachusetts Institute of Technology	X	X	X
Matsushita Electric Industrial Co., Ltd.	X	X	X
Mitsubishi Electric Corporation			

Company	V	A	S
National Transcommunications Limited			
NEC Corporation		X	
Nippon Hoso Kyokai	X		
Nippon Telegraph and Telephone	X		
Nokia Research Center	X		
Norwegian Telecom Research	X		
Philips Consumer Electronics	X	X	X
OKI			
Qualcomm Incorporated	X		
Royal PTT Nederland N.V., PTT Research (NL)	X	X	X
Samsung Electronics			
Scientific Atlanta	X	X	X
Siemens AG	X		
Sharp Corporation			
Sony Corporation			
Texas Instruments			
Thomson Consumer Electronics			
Toshiba Corporation	X		
TV/Com	X	X	X
Victor Company of Japan Limited			

Annex J
(Informative.)

Systems conformance and real-time interface

K.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 Part of this Recommendation | International standard. These specifications include, among other requirements, a System Target Decoder (T-STD and P-STD) which specifies the behavior 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 specification.

In addition, there is a real-time interface specification for input of Transport Streams and Program Streams to a decoder. This specification allows standardization of the interface between MPEG decoders and adaptors 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 specification covers the real-time delivery behavior 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 K
(Informative.)

Interfacing Jitter-Inducing Networks to MPEG-2 Decoders

K.0 Introduction

In this Annex the expression system stream will be used to refer to both MPEG-2 transport streams and MPEG-2 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 Section 2.

Three examples of network encoding to enable the building of jitter-smoothing network adapters are discussed in Section 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.

Section 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 Section 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 Section 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.

K.1 Network compliance models

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

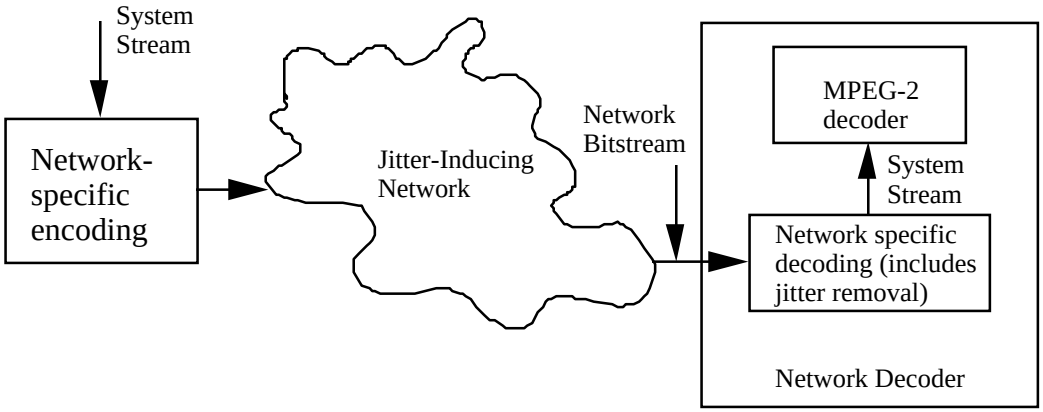

Figure K-1 -- Sending system streams over a jitter-inducing network

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 MPEG-2 decoder. The MPEG-2 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-MPEG-2 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.

K.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 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.

