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

ISO/IEC JTC1/SC29/WG11N0701
MPEG94/
March 1994

Source : **Systems Subgroup**
Title : **ISO/IEC 13818-1 (MPEG-2 Systems)**
Purpose : **Review**
Status : **Interim edited CD**

The following is an interim version of the systems specification for the purpose of documenting the agreements reach at the March 1994 Paris meeting and the editing changes submitted by national body members and approved by the systems subgroup.

Some significant technical matters were resolved but are not fully documented in this document. Examples include:

- **T-STD - leaky bucket model**
- **PES STD to constrain PES overhead**
- **Splicing support**
- **PCR tolerance**
- **Remultiplexing support**

TITLE PAGE PROVIDED BY ISO

1743 Mon 24 Mar 2003

CONTENTS

Page

CONTENTS ii
List of Figures iv
List of Syntax Tables v
List of Equations vii
FOREWORD viii
INTRODUCTION - PART 1 SYSTEMS ix
0.1 Transport Stream xi
0.2 Program Stream xiii
0.3 Conversion between Transport Stream and Program Stream xiii
0.5 Timing model xiv
0.6 Conditional access xv
0.7 Multiplex-wide Operations xv
0.8 Individual stream operations xv
0.8 1 De-multiplexing xv
0.8 2 Synchronization xvi
0.8 3 Relation to compression layer xvi
0.9 System reference decoder xvi
0.10 Applications xvii
Section 1: General 1
1.1 Scope 1
1.2 References 1
2 TECHNICAL ELEMENTS 3
2.1 Definitions 3
2.2 Symbols and abbreviations 5
2.2.1Arithmetic operators 5
2.2.2 Logical operators 6
2.2.3 Relational operators 6
2.2.4 Bitwise operators 7
2.2.5 Assignment 7
2.2.6 Mnemonics 7
2.2.7 Constants 8
2.3 Method of describing bit stream syntax 8
2.4 Bitstream requirements 9
2.4.1 Transport Stream coding structure and parameters 9
2.4.2 Transport Stream system target decoder 11
2.4.3 Specification of the system Transport Stream syntax 16
2.4.4 Program Stream coding structure and parameters 29
2.4.5 Program Stream system target decoder 30
2.4.6 Specification of the Program Stream syntax 33
2.4.7 Program specific information 43
2.4.8 Program and elementary stream descriptors 52

2.4.9 Restrictions on the multiplexed stream semantics 60
2.4.10 Constrained system parameter stream 63
2.4.11 Compatibility with ISO/IEC 11172 65

Annexes

A Digital Storage Medium Command and Control [DSM CC] 66
B ISO/IEC 13818 Systems Timing Model and Application Implications 77
C Program Specific Information 87
D Service Information Descriptors 94
E Data Transmission Applications 96
F Graphics of Syntax for ISO/IEC 13818 97
G General Information 102
H List of companies having provided patent statements for ISO/IEC 13818 103

List of Figures

0-1 -- Simplified overview of ISO/IEC 13818 scope x
0-2 -- Prototypical transport demultiplexing and decoding example xii
0-3 -- Prototypical transport multiplexing example xii
0-4 -- Prototypical transport to program stream conversion xii
0-5 -- Prototypical program stream decoder xiii
2-6 -- ISO/IEC 13818 Transport Stream system target decoder notation 11
2-7 -- ISO/IEC 13818 Program Stream system target decoder notation 30
A-1 -- Configuration of DSM CC application 68
A-2 -- DSM CC bitstream decoded as a standalone bitstream 69
A-3 -- DSM CC bitstream decoded as part of the system bitstream 69
B-1 -- Constant delay model 77
B-2 -- STC recovery using PLL 81
C-1 -- Program and network mapping relationships 89
F-1 -- Transport Stream syntax diagram 97
F-2 -- PES packet syntax diagram 98
F-3 -- Conditional access section diagram 98
F-4 -- TS program map section diagram 99
F-5 -- Private section diagram 100
F-6 -- Program Stream diagram 101
F-7 -- Program Stream map diagram 101

List of Syntax Tables

2-1 -- Next start code 9
2-2 -- ISO/IEC 13818 Transport Stream 16
2-3 -- ISO/IEC 13818 transport header 17
2-4 -- Scrambling control values 18
2-5 -- Adaptation field control values 18
2-6 -- Transport Stream adaptation header 19
2-7 -- PES Packet 21
2-8 -- PES scrambling control values 24
2-9 -- Trick mode control values 26
2-10 -- Coefficient selection values 27
2-11 -- Freeze frame control values 27
2-12 -- Fast reverse control values 28
2-13 -- Program specific information 30
2-15 -- Program association section 32
2-16 -- table_id assignment values 33
2-17 -- Conditional access section 34
2-18 -- Transport Stream program map section 36
2-19 -- Private section 38
2-20 -- ISO/IEC 13818 Program Stream 43
2-21 -- ISO/IEC 13818 Program Stream pack 43
2-22 -- Program Stream pack header 44
2-23 -- Program Stream system header 45
2-24 -- Stream_id table 47
2-25 -- Program Stream map 49
2-26 -- Stream type assignments 50
2-27 -- PES packet header for Program Stream directory 51
2-28 -- PES packet payload for Program Stream directory 52
2-29 -- Intra_coded indicator 53
2-30 -- Coding_parameters indicator 53
2-31 -- Stream descriptors 54
2-32 -- Video stream descriptor 54
2-33 -- Video coding version 54
2-34 -- Audio stream descriptor 55
2-35 -- Audio coding version 55
2-36 -- Hierarchy descriptor 55
2-37 -- Hierarchy descriptor values 56
2-38 -- Registration descriptor 56
2-39 -- Registration authority committee 56
2-40 -- Data stream alignment descriptor 59
2-41 -- Video stream alignment values 60
2-42 -- Audio stream alignment values 60
2-43 -- Target background grid descriptor 60
2-44 -- Video window descriptor 61
2-45 -- Conditional access descriptor 61
2-46 -- ISO 639 language descriptor 62
2-47 -- System clock descriptor 63
2-48 -- Multiplex buffer utilization descriptor 63
A-1 -- ISO/IEC 13818 DSM CC 76
A-2 -- Command_id assigned values 76
A-3 -- DSM_CC control 77
A-4 -- Select mode assigned values 78
A-5 -- DSM CC Acknowledgement 79
A-6 -- Time code 80
A-1 -- Program association table bandwidth usage (bps) 96

A-2 -- Program table bandwidth usage (bps) 96
A-1 -- Program descriptor example 98
A-2 -- Program sub-descriptor 99
A-1 -- PES packet header example 100
A-1 -- List of companies supplying patent statements 107

List of Equations

2-1 -- PCR base equation 13
2-2 -- PCR extension equation 13
2-3 -- Program Clock Reference equation 13
2-4 -- Calculation of input arrival time 13
2-5 -- Transport rate calculation 14
2-6 -- Latency calculation 14
2-7 -- System information main buffer transfer rate 15
2-8 -- Calculation of presentation timestamp 25
2-9 -- Calculation of decode timestamp 25
2-10 -- Calculation of elementary stream clock reference 26
2-11 -- Calculation of elementary stream clock reference extension 26
2-12 -- Calculation of elementary stream clock reference 26
2-13 -- Buffer size for audio stream 29
2-14 -- Buffer size for video stream 29
2-15 -- System clock reference base calulation 42
2-16 -- System clock reference extension 42
2-17 -- System clock reference calculation 42
2-18 -- Arrival time calculation 42
2-19 -- System clock reference base calculation 44
2-20 -- System clock reference extension calculation 44
2-21 -- System clock reference calculation 44
2-22 -- System clock reference interval calculation 46
2-23 -- Ratio of system clock frequency and audio sample rate 46
2-24 -- Ratio of system clock frequency to video picture rate 46
2-25 -- Clock accuracy determination 63
2-26 -- Multiplex buffer utilization calculation 64
2-27 -- Packet rate calculation 69
2-28 -- Maximum packet rate calculation 69
2-29 -- Sample rate locking in Transport Stream 70
2-30 -- Ratio of system clock frequency to video picture rate 70

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 x . The video and audio data is encoded as described in Parts 2 and 3 of this Recommendation | International Standard. The resulting compressed elementary streams are packetized to produce **PES packets**. Information needed to use PES packets independent 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 a **Program Streams**. This systems standard covers those processes to the right of the vertical dashed line.

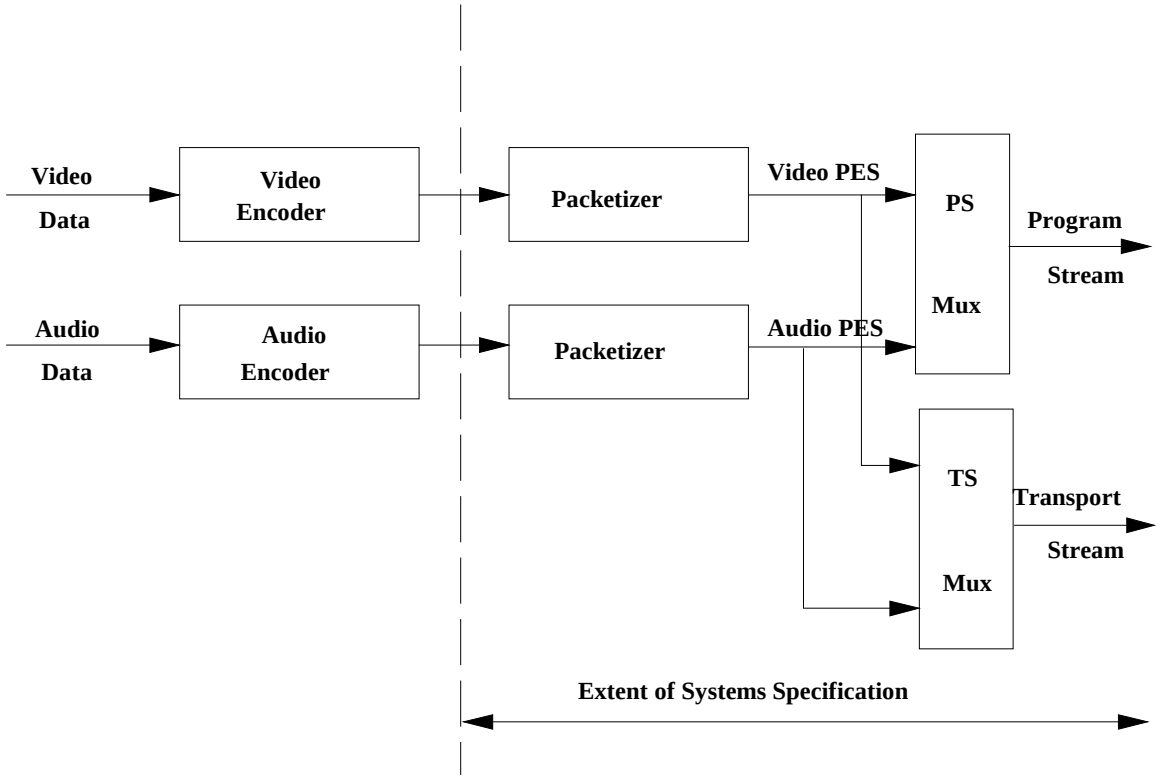


Figure 0-1 -- Simplified overview of ISO/IEC 13818 scope

The **Program Stream** is analogous and similar to ISO/IEC 11172 Systems Multiplex. It results from combining one or more streams of PES packets, which have a common time base, into a single stream. The Program Stream definition can also be used to encode multiple audio and video elementary streams into multiple Program Streams, all of which have a common time base.

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 ISO/IEC 13818 coded data and other data in environments in which significant errors may occur. Such errors may be manifested as bit value errors or loss of packets.

The ISO/IEC 13818 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 xii .

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 xii .

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 xii .

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 xii and figure 0-3 on page xii illustrate prototypical demultiplexing and decoding systems which takes as input an ISO/IEC 13818 Transport Stream. Figure 0-2 on page xii illustrates the first case, where a Transport Stream is directly demultiplexed and decoded. ISO/IEC 13818 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 Transport Stream decoder either apply to the entire ISO/IEC 13818 Transport Stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The ISO/IEC 13818 transport system layer is divided into two sub-layers, one for multiplex-wide operations (the transport packet layer), and one for stream-specific operations (the PES packet layer).

A prototypical audio/video Transport Stream decoder is also depicted in figure 0-2 on page xii to illustrate the function of a decoder. The architecture is not unique -- some Transport Stream system decoder functions, such as decoder timing control, might equally well be distributed among elementary stream decoders and the Data Link Specific Decoder -- but this figure is useful for discussion. Likewise, indication of errors detected by the data link specific decoder to the individual audio and video decoders may be performed in various ways and such communications paths are not shown in the diagram. The prototypical decoder design does not imply any normative requirement for the design of an ISO/IEC 13818 Transport Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

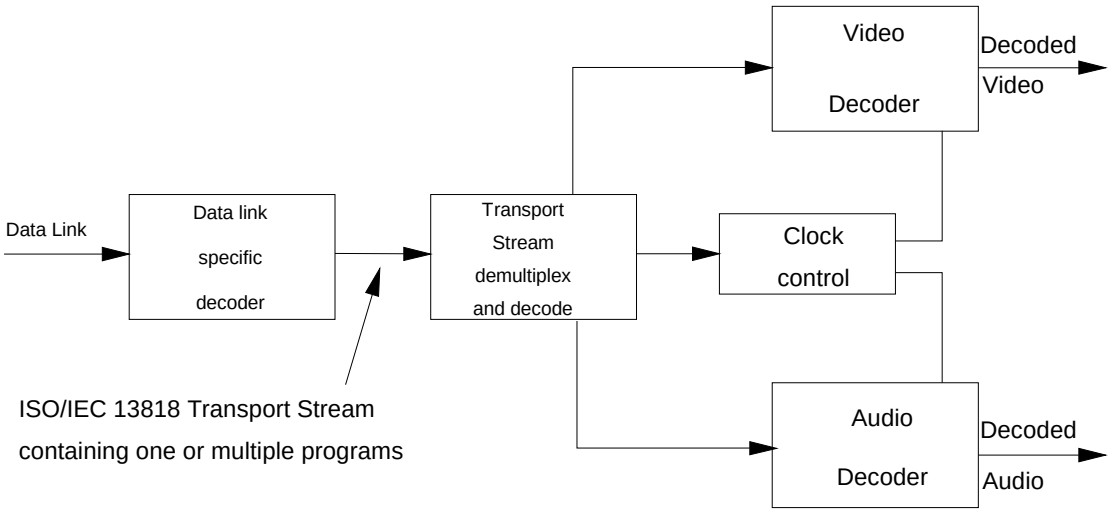


Figure 0-2 -- Prototypical transport demultiplexing and decoding example

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 the case shown in figure 0-3 the remultiplexing operation will include the correction of Program Clock Reference (PCR) timestamps to account for the change in arrival of timestamps.

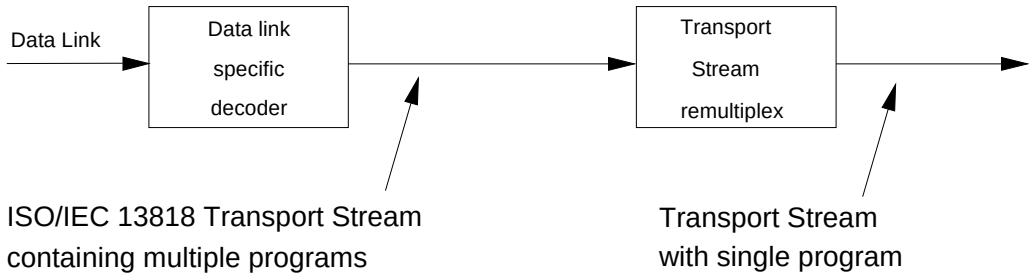


Figure 0-3 -- Prototypical transport multiplexing example

Figure 0-4 below illustrates a case in which an ISO/IEC 13818 multi-program Transport Stream is first demultiplexed then converted into an ISO/IEC 13818 Program Stream.

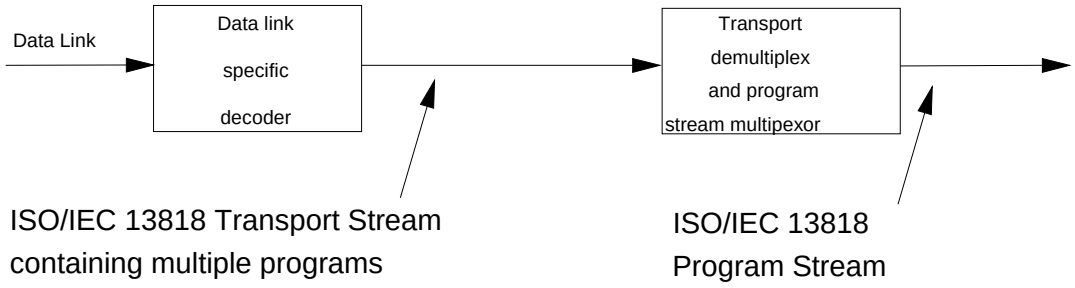


Figure 0-4 -- Prototypical transport to program stream conversion

Figure 0-3 on page xii and figure 0-4 on page xii indicate that it is possible and reasonable to convert between different types and configurations of ISO/IEC 13818 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

As in the case of Transport Stream, a prototypical audio/video Program Stream decoder system is depicted in figure 0-5 to illustrate the function of a decoder. The architecture is not unique -- System Decoder functions including decoder timing control might equally well be distributed among elementary stream decoders and the Medium Specific Decoder -- but this figure is useful for discussion. The prototypical decoder design does not imply any normative requirement for the design of an ISO/IEC 13818 Program Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

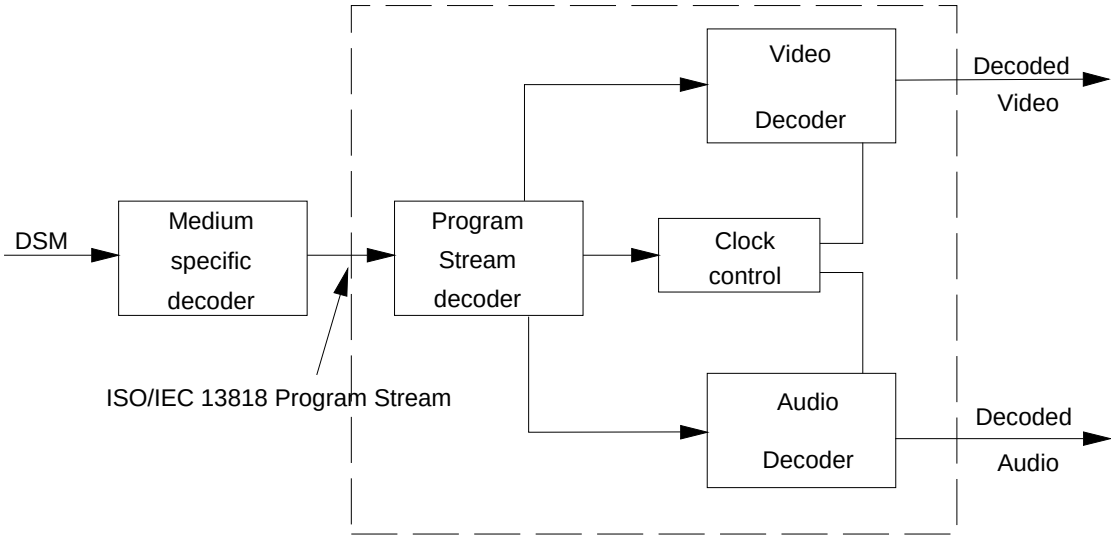


Figure 0-5 -- Prototypical program stream decoder

The prototypical ISO/IEC 13818 Program Stream decoder 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 on a digital storage medium (DSM), or network, in some medium-specific format. The medium specific format is not governed by this Recommendation | International Standard, nor is the medium-specific decoding part of the prototypical ISO/IEC 13818 Program Stream decoder.

