AVC-502

MPEG 2 Systems

Working Draft

11 JUNE 1993

INTRODUCTION - PART 1: SYSTEMS

The systems specification addresses the problem of combining one or more data streams from the video and audio parts of this proposed standard as well as other data into a single or multiple streams which are suitable for storage or transmission. System coding following the syntactical and semantic rules imposed by this specification provides the necessary and sufficient information to enable enable synchronized playback without overflow or underflow of decoder buffers under a wide range of stream retrieval or receipt conditions and with various forms of system coding.

System coding is specified in two forms: the Program Stream and the Transport Stream. Each is optimized for a different set of applications. Both the program and transport streams defined in this specification provide system level coding which is necessary and sufficient to sychronize the decoding and presentation of the video and audio information, while ensuring that coded data buffers in the decoders do not overflow or underflow. Such 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 program stream is analagous and similar to ISO 11172-1 (MPEG Systems) and combines one or more elementary streams 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 which all have a common time base and which, like the single program stream, can be decoded with synchronization between the various elementary streams. The program stream is designed for use in relatively error-free environments and for applications which involve processing, possibly in software, of the streams. Program stream packets may be of variable and relatively great length.

The transport stream combines one or more elementary streams with one or more distinct time bases into a single stream. 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 of fixed and relatively short length.

As the program and transport streams are designed for different applications, their definitions do not follow strictly 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 program stream is straightforward, but not all of the fields of the program stream are contained literally 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 wide 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 multiplexers or demultiplexers. However, bitstream properties do impose functional and performance requirements on encoders, decoders, multiplexers and demultiplexers. 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, multiplexers and demultiplexers.

I.1 Timing Model

MPEG 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 the means of creating them.

I.2 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.

I.3 Program Stream

A prototypical audio/video program stream decoder system is depicted in Figure I-I.1 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 MPEG 2 Program Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

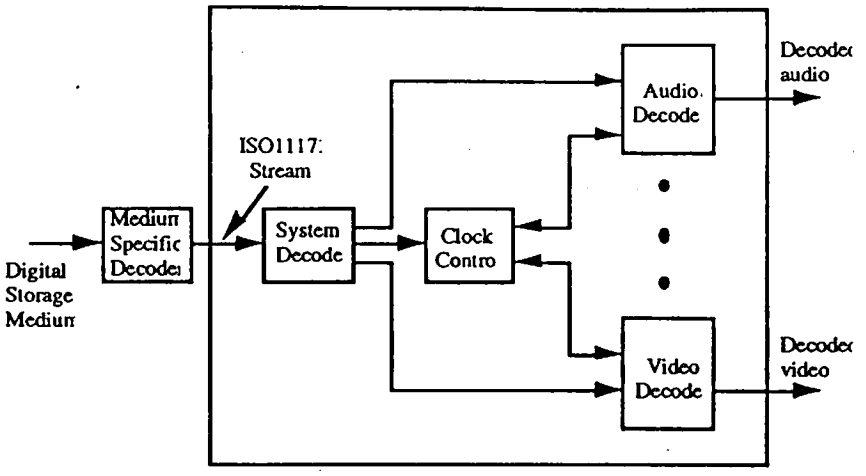

Figure 1-I.1 -- Prototypical MPEG 2 Program Stream Decoder

The prototypical MPEG 2 Program Stream decoder shown in Figure 1-I.1 is composed of System, Video, and Audio decoders conforming to Parts 1, 2, and 3, respectively, of this 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 International Standard, nor is the medium-specific decoding part of the prototypical MPEG 2 Program Stream decoder.

The prototypical decoder accepts as input an MPEG 2 Program Stream and relies on a System Decoder to extract timing information from the stream. The System 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 System 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.

MPEG 2 Program streams are constructed in two layers: a system layer and a compression layer. The input stream to the System 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 System Decoder either apply to the entire MPEG 2 Program stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The MPEG 2 Program system layer is divided into two sub-layers, one for multiplex-wide operations (the pack layer), and one for stream-specific operations (the packet layer).

I.4 Transport Stream

The transport stream is a stream definition which is tailored for communicating or storing one or more programs of MPEG coded data and other data in environments which significant errors may occur. Such errors may be manifested as bit value errors or loss of packets.

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.
2. Extract the transport packets from one program within the transport stream and produce as output a different transport stream with only that one program.
3. Extract the transport packets of one or more programs from one or more transport streams and produce as output a different transport stream.
4. Extract the contents of one program from the transport stream and produce as output a program stream containing that one program.