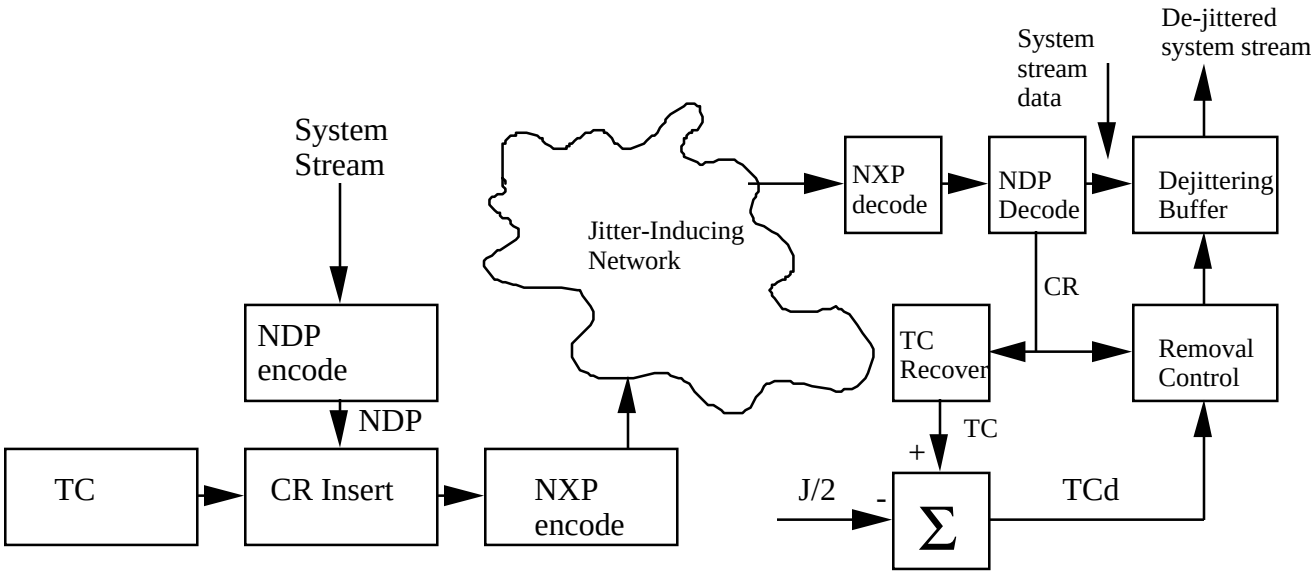

Figure K-2 -- Jitter smoothing using network-layer timestamps

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 samples (CRs) taken from a time clock (TC). The CRs and the TC are analagous 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.

K.3 Example decoder implementations

K.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.

K.3.2 Integrated decoder

The example presented in Section 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 section 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 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 2.

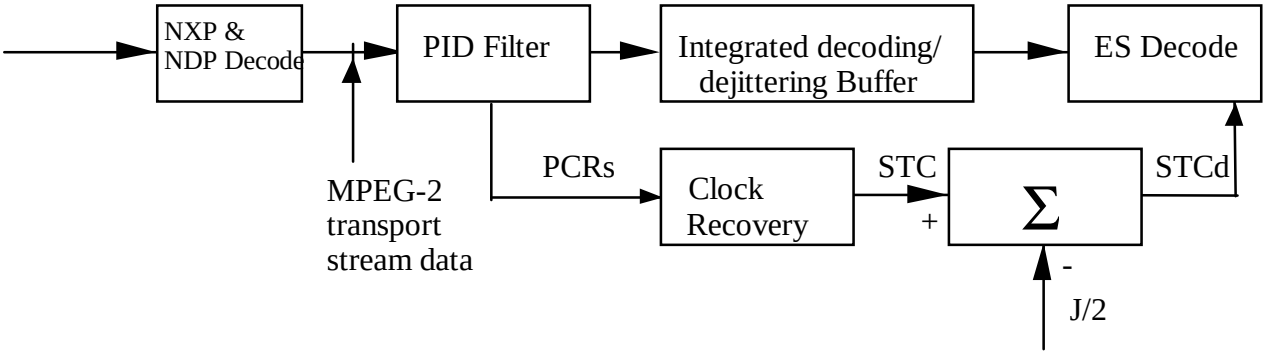

Figure K-3 -- Integrated dejittering and MPEG-2 decoding

The incoming network packets are reassembled into MPEG-2 transport stream data by the NXP and NDP decode functions. The jittered MPEG-2 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 seconds 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

$$B_{size} = B_{dec} + B_{mux} + B_{OH} + 512 + B_J$$
$$= B_n + 512 + B_J$$

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 L
(Informative.)

Splicing Transport Streams

L.0 Introduction

For the purpose 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 of this Specification). 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 (N.B: 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 of this Specification 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.

L.1 The different types of splicing point

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

L.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 time stamp values.

L.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~~au~~ field. This DTS value is expressed in the time base which is valid in the splicing packet.

- 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.

L.2 Decoder behaviour on splices

L.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 (section 2.7.6 of this Recommendation | International Standard) 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 this Recommendation | International Standard.

L.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 part 2 of this Recommendation | International Standard, 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 Part 2 of this Recommendation | International Standard 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 this Recommendation | Internationl Standard.

L.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 delived 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 neighbourhood 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 table 2-8 through table 2-17 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-STD 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:

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