The prototypical decoder accepts as input an ISO/IEC 13818 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 Medium Specific Decoder. The Video and Audio Decoders are synchronized with each other and with the DSM using this timing information.

ISO/IEC 13818 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 Program Stream Decoder either apply to the entire ISO/IEC 13818 Program Stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The ISO/IEC 13818 Program 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 is 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 clause 2.4.1 on page 9 and clause 2.5.1 on page 39 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.

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

Not all **Transport Streams** will have been formed from **Program Streams** but it is possible to create a valid **Program Stream** from a valid **Transport Stream**.

0.4 Packetized Elementary Stream

Transport Streams and **Program Streams** are each logically constructed from PES packets, as indicated in the syntax definitions in clause 2.4.3.6 on page 21. 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 clause 2.4.3.7 on page 24. 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

ISO/IEC 13818 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. 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 DSM or data link, the adjustment of clocks, and the management of buffers. The tasks are intimately related. If the rate of data delivery off the data link or DSM 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 source 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 data source, 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 data source. 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. These topics are discussed next.

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 ISO/IEC 13818 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 clause 2.5.1 on page 39 and clause 2.5.2 on page 40 of ISO/IEC 13818-1. For Transport Streams the packet length is 188 bytes. Both fixed and variable PES packet length are allowed, but 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 effected with presentation time stamps (PTS) in the Program and Transport bit 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 DSM or channel clock, or it may be some external clock.

Each program in a Transport Stream, which contains multiple programs, has its own time base. The time bases of different programs within such a stream may be different.

Because presentation time-stamps (PTS) 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 data source is achieved through the use of the SCR in the Program Stream and by the equivalent PCR in the Transport Stream. The SCR and PCR are time stamps encoding the timing of the bit stream itself in terms of the same time base as is 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. It is possible to have only one PCR for some or all programs in a multi-program transport. See clause 2.4.4 on page 29, 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. Multiple programs may share a common set of PCRs.

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, a video packet may start at any byte in the video stream. However, time stamps encoded in PES packet headers apply to presentation times of compression layer constructs (namely, presentation units).

0.9 System reference decoder

Part 1 of ISO/IEC 13818 employs a "System Target Decoder," (STD), one for Transport Streams (see clause 2.4.2 on page 10) referred to as "Transport- System Target Decoder"(T-STD) and one for Program Streams (see clause 2.5.2 on page 40) 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 ISO/IEC 13818 fields (for example, buffer sizes) each ISO/IEC 13818 stream leads to its own parameterization of the STD. It is up to encoders to ensure that bit streams they produce will forward play in normal speed on corresponding STDs. 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 will simply select the most appropriate stream.

Modern data communications networks may be capable of supporting ISO/IEC 13818 audio and video. A real time transport protocol is required. The Program Stream may 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 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 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 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 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 ISO/IEC 13818 multiplexed bit stream, whether a Program Stream or a Transport 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 clause 2.5.7 on page 66 of this part.

1.2 References

The following 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 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 standards indicated below. Members of IEC and ISO maintain registers of currently valid Recommendation | International Standards.

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 13818-2:1994 *Information technology - Coding of moving pictures and associated audio - Part 2: Video.*

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

CCIR Recommendation 601-2 *Encoding parameters of digital television for studios.*

CCIR Report 624-4 *Characteristics of systems for monochrome and colour television.*

CCIR Recommendation 648 *Recording of audio signals.*

CCIR Report 955-2 *Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of Advanced Digital System II.*

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 908:198, C*D Digital Audio System*

ISO/CD 13522;1993 *Information Technology - Coded Representation of Multimedia and Hypermedia Information Objects*. (MHEG)

ISO/CD 639-2, *Terminology - Code for Presentation of Names of Languages Part 2 Alpha-3 Code*

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 compressed audio an access unit is an Audio Access Unit. In the case of compressed video an access unit is the coded representation of a picture.

2.1.2 bitrate: The rate at which the compressed bit stream is delivered from the storage medium or data-link 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 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]: An ISO/IEC 13818 Transport Stream or Program Stream for which the constraints defined in Part 1clause **Error! Reference source not found.** on page **Error! Bookmark not defined.** 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 entitlement control message; ECM: Entitlement Control Messages are private conditional access information which specify control words and possibly other, typically stream-specific, scrambling and and/or control parameters.

2.1.17 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.

2.1.18 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.

2.1.19 elementary stream; ES [system]: A generic term for one of the coded video, coded audio or other coded bit streams.

2.1.20 encoder: An embodiment of an encoding process.

2.1.21 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.

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

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

2.1.24 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.

2.1.25 ISO/IEC 13818 (multiplexed) stream [system]: A bit stream composed of 0 or more elementary streams combined in the manner defined in Part 1 of this Recommendation | International Standard.

2.1.26 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.

2.1.27 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 clause 2.5.3.3 on page 43 this Recommendation | International Standard.

2.1.28 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 clause 2.4.3.6 on page 21 this Recommendation | International Standard.

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

2.1.30 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 clause 2.4.3 on page 16.

2.1.31 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.

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

2.1.33 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 clause 2.4.3.6 on page 21.

2.1.34 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.

2.1.35 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.

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

2.1.37 program [system]: A program is a collection of elementary streams with a common time base.

2.1.38 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 clause 2.4.4 on page 29. One case of PSI, the non-mandatory network information table, is privately defined.

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

2.1.40 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'.

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

2.1.42 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 code consists of a 24 bit prefix (0x0000001) and an 8 bit stream_id as shown in table 2-24 on page 47

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

2.1.44 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.

2.1.45 system header [system]: The system header is a data structure defined in clause 2.5.3.5 on page 45 of this Recommendation | International Standard that carries information summarizing the system characteristics of the ISO/IEC 13818 multiplexed stream.

2.1.46 system target decoder; STD [system]: A hypothetical reference model of a decoding process used to describe the semantics of an ISO/IEC 13818 multiplexed bit stream.

2.1.47 time-stamp [system]: A term that indicates the time of an event.

2.1.48 Transport Stream packet header [system]: A data structure used to convey information about the Transport Stream payload.

2.1.49 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 x > 0
0 x == 0
-1 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 VLC codes are written.

window Number of actual time slot in case of block_type==2, 0 ≤ 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 clause 2.4.1 on page 9 and clause 2.5.1 on page 39. 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.
data_element	
. . .	
} while (condition)	The data element is repeated until the condition is not true. If the
if (condition) {	condition is true, then the first group of data elements occurs next in the
data_element	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++) {	The group of data elements occurs n times. Conditional constructs within
data_element	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 11 or figure 2-7 on page 40 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 Transport Stream coding layer allows one or more groups of one or more elementary streams to be combined into a single stream. A group of elementary streams, with a common system_clock_frequency time base, is called a program. Data from each elementary stream are encoded and multiplexed together with information that allows elementary streams within a program to be replayed in synchronism.

ISO/IEC 13818 multiplexed Transport Stream

An ISO/IEC 13818 Transport Stream consists of one or more programs each containing one or more elementary streams and other streams multiplexed together. Each elementary stream consists of access units, which are the coded representation of presentation units. The presentation unit for a video elementary stream is a picture. The corresponding access unit includes all the coded data for the picture. If the picture

is the first of a group of pictures (GOP) then the access unit also includes any preceeding data from that GOP starting with the first byte of the group_start_code. Group_start_code is defined in part 2 of this Recommendation | International Standard. If the picture is the first coded picture after a sequence header, then the access unit also includes the sequence header starting with the first byte of the sequence_header_code. Sequence_header_code is defined in part 2 of this Recommendation | International Standard. Otherwise a video access unit commences with the first byte of a picture_start_code which is defined in part 2 of this Recommendation | International Standard. The sequence_end_code is included in the Access Unit containing the last coded picture of a sequence. (See Part 2 of this Recommendation | International Standard for the definition of the sequence_end_code). The presentation unit for an audio elementary stream is the set of samples that corresponds to samples from an audio frame. The corresponding audio access unit commences with the first byte of the audio frame synchronization word. (see Part 3 of this Recommendation | International Standard for the definition of an audio frame).

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 to which the packet data belongs. The PES packet header may contain decoding and/or presentation time-stamps (DTS and PTS) that refer to the first access unit that commences in the packet. The PES packet header also contains a number of flags opening up a range of optional fields. The packet data 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 17 . 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
- Network Information Table
- Conditional Access Table

These tables contain the necessary and sufficient information to demultiplex and present programs. The Program Map Table, in table 2-18 on page 36 , 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.

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 (see table 2-17 on page 34 of the PSI.

2.4.2 Transport Stream system target decoder

The semantics of the Transport Stream specified in clause 2.4.3 on page 16 and the constraints on these semantics specified in clause 2.5.7 on page 66 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 Transport Stream system target decoder (T-STD).

The T-STD is a conceptual model used to define these terms precisely and to model the decoding process during the construction of ISO/IEC 13818 Transport Streams. The T-STD is defined only for this purpose.

Neither the architecture of the T-STD nor the timing described precludes uninterrupted, synchronized playback of ISO/IEC 13818 Transport Streams from a variety of decoders with different architectures or timing schedules.

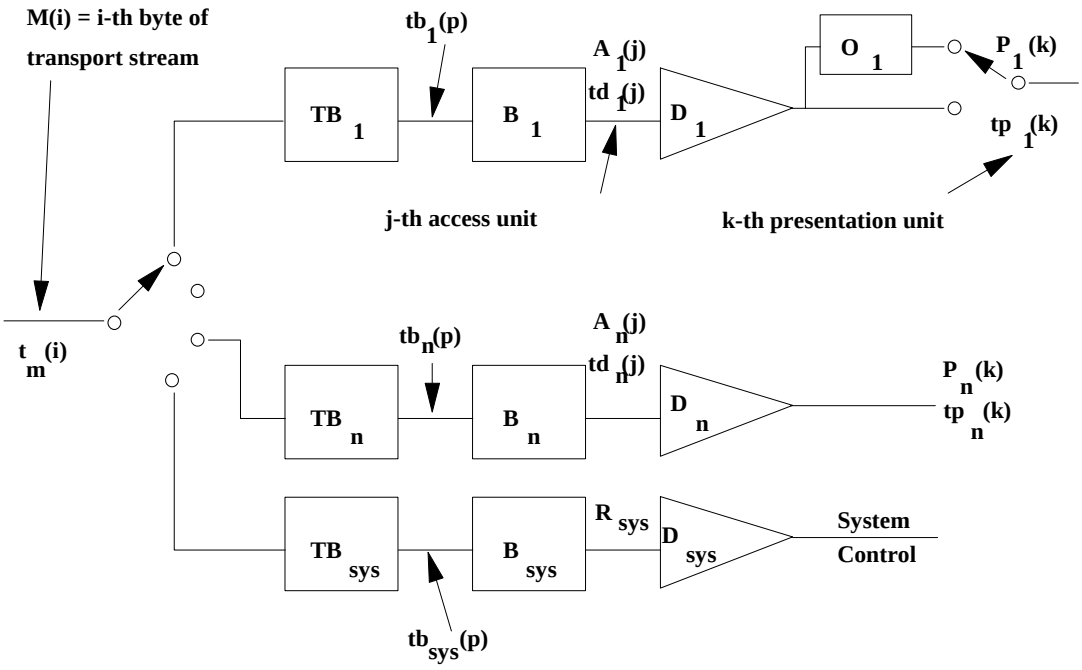


Figure 2-6 -- ISO/IEC 13818 Transport Stream system target decoder notation

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 ISO/IEC 13818 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 ISO/IEC 13818 Transport Stream

M(i) is the i^{th} byte in the Transport Stream.

$t_m(i)$ indicates the time in seconds at which the i^{th} byte of the ISO/IEC 13818 Transport Stream enters the system target decoder. The value $t_m(0)$ is an arbitrary constant.

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

$A_n(j)$ is the j^{th} access unit in elementary stream n. Note that access units are 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.

$tb_n(p)$ indicates the time, measured in seconds, at which the p^{th} Transport Stream packet of the M^{th} Transport Stream enters buffer B_n.

$tb_{sys}(p)$ indicates the time, measured in seconds, at which the p^{th} Transport Stream packet of the M^{th} Transport Stream enters buffer B_{sys}.

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

$tp_n(k)$ 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 main buffer in the system target decoder for elementary stream n.

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

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 the main buffer in the system target decoder, measured in bytes, for system information for the program that is in the process of being decoded.

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

TBS_{sys} is the size of the input buffer in the system target decoder, measured in bytes, for system information for the program that is in the process of being decoded.

TB_n the transport buffer in the system target decoder for elementary stream n.

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

T_{tp} indicates the length of the period during which one Transport Stream packet arrives at the input of the system target decoder.

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}.

Res_n is the rate of the elementary stream n.

L_n is the latency time after which one half of each Transport Stream packet enters TB_n before that Transport Stream packet leaves TB_n.

System clock frequency

Timing information referenced in T-STD is carried by several data fields defined in this Recommendation | International Standard. Refer to clause 2.4.3.4 on page 19, and clause 2.4.3.6 on page 21. 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 - 1 350 ≤ system_clock_frequency ≤ 27 000 000 + 1 350

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 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 (300×2^{33}/system_clock_frequency). 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.

Input to the Transport Stream system target decoder

Input to the Transport Stream system target decoder (T-STD) is an ISO/IEC 13818 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 ISO/IEC 13818 Transport Stream enters the system target decoder at a piece-wise constant rate. The i^{th} byte, M(i), enters at time $t_m(i)$. The time at which this byte enters the system target decoder 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. The PCR is encoded in equation 2-3 in two parts; one, in units of 1/300 times the system clock frequency, called PCR_base(equation 2-1), and one, called PCR_ext (equation 2-2), in units of the system clock frequency. The value encoded in the PCR(i') field indicates time $t_m(i')$, where i' refers to the byte containing the last bit of the PCR_base field, M(i').

Specifically:

$$PCR\_base(i') = ((system\_clock\_frequency \times t_m(i'))/300)\%2^{33} \quad (2\text{-}1)$$

$$PCR\_ext(i') = NINT(system\_clock\_frequency \times t_m(i'))\%(2^{33} \times 300) - PCR\_base(i') \times 300 \quad (2\text{-}2)$$

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

The input arrival time, $t_m(i)$ is shown in equation 2-4 below, for all other bytes shall be constructed 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 PCR fields of the same program divided by the difference between the time values encoded in these same two PCR fields.

$$t_m(i) = \frac{PCR(i'')}{system\_clock\_frequency} + \frac{i - i''}{transport\_rate} \quad (2\text{-}4)$$

Where:

i"	is the index of the byte containing the last bit of the program_clock_reference base field in the Transport Stream packet adaptation field.
i	is the index of any byte in the Transport Stream.
PCR(i")	is the time encoded in the program_clock_reference base and extension fields in units of the system clock.

$$transport\_rate(i) = \frac{((i'-i'')\times system\_clock\_frequency)}{PCR(i') - PCR(i'')} \quad (2\text{-}5)$$

where

i"	is the index of the byte containing the last bit of the latest program_clock_reference base preceding the PCR_base field at i', in the Transport Stream packet with the same PID as that containing i'.
and	$i'' < i \leq i'$

Buffering

Complete Transport Stream packets containing data from elementary stream n, as indicated in the PID, are passed to the transport buffer for stream n, TB_n. Transfer of byte M(i) from the system target decoder input to TB_n is instantaneous, so that byte M(i) enters the buffer for stream n, of size TBS_n, at time $t_m(i)$.

Bytes present in the Transport Stream packet, which are part of the PES packet or its contents are delivered to the main buffer B_n. Other bytes are not, and may be used to control the system.

Each complete Transport Stream packet which has entered TB_n is removed instantaneously and immediately placed in B_n at the time specified as latency L_n following the time when one-half of the Transport Stream packet has entered TB_n. The value of L_n is defined in equation 2-6 below. The Transport Stream packet time T_{tp} is 188 divided by the Transport Stream rate in bytes per second, as calculated above using the Program Clock Reference (PCR). Each byte of each Transport Stream packet is moved from TB_n to B_n at the specified time if and only if there is space available in B_n. If there is no space in B_n the remaining Transport Stream packet data bytes remains in TB_n; these remaining bytes are moved as soon as space is available in B_n.

Each elementary stream has a maximum elementary stream rate $R_{es}(max)$, which is indicated within the bit streams which are specified in Parts 2 and 3 of this Recommendation | International Standard. In the case of constant bitrate operations $R_{es}(max) \equiv R_{es}$. The latency L_n is specified by the following:

$$if\left(\frac{transport\_rate}{R_{es}(\max)}\right) < 1.2 \quad \text{then}$$

$$L_n = \min\left[4ms, 2\times T_{tp}\right]$$

else

$$L_n = \min\left[4ms, \frac{2\times 188}{R_{es}(\max)}\right] \quad (2\text{-}6)$$

Note - $\frac{2\times 188}{R_{es}(\max)} \equiv \frac{2\times transport\_rate}{R_{es}(\max)} \times T_{tp}$

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 2-15 on page 32)as having the PMT_PID value for the selected program.

The transport buffer size is fixed at 512 bytes.

The main buffer sizes BS_1 through BS_n are defined as follows. In the case of constant bitrate operations $R_{es}(max) \equiv R_{es}$. A theoretical effective rate R_{eff} is calculated as the elementary stream rate, $R_{es}(max)$, multiplied by 188/184. A portion BS_{mux} of the buffer size BS_n allocated for multiplex buffering is defined as $BS_{mux} = R_{eff} \times 4$ milliseconds. Another portion, $BS_n(dec)$, of the buffer size allocated for decoding is the size of the Video Buffering Verifier, as specified in Part 2 of this Recommendation | International Standard, in the case of video elementary stream data, or the size of one audio access unit, equivalently a coded audio frame, as specified in Part 3 of this Recommendation | International Standard, in the case of audio elementary stream data. The size BS_n of buffer B_n is equal to the sum $BS_{mux} + BS_n(dec)$.