Figures 1-I.2, 1-I.3, and 1-I.4 illustrate prototypical demultiplexing and decoding systems which takes as input an MPEG 2 Transport Stream. Figure 1-I.2 illustrates the first case, where a transport stream is directly demultiplexed and decoded. As in the case of program streams, 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 MPEG 2 Transport Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

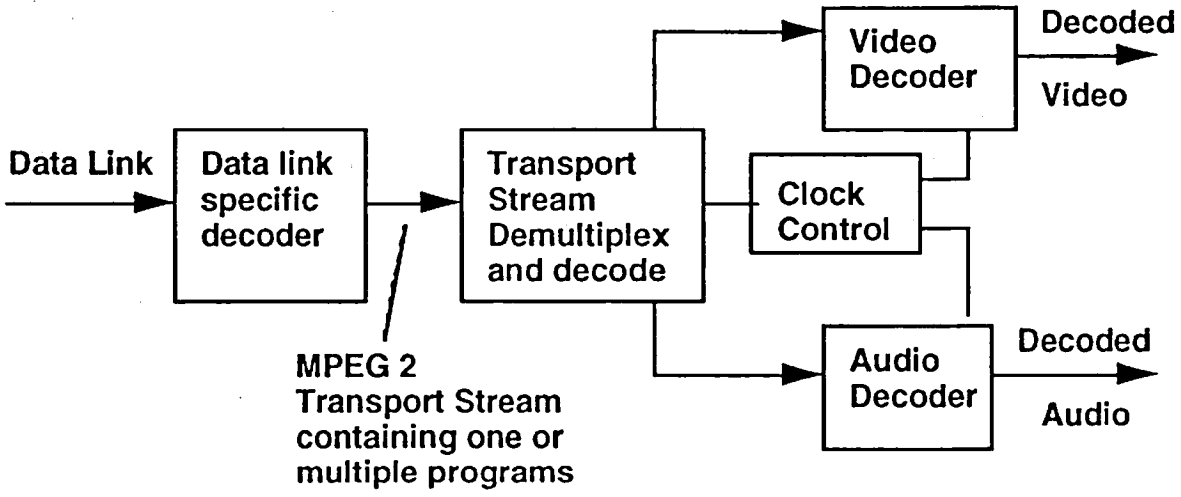

Figure 1-I.2

Figure 1-I.3 illustrates the second case, where a transport stream containing multiple programs is converted into a transport stream containing a single program:

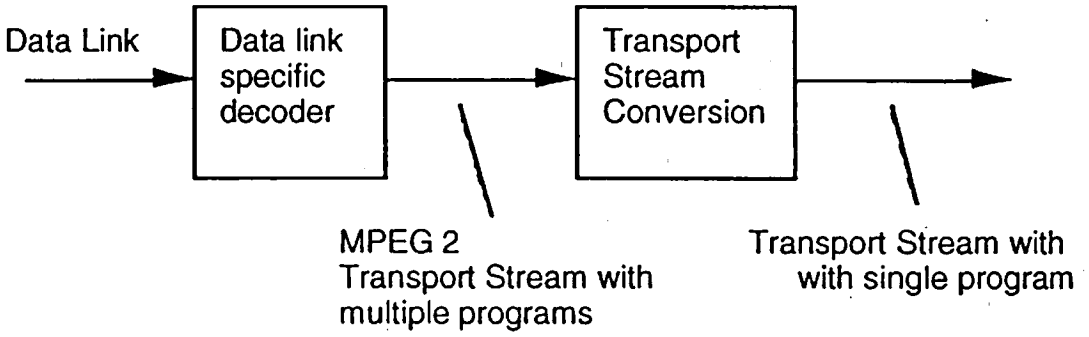

Figure 1-I.3

Figure 1-I.4 illustrates the fourth case, where a transport stream containing one or more programs is converted into a program stream:

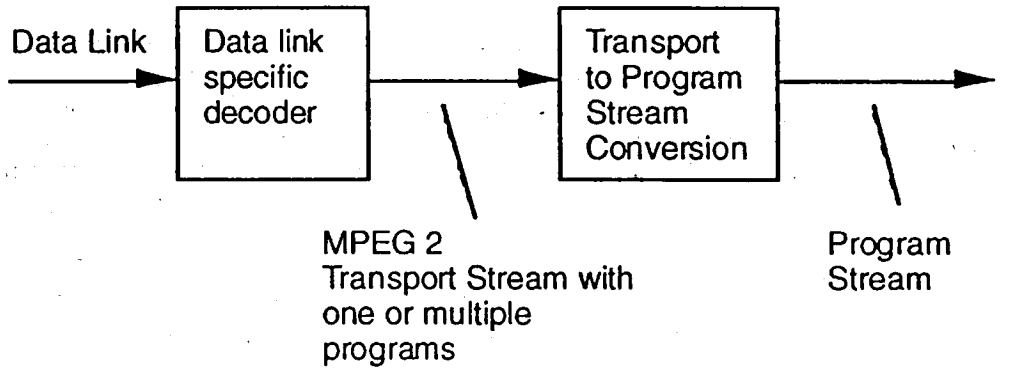

Figure 1-I.4

Figures 1-I.3 and 1-I.4 indicate that it is possible and reasonable to convert between different types and configurations of MPEG 2 streams. The fact that this is so results from the design of the Transport and Program streams, as embodied in the Normative Requirements of this part of the standard. 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 demultiplexers or decoders include all of these functions.

The MPEG 2 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. Such implementations are not required by this standard.

I.5 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 system 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 contain information which specifies the times at which each byte is intended to enter a system 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 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 program stream and transport stream each contain information which identifies the pertinent characteristics of and relationships between the elementary streams which constitute each program. Such information includes the language spoken in audio channels and the relationship between video streams when multi-layer video coding is implemented.

I.6 Individual Stream Operations (Packet Layer)

The principal stream-specific operations are 1) demultiplexing, and 2) synchronizing playback of multiple elementary streams. These topics are discussed next.

I.6.1 Demultiplexing

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 MPEG 2 audio and video streams. The streams are temporally subdivided into packets, and the packets are serialized. A 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 in Clause (2.4.3.3 in ISO 11172-1) and in Clauses (2.4.5 and 2.4.6 in ISO 11172-1). In the transport stream only fixed length packets are allowed.

Note: the packet length is not yet determined; it may be allowed to vary between different instances of transport stream, and the length may or may not be indicated within the transport stream itself. There may be implications to the ability to convert the lengths of transport packets where transport packets contain scrambled data, as well as others.

On decoding, demultiplexing is required to reconstitute elementary streams from the multiplexed Program stream. stream_id codes in program stream packet headers, and Packet ID codes combined with the transport stream ID table in the transport stream, make this possible.

I.6.2 Synchronization

Synchronization among multiple elementary streams is effected with presentation time stamps in the Program and Transport bitstreams. The time stamps are in units of 90kHz. Playback of N elementary streams is synchronized by adjusting the playback of all streams to a master time base rather than by adjusting the playback 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.

In transport streams which contain multiple programs each program has its own time base, and 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 packet layer of both the transport 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 System Clock Reference (SCR) in the Program stream and by the equivalent Program Clock Reference (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.

1.6.3 Relation to Compression Layer

The packet layer is independent of the compression layer in some senses, but not in all. It is independent in the sense that packets need not start at compression layer start codes, as defined in parts 2 and 3. For example, a video packet may start at any byte in the video stream. However, time stamps encoded in packet headers apply to presentation times of compression layer constructs (namely, presentation units).

I.7 System Reference Decoder

Part 1 of MPEG 2 employs a "System Target Decoder," (STD) to provide a formalism for timing and buffering relationships. Because the STD is parameterized in terms of MPEG 2 fields (for example, buffer sizes) each MPEG 2 stream leads to its own parameterization of the STD. It is up to encoders to ensure that bitstreams they produce will play in normal speed, forward play 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.

1 GENERAL NORMATIVE ELEMENTS

1.1 Scope

This part of MPEG 2 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 International Standard. The system layer supports five basic functions: 1) the synchronization of multiple compressed streams on playback, 2) the interleaving of multiple compressed streams into a single stream, 3) the initialization of buffering for playback start up, 4) continuous buffer management, and 5) time identification.

An MPEG 2 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 2.4 in MPEG 1) of this part.

1.2 References

The following standards contain provisions which, through reference in this text, constitute provisions of this 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 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 International Standards.

Recommendations and reports of the CCIR, 1990
XVIIth Plenary Assembly, Dusseldorf, 1990
Volume XI - Part 1
Broadcasting Service (Television)
Rec 601-2 "Encoding parameters of digital television for studios"

CCIR Volume X and XI Part 3
Recommendation 648: Recording of audio signals.

CCIR Volume X and XI Part 3
Report 955-2: Sound broadcasting by satellite for portable and mobile receivers, including Annex IV Summary description of advanced digital system II.

IEEE Draft Standard "Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform", P1180/D2, July 18, 1990.

IEC Publication 908:198, "CD Digital Audio System"

2 TECHNICAL NORMATIVE ELEMENTS

2.1 Definitions

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

access unit [system]: 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.

bitrate: The rate at which the compressed bitstream is delivered from the storage medium to the input of a decoder.