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

For each elementary stream buffer, all the data for the access unit that has been in the input buffer, $A_n(j)$, longest is removed instantaneously at its decoding time $td_n(j)$. In the case of an ISO/IEC 13818 video stream the decoding time, $td_n(j)$, may be derived from subclauses C.9 to C.11 of annex C of ISO/IEC 13818-2. The data defined as belonging to an ISO/IEC 13818 video access unit for this purpose is defined in subclause C.5 of annex C of ISO/IEC 13818-2. In the case of an ISO/IEC 13818 audio stream, access units consist of audio frames. 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.

When the **low_delay** flag in the video sequence extension is set to '1' (subclause 6.2.2.3 of ISO/IEC 13818-2) the VBV buffer may underflow. In this case when the T-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. Overflow of buffer B_n shall not occur.

VBV buffer underflow is allowed to occur continuously without limit. The T-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.

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}.

$$R_{sys} = \max\left[80kbps, transport\_rate(i) / 500\right] \quad (2\text{-}7)$$

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

Decoding

Elementary streams buffered in B_1 through B_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 system target decoder. 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 initally has only I- and P-pictures but later includes B-pictures should include re-ordering delay starting at th beginning of the stream.

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

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 11 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.

2.4.3 Specification of the Transport Stream syntax and semantics

The following syntax describes a stream of bytes.

2.4.3.1 ISO/IEC 13818 Transport Stream

Table 2-2 -- ISO/IEC 13818 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 -- ISO/IEC 13818 transport header

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).

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 a value of '1' by entities external to the transport layer. When set to a value of '1' this bit shall not be reset to a value of '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. One and only one PES packet may start in any Transport Packet. This also applies to private streams of stream_type 6(refer to table 2-26 on page 50).

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 clause 2.4.4.1 and clause 2.4.4.2 on page 31. This also applies to private streams of stream_type 5(refer to table 2-26 on page 50).

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 priority field may be coded regardless of the PID or within one PID only. This field may be changed by data link 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 2-15 on page 32). PID value 0x0001 is reserved for Conditional Access Table (table 2-17 on page 34). PID values 0x0002-0x000F are reserved. PID value 0x1FFF is reserved for null packets.

transport_scrambling_control -- This 2 bit field indicates the scrambling of the Transport Stream packet payload. The Transport Stream packet header, including 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 2-4 -- 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 2-5 -- 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

ISO/IEC 13818 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 ISO/IEC 13818 Transport Streams, duplicate packets may be sent as 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. 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 , 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.

2.4.3.4 Adaptation field

Table 2-6 -- 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**
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**
	9	**uimsbf**
original_program_clock_reference_extension		
}		
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**
for (i=0;i<adaptation_field_extension_length;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 field. The length shall not exceed 183 bytes. The value '0' is permitted, to support the stuffing of a single byte at this level.

discontinuity_indicator -- This is a 1 bit field which when set to '1' indicates that the next PCR in a Transport Stream packet with the same PID represents a sample of a new system time clock for the associated program. When there is a discontinuity a Transport Stream packet with the first PCR of the new

time-base shall have this bit set to '1'. This bit may also be set in prior packets preceding the discontinuity. When set to a value of '0' no discontinuity in PCR fields shall occur in the following the Transport Stream packets of the same PID value.

random_access_indicator -- The random_access_indicator is a 1 bit field. When set to '1', it indicates that the next PES packet in this Transport Stream packet or in subsequent Transport Stream packets with the same PID shall contain a PTS field and an elementary stream access point. 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

In Transport Stream packets where the random_access_indicator is set to a value of '1', there shall be an adaptation field which shall contain at least the program_clock_reference_base and program_clock_reference_extension fields. When set to a value of '0' it is not defined whether a random access point occurs or not. The meaning of the flag is not defined for private data.

elementary_stream_priority_indicator -- The elementary_stream_priority_indicator is a single bit field. It indicates, within this 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 a value of '1'.

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

OPCR_flag -- The OPCR_flag is a 1 bit flag. A '1' indicates that the adaptation_field contains an OPCR field. 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. This feature is included to allow for future growth of the adaptation field. 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) as defined in (equation 2-3 on page 13), is a 42 bit field coded in two parts; one, in units of 1/300 multiplied by the system clock frequency (90kHz) called program_clock_reference_base (equation 2-1 on page 13), is a 33 bit field and one called program_clock_reference_extension (equation 2-2 on page 13), is a 9 bit field in units of system clock frequency (27MHz). Its presence is indicated by the PCR_flag. 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.

The program_clock_reference_extension field indicates the number of periods of a 27MHz clock after a 90kHz period start. The value of this field will be between 0 and 299. When the value equals 299, the program_clock_reference_extension will wrap around to zero and simultaneously the lsb (least significant bit) of the program_clock_reference_base field will increment by 1. For Transport Stream packets containing video or audio elementary streams, if a PCR field is present in the adaptation field, that PCR field must be valid for the elementary stream contained in its Transport Stream packet.

original_program_clock_reference_base; original_program_clock_reference_extension -- The optional original program clock reference (OPCR) is a 42 bit field coded in two parts; one, in units of 1/300 multiplied by the system clock frequency (90kHz) called original_program_clock_reference_base is a 33 bit field and one called original_program_clock_reference_ext is a 9 bit field in units of system clock frequency (27MHz). OPCR indicates the intended time of arrival of the byte containing the last bit of the original_program_clock_reference_base reference field at the system target decoder in a single program Transport Stream. If it is present in a multi-program Transport Stream, it may be ignored by the decoder. Further, it may not be modified by any multiplexor or decoder. The OPCR use is for recovery of the original single-program Transport Stream with its identical time stamps. The OPCR will only be valid if, e.g. by private arrangements, the original single program Transport Stream is reconstructed exactly.

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 contain adaptation fields are excluded. The splicing point occurs immediately after the transfer to buffer B_n in the STD model of the Transport Stream packet in which the splice_countdown reaches the value 0. At the splicing point, the contents of buffer B_n shall be less than or equal to 1/8 of the size BS_n of buffer B_n. In the Transport Stream packet where the splice_countdown value reaches 0, the last byte of the Transport Stream packet payload shall be the last byte of an access unit. Transport Stream packets with the same PID, which follows, may contain data from a different elementary stream of the same type.

The first byte of the payload of the next Transport Stream packet shall be the first byte of an access point of the elementary stream contained in Transport Stream packets with that PID value. Thus the previous access unit aligns with the packet boundary or is "stuffed" 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.

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.

adaptation_field_extension_length -- The adaptation_field_extension_length is an 8 bit field. It indicates the length of the extended adaptation field data following the end of this field.

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

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 2-7 -- PES Packet

Syntax	No. of Bits	Mnemonic
PES_packet() {		
packet_start_code_prefix	**24**	**bslbf**
stream_id	**8**	**uimsbf**
PES_packet_length	**16**	**uimsbf**
if (stream_id != private_stream_2 && stream_id != padding_stream) {		
'10'	**2**	**bslbf**
PES_scrambling_control	**2**	**bslbf**
PES_priority	**1**	**bslbf**
data_alignment_indicator	**1**	**bslbf**
copyright	**1**	**bslbf**

Syntax	No. of Bits	Mnemonic
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**
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 == '000') {		
field_id	**2**	**bslbf**
intra_slice_refresh	**1**	**bslbf**
frequency_truncation	**2**	**bslbf**
}		
else if(trick_mode_control == '001'){		

Syntax	No. of Bits	Mnemonic
field_rep_cntrl	**5**	**uimsbf**
}		
else if(trick_mode_control == '010') {		
field_id	**2**	**uimsbf**
reserved	**3**	**bslbf**
}		
else if (trick_mode_control == '011') {		
field_id	**2**	**bslbf**
intra_slice_refresh	**1**	**bslbf**
frequency_truncation	**2**	**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_field_flag	**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()		
}		
if(program_packet_sequence_counter_flag== '1'){		
marker_bit	**1**	**bslbf**
program_packet_sequence_counter	**7**	**uimsbf**
marker_bit	**1**	**bslbf**
original_stuff_length	**7**	**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_field_flag == '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<N;i++) {		
stuffing_byte	**8**	**bslbf**
}		
for (i=0;i<N;i++) {		
PES_packet_data_byte	**8**	**bslbf**
}		

Syntax	No. of Bits	Mnemonic
}		
else if (stream_id == private_stream_2) {		
for (i=0;i<PES_packet_length;i++) {		
PES_packet_data_byte	**8**	**bslbf**
}		
}		
else if (stream_id == padding_stream) {		
for (i=0;i<N;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 -- The stream_id specifies the type and number of the elementary stream as defined by the stream_id table 2-24 on page 47 .

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 indicates the scrambling mode of the PES packet payload.

Table 2-8 -- PES scrambling control values

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

PES_priority -- 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. A value of zero shall not coded when the PES priority is carried in a Program Stream.

data_alignment_indicator -- A 1 bit flag, when set to '1' indicates that the PES packet header is immediately followed by the access unit data type as defined in the data_stream_alignment_descriptor in table 2-42 on page 59. When set to a value of '0' it is not defined whether any such alignment occurs or not.

copyright -- If this bit is set to '1', it indicates that the contents of the associated PES packet payload is copyrighted. When set to a value of '0' this indicates that the material is not copyrighted.

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 -- A 2 bit flag. If the PTS_DTS_flags field equals '10', a PTS field is present in the PES packet header. If the PTS_DTS_flags field equals '11', both a PTS and DTS field are present in the PES

packet header. If the PTS_DTS_flags field equals '00' neither a PTS nor a DTS field is present in the PES packet header. The value '01' is forbidden.

ESCR_flag -- A 1 bit flag, when set to '1' indicates that an ESCR field is 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 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 inidcates that no ES_rate field is present.

DSM_trick_mode_flag -- a 1 bit field when set to '1' it indicates the presence of an 8 bit field used for trick_mode_control field. When set to a value of '0' it indicates that this field is not present.

additional_copy_info_flag -- A 1 bit flag 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 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, when set to '1' indicates that an extension field of flags 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) -- The PTS is a 33-bit number coded in three separate fields. It indicates the intended time of presentation in the system target decoder of the presentation unit that corresponds to the first access unit that commences in the packet. The value of PTS is measured in the number of periods of a 90kHz system clock with a tolerance specified in clause 2.4.2 on page 10. Using the notation of clause 2.4.2 on page 10 the value encoded in the presentation_time_stamp is:

$$PTS = NINT(system\_clock\_frequency \times (tp_n(k)))\%2^{33} \quad (2\text{-}8)$$

where

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

$P_n(k)$ is the presentation unit corresponding to the first access unit that commences in the packet data. An access unit commences in the packet if the first byte of the access unit is contained in the packet data. (see 2.4.1 for a definition of the first byte of audio and video access units).

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. Refer to clause 2.5.7.5.

DTS (decoding_time_stamp) -- The DTS is a 33-bit number coded in three separate fields. It indicates the intended time of decoding in the system target decoder of the first access unit that commences in the packet. The value of DTS is measured in the number of periods of a 90kHz system clock with a tolerance specified in clause 2.4.2 on page 10. Using the notation of clause 2.4.2 on page 10 the value encoded in the decoding_time_stamp is:

$$DTS = NINT(system\_clock\_frequency \times (td_n(j)))\%2^{33} \quad (2\text{-}9)$$

where

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

$A_n(j)$ is the first access unit that commences in the packet data. An access unit commences in the packet if the first byte of the access unit is contained in the packet data. (see 2.4.1 for a definition of the first byte of audio and video access units).

Refer to clause 2.5.7.6

ESCR -- The elementary_stream_clock_reference (ESCR) is an optional 42 bit field coded as a 33 bit base field (ESCR_base) with a 9 bit extension field (ESCR_ext). It indicates the intended time of arrival of the byte containing the last bit of the ESCR_base at the input of the system target decoder when present in a PES Stream. The value of the ESCR_base is measured in the number of periods of a 90kHz system clock with a tolerance specified in clause 2.4.2 on page 10.

Specifically:

$$ESCR\_base(i') = ((system\_clock\_frequency * t_m(i'))/300)\%2^{33} \quad (2\text{-}10)$$

$$ESCR\_ext(i') = NINT(system\_clock\_frequency * t_m(i'))\%(2^{33} * 300) - ESCR\_base(i') * 300 \quad (2\text{-}11)$$

$$ESCR(i') = ESCR\_base(i') + ESCR\_ext(i') \quad (2\text{-}12)$$

The ESCR_extension field indicates the number of periods of a 27MHz clock after a 90kHz period start. The value of this field will be between 0 and 299. When the value equals 299, the ESCR_ext will wrap around to 0 and simultaneously the ESCR_base field will increment by 1. Refer to clause 2.5.7.4.

ES_rate -- The elementary_stream_rate (ES_rate) field is a positive 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 rounded upwards. 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 in clause 2.5.2 on page 40 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 three (3) bit field used to indicate the mode in which the DSM operates. The values for each mode is in the table below.

Table 2-9 -- Trick mode control values

value	description
'000'	fast forward
'001'	slow motion
'010'	freeze frame
'011'	fast reverse
'1xx'	reserved

fast forward-When the value '000' is coded, then the DSM is in the fast forward mode. The DSM will ensure that the buffers in the STD model do not overflow. The DSM will provide each picture with valid timestamps. During the fast forward mode, the system time clock in the STD model is continuous. However, at the transition to another mode, a discontinuity in the system time clock will occur. When

intra_slice_refresh bit is set in the video elementary stream (see Part 2 of this Recommendation | International Standard), it indicates that each picture is composed of intra slices with possible gaps between them. The decoder may replace the missing slices by repeating the co-sited slices of previously decoded picture(s). The field_id flag is to be interpreted in the same ways as for the freeze frame mode.

The field "frequency_truncation" provides an indication of coefficient selection done by the DSM in the process of generating the bitstream in fast play modes.

Table 2-10 -- Coefficient selection values

Value	Description
'00'	Only DC coefficients are sent
'01'	the first three coefficients in scan order on average
'10'	the first six coefficients in scan order on average
'11'	all coefficients may be sent

slow motion-If the value of '001' is encoded, then the DSM is in slow motion mode. The DSM will provide a picture with valid time stamps to the decoder. In the slow motion mode, the PTS indicates the value of the system time clock at which the first field from the associated picture is to be displayed in the STD model. During the slow motion mode, the system time clock in the STD model is continuous. However, at the transition to another mode, a discontinuity in the system time clock will occur. The field_rep_cntrl value is used to indicate to the decoder how many field times it needs to repeat the display of field #1 of the received picture as both top and bottom fields alternatingly. The same value also indicates to the decoder how often field #2 has to be repeated after the repetition of field #1 has been completed. For progressive pictures the decoder interprets the value specified by field_rep_cntrl field in terms of complete frames. The DSM will then provide the next picture to the decoder with the correct presentation time stamp (i.e. taking into account how many times the fields of the previous picture were repeated) along with the appropriate value of the field_rep_cntrl field. If the value of field_rep_cntrl is 0, this is be interpreted in the same way as freeze frame, field_id = ' 10'. The field_rep_cntrl field shall not be coded with a value indicating a number of field repetitions less than in the case of playback at normal speed.

freeze frame-If the value '010' is encoded, then the DSM is in freeze frame mode. In the freeze frame mode, the PTS indicates the value of the system time clock at which the first field from the associated picture to be frozen is to be displayed in the STD model. During the freeze frame mode, the system time clock in the STD model is continuous. However, at the transition to another mode, a discontinuity in the system time clock will occur. The DSM will provide a picture with valid time stamps to the decoder. The field_id flag is only meaningful when dealing with interlaced pictures. In this case these bits are to be interpreted as follows:

Table 2-11 -- Freeze frame control values

value	description
'00'	display from field 1 only
'01'	display from field 2 only
'10'	display complete frame
'11'	reserved

In this definition, displaying from a single field implies repeating the specified field as top and bottom fields alternately until another mode is specified. The DSM temporarily stops providing additional data to the decoder. The decoder buffer may under-flow in this mode.

fast reverse-When the value '011' is encoded, then the DSM is in the fast reverse mode. The DSM will ensure that the buffers in the STD model do not overflow. The DSM will provide each picture with valid timestamps. During the fast reverse mode, the system time clock in the STD model is continuous. However, at the transition to another mode, a discontinuity in the system time clock will occur. When intra_slice_refresh bit is set, it indicates that each picture is composed of intra slices with possible gaps between them. The decoder may replace the missing slices by repeating the co-sited slices of previously

decoded picture(s). The field_id flag is to be interpreted in the same ways as for the freeze frame mode. The "frequency_truncation" field provides an indication of coefficient selection done by the DSM in the process of generating the bitstream in fast play modes.

Table 2-12 -- Fast reverse control values

Value	Description
'00'	Only DC coefficients are sent
'01'	the first three coefficients in scan order on average
'10'	the first six coefficients in scan order on average
'11'	all coefficients may be sent

additional_copy_info: This field contains private data relating to copyright information.

previous_PES_packet_CRC -- The previous_PES_packet_CRC is a 16 bit polynomial given by the equation:

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

The message data to be error checked is processed with the check polynomial by dividing the message data by previous_PES_packet_CRC. This field is calculated over the PES packet data bytes in the previous PES packet being transported in the same PID. The initial vector for the CRC computation shall be 0xFFFF. The resulting 16 bit remainder is placed directly into the previous_PES_packet_CRC field.

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 steam decoders. It is calculated only over the data bytes because PES packet header data can be modified during transport.

PES_private_data_flag -- is a 1 bit flag, 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 when set to '1' indicates that an ISO/IEC 11172 pack header or an ISO/IEC 13818 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 a value '0' it indicates that no pack header is present in the PES header.

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

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

PES_extension_field_flag -- This is a 1 bit field, when set to '1' indicates the presence of the PES_extension_field.

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 a 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 in the program multiplex. This allows an application to retrieve the original PES packet sequence of an ISO/IEC 13818 Program 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.

original_stuff_length -- This 7 bit field specifies the number of stuffing bytes used in the ISO/IEC 11172 packet header.

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 digital storage medium. It is discarded by the decoder. No more than 32 stuffing bytes shall be present in 1 PES packet header.

P-STD_buffer_scale[Program Stream only] -- The P-STD_buffer_scale is a 1 bit field that 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[Program Stream only] --The P-STD_buffer_size is a 13-bit unsigned integer defining 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)

$$BS_n = P\text{-}STD\_buffer\_size \times 128; \quad (2\text{-}13)$$

else

$$BS_n = P\text{-}STD\_buffer\_size \times 1024; \quad (2\text{-}14)$$

The encoded value of the P-STD buffer size takes effect immediately when the P-STD_buffer_size field is received by the ISO/IEC 13818 System Target Decoder. Refer to 2.4.9.7.

PES_extension_field_length -- This 7 bit field specifies the length, in bytes, of the data following this field in the PES_extension_field.

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. 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' that can be inserted by the encoder, for example, to meet the requirements of the DSM. It is discarded by the decoder.

2.4.4 Program specific information

Program Specific Information (PSI) includes both ISO/IEC 13818 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 or elementary streams or parts thereof may be conditionally accessed. However, Program Specific Information shall not be scrambled.

In Transport Streams Program Specific Information (PSI) 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 ISO/IEC 13818 Transport Stream packets.

Table 2-13 -- Program specific information

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

ISO/IEC 13818-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 ISO/IEC 13818-defined PSI tables into Transport Stream packets.

Along with ISO/IEC 13818-defined PSI tables, it is possible to carry private data tables. The means by which private information is carried within Transport Stream packets shall be private. It may be structured in the same manner used for carrying of ISO/IEC 13818-defined PSI tables, such that the syntax for mapping this private data is identical to that used for the mapping of ISO/IEC 13818-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. 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 ISO/IEC 13818.

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 **Error! Reference source not found.** on page **Error! Bookmark not defined.**.

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 byte 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.

All Transport Streams shall contain one or more Transport Stream packets with PID value 0x0000. These Trnsport 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.

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.

All Transport Streams shall contain one or more Transport Stream packets with PID values which are labled under the program association table as Transport Stream packets containing TS program map sections. Every program listed in the program association table shall be described in a single TS program map section. Every program shall be fully described within the Transport Stream itself. 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 PMT_PID for that specific program. All Transport Stream packets which carry a given Program Map section shall have the same PID value.

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 ISO/IEC 13818-defined PSI table is 1K(1024 bytes). The maximum number of bytes in a private_section is 4K (4096 bytes).

There are no restrictions on the occurrence of start codes, sync bytes or other bit patterns in PSI data, whether ISO/IEC 13818 or private.

2.4.4.1 Pointer

The pointer_field syntax is defined in table 2-14 below.

Table 2-14 -- 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 Stream types

The stream types specified in table 2-31 are valid for both Transport Stream and Program Stream Program Specific Information.

2.4.4.4 Specification of the 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.

Program number 0x0000 is reserved to specify the network PID. This identifies the Transport Stream packets which carry the Network Information Table.

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

Table 2-15 -- 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**
}		

2.4.4.5 Program association table

2.4.4.6 Table_id assignments

The table_id field identifies the content of a Transport Stream PSI section as shown in table 2-16 below.

Table 2-16 -- table_id assignment values

value	description
0x00	program_association_section
0x01	conditional access_section
0x02	program_map_section
0x03-0x3F	ISO/IEC 13818 reserved
0x40-0xFE	User private
0xFF	forbidden

2.4.4.7 Sematics for fields in program association section

table_id -- This is an 8 bit field, which shall be set to 0x00 as shown in table 2-16 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.

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 case 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.

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 not take values reserved for other purposes. 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 not take the values reserved for other purposes.

CRC_32 -- CRC_32 is a 32 bit polynomial given by the equation:

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

The message data to be error checked is processed with the check polynomial by dividing the message data by CRC_32. The initial vector for the CRC computation shall be 0xFFFFFFFF. The resulting 32 bit remainder is placed directly into the CRC_32 field.

2.4.4.8 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.

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

Table 2-17 -- 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**
}		

2.4.4.9 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 2-16 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.

version_number -- This 5 bit field is the version number of the whole conditional access table. The version number shall be incremented by 1 when a change in the information carried within the CA table occurs. When it reaches 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 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 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 -- CRC_32 is a 32 bit polynomial given by the equation:

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

The message data to be error checked is processed with the check polynomial by dividing the message data by CRC_32. The initial vector for the CRC computation shall be 0xFFFFFFFF. The resulting 32 bit remainder is placed directly into the CRC_32 field.

2.4.4.10 Program Map Table

The Program Map Table provides the mappings between program numbers and the elementary streams 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 privately selected. 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 .

Table 2-18 -- 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<P;i++) {		
stream_type	**8**	**uimsbf**
reserved	**3**	**bslbf**
elementary_PID	**13**	**uimsnf**
reserved	**4**	**bslbf**
ES_info_length	**12**	**uimsbf**
for (i=0; i<Q; i++) {		
descriptor()		
}		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.11 Semantics 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.

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.

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.

version_number -- This 5 bit field is the version number of the TS_program_map_section. The version number shall be incremented by 1 when a change in the information carried within the section occurs. Upon reaching the value 31, it wraps around to 0. 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 indicator, 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 packet 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.

program_info_length -- This is a twelve 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 elementary stream or payload carried within the packets with the PID whose value is specified by the elementary_PID. The values of stream_type are specified in table 2-26 on page 50.

elementary_PID -- This is a 13 bit field specifying the PID of the Transport Stream packets which carry the associated elementary stream or payload.

ES_info_length -- This is a twelve bit field, the first two bits of which shall be '00'. It specifies the number of bytes of the descriptors of the associated elementary stream immediately following the ES_info_length field.

CRC_32 -- CRC_32 is a 32 bit polynomial given by the equation:

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

The message data to be error checked is processed with the check polynomial by dividing the message data by CRC_32. The initial vector for the CRC computation shall be 0xFFFFFFFF. The resulting 32 bit remainder is placed directly into the CRC_32 field.

2.4.4.12 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 to enable an ISO/IEC 13818 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 first four fields 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 2-19 -- 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**
}		
}		

2.4.4.13 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 2-16 on page 33).

section_syntax_indicator -- This is a 1 bit indicator. Set to '1', it indicates that the private section header follows the generic section header syntax beyond the private_section_length field. Set to '0', it indicates that the private_data_bytes are immediately following the private_section_length field.

private_indicator -- This is a 1 bit user definable flag that shall not be specified by 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.

private_data_byte -- The private_data_byte field is user definable and shall not be specified by ISO 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 when a change in the information carried within the private_section occurs. Upon reaching value 31, it wraps around to 0. 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 indicator, which when set to '1' indicates that the private_section sent is the currently applicable section. 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 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 -- CRC_32 is a 32 bit polynomial given by the equation:

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

The message data to be error checked is processed with the check polynomial by dividing the message data by CRC_32. The initial vector for the CRC computation shall be 0xFFFFFFFF. The resulting 32 bit remainder is placed directly into the CRC_32 field.

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.

An ISO/IEC 13818 Program Stream consists of one or more elementary streams from one program multiplexed together. Each elementary stream consists of access units, which are the coded representation of presentation units. The presentation unit for a video elementary stream is a picture. The corresponding access unit includes all the coded data for the picture. The access unit containing the first coded picture of a group of pictures also includes any preceding data from that group of pictures, as defined in Part 2 of this Recommendation | International Standard starting with the group_start_code. The Access Unit containing the first coded picture after a sequence header, as defined in Part 2 of this Recommendation | International Standard also includes that sequence header. The sequence_end_code is included in the Access Unit containing the last coded picture of a sequence. (See Part 2 for the definition of the sequence_end_code). The presentation unit for an audio elementary stream is the set of samples that corresponds to samples from an audio frame (see Part 3 of this Recommendation | International Standard for the definition of an audio frame).

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 (see table 2-24 on page 47) to which the packet data belongs. The PES packet header may contain decoding and/or presentation time-stamps (DTS and PTS) that refer to the first access unit that commences in the packet. 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 clause 2.5.3 on page 43 and the constraints on these semantics specified in clause 2.5.7 on page 66 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 ISO/IEC 13818 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 ISO/IEC 13818 Program Streams from a variety of decoders with different architectures or timing schedules.

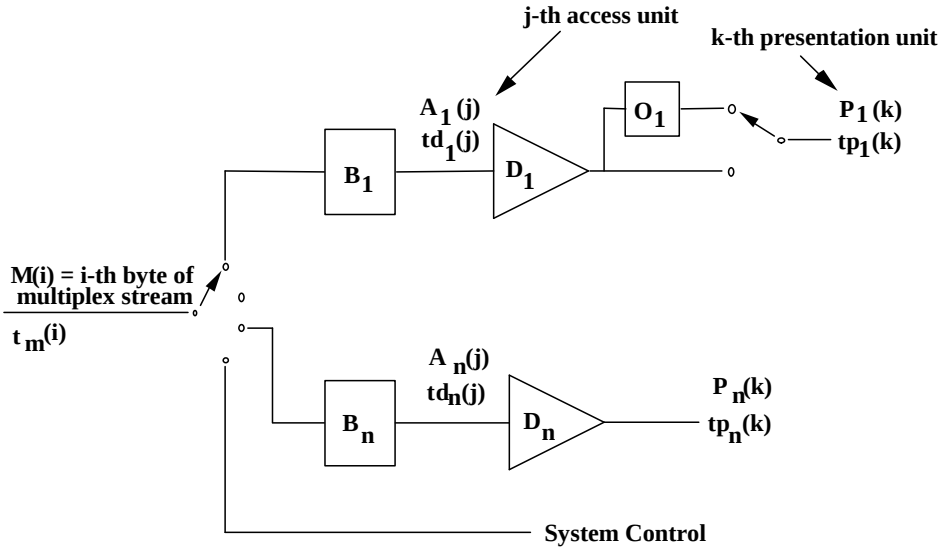


Figure 2-7 -- ISO/IEC 13818 Program Stream system target decoder notation

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 ISO/IEC 13818 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.

M(i) is the i^{th} byte in the Program Stream.

$t_m(i)$ indicates the time in seconds at which the i^{th} byte of the ISO/IEC 13818 Program Stream enters the system target decoder. The value $t_m(0)$ 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 SCR field.

$A_n(j)$ is the j^{th} access unit in elementary stream n. Note that access units are 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.

$tp_n(k)$ 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.

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 clause 2.5.3.3 on page 43, and clause 2.4.3.6 on page 21. 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 - 1 350 <= system_clock_frequency <= 27 000 000 + 1 350

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). 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 the system clock frequency divided by 300 for PTS and DTS.

Input to the Program Stream system target decoder

Data from the ISO/IEC 13818 Program Stream enters the system target decoder. The i^{th} byte, M(i), enters at time $t_m(i)$. 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 encoded in the pack header. The SCR, as

defined in equation 2-17 , is coded in two parts; one, in units of 1/300 × the system clock frequency, called SCR_base (equation 2-15), and one, called SCR_ext (equation 2-16), in units of the system clock frequency. The value encoded in the SCR(i') field indicates time $t_m(i')$, where i' refers to the byte containing the last bit of the SCR_base field, M(i').

Specifically:

$$SCR\_base(i') = ((system\_clock\_frequency \times t_m(i'))/300)\%2^{33} \quad (2\text{-}15)$$

$$SCR\_ext(i') = NINT(system\_clock\_frequency \times t_m(i'))\%(2^{33} \times 300) - SCR\_base(i') \times 300 \quad (2\text{-}16)$$

$$SCR(i') = SCR\_base(i') \times 300 + SCR\_ext(i') \quad (2\text{-}17)$$

The input arrival time, $t_m(i)$, as given in equation 2-18, 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.

$$t_m(i) = \frac{SCR(i')}{system\_clock\_reference} + \frac{i - i'}{program\_mux\_rate \times 50} \quad (2\text{-}18)$$

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 clause 2.5.3.3 on page 43.

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.

Buffering

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

Bytes present in the pack header, system headers, or PES packet headers of the ISO/IEC 13818 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 2-13 and equation 2-14 on page 29.

At the decoding time, $td_n(j)$, all the data for the access unit that has been in the input buffer longest ($A_n(j)$) is removed instantaneously. In the case of an ISO/IEC 13818 video stream the decoding time, $td_n(j)$, may be derived from subclauses C.9 to C.11 of annex C of ISO/IEC 13818-2. The data defined as belonging to an ISO/IEC 13818 video access unit for this purpose is defined in subclause C.5 of annex C of ISO/IEC 13818-2. In the case of an ISO/IEC 13818 audio stream access units consist of audio frames and are decoded at a time defined in ISO/IEC 13818-3. As the access unit is removed from the buffer it is instantaneously decoded to a presentation unit.

When the **low_delay** flag in the video sequence extension is set to '1' (subclause 6.2.2.3 of 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. Overflow of buffer B_n shall not occur.

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.

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. Such a stream is called a PES stream. The P-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 the demultiplexor sends data to only one elementary stream buffer.

Decoding and presentation

Decoding and presentation in the Program Stream system target decoder (P-STD) are exactly the same as defined for the Transport Stream system target decoder (T-STD), in clause 2.4.2 on page 10.

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 2-20 -- ISO/IEC 13818 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 ISO/IEC 13818 Program Stream.

2.5.3.3 Pack layer of Program Stream

Table 2-21 -- ISO/IEC 13818 Program Stream pack

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

Table 2-22 -- 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 -- The system_clock_reference (SCR) is a 42 bit number coded in four separate fields. It indicates the intended time of arrival of the byte containing the last bit of the system_clock_reference base (SCR_base) field at the input of the system target decoder. The value of the SCR_base is measured in the number of periods of a 90kHz clock with a fractional tolerance specified in clause 2.5.2 on page 40. The SCR_ext is measured in the number of periods of a 27MHz system clock with a tolerance specified in clause 2.5.2 on page 40. Using the notation of clause 2.5.2 on page 40 the value encoded in the system_clock_reference is:

$$SCR\_base(i') = \left((system\_clock\_frequency * t_m(i')) / 300 \right) \% 2^{33} \quad (2\text{-}19)$$

$$SCR\_ext(i') = NINT\left(system\_clock\_frequency * t_m(i')\right) \% \left(2^{33} * 300\right) - SCR\_base(i') * 300 \quad (2\text{-}20)$$

$$SCR(i') = SCR\_base(i') * 300 + SCR\_ext(i') \quad (2\text{-}21)$$

for i such that $t_m(i)$ is the last byte of the coded system_clock_reference field.

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

program_mux_rate -- This is a positive integer specifying the rate at which the P-STD receives the ISO/IEC 13818 Program Stream during the pack in which it is included. The value of program_mux_rate is

measured in units of 50 bytes/second rounded upwards. 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 clause 2.5.2 on page 40 of this Recommendation | International Standard. The value encoded in the program_mux_rate field may vary from pack to pack in an ISO/IEC 13818 program multiplexed stream.

pack_stuffing_length -- A non-negative integer specifying the number of stuffing bytes which follow this field. The minimum value is 0 and the maximum value is 7.

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 digital storage medium. It is discarded by the decoder. In each pack header no more than 7 stuffing bytes shall be present in 1 system header.

2.5.3.5 System header

Table 2-23 -- 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**
reserved_byte	**8**	**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 inidcates 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 -- The rate_bound is an integer value greater than or equal to the maximum value of the mux_rate field coded in any pack of the ISO/IEC 13818 Program Stream. It may be used by a decoder to assess whether it is capable of decoding the entire stream.

audio_bound -- The audio_bound is an integer in the inclusive range from 0 to 32 greater than or equal to the maximum number of audio streams in the ISO/IEC 13818 Program Stream of which the decoding processes are simultaneously active. For the purpose of this Clause, the decoding process of an ISO/IEC 13818 audio stream is active, if the STD buffer is not empty, or if the decoded Access Unit is being presented in the 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 ISO/IEC 13818 stream shall adhere to the following linear equation:

$$SCR(i) = NINT(c1 * i + c2)\%2^{33} \qquad (2\text{-}22)$$

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 multiplexed ISO/IEC 13818 stream of the byte containing the last bit of any system_clock_reference field in the stream.

CSPS_flag -- The CSPS_flag is a 1 bit flag. If its value is set to '1' the ISO/IEC 13818 Program Stream meets the constraints defined in clause 2.5.7.9 on page 69 of this Recommendation | International Standard.

system_audio_lock_flag -- The system_audio_lock_flag is a 1 bit flag indicating that there is a specified, constant rational relationship between the audio sampling rate and the system clock frequency in the system target decoder. Clause 2.5.2 on page 40 defines system_clock_frequency and the audio sampling rate is specified in Part 3 of this Recommendation | International Standard. The system_audio_lock_flag may only be set to '1' if, for all presentation units in all audio elementary streams in the ISO/IEC 13818 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.

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

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

Nominal audio sampling frequency (kHz)	16	32	22,05	44,1	24	48
Ratio 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 flag indicating that there is a specified, constant rational relationship between the video picture rate and the system clock frequency in the system target decoder. Clause 2.4.5 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 ISO/IEC 13818 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.

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

Nominal picture rate (Hz)	23,976	24	25	29,97	30	50	59,94	60
Ratio 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 an integer in the inclusive range from 0 to 16 greater than or equal to the maximum number of ISO/IEC 13818-2 streams in the ISO/IEC 13818 Program Stream of which the decoding processes are simultaneously active. For the purpose of this Clause, the decoding process of an ISO/IEC 13818 video stream is active if the STD buffer is not empty, or if the decoded Access Unit is being presented in the STD model, or if the reorder buffer is not empty.

reserved_byte -- This byte is reserved for future use by ISO/IEC. Until otherwise specified by ISO/IEC it shall have the value '1111 1111'.

stream_id -- The stream_id 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 ISO/IEC 13818 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 ISO/IEC 13818 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 the following table. This table is used also to identify the stream coding and elementary stream number indicated by the stream_id defined in below in table 2-24 below.

Table 2-24 -- Stream_id table

stream_id	stream coding
1011 1100	program stream map
1011 1101	private_stream_1†
1011 1110	padding stream
1011 1111	private_stream_2‡
110x xxxx	ISO/IEC 13818-3 audio stream or ISO/IEC 11172-3 audio stream - number xxxxx
1110 xxxx	ISO/IEC 13818-2 video stream or ISO/IEC 11172-2video stream - number xxxx
1111 0000	ECM
1111 0001	EMM
1111 0010	DSM CC
1111 0011	ISO/IEC 13522 stream
1111 xxxx	reserved data stream - number xxxx '0100'-'1110' are reserved
1111 1111	program stream directory

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

† PES packets of type private_stream_1 follow the same PES packet header syntax as those for ISO/IEC 13818 audio and video streams.

‡ PES packets of type private_stream_2 are similar to private_stream_1 except that no syntax is specified after PES_packet_length field.

Each elementary stream present in the ISO/IEC 13818 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 STD_buffer_size_bound is a 13 bit unsigned integer defining a value greater than or equal to the maximum System Target Decoder input buffer size, BS_n, over all packets for stream n in the ISO/IEC 13818 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 clause 2.4.3.6 on page 21.

2.5.3.8 ISO/IEC 13818 Pack header field

As described in clause 0.3 on page xiv, conversion between Transport Streams and Program Streams is possible by means of PES packets. The ISO/IEC 13818 Pack Header Field, when present in a PES packet, contains pack_header data which may also include system_header data. In the conversion from a Transport Stream to a Program Stream the ISO/IEC 13818 Pack Header Field is removed from all PES packets with PIDs from one program, and placed ahead of the associated PES packet header in the data stream as defined in clause 2.5.3.3 on page 43 . The pack_header_field_flag value shall then be set to '0' indicating that the ISO/IEC 13818 Pack Header Field is no longer present the PES packet header. The result is a Program Stream.