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

channel: A digital medium that stores or transports an ISO 11172 stream.

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

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

constant bitrate: Operation where the bitrate is constant from start to finish of the compressed bitstream.

constrained system parameter stream (CSPS) [system]: An ISO 11172 multiplexed stream for which the constraints defined in Part 1 Clause 2.4.6 apply.

CRC: Cyclic redundancy code.

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

decoded stream: The decoded reconstruction of a compressed bitstream.

decoder: An embodiment of a decoding process.

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

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

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

editing: The process by which one or more compressed bitstreams are manipulated to produce a new compressed bitstream. Conforming edited bitstreams must meet the requirements defined in this International Standard.

elementary stream [system]: A generic term for one of the coded video, coded audio or other coded bitstreams.

encoder: An embodiment of an encoding process.

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

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

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

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

MPEG 2 (multiplexed) stream [system]: A bitstream composed of zero or more elementary streams combined in the manner defined in Part 1 of this Working Draft.

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 Working Draft.

pack [system]: A pack consists of a pack header followed by one or more packets. It is a layer in the system coding syntax described in this Working Draft.

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

packet header [system]: The data structure used to convey information about the elementary stream data contained in the packet data.

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 this Working Draft.

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.

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

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

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

reserved: The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO defined extensions.

side information: Information in the bitstream necessary for controlling the decoder.

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

start codes [system and video]: 32-bit codes embedded in that coded bitstream that are unique. They are used for several purposes including identifying some of the layers in the coding syntax.

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.

system header [system]: The system header is a data structure defined in Part 1 of this International Standard that carries information summarising the system characteristics of the ISO 11172 multiplexed stream.

system target decoder; STD [system]: A hypothetical reference model of a decoding process used to describe the semantics of an ISO 11172 multiplexed bitstream.

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

variable bitrate: Operation where the bitrate varies with time during the decoding of a compressed bitstream.

2.2 Symbols and Abbreviations

The mathematical operators used to describe this 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 zero.

2.2.1 Arithmetic Operators

+	Addition.
-	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
- -	Decrement.
*	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. 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 zero 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( )	$Sign(x) = \begin{cases} 1 & x > 0 \\ 0 & x == 0 \\ -1 & x < 0 \end{cases}$
NINT ( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from zero.
sin	Sine.
cos	Cosine.
exp	Exponential.

$\sqrt{}$ 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.

$\geq$ Greater than or equal to.

< Less than.

$\leq$ Less than or equal to.

= Equal to.

$\neq$ 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 zero 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 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 subband samples in audio Layer III.

main_data — The main_data portion of the bitstream contains the scalefactors, Huffman encoded data, and ancillary information.

main_data_beg — This gives the location in the bitstream 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 one 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 scalefactors.

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.

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 is described in Clauses (?) (Clause 2.4.3 in MPEG 1) 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 (Clause 2.4.4 in MPEG 1). 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
   data_element in the data stream. This repeats until the condition is not true.
      . . .
}

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

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

} incremented to one 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 (Clause 2.4.3) implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bitstream. Actual decoders must include a means to look for start codes 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.

Syntax	No. of bits	Identifier
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, zero stuffing bits are present. After that any number of zero bytes may be present before the start-code. Therefore start-codes are always byte aligned and may be preceded by any number of zero stuffing bits.

2.4 Requirements

2.4.1 Coding Structure and Parameters

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

Program stream and transport stream

An MPEG 2 program stream consists of one or more elementary streams from one program multiplexed together. An MPEG 2 transport stream consists of one or more elementary streams from one or more programs 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 (Clause 2.4.2.4 in Part 2 of this MPEG 1) starting with the group_start_code. The Access Unit containing the first coded picture after a sequence header, as defined in (Clause 2.4.2.3 in Part 2 of MPEG 1), also includes that sequence header. The sequence_end_code is included in the Access Unit containing the last coded picture of a sequence. (See Clause 2.4.2.2 in Part 2 of MPEG 1 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 Clauses 2.4.3.1, 2.4.2.1, and 2.4.2.2 in Part 3 of MPEG 1 for the definition of an audio frame).

Data from elementary streams is stored in packets. A packet consists of a packet header followed by packet data.

The program stream packet header begins with a 32-bit start-code that also identifies the stream to which the packet data belongs. The 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 packet data contains a variable number of contiguous bytes from one elementary stream.

The transport stream packet header begins with an 8-bit start code which is used to identify the start of packets and a 16 bit Packet ID code which identifies the stream to which the packet data belongs. The 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 packet data contains a fixed number of contiguous bytes from one elementary stream.