When converting from a Program Stream to a Transport Stream the pack_header data, including the system_header data if present, is moved from its position in the Program Stream and placed within the PES packet header as defined in clause 2.4.3.6 on page 21. The pack_header_field_flag value shall then be set to '1' indicating the ISO/IEC 13818 Pack Header Field is present in the PES packet header. The result is a stream of PES packets which may be used in a single or multi-program transport multiplex.

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 in a PES packet when the stream_id value is 0xBC.

Definition for the descriptor() fields may be found in clause 2.5.6 on page 53 of this Recommendation | International Standard.

2.5.4.1 Syntax of Program Stream map

Table 2-25 -- 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<N;i++) {		
stream_type	**8**	**uimsbf**
elementary_stream_id	**8**	**uimsbf**
elementary_stream_info_length	**16**	**uimsbf**
for (i=0;i<N;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 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 1024 (1K).

current_next_indicator -- A 1 bit indicator, 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 whenever the definition of the Program Stream Map changes. Afer reaching the value 31, it wraps around to 0. 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 2-26 -- Stream type assignments

Value	Description
0x00	ISO/IEC Reserved
0x01	ISO/IEC 11172 Video
0x02	ISO/IEC 13818 Video
0x03	ISO/IEC 11172 Audio
0x04	ISO/IEC 13818 Audio
0x05	ISO/IEC 13818 private_sections
0x06	ISO/IEC 13818 PES packets containing private data
0x07	ISO/IEC 13522 MHEG
0x08	ISO/IEC 13818 DSM CC
0x09	ISO/IEC 13818 Private data
0x0A-0x7F	ISO/IEC 13818 Reserved
0x80-0xFF	User Private

elementary_stream_id -- The elementary_stream_id is an 8 bit field indicating the value of the stream_id field in the PES packet header 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 -- CRC_32 is a 32 bit polynomial given by the equation:

$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

The message data to be error checked is processed with the check polynomial by dividing the message data by CRC_32. The initial vector for the CRC computation shall be 0xFFFFFFFF. The resulting 32 bit remainder is placed directly into the CRC_32 field.

2.5.5 Program Stream directory

2.5.5.1 Syntax of the PES packet header for the Program Stream directory

The stream_id '1111 1111' is the directory_stream_id and denotes a PES packet containing a portion of an ISO/IEC 13818 directory as defined in this clause.

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 payload of each directory PES packet is defined as below. In packets identified by a directory_stream_id the following PES packet header fields shall have the following values:

Where 'x' indicates that the correct value as defined in the semantics of clause 2.4.3.6 on page 21 shall be inserted.

Table 2-27 -- PES packet header for Program Stream directory

PES packet header Variable	Assigned Value
packet_start_code_prefix	'0000 0000 0000 0000 0000 0001'
stream_id	'1111 1111'
PES_packet_length	**'nn'**
	'10'
PES_scrambling_control	**'xx'**
PES_priority	**'x'**
data_alignment_indicator	'0'
copyright	**'x'**
original_or_copy	**'x'**
PTS_DTS_flags	'00'
ESCR_flag	'0'
ES_rate_flag	'0'
DSM_trick_mode_flag	'0'
additional_copy_info_flag	'0'
PES_CRC_flag	'0'
PES_extension_flag	'0'
PES_header_data_length	**'xxxx xxxx'**

The rest of the PES packet header shall follow the syntax and semantics as defined in clause 2.4.3.6 on page 21 and clause 2.4.3.7 on page 24.

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 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 requires to reference shall be referenced by the first byte of the picture header.

Note: 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 part 3 of this Recommendation | International standard shall be the syncword of the audio frame.

Note: 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.

2.5.5.2 Syntax of the Program Stream directory

Table 2-28 -- PES packet payload for Program Stream directory

Syntax	No of bits	Mnemonic
Directory_PES_Packet_Payload(){		
number_of_access_units	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
next_directory_offset[44..30]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
next_directory_offset[29..15]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
next_directory_offset[14..0]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
for(i=0; i < number_of_access_units; i++){		
packet_stream_id	**8**	**uimsbf**
PES_header_position_offset[44..30]	**15**	**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**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
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**
}		
}		

2.5.5.3 Semantic definition of fields in Program Stream directory

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

next_directory_offset - This 45-bit unsigned integer gives the byte address of the first byte of the next PES packet that contains Program Stream Directory information. The first byte of a Program Stream has byte address 0x0.

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

PES_header_position_offset - This 45-bit unsigned integer gives the byte address of the first byte of the PES packet containing the access unit referenced relative to the first byte of the Program Stream.

reference_offset - This 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.

presentation_time_stamp - This is the PTS of the access unit that is referenced. The semantics of the coding of the PTS field are as described in clause 2.4.3.6 on page 21.

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'.

Table 2-29 -- Intra_coded indicator

Value	Meaning
0	Not Intra
1	Intra

coding_parameters_indicator - This 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 2-30 -- 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.

2.5.6 Program and elementary stream descriptors

Program and Elementary stream descriptors are structures which may be used to extend the definitions of programs and elementary streams. All stream descriptors have a format which begins with an 8 bit tag value. It is followed by an 8 bit descriptor length and data fields.

Table 2-31 on page 54below provides the ISO/IEC 13818 defined, ISO/IEC 13818 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 2-31 -- Stream 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
14-63	n/a	n/a	ISO/IEC 13818 Reserved
64-255	n/a	n/a	User Private

2.5.6.1 Semantic definition of fields in stream descriptor

The following semantics apply to the descriptors defined in 2.4.8.

descriptor_tag -- The descriptor_tag is an 8 bit field which identifies each descriptor. Its meaning is given in table 2-31 . Some values have normative meaning, some are reserved for future ISO/IEC use and the remaining values may be user defined.

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

2.5.6.2 Video stream descriptor

The video stream descriptor provides basic information which identifies the coding version of a video elementary stream.

Table 2-32 -- Video stream descriptor

Syntax	No. of bits	Mnemonic
video_stream_descriptor(){		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
video_coding_version	**8**	**uimsbf**
}		

2.5.6.3 Semantic definitions of fields in video stream descriptor

video_coding_version -- The video_coding_version is an 8 bit field which specifies the type of video coding used for the associated elementary stream. It's values are shown in the table 2-33 below.

Table 2-33 -- Video coding version

video_coding_version	description
TBD	**TBD**

2.5.6.4 Audio stream descriptor

The audio stream descriptor provides basic information which identifies the coding version of an audio elementary stream.

Table 2-34 -- Audio stream descriptor

Syntax	No. of bits	Mnemonic
audio_stream_descriptor(){		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
audio_coding_version	**8**	**uimsbf**
}		

2.5.6.5 Semantic definition of fields in audio stream descriptor

audio_coding_version -- The audio_coding_version is an 8 bit field which specifies the type of audio coding used for the associated elementary stream. Its values are shown in the table 2-35 below.

Table 2-35 -- Audio coding version

audio_coding_ version	description
TBD	**TBD**

2.5.6.6 Hierarchy descriptor

The hierarchy descriptor provides identifying information for the components (streams) of a hierarchically-coded video.

Table 2-36 -- Hierarchy descriptor

Syntax	No. of bits	Mnemonic
hierarchy_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
hierarchy_type	**4**	**uimsbf**
hierarchy_layer	**6**	**uimsbf**
hierarchy_embedded_layer	**6**	**uimsbf**
}		

2.5.6.7 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 2-37 on page 56below.

Table 2-37 -- Hierarchy descriptor values

Value	Description
0	reserved
1	Spatial Scalability
2	SNR Scalability
3	Temporal Scalability
4	Data partitioning
5-14	reserved
15	Base layer

Hierarchy_layer -- The hierarchy_layer is a 6 bit field that identifies the relative position of the associated elementary stream in the overall hierarchy of video layers. The lowest (base layer) is assigned a value of 0, the next is assigned a value 1, an so on.

Hierarchy_embedded_layer -- The hierarchy_embedded_layer is a 6 bit field that identifies the hierarchy position of the video elementary stream that needs to be accessed for decoding of the elementary steam described by the descriptor containing this field.

2.5.6.8 Registration descriptor

The registration_descriptor provides a method of indicating a registration_id, which has been obtained from recognized bodies to enable signalling of certain formats. This enables the unambiguous signalling of formats for private streams. This registration_id may apply to programs or streams within programs.

Table 2-38 -- Registration descriptor

Syntax	No. of bits	Identifier
registration_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
registered_id	**8 ... 2040**	**uimsbf**
}		

2.5.6.9 Semantic definition of fields in registration descriptor

RAC --This integer indicates the registration authority committee that registers the meanings associated with the following Registered_ID. An initial table of RACs is table 2-39 below.

Table 2-39 -- Registration authority committee

RAC	Description
0	Not Registered
1	ISO/IEC 13818
2	ISO/ITU
3	IEEE
4	ETSI
5	CENELEC
6	CEN
7-65535	Reserved

registered_id -- This byte string provides a globally unique label. The registered_id value consists of data bytes; these data bytes are structured into an initial datum and zero or more additional datum. All of which are encoded using the ASN.1 Basic Encoding Rules (BER), as illustrated in figure 2-8 below.

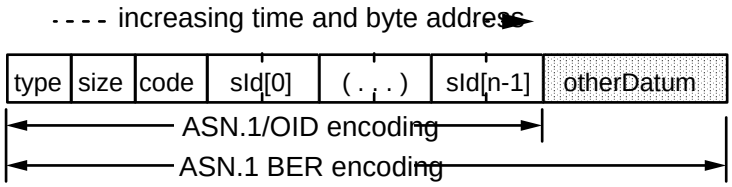


Figure 2-8 -- ISO/ITU identifier string

The initial datum is an ASN.1 encoded hierarchical object identifier. The hierarchial identifier is encoded according to the Basic Encoding Rules specified by ISO 8825[1] and restricted according to the Distinguished Encoding Rules specified by CCITT X.509 section 8.7[2].

The format of the first datum can be summarized as follows. The type byte identifies the first datum as an object identifier and shall have the value of type=6. The size byte specifies the number of additional bytes (after the size byte value) within the first datum and shall be less than the decimal value of 127 (hexadecimal value of "7F"). The code value specifies the organization responsible for specifying the initial subidentifier value, as summarized within table 2-40 below. The subidentifiers are variable-length encoded positive integers. The subidentifiers within the first datum are hierarchical, in the sense that each subidentifier is administered by the organization affiliated with the previous subidentifier value.

The encoding rules are open and universal.

[1]International Standard ISO 8824/8825 (CCITT X.209/209), Abstract Syntax Notation 1 (ASN.1)

[2]International Standard CCITT X.509, The Directory Authentication Framework, Section 8.7 (Distinguished Encoding Rules)

Table 2-40 -- Identifier code field values

Organization	Code	Name	Description
ITU	0	recommendation	Committees
	1	question	Study Groups
	2	administration	country PTT's (country code)
	3	network operator	X.121 organizations
	4-39		reserved
ISO	40	standard	ISO Standards
	41	registration authority	ISO Authorities
	42	member body	member bodies(country code)
	43	identified organization	organizations
	44-79		reserved
joint ISO ITU (delegated to ANSI)	>= 80		reserved

When they exist, the encoding of the second-through-final datum consists of a 2-byte tag, as illustrated in figure 2-9. The type byte specifies the datum type. For example, an octet (byte) string would be identified by the type=3 value. The second byte specifies the number of additional bytes within the datum.

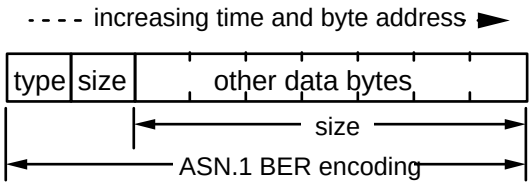


Figure 2-9 -- Other ASN.1 encoded datums

Note that each identified registration authority is responsible for specifying which type value that is used, so that their assigned values can be represented in their most natural form (subject to the constraints of ASN.1 encoding rules). table 2-41 illustrates the commonly used values of ISO OID values.

Table 2-41 -- Registration authority identifiers

ISO OID	Description
TBD	MPEG
TBD	IEEE
TBD	ETSI
TBD	CENELEC
TBD	CEN

Editorial note:
The following OID values are in the process of being identified. Any OID not identified by the publication date should be deleted.

For further information on the content and format of the datum following these OID values (or the remainder of the OID) contact the following organizations:

IEEE Standards Department
445 Hoes Lane

Piscataway, NJ 08855-1331
USA
Phone: +1 (908) 562-3813
FAX: +1 (908) 562-1571

European Telecommunications Standards Institute
F-06921 Sofia Antipolis Cedex
FRANCE
Phone: + 33 92 94 42 00
Fax: + 33 93 65 47 16

European Committee for Electrotechnical Standardization
CENELEC:
rue de Stassart, 35
B-1050 Bruxelles
BELGIUM
Phone: +32 2 519 6871
Fax: +32 2 519 69 19

European Committee for Standardization
CEN
rue de Stassart, 36
B-1050 Bruxelles
BELGIUM
Phone: +32 2 519 6811
Fax: + 32 2 519 68 19

Other organizations with an assigned ISO OID may define the format of their following datum (or the remainder of the OID).

2.5.6.10 Data stream alignment descriptor

The data stream alignment descriptor describes which type of alignment is present in the associated elementary stream. If data_alignment_indicator in the PES packet header is set to '1' it shall indicate the type of alignment specified in this descriptor.

Table 2-42 -- 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**
}		

2.5.6.11 Semantics of fields in data stream alignment descriptor

alignment_type -- The following data_stream_alignment_descriptor table describes the video alignment type when the data_alignment_indicator in the PES packet header has a value of '1'.

Table 2-43 -- Video stream alignment values

alignment type	Description
00	reserved
01	Slice, picture, GOP, or SEQ
02	picture, GOP, or SEQ
03	GOP, or SEQ
04	SEQ
05-FF	reserved

The following data_stream_alignment_descriptor table describes the audio alignment type when the data_alignment_indicator in the PES packet header has a value of '1'.

Table 2-44 -- Audio stream alignment values

alignment type	Description
00	reserved
01	Audio frame
02-FF	reserved

2.5.6.12 Target background grid descriptor

The target background grid descriptor is used to describe the display format background grid for windowing operations.

Table 2-45 -- 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**
pel_aspect_ratio	**4**	**uimsbf**
}		

2.5.6.13 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.

pel_aspect_ratio -- Pel aspect ratio of the target background grid value is as defined in the sequence header in video. See Part 2 of this Recommendation | International Standard and Part 2 of ISO 11172.

2.5.6.14 Video window descriptor

The video window descriptor is used to describe the window characteristics of the associated elementary stream. Its values reference the target background grid descriptor for the same stream.

Table 2-46 -- 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**
}		

2.5.6.15 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 window on the target background grid for display as defined in the target_background_grid_descriptor.

vertical_offset -- The value indicates the vertical position of the top left pixel of the current video window on the target background grid for display as defined in the target_background_grid_descriptor.

window_priority -- The value indicates how windows overlap. A value of 1 being lowest priority and a value of 15 is the highest priority, e.g. windows with priority 15 are always visible.

2.5.6.16 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 must be present for the program containing that elementary stream. If any system-wide management information exists within a Transport Stream, a CA descriptor must be present in the appropriate map section.

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 extended program information indicates applicability to the entire program. In the same case, its presence as extended ES information indicates applicability to the associated elementary stream. 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 private.

Table 2-47 -- 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**
}		
}		

2.5.6.17 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.

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. the program map table or the CA table in the Transport Stream or the stream_id field in the Program Stream.

2.5.6.18 ISO 639 language descriptor

The language descriptor is used to specify the language of the associated elementary stream.

Table 2-48 -- 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**
}		

2.5.6.19 Semantic definition of fields in ISO 639 language descriptor

ISO_639_language_code -- Identifies the language or languages used by the associated elementary stream. 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 and inserted in order into the 24 bit field. In the case of multilingual audio streams the value of descriptor_length is 1 greater than 3 times the number of languages carried in the associated audio stream. The sequence of ISO_639_language_code shall reflect the content of the audio stream.

audio_type -- The audio_type is an 8 bit field which specifies the audio type defined by the following table.

Table 2-49 -- Audio type values

Value	Description
0x00	reserved
0x01	clean_effects
0x02	hearing_impaired
0x03	visual_impaired_commentary
0x40-0xFF	reserved

clean_effects -- This field indicates that the referenced elementary stream has no language.

hearing_impaired -- This field indicates that the referenced elementary stream is prepared for the hearing imparied.

visual_impaired_commentary -- This field indicates that the referenced elementary stream is prepared for the visually impaired viewer.

2.5.6.20 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 is set. The decoder optionally can use the same external reference if it is available.

If the system clock is more accurate than the 50 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 50 ppm.
When both parts of the descriptor are used, the clock accuracy pertains to the external reference clock.

Table 2-50 -- 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**
}		

2.5.6.21 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.

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.

2.5.6.22 Multiplex buffer utilization descriptor

Table 2-51 -- Multiplex buffer utilization descriptor

Syntax	No. of bits	Mnemonic
multiplex_buffer_utilization_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
multiplex_buffer_utilization	**8**	**uimsbf**
}		

2.5.6.23 Semantic definition of fields in multiplex buffer utilization descriptor

multiplex_buffer_utilization -- The multiplex_buffer_utilization is an 8 bit field specifying the upper bound of the portion given by ...

$$\frac{multiplex\_buffer\_utilization + 1}{256} \quad (2\text{-}26)$$

...of the buffers BS_{mux} in a T-STD that may have been utilized in the multiplexing of this program. This upper bound shall not be exceeded.

2.5.6.24 Copyright descriptor

The copyright_descriptor provides a method of indicating a copyright_id, which has been obtained from recognized bodies to enable audio-visual works identification. This copyright_id may apply to programs or elementary streams within programs.

Table 2-52 -- Copyright descriptor

Syntax	No. of bits	Identifier
copyright_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
copyright_id	**8 ... 2040**	**uimsbf**
}		

2.5.6.25 Semantic definition of fields in copyright descriptor

copyright_id -- This byte string provides a globally unique label. The copyright_id value consists of data bytes; these data bytes are structured into an initial datum and zero or more additional datum. All of which are encoded using the ASN.1 Basic Encoding Rules (BER), as illustrated in figure 2-10 below.

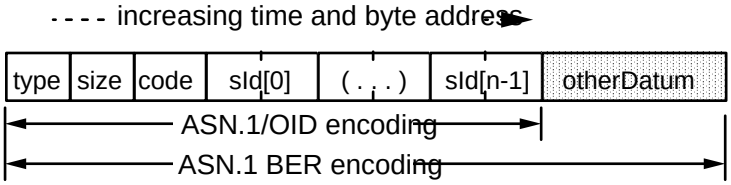


Figure 2-10 -- ISO/ITU identifier string

The initial datum is an ASN.1 encoded hierarchical object identifier. The hierarchial identifier is encoded according to the Basic Encoding Rules specified by ISO 8825[3] and restricted according to the Distinguished Encoding Rules specified by CCITT X.509 section 8.7[4].

The format of the first datum can be summarized as follows. The type byte identifies the first datum as an object identifier and shall have the value of type=6. The size byte specifies the number of additional bytes (after the size byte value) within the first datum and shall be less than the decimal value of 127 (hexadecimal value of "7F"). The code value specifies the organization responsible for specifying the initial subidentifier value, as summarized within table 2-53 below. The subidentifiers are variable-length encoded positive integers. The subidentifiers within the first datum are hierarchical, in the sense that each subidentifier is administered by the organization affiliated with the previous subidentifier value.

The encoding rules are open and universal.

[3]International Standard ISO 8824/8825 (CCITT X.209/209), Abstract Syntax Notation 1 (ASN.1)

[4]International Standard CCITT X.509, The Directory Authentication Framework, Section 8.7 (Distinguished Encoding Rules)

Table 2-53 -- Identifier code field values

Organization	Code	Name	Description
ITU	0	recommendation	Committees
	1	question	Study Groups
	2	administration	country PTT's (country code)
	3	network operator	X.121 organizations
	4-39		reserved
ISO	40	standard	ISO Standards
	41	registration authority	ISO Authorities
	42	member body	member bodies(country code)
	43	identified organization	organizations
	44-79		reserved
joint ISO ITU (delegated to ANSI)	>= 80		reserved

When they exist, the encoding of the second-through-final datum consists of a 2-byte tag, as illustrated in figure 2-11. The type byte specifies the datum type. For example, an octet (byte) string would be identified by the type=3 value. The second byte specifies the number of additional bytes within the datum.

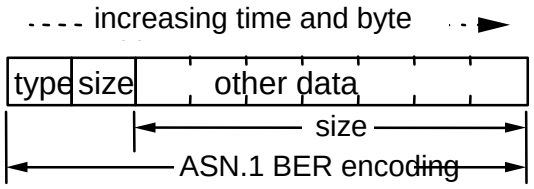


Figure 2-11 -- Other ASN.1 encoded datums

Note that each identified copyright authority is responsible for specifying which type value that is used, so that their assigned values can be represented in their most natural form (subject to the constraints of ASN.1 encoding rules). table 2-54 illustrates the commonly used values of ISO OID values.

Table 2-54 -- Copyright authority identifiers

ISO OID	Description
TBD	MPEG
TBD	IEEE
TBD	ETSI
TBD	CENELEC
TBD	CEN

Editorial note:
The following OID values are in the process of being identified. Any OID not identified by the publication date should be deleted.

For further information on the content and format of the datum following these OID values (or the remainder of the OID) contact the following organizations:

IEEE Standards Department
445 Hoes Lane
Piscataway, NJ 08855-1331
USA

Phone: +1 (908) 562-3813
FAX: +1 (908) 562-1571

European Telecommunications Standards Institute
F-06921 Sofia Antipolis Cedex
FRANCE
Phone: + 33 92 94 42 00
Fax: + 33 93 65 47 16

European Committee for Electrotechnical Standardization
CENELEC:
rue de Stassart, 35
B-1050 Bruxelles
BELGIUM
Phone: +32 2 519 6871
Fax: +32 2 519 69 19

European Committee for Standardization
CEN
rue de Stassart, 36
B-1050 Bruxelles
BELGIUM
Phone: +32 2 519 6811
Fax: + 32 2 519 68 19

Other organizations with an assigned ISO OID may define the format of their following datum (or the remainder of the OID).

2.5.7 Restrictions on the multiplexed stream semantics

2.5.7.1 Buffer management

The ISO/IEC 13818 Program Stream, M(i), in the notation described in clause 2.5.2 on page 40, and the PESpackets defined in clause 2.4.3.6 on page 21 shall be constructed and $t_m(i)$ 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 $t=t_m(0)$.

$F_n(t)$ 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 VBV buffer underflow occurs; see clause 2.5.2 on page 40.

The ISO/IEC 13818 Transport Stream, M(i), in the notation described in clause 2.4.2 on page 10 shall be constructed and $t_m(i)$ shall be chosen so that the input buffers of size BS_1 through BS_n do not underflow or overflow, and the transport buffers TB_1 through TB_n do not overflow in the transport system target decoder. That is:

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

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

$F_n(t)$ is the instantaneous fullness of T-STD main buffer B_n, and Ft_n is the instantaneous fullness of the T-STD transport buffer TB_n.

An exception to this condition is that the buffers BS_n may underflow when "skipped picture" occur; see clause 2.4.2 on page 10.

The system main buffer B_{sys} may underflow. TB_{sys} shall not overflow.

For all ISO/IEC 13818 Program Streams and Transport 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:

$$td_n(j) - t_m(i) \leq 1$$

For all bytes M(i) contained in access unit j.

2.5.7.2 Frequency of coding the system_clock_reference

The ISO/IEC 13818 Program Stream, M(i), shall be constructed so that the time interval between the bytes containing the last bit of system_clock_reference fields in successive packs shall be less than or equal to 0,7 seconds. Thus:

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

for all i and i' where M(i) and M(i') are the bytes containing the last bit of consecutive system_clock_reference fields.

2.5.7.3 Frequency of coding the program_clock_reference

The ISO/IEC 13818 Transport Stream, M(i), shall be constructed so that the time interval between the byte containing the last bit of program_clock_reference fields in successive occurrences of the program_clock_reference for each program shall be less than or equal to 0,1 seconds. Thus:

$$\left|t_m(i) - t_m(i')\right| \leq 0{,}1\text{sec}$$

for all i and i' where M(i) and M(i') are the bytes containing the last bit of consecutive program_clock_reference fields for each program.

2.5.7.4 Frequency of coding the elementary_stream_clock_reference

The ISO/IEC 13818 Program Stream and Transport Stream, M(i), shall be constructed so that if the elementary_stream_clock_reference field is coded in any PES packets containing a given elementary stream the time interval between the bytes containing the last bit of successive elementary_stream_clock_reference fields shall be less than or equal to 0,7 seconds. In PES Streams the ESCR encoding is required with the same interval. Thus:

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

for all i and i' where M(i) and M(i') are the byte containing the last bits of consecutive elementary_stream_clock_reference 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.

2.5.7.5 Frequency of presentation_time_stamp coding

The ISO/IEC 13818 Program Stream and Transport Stream, M(i), 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 discontinuity exists in elementary stream n between $P_n(k)$ and $P_n(k'')$.

2.5.7.6 Conditional coding of time stamps

For each elementary stream of an ISO/IEC 13818 Program Stream or Transport Stream, the presentation_time_stamp (PTS) shall be encoded in the PES packet in which the first access unit of that elementary stream commences. For the purposes of this subclause a video access unit commences in a PES packet if the first byte of the picture_start_code is present in the PES packet data (see Part 2 of this Recommendation | International Standard and Part 2 of ISO 11172). An audio access unit commences in a PES packet if the first byte of the synchronization word of the audio frame is present in the PES packet data (see Part 3 of this Recommendation | International Standard and Part 3 of ISO 11172).

A discontinuity exists at the start of presentation unit $P_n(k)$ in an elementary stream n if the presentation time $tp_n(k)$ is greater than the largest value permissible given the specified tolerance on the system_clock_frequency. If a discontinuity exists in any elementary audio or video stream in the ISO/IEC 13818 Program Stream or Transport Stream then a presentation_time_stamp shall be encoded referring to the first access unit after each discontinuity.

Presentation_time_stamps may be present in any PES packet header with the following exception. If no access unit commences in the PES packet data, the presentation_time_stamp shall not be present in the PES packet header. If a presentation_time_stamp is present in a PES packet header it shall refer to the presentation unit corresponding to the first access unit that commences in the PES packet data.

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

a) A presentation_time_stamp is present in the PES packet header
b) The decoding time differs from the presentation time.

2.5.7.7 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.

2.5.7.8 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 ISO/IEC 13818 Program Stream. The values encoded in all the system headers in the ISO/IEC 13818 Program Stream shall be identical.

2.5.7.9 Constrained system parameter Program Stream

An ISO/IEC 13818 Program Stream is a "constrained system parameters stream" (CSPS) if it conforms to the bounds specified in this subclause. ISO/IEC 13818 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 clause 2.5.3.5 on page 45. The CSPS is a subset of all possible ISO/IEC 13818 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 program_mux_rate field is less than or equal to 4 500 000 bits/second. For higher bit-rates the CSPS packet rate is bounded by a linear relation to the value encoded in the program_mux_rate field.

Specifically, for all packs p in the ISO/IEC 13818 Program Stream,

$$NP <= (tm(i') - tm(i)) \times 300 \times \max\left[1, \frac{R_{\max}}{4.5 \times 10^6}\right] \quad (2\text{-}27)$$

where

$$R_{\max} = 8 \times 50 \times rate\_bound\left(\frac{bits}{\sec}\right) \quad (2\text{-}28)$$

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 2-20 on page 43 and semantics in clause 2.5.3.2 on page 43.

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

$t_m(i')$ is the time, measured in seconds, encoded in the system_clock_reference for pack p+1, immediately following pack p, or in the case of the final pack in the ISO/IEC 13818 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

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

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

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

R_{vmax} is the peak video bit rate

2.5.7.10 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. Clause 2.4.2 on page 10 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 Part 3 of this Recommendation | International Standard or in Part 3 of ISO 11172. For all presentation units in all audio elementary streams in the ISO/IEC 13818 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.

$$SCASR = \frac{system\_clock\_frequency}{audio\_sample\_rate\_in\_the\_STD} \qquad (2\text{-}29)$$

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

Nominal audio sampling frequency (kHz)	16	32	22,05	44,1	24	48
Ratio 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 ISO/IEC 13818 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.

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

Nominal picture rate (Hz)	23,976	24	25	29,97	30	50	59,94	60
Ratio 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.

2.5.8 Compatibility with ISO/IEC 11172

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

Annex A
(normative)

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 ISO/IEC 13818 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 ISO/IEC 13818 DSM Control Command require access to an ISO/IEC 13818 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 ISO/IEC 13818 bitstreams from different DSM. This brings many difficulties to the interface design of an ISO/IEC 13818 or ISO 11172 application system. To overcome this difficulty, a set of common DSM control commands, which is independent of the specific DSM used, is required.

A.0.2 Future applications

Beyond the immediate applications supported by the current DSM control command, 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 at 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 ISO/IEC 13818 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 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 ISO/IEC 13818 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 ISO/IEC 13818 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 ISO/IEC 13818 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 ISO/IEC 13818 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 ISO/IEC 13818 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 ISO/IEC 13818 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 ISO/IEC 13818 bitstream and is also [optionally] capable of generating a ISO/IEC 13818 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 ISO/IEC 13818 bitstreams if an out of band channel is not available.

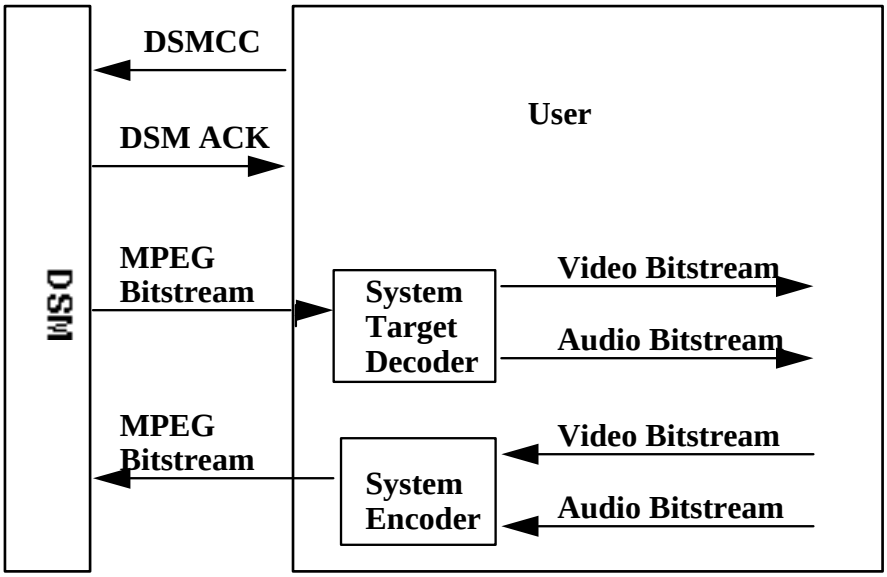


Figure A-1 -- Configuration of DSM CC application

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 ISO/IEC 13818 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 74. 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 ISO/IEC 13818 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.

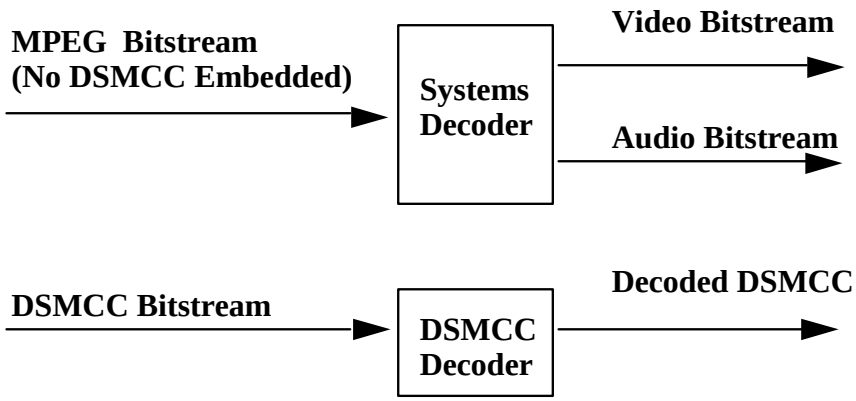


Figure A-2 -- DSM CC bitstream decoded as a standalone bitstream

For some applications, it is desirable to transmit the DSM CC in a ISO/IEC 13818 systems bitstream so that some features of the ISO/IEC 13818 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 section 2.4.4.4 of the system part of the IS. 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 3.

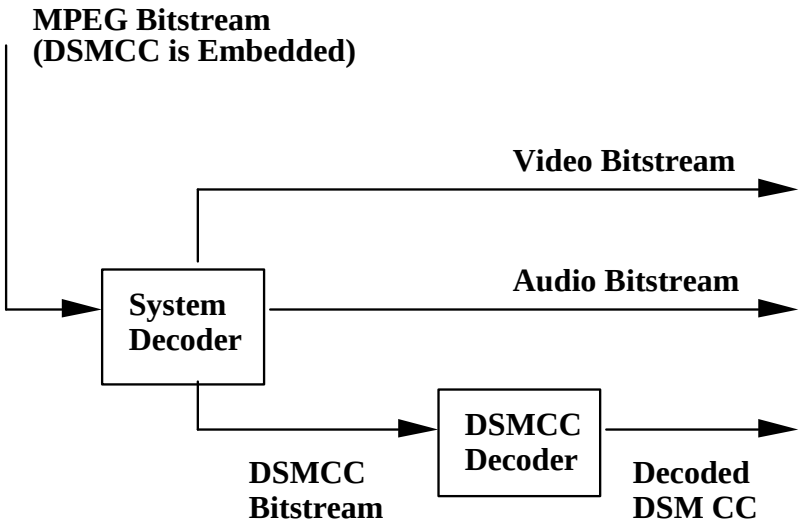


Figure A-3 -- DSM CC bitstream decoded as part of the system bitstream

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 0-24 on page 47 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 ISO/IEC 13818 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, ISO/IEC 13818-1 Program Stream or ISO/IEC 13818-1 Transport stream.

A.2.1.4 DSM CC server -- A system, either local or remote, used to store and/or retrieve an ISO/IEC 13818 bitstream.

A.2.1.5 Point of random access -- A point in an ISO/IEC 13818 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 to the last access unit provided from the DSM from the currently selected ISO/IEC 13818 bitstream. If no access unit has been provided from this ISO/IEC 13818 bitstream or 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 ISO/IEC 13818 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 DSM_start_code;
- The actual control command or acknowledgment shall immediately follow the DSM_start_code;
- 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 ISO/IEC 13818 systems bitstream. This may include manipulating the trick mode bits defined in the system part of the IS.

Table A-1 -- ISO/IEC 13818 DSM CC

Syntax	No. of bits	Mnemonic
DSM_CC() {		
DSM_CC_start_code	**32**	**bslbf**
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

DSM_start_code -- The DSM_start_code is a 32-bit string '0000 0000 0000 0000 0000 0001 1111 0010' (0000 01F2 in hexadecimal). It identifies the beginning of a DSM CC bitstream.

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 == ' ') {		
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 string that is reserved for future extension of the ISO/IEC 13818 DSM control command. 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 ISO/IEC 13818 bitstream is to be selected. It is the DSM server's responsibility to map the names of the ISO/IEC 13818 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 ISO/IEC 13818 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.

playback_ack -- This is a 1-bit code. When it is set to '1', it specifies that the ack() command is to acknowledge a playback 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 command 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 ISO/IEC 13818 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 ISO/IEC 13818 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
(informative)

ISO/IEC 13818 Systems Timing Model and Application Implications

B.0 Introduction

The ISO/IEC 13818 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 ISO/IEC 13818 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 ISO/IEC 13818 Systems timing model in detail, and gives some suggestions for implementing decoder systems to suit some typical applications.

A.0.1 Timing Model

ISO/IEC 13818 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:

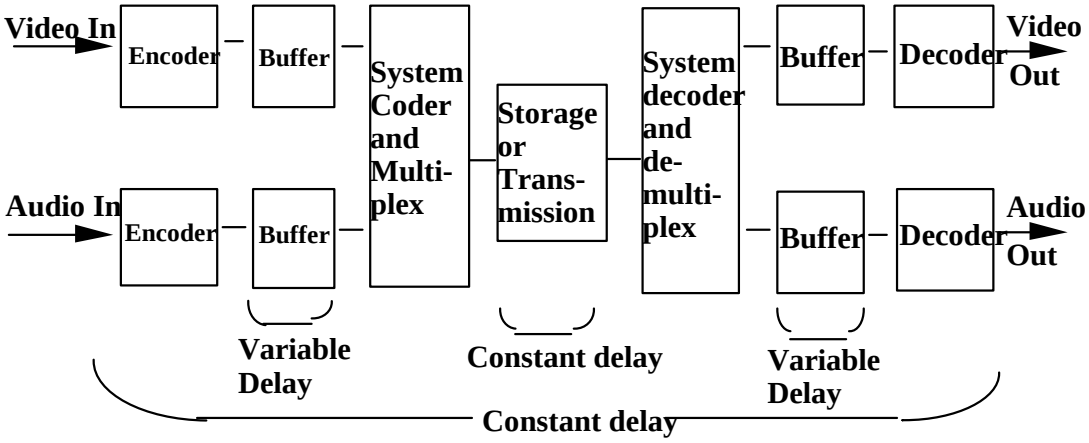


Figure A-1 -- Constant delay model

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[5], 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

[5]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.