Program stream packets are organized in packs. A pack commences with a pack header and is followed by zero or more 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.

The transport stream contains global system information in specially designated packets.

2.4.2 System Target Decoder

NOTE: This section needs to be expanded and edited to include the Program Clock Reference (PCR) and additional STD buffers that are relevant to the Transport Stream. As it stands the following section is taken directly from ISO 11172-1.

The semantics of the multiplexed stream specified in Clause 2.4.4 and the constraints on these semantics specified in Clause 2.4.5 require exact definitions of decoding events and the times at which these events occur. The definitions needed are set out in this International Standard using a hypothetical decoder known as the system target decoder (STD).

The STD is a conceptual model used to define these terms precisely and to model the decoding process during the construction of ISO 11172 streams. The STD is defined only for this purpose. Neither the architecture of the STD nor the timing described precludes uninterrupted, synchronized play-back of ISO 11172 multiplexed streams from a variety of decoders with different architectures or timing schedules.

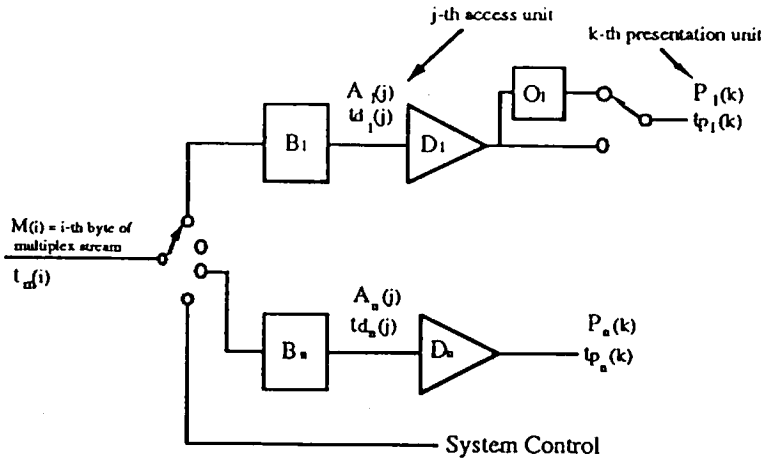

Figure 1-1 Diagram of System Target Decoder

Notation

The following notation is used to describe the system target decoder and is partially illustrated in Figure 1-1.

i, i' are indices to bytes in the MPEG 2 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 ISO 11172 multiplexed stream.

tm(i) indicates the time in seconds at which the i^{th} byte of the MPEG 2 multiplexed stream enters the system target decoder. The value tm(0) is an arbitrary constant.

SCR(i) is the time encoded in the SCR field measured in units of the 90kHz 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 elementary stream n.

System Clock Frequency

NOTE: There is a proposal under consideration that the system_clock_frequency be set to 27 MHz while retaining a sufficient code length in the PCR, PTS, and DTS in the transport stream that the time modulus is greater than 24 hours.

Timing information is carried by several data fields defined in this International Standard in (sub-Clauses 2.4.3 and 2.4.4). 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:

90 000 - 4.5 <= system_clock_frequency <= 90 000 + 4.5

rate of change of system_clock_frequency with time <= $250 * 10^{-6}$ Hz/s

The notation "system_clock_frequency" is used in several places in this International 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 (2^33/system_clock_frequency). This is due to the 33-bit encoding of timing information.

Input to the System Target Decoder

Data from the MPEG 2 multiplexed stream enters the system target decoder. The i^{th} byte, M(i), enters at time tm(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 value encoded in the SCR(i') field indicates time tm(i'), where i' refers to the last byte of the SCR field, M(i').

Specifically:

$$SCR(i') = NINT (system\_clock\_frequency * tm(i')) \% 2^{33}$$

The input arrival time, tm(i), for all other bytes shall be constructed from SCR(i') and the rate at which data arrive, where the arrival rate within each pack is the value represented in the mux_rate field in that pack's header (see Clauses 2.4.3.2 and 2.4.4.2) .

$$tm(i) = \frac{SCR(i')}{system\_clock\_frequency} + \frac{i - i'}{(mux\_rate * 50)}$$

Where:

i' is the index of the final byte of the system_clock_reference 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 field in units of the system clock.

mux_rate is a field defined in (Clauses 2.4.3.2 and 2.4.4.2.)

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 system target decoder.

Buffering

The 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 tm(i).