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_lock_flag and system_video_lock_flag 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 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.

A.0.2 Audio and Video Presentation Synchronization

Within the coding of ISO/IEC 13818 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 ISO/IEC 13818 bit stream are called "Access Units", abbreviated AU. An audio access unit is abbreviated AAU, and a video access unit is abbreviated VAU. In part 3 of this Standard (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 that B pictures are present. 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 video streams where B pictures do not exist, 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.

A.0.3 System Time Clock recovery in the decoder

Within the ISO/IEC 13818 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 that 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 SCR 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 ISO/IEC 13818 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.

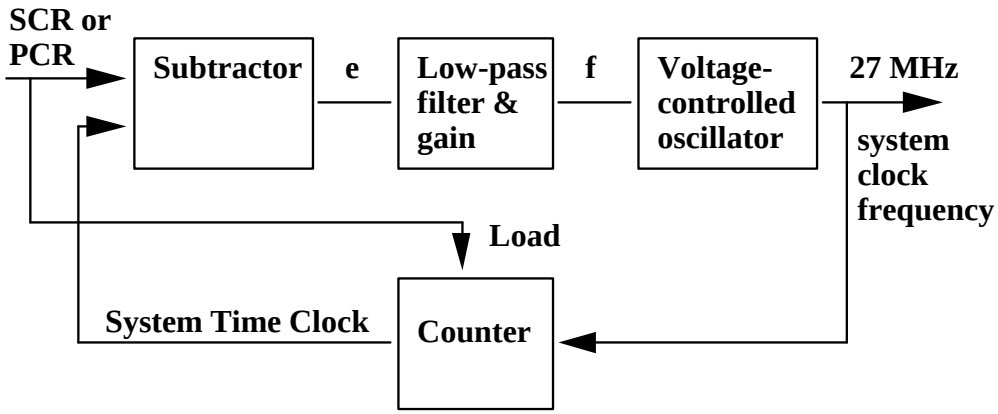


Figure A-2 -- STC recovery using PLL

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 designed to meet application requirements, subject to constraints of decoder buffer size and delay.

A.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 A-2 on page 85, 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 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.

This jitter in the delivery of SCRs and PCRs can be corrected, in some cases, by changing the value of the SCR or PCR to reflect the variation in delay that is experienced by the field. In order to ensure that SCR or PCR values are correct after re-multiplexing these values must be corrected to account for variations in delivery time caused by the re-multiplexing. The correction is accomplished by adding a correction term to the value of the SCR or PCR; and this correction term is equal to the difference between the actual delay experienced by that clock reference field and the average delay experienced by all data from the same program. In the case of Transport Streams such correction of PCRs is necessary when a Transport Stream is re-multiplexed, creating a new Transport Stream from one or more Transport Streams. If the PCRs or SCRs are not corrected, they are not strictly correct, and they may or may not be handled properly by a decoder, depending on the design of the decoder and its presentation timing requirements.

A.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 on 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.

A.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 A-2 on page 85, 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 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 233 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 A-2 on page 85 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 A-2 on page 85 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.

A.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.

A.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.

A.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 the 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 A-2 on page 85. 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 C
(informative)

Program Specific Information

C.0 Description of the Program Specific Information (PSI)

Part 1 of this standard provides a method for describing the contents of ISO/IEC 13818 Transport Streams for the purpose of demultiplexing and presentation of programs. The coding specification accommodates this function through the Program Specific Information or PSI. This annex discusses the use of ISO/IEC 13818 Program Specific Information (PSI).

The PSI may be thought of as belonging to four tables: 1) Program Association Table (PAT) 2) Program Map Table (PMT), 3) Network Information Table (NIT) and 4) Conditional Access Table (CAT). The contents of the PAT, PMT and CAT are specified by ISO/IEC 13818. The NIT is a private table, but the PID value of the Transport Stream packetTransport Stream packets which carry it is specified in the PAT.

While these structures may be thought of as simple tables, they must be partitioned before they are sent in ISO/IEC 13818 Transport Stream packetTransport 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 Packet payloads.

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). To successfully demultiplex a program, a decoder must be notified of both the program number and the Transport Stream ID. The Transport Stream mapping may be accomplished via the Network Information Table. For example, in a North American CATV system, each one of many 6 MHz channels will carry a single Transport Stream. The mapping required would thus be:

Subscriber-selected Service Number ⇒ (6 MHz channel frequency, program number)

In this case, the Network Information Table could contain the above mapping. The NIT would contain a correspondence between the 6 MHz CATV channel and a **transport_stream_ID**. Note that the Network Information Table may be stored in decoder non-volatile memory to reduce channel hopping time. In this case, it need be transmitted only often enough to support timely decoder staging (setup) operations. The contents of the NIT are private.

If program mappings are static or quasi-static, they may also be stored in the decoder. The trade-off between the amount of storage required and the desired impact on channel hopping may be made by the decoder manufacturer.

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. Or, if pure out-of-band signaling is desired, the PAT, NIT, PMT, and the CAT may be sent in a separate signaling channel. It should be noted that this signaling channel must carry this data in ISO/IEC 13818 Transport Stream format.

The PSI tables are mapped into Transport Stream packetTransport Stream packets via a structure called a section. Because each section has a **table_ID** field in its header, sections from PSI tables and private data may be mixed in Transport Stream packetTransport Stream packets of the same PID value or even in the same Transport Stream packetTransport Stream packet. This is only possible for the Program Map Table, however, since private sections may not be mapped into the PAT or CAT. It is required that all PAT sections be mapped into Transport Stream packetTransport Stream packets with PID=0 and all CA sections be mapped into packets with PID=1. PMT sections may be mapped into packets of user-selected PID value. Likewise, the PID for the NIT-bearing Transport Stream packetTransport Stream packets may be user-selected, but must be pointed to by "Program zero" in the PAT.

The contents of any CA parameter streams are entirely private, but EMMs and ECMs must also be sent in ISO/IEC 13818 Transport Stream packetTransport Stream packets to be normative.

A.1 The Relationships Of PSI Structures

Figure A-1 on page 93 shows an example of the relationship between the four PSI structures and the Transport Stream. Other different examples are possible, but the figure shows the primary connections.

The PSI tables map into the Transport Stream via a generic mapping structure and by using PID pointers to locate data in specific Transport Stream packetTransport Stream packets.

In the following sections, each PSI table is described.

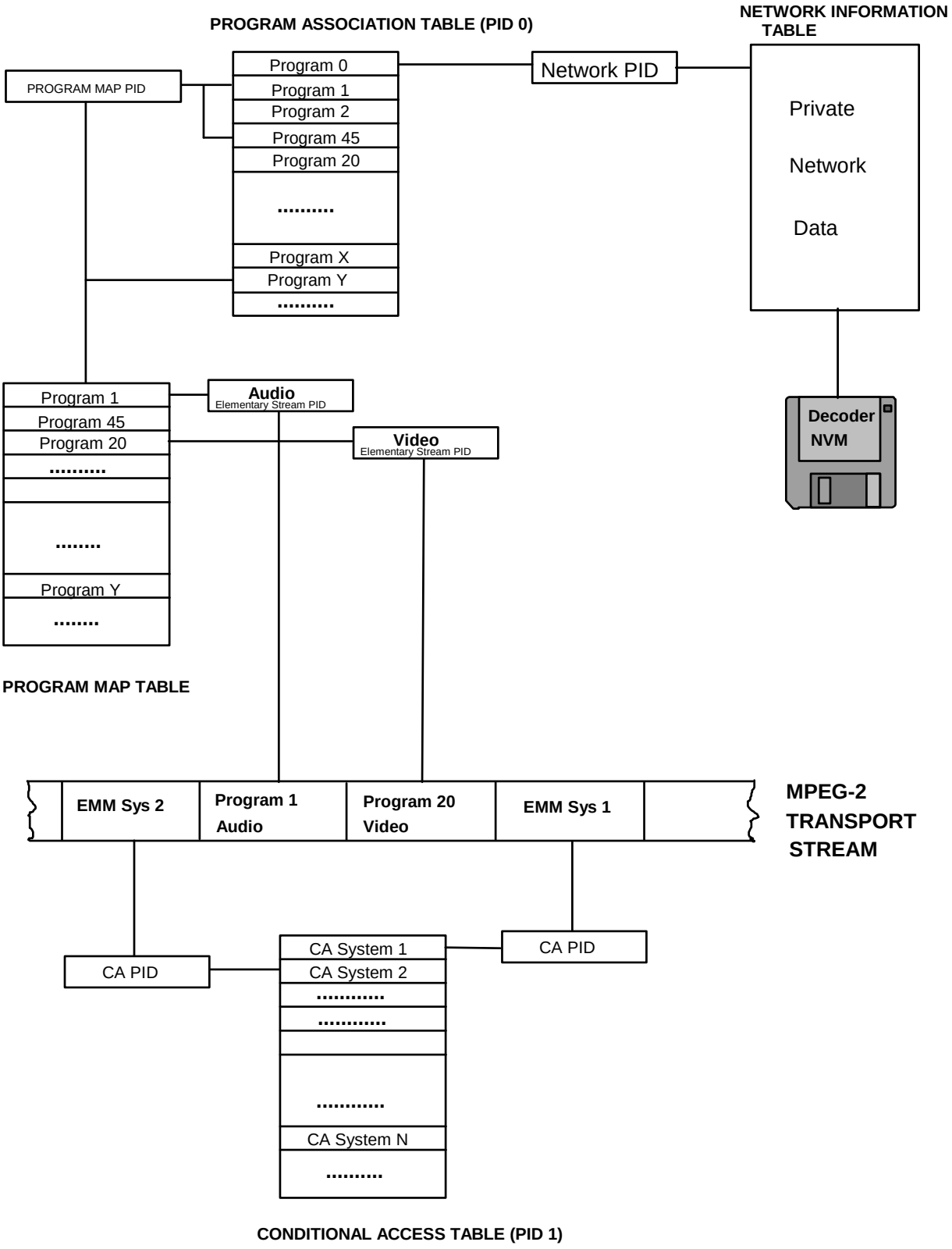


Figure A-1 -- Program and network mapping relationships

A.1.1 Program Association Table

The Program Association Table gives the correspondence between a program number and the PID of the Transport Stream packetTransport Stream packets that carry the definition of that program. The PAT may

be partitioned into up to 255 sections before it is mapped into Transport Stream packetTransport 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 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.

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

The Program Association Table is always transmitted without encryption.

A.1.2 Program Map Table

The Program Map Table provides the mapping between a program number and the elementary streams that comprise it. This table is presented in packets having one or more privately-selected PID values. These packets may contain other private structures as defined by the **table_ID** field.

ISO/IEC 13818-1 requires a minimum of program identification: program number, PCR PID, elementary stream types and elementary stream PIDs. Additional information for either programs or elementary streams may be conveyed by use of the **descriptor()** construct.

Private data may also be sent in PMT Transport Stream packetTransport Stream packets. 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 single section or whether they applicable to many sections as members of a larger private table.

There are several normative descriptors defined by ISO/IEC 13818. 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 section of these private descriptors may, of course, be privately defined.

One descriptor (the **CA_descriptor()**), may be used to indicate the location (PID value of Transport Stream packetTransport Stream packets) of ECM data associated with elementary streams.

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

A.1.3 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 packetTransport Stream packets containing EMM and CA parameters (if present) will, in general, be encrypted (scrambled).

A.1.4 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_ID**s, channel frequencies, satellite transponder numbers, etc.

A.2 Bandwidth Utilization and Signal Acquisition Time

Any implementation of an ISO/IEC 13818 bitstream must make reasonable bandwidth demands for PSI information and should promote fast signal acquisition. This section analyses this issue and gives some broadcast application examples.

The packet-based nature of the ISO/IEC 13818 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, which include at least: FDM tuning slew time, demultiplexing time, I-frame occurrence rate and scrambling key retrieval and processing.

This section examines both the bandwidth and signal acquisition time impacts of the PSI syntax clause 2.4.8. It is assumed that the Conditional Access Table does not need to be received dynamically at every program change. This is also true of the private EMM streams. This is because these streams do not contain the quickly-varying ECM components used for elementary stream scrambling (encryption).

Also, in the discussion below, it is assumed that the ECM parameters needed for service descrambling are sent in transport adaptation fields of each service stream. Having a separately labeled PID stream to carry this data is possible and would increase the bandwidth usage only slightly, but might increase signal acquisition time because of the extra demultiplexing operation. The magnitude of this additional acquisition delay is dependent on how the ECM data is distributed in the separately labeled PID stream(s). Using a different stream for each single-service ECM component could make the delay negligible because the ECM-bearing packet could be sent exactly when needed in conjunction with the service-bearing Transport Stream packetTransport Stream packets. If, however ECM components for more than one service are packed into a single Transport Stream packetTransport Stream packet, it will either increase acquisition delay by not being available in a timely fashion, or must be transmitted multiple times to match the Transport Stream scrambling boundaries of multiple programs. Sending ECM parameters in elementary-stream-bearing packet adaptation fields avoids both the bandwidth and acquisition problems.

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 included. All programs are composed of two elementary streams. Program associations are 4 bytes long, while the minimal program map is 25 bytes long. There is additional overhead associated with , version numbers, section lengths, the pointer field etc. This will be on the order of 1-3% of the total PSI bandwidth usage in sections of moderate to maximum length (a few hundred bytes to 1024 bytes) and will thus be ignored here.

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

Table A-1 -- 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: The numbers in the table do not change until the last column, 46 program associations fit into one Transport Stream packetTransport Stream packet.

Table A-2 -- Program table bandwidth usage (bps)

		Number of Programs Per Transport Stream				
		1	**5**	**10**	**32**	**128**
Frequency of P 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 25 sec^{-1} for the two PSI Tables, would yield a worst case contribution to the signal acquisition time of approximately **80 ms.** This would only occur when the required PAT data was "just missed" and then, once the PAT was acquired and decoded, the required PMT data was also "just missed". This doubling of the worst case acquisition time is one disadvantage of the extra level of indirection introduced by the PAT structure. This effect could be reduced by coordinated transmission of related PAT and PMT packets. Presumably, the advantage that this approach offers for "drop/add" re-multiplexing operations is compensatory.

With the 25 sec^{-1} PSI frequency, the following examples may be constructed (all examples leave ample allowance for various datalink, FEC and routing overhead):

6 MHz CATV channel:

five 5.2-Mbps programs: 26.5 Mbps (includes transport overhead)
total PSI bandwidth: 75.2 Kbps
CA bandwidth: 500 Kbps

total ISO/IEC 13818 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
CA bandwidth: 500 Kbps

total ISO/IEC 13818 transport bandwidth: 128.2 Mbps

PSI Overhead: **0.18%**

C-band satellite transponder:

128 256-Kbps audio programs:	33.5 Mbps (includes transport overhead)
total PSI bandwidth:	826.4 Kbps
CA bandwidth:	500 Kbps

total ISO/IEC 13818 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 quite comfortable in all cases. Higher transmission rates (than 25 sec^{-1}) for the PSI data may be used to decrease the impact on channel hopping time with only modest bandwidth increases.

Annex D
(informative)

Service Information Descriptors

D.0 Introduction

Descriptors are necessary for Service Information. Among these descriptors, are the following:

Service Name
Service reference
Program name
Program Type
Program timing
Teletext elementary streams
transparent elementary streams
etc.

A Program description may need about hundred different fields. That is to say hundred descriptors, would be needed. Open solutions are needed in order to save the number of descriptor tags required.

Solution using two kinds of descriptors -

1. "simple" descriptors identical to the descriptors already defined in the present PSI syntax.

2. "composite" descriptors built on sub_descriptors, in order to build a hierachy of descriptors on two levels (*and more if needed*).

The distinction between "simple" descriptors and "composite" descriptors may be done on the value of the tag, some tag values being assigned to "composite" descriptors tags.

The "composite" descriptors are built on sub_descriptors. None of the sub_descriptors will be mandatory, as are already the descriptors.

Among these sub_descriptor the same thing may be said, if it seems to be interesting: some of them may be "simple" sub_descriptors and some others may be "composite" ones.

An example for a program descriptor is shown below, on a "composite" descriptor basis .This example uses one MPEG2 descriptor_tag to describe a program. That "composite" structure allows to have an open program description.

A.1 Program descriptor definition

The Program descriptor structure carries some of the necessary information needed to select a program. All information is optional and is coded in variable length structures called the program_sub_descriptors

Table A-1 -- Program descriptor example

Syntax	No. of bits	Identifier
Program_descriptor(){		
program_descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
for(i=0;i<N;i++){		
program_sub_descriptor()		
}		
}		

The program_descriptor_tag value is found in the list of descriptor_tag values assigned to user private descriptors in the stream descriptor table 0-31 on page 54

Table A-2 -- Program sub-descriptor

Syntax	No. of bits	Identifier
program_sub_descriptor() {		
sub_descriptor_tag	**8**	**uimsbf**
sub_descriptor_length	**8**	**uimsbf**
for (i=0; i<N; i++) {		
sub_descriptor_byte	**8**	**uimsbf**
}		
}		

The sub_descriptor_tag values and the sub_descriptor_bytes are user private defined.

Annex E
(informative)

Data Transmission Applications

E.0 CONSIDERATIONS

- ISO/IEC 13818 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 Stream packet carrying data PES packets may be lost.

A.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 A-1 -- PES packet header example

PES packet headers 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_length	0x00

Annex F
(informative)

Graphics of Syntax for ISO/IEC 13818

F.0 Introduction

This part of the standard is an informative annex presenting graphically the ISO/IEC 13818 Transport Stream and Program Stream syntax . This part in no way replaces the preceeding 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 hatched lines. Fields length are indicated in bits.

A.0.5 Transport Stream syntax

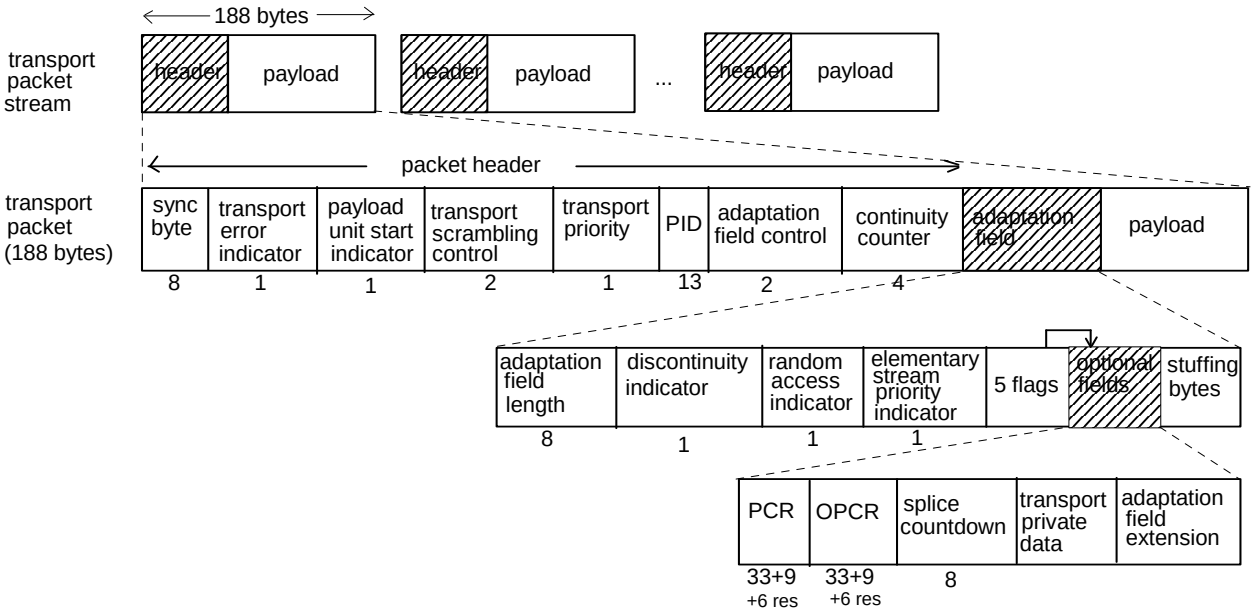


Figure A-1 -- Transport Stream syntax diagram

A.0.6 PES packet

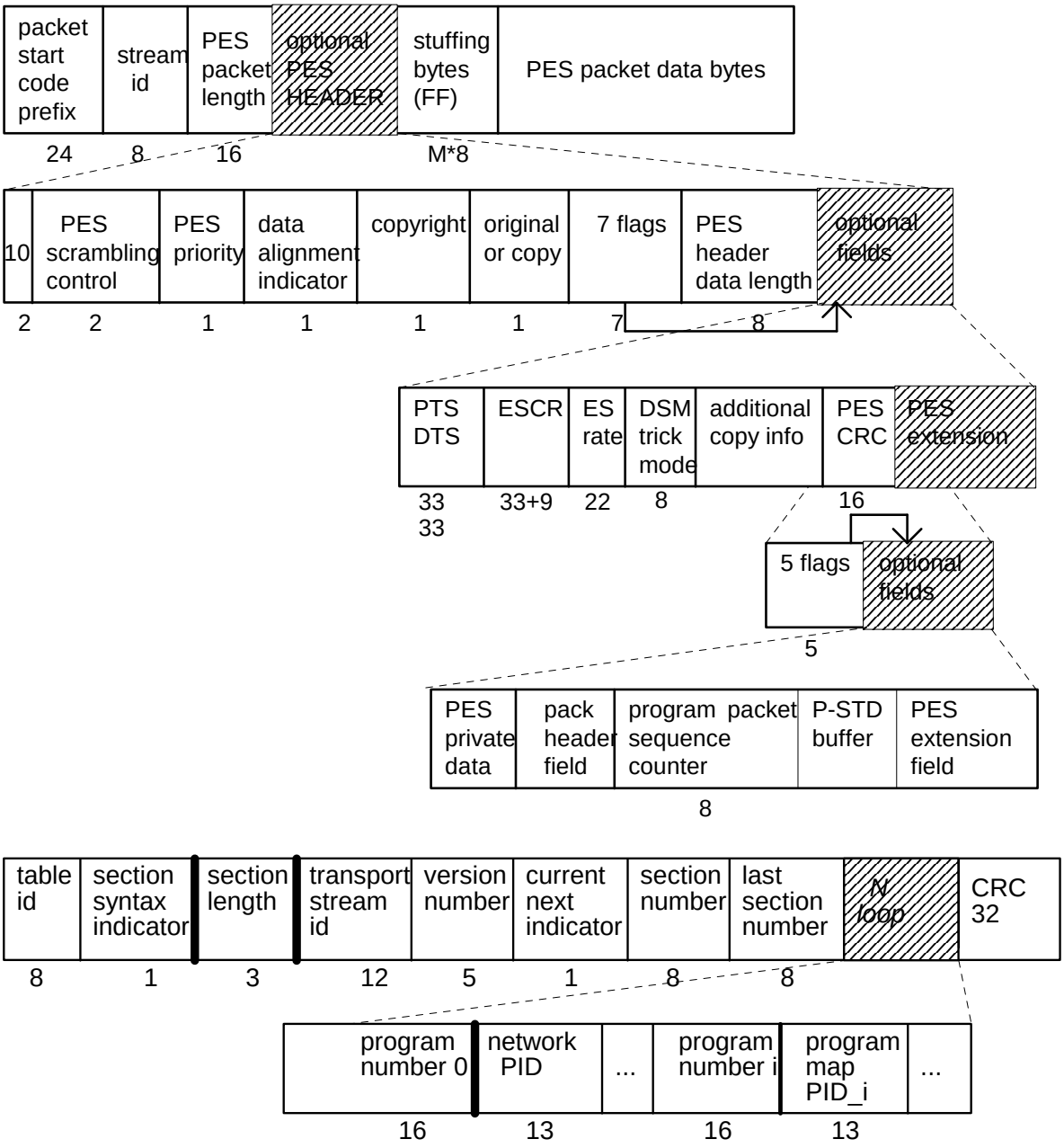


Figure A-2 -- PES packet syntax diagram

A.0.7 CA section

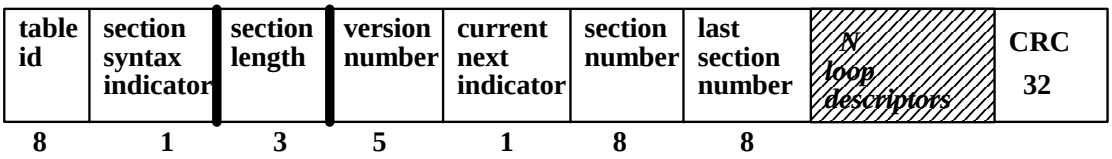


Figure A-3 -- Conditional access section diagram

A.0.8 TS program map section

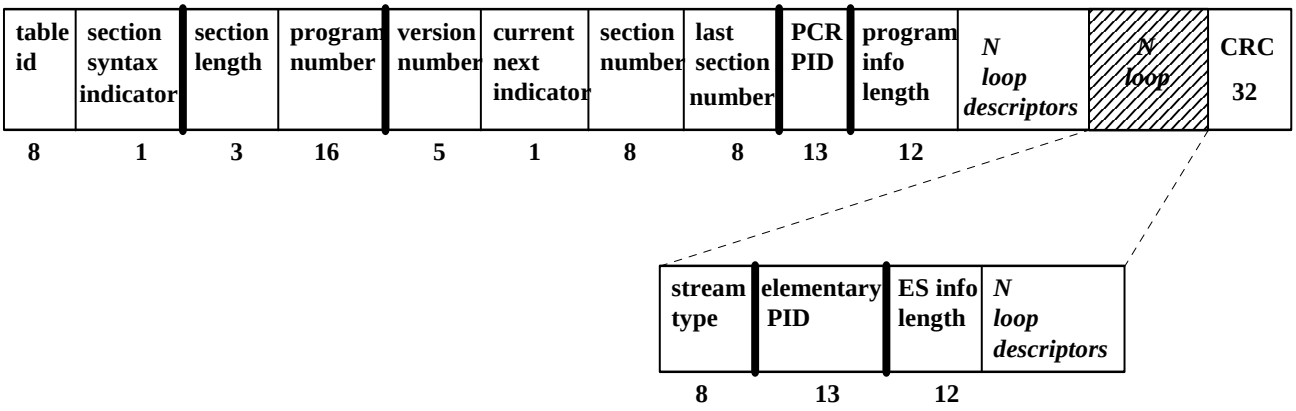
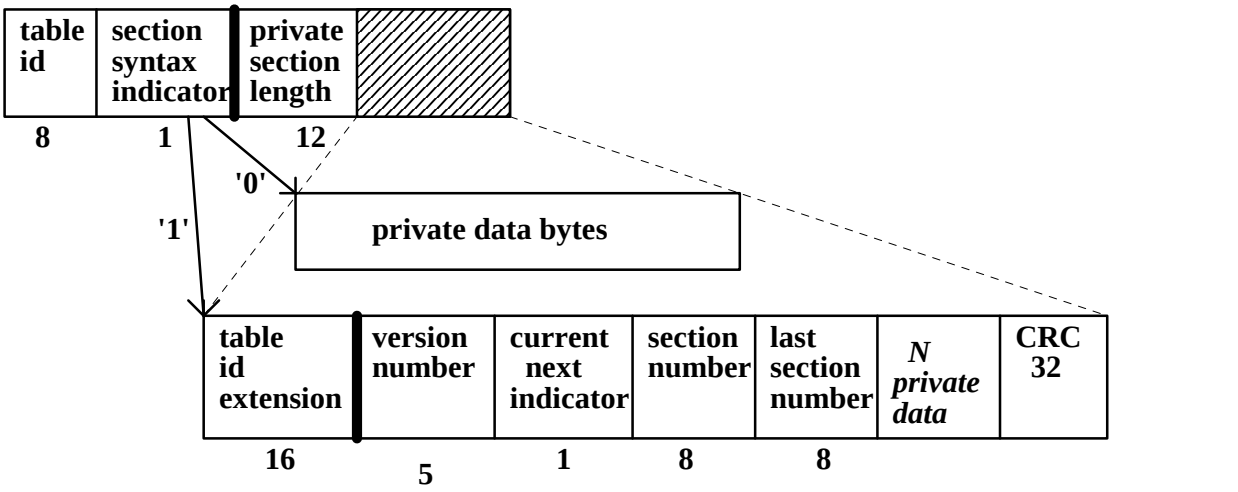


Figure A-4 -- TS program map section diagram

A.0.9 Private section



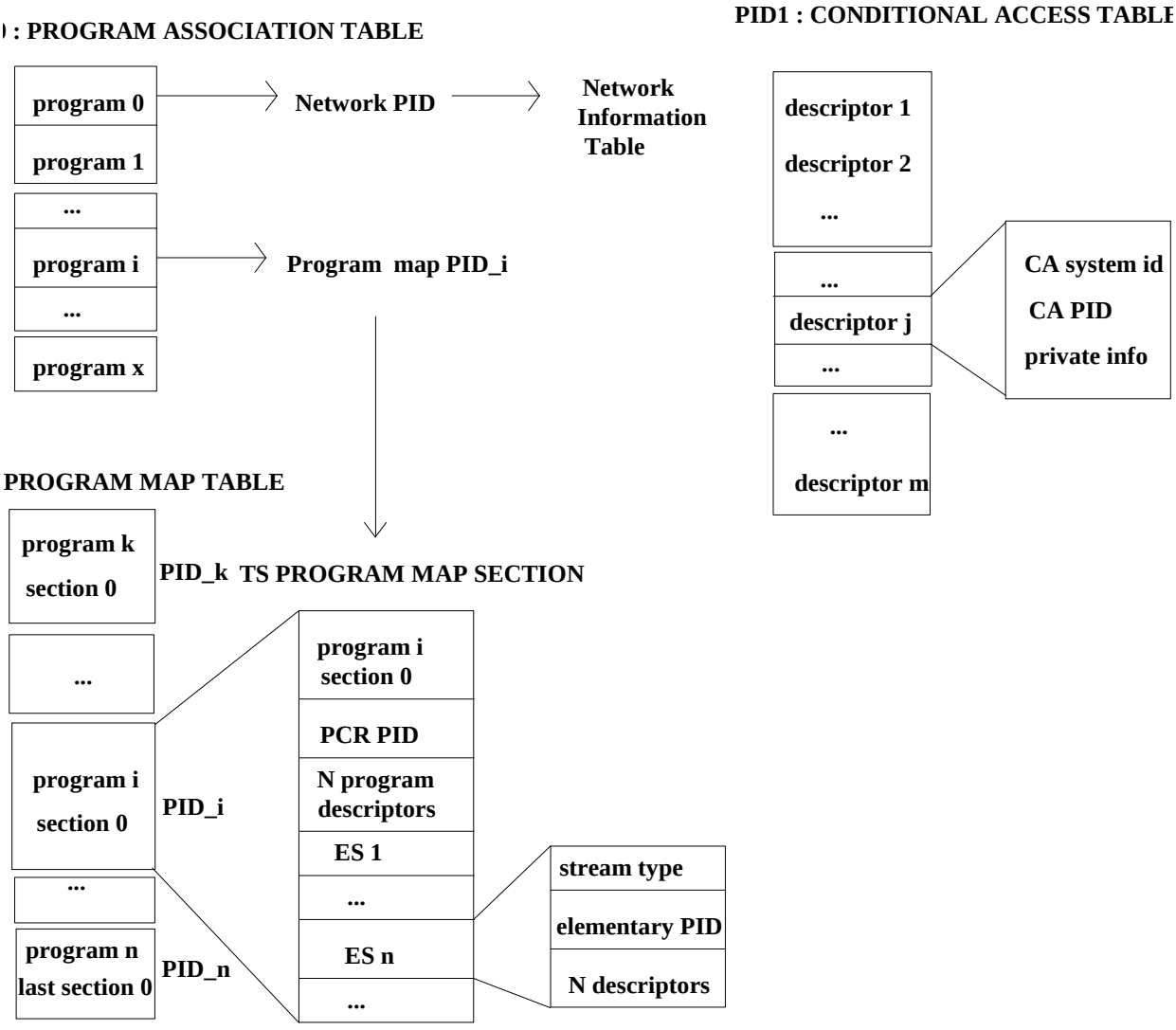


Figure A-5 -- Private section diagram

A.0.10 Program Stream

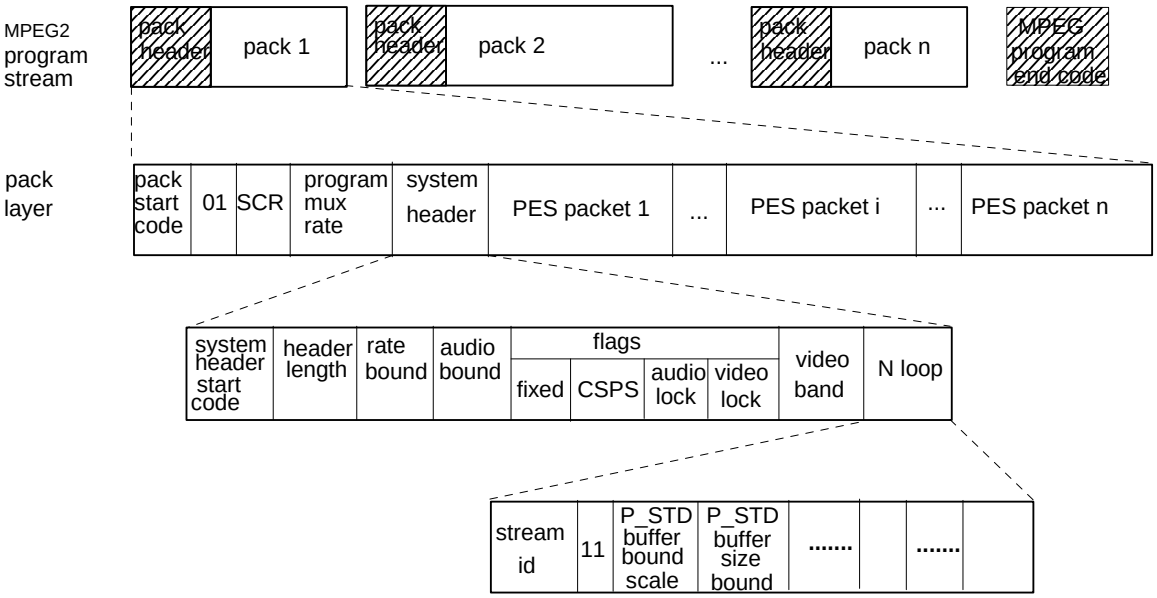


Figure A-6 -- Program Stream diagram

A.0.11 Program Stream map

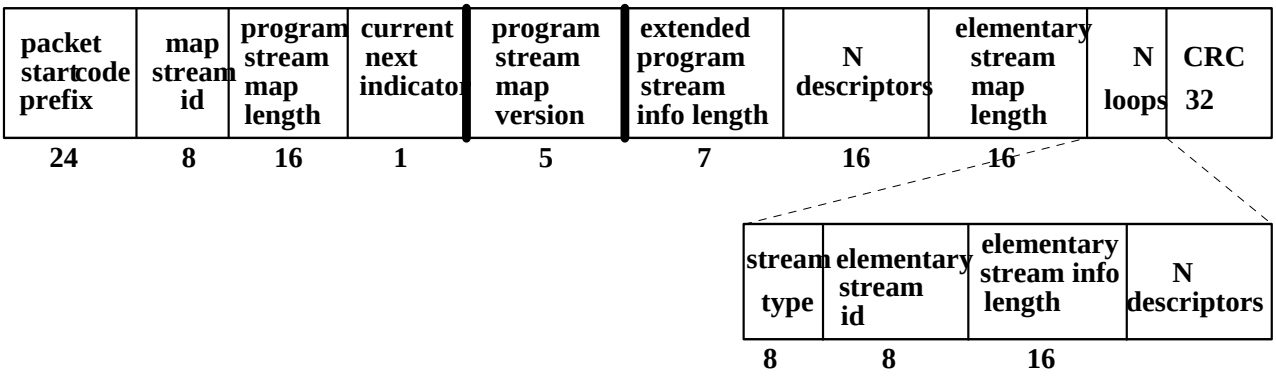


Figure A-7 -- Program Stream map diagram

Annex G
(informative)

General Information

G.0 General Information

A.0.1 Sync Byte Emulation

In the choice of PID values it is recommend 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 Stream packetTransport Stream packets.

A.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 buffer when the time arrives to remove those bits for instantaneous decoding and display. When this case arises, 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 the STD buffer, these bits are removed and picture_next is displayed. If all the bits of picture_next are not in the STD buffer, 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 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.

A.0.3 Selection of PID Values

Applications are encouraged to use low numbered PID values and group values together as much as possible.

Annex H
(informative)

List of companies having provided patent statements for ISO/IEC 13818

H.0 Companies having provided patent statements for ISO/IEC 13818

The user's attention is called to the possibility that - for some of the processes specified in this part of ISO/IEC 13818 - conformance with this Recommendation | International Standard may require use of an invention covered by patent rights.

By publication of this part of ISO/IEC 13818, 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 A-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			
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			