Bytes present in the pack, system or packet headers of MPEg 2 stream but not part of the packet data (for example the SCR, DTS, PTS, packet_length fields, etc.- see Clause 2.4.3) 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 parameters in the syntax (see sub-Clauses 2.4.3 and 2.4.4).

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 a video elementary stream, group of picture and sequence header data that precedes the picture is removed at the same time. In the case of the first coded picture of a video sequence, any zero bit or byte stuffing immediately preceding the sequence header is removed at the same time. Note that this only applies to the first picture of a video sequence and not to additional occurences of a sequence header within a video sequence. As the access unit is removed from the buffer it is instantaneously decoded into a presentation unit.

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 video decoding to store I-pictures and P-pictures while the sequence of presentation units is reordered before presentation.

In the case of a video elementary stream, some access units may not be stored in presentation order. These access units will need to be reordered before presentation. In particular, an I-picture or a P-picture stored before one or more B-pictures must be delayed in the reorder buffer, O_n, of the system target decoder before being presented. It 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.

If $P_n(k)$ is an I-picture or a P-picture that needs to be reordered before presentation, it is stored in O_n after being decoded and the picture previously stored in O_n is presented. Subsequent B-pictures are decoded and presented without reordering.

The time at which a presentation unit $P_n(k)$ is presented to the viewer is $tp_n(k)$. For presentation units that are not reordered, $tp_n(k)$ is equal to $td_n(j)$ since the access units are decoded instantaneously. For presentation units that are reordered $tp_n(k)$ 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.

Part 2 Clause 2.4.1 of this International Standard explains reodering 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 system target decoder the display of a video presentation unit (a picture) occurs instantaneously at its presentation time, $tp_n(k)$.

In the system target decoder 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 Program System Stream Syntax

The following syntax describes a stream of bytes.

2.4.3.1 Packetized Elementary Stream

The packetized elementary stream or PES stream, consists of PES packets , etc See MPEG-2 System WD proposal June 9, 1993

Syntax	No. of Bits	Identifier
packet() {		
packet_start_code_prefix	24	bslbf
stream_id	8	uimsbf
packet_length	16	uimsbf
if (stream_id != private_stream_2) {		
while (nextbits() == '10') {		
'10'	2	bslbf
scrambling_control_flags	2	bslbf
vital_data_flag	1	bslbf
ESCR_flag	1	bslbf
ES_rate_flag	1	bslbf
reserved	3	bslbf
packet_continuity_counter	8	uimsbf
if(ESCR_flag=='1'){		
elementary_SCR[32..30]	3	bslbf
marker_bit	1	bslbf
elementary_SCR[29..15]	15	bslbf
marker_bit	1	bslbf
elementary_SCR[14..0]	15	bslbf
marker_bit	1	uimsbf
'0001'	4	bslbf
elementary_SCR[32..30]	3	bslbf
marker_bit	1	bslbf
elementary_SCR[29..15]	15	bslbf
marker_bit	1	bslbf
elementary_SCR[14..0]	15	bslbf
marker_bit	1	bslbf
elementary_SCR_extension	9	bslbf
Note: elementary_SCR_extension is coded in units of 27MHZ		
}		
if (ES_rate_flag == '1') { → marker bit	1	
ES_rate	22	uimsbf
Note: ES_rate has similar semantics to mux_rate		
reserved → marker bit	1	bslbf
}		
}		
while(nextbits()=='11'		
stuffing_byte	8	bslbf
if(nextbits()=='01'{		
'01'	2	bslbf
STD_buffer_scale	1	bslbf
STD_buffer_size	13	uimsbf
Note: STD_buffer_size applies only to PES streams		
}		

if(nextbits()=='0010'{		
'0010'	4	bslbf
presentation_time_stamp[32.30]	3	bslbf
marker_bit	1	bslbf
presentation_time_stamp[29.15]	15	bslbf
marker_bit	1	bslbf
presentation_time_stamp[14..0]	15	bslbf
marker_bit	1	bslbf
}		
else if(nextbits()=='0011'{		
'0011'	4	bslbf
presentation_time_stamp[32.30]	3	bslbf
marker_bit	1	bslbf
presentation_time_stamp[29.15]	15	bslbf
marker_bit	1	bslbf
presentation_time_stamp[14..0]	15	bslbf
marker_bit	1	bslbf
'0001'	4	bslbf
decoding_time_stamp[32.30]	3	bslbf
marker_bit	1	bslbf
decoding_time_stamp[29.15]	15	bslbf
marker_bit	1	bslbf
decoding_time_stamp[14..0]	15	bslbf
marker_bit	1	bslbf
}		
else		
'0000 1111	8	bslbf
}		
for(i=0; i,N; i++){		
packet_data_byte	8	bslbf
}		
}		

2.4.3.2 Program Stream

Syntax	No. of bits	Identifier
iso11172_stream() {		
do {		
pack()		
} while (nextbits() == pack_start_code)		
MPEG_program_end_code	32	bslbf
}		

2.4.3.2.1 Pack Layer of Program Stream

Pack

Syntax		No. of bits	Identifier
pack() {			
pack_start_code		32	bslbf
'0010'		4	bslbf
system_clock_reference	[32..30]	3	bslbf
marker_bit		1	bslbf
system_clock_reference	[29..15]	15	bslbf
marker_bit		1	bslbf
system_clock_reference	[14..0]	15	bslbf
marker_bit		1	bslbf
marker_bit		1 < 9	bslbf
mux_rate		22	uimsbf
marker_bit		1	bslbf
if (nextbits() == system_header_start_code)			
system_header ()			
while (nextbits() == packet_start_code_prefix)			
packet()			
}			

SCR extension

NOTE: ECM data may belong in the Pack header

2.4.3.2.2 Packet Layer of Program Stream

The packet layer of the program stream is defined by the Packetized Elementary Stream layer 2.4.3.1

2.4.3.2 System header

Syntax	No. of Bits	Identifier
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
STD_buffer_bound_scale	1	bslbf
STD_buffer_size_bound	13	uimsbf
STD_buffer_size_increase	8	uimsbf
Note: STD_buffer_size_increase denotes additional STD buffer size due to the placement of PES packets in a Program stream		
}		
}		

link to prog. descriptor

2.4.3.2.4 Program Stream Description Table

Note: This table is intended to be included in distinct packets with an identifiable packet ID, and not within the same packets which contain elementary stream data. (In case of prog. stream in system header)

Stream description table

Syntax	No. of Bits	Identifier
stream_description_table() {		
while(nextbits()=='1110'){		
video_stream_id	8	uimsbf
video_hierarchy	4	uimsbf
base_video_stream_id	8	uimsbf
ECM_substream	4	uimsbf
EMM_substream	4	uimsbf
version_registration_authority	16	uimsbf
version_code	16	uimsbf
while(nextbits()=='110')		
audio_stream_id	8	uimsbf
audio_channels_select	16	bslbf
base_video_stream_id	4	uimsbf
language_registration_authority	8	uimsbf
language_code	8	uimsbf
clean_effects	4	uimsbf
conditional_access_ECM_substream	4	uimsbf
conditional_access_EMM_substream	4	uimsbf
version_registration_authority	16	uimsbf
version_code	16	uimsbf
}		

** NOTE: the Registration Authority field requires further discussion. The use of the name "Registration Authority" is meant to imply that National Bodies and other groups collaborating with ISO/IEC JTC1/sc29/WG11 are valid registration authorities. One value of this field is reserved to indicate ISO.*

Note: the name "Version Registration Authority" as a replacement for the name "Version Country Code", and the changes of the sizes of the registration authority and version code fields from 8 to 16 bits are proposed by the Systems group chairman and not yet agreed by the committee nor an Ad Hoc group.

The use of an EMM_substream also requires further discussion.

2.4.3.2.5 Program Stream Directory

Note: The following section is intended to include a program stream directory as proposed in January 1993 and is intended to be optional and to be carried in a specified stream.

Program Stream Directory

Syntax	No. of bits	Identifier
program_stream_directory() {		
}		

2.4.3.3 Packet Layer

Syntax	No. of Bits	Identifier
packet() {		
packet_start_code_prefix	24	bslbf
stream_id	8	uimsbf
packet_length	16	uimsbf
if (stream_id != private_stream_2) {		
while (nextbits() == '10') {		
'10'	2	bslbf
scrambling_control_flags	2	bslbf
vital_data_flag	1	bslbf
reserved	3	bslbf
	8	uimsbf
packet_continuity_counter		
}		
while (nextbits()=='11')		
stuffing_byte	8	bslbf
if (nextbits () == '01') {		
'01'	2	bslbf
STD_buffer_scale	1	bslbf
STD_buffer_size	13	uimsbf
}		
if (nextbits() == '0010') {		
'0010'	4	bslbf
	3	bslbf
presentation_time_stamp[32..30]		
marker_bit	1	bslbf
	15	bslbf
presentation_time_stamp[29..15]		
marker_bit	1	bslbf
	15	bslbf
presentation_time_stamp[14..0]		
marker_bit	1	bslbf
}		
else if (nextbits() == '0011') {		
'0011'	4	bslbf
	3	bslbf
presentation_time_stamp[32..30]		
marker_bit	1	bslbf
	15	bslbf
presentation_time_stamp[29..15]		
marker_bit	1	bslbf

presentation_time_stamp[14..0]	15	bslbf
marker_bit	1	bslbf
'0001'	4	bslbf
decoding_time_stamp[32..30]	3	bslbf
marker_bit	1	bslbf
decoding_time_stamp[29..15]	15	bslbf
marker_bit	1	bslbf
decoding_time_stamp[14..0]	15	bslbf
marker_bit	1	bslbf
}		
else		
0000 1111'	8	bslbf
}		
for (i = 0; i < N; i++) {		
packet_data_byte	8	bslbf
}		
}		

2.4.4 Specification of the System Transport Stream Syntax

The following syntax describes a stream of bytes.

2.4.4.1 MPEG 2 Transport stream

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

```
            adaptation_field()
        }
        if((adaptation_field_flag & 0x01) == '01') {
          for (i=0;i<N;i++){
            packet_data_byte                    8          bslbf
        }
        }
    }
```

Notes : The use of the 2 bit adaptation _field_flag is as follows:

0	*0*	*Reserved*
0	*1*	*Payload Only*
1	*0*	*AH only*
1	*1*	*AH & Payload*

Also note the PID field is a 16 bit value which is byte aligned

Note: The following has been revised in accordance with the agreement of the Ad Hoc Group meeting in Atlanta, May 1993, to make the transport Istream definition a "two layer synchronous" structure, with aligned PES packets containing the coded data.

Semantic Constraints

1. packet_data_bytes consists of contiguous segments of Packetized Elementary Stream (PES) packets.

2. If a PES packet starts within a Transport Packet, the PES packet's Packet Start Code Prefix occurs immediately following the packet_id, or the adaption_field() if one is present.

A clear syntactic description of these constraints is needed. One possible syntactic element that would help would be a bit in the transport packet header indicating that a PES packet begins within the transport packet.

2.4.4.2 Transport Packet Layer

Syntax	No. of bits	Identifier
transport_packet(){		
sync_byte	8	bslbf
packet_error_indication	1	bslbf
PES_start	1	bslbf
priority	1	bslbf
PID	13	uimsbf
scrambling_control_field	2	bslbf
adaptation_field_flag	2	bslbf
continuity_counter	4	uimsbf
if((adaptation_field_flag & 0x10)== '10') {		
adaptation_field()		
}		
if((adaptation_field_flag & 0x01) == '01') {		
for (i=0;i<N;i++){		
packet_data_byte	8	bslbf
}		
}		
}		

Notes : The use of the 2 bit adaptation _field_flag is as follows:

0	*0*	*Reserved*
0	*1*	*Payload Only*
1	*0*	*AH only*
1	*1*	*AH & Payload*

Also note the PID field is a ~~16~~ 13 bit value which is byte aligned

Note: The following has been revised in accordance with the agreement of the Ad Hoc Group meeting in Atlanta, May 1993, to make the transport Istream definition a "two layer synchronous" structure, with aligned PES packets containing the coded data.

Semantic Constraints

packet_data_bytes consists of contiguous segments of Packetized Elementary Stream (PES) packets.

2. If a PES packet starts within a Transport Packet, the PES packet's Packet Start Code Prefix occurs immediately following the packet_id, or the adaption_field() if one is present.

A clear syntactic description of these constraints is needed. One possible syntactic element that would help would be a bit in the transport packet header indicating that a PES packet begins within the transport packet.

3. The PID values 0 and 1 are reserved for program map table & C.A. table respectively.

4. PID(8191) ~~for stuffing~~ reserved for null-packets

5. PID values 2..7 & 1FF8-1FFE reserved.

2.4.4.3 Adaptation field

Syntax	No. of Bits	Identifier
adaptation_field() {		
adaptation_field_length	8	uimsbf
PCR_flag	1	bslbf
~~elementary_stream_rate_flag~~	~~1~~	~~bslbf~~
private_flag	1	bslbf
discontinuity_flag	1	bslbf
random_access_flag	1	bslbf
reserved	~~3~~ 4	bslbf
if(PCR_flag==1) {		
program_clock_reference	33	uimsbf
reserved	6	bslbf
program_clock_reference_extension	9	uimsbf
~~delta_program_clock_reference~~	~~33~~	~~uimsbf~~
~~reserved~~	~~6~~	~~bslbf~~
~~delta_program_clock_reference_extension~~	~~9~~	~~uimsbf~~
~~*Note: meaning of delta PCR is not defined*~~		
} elem stream		
~~if(program_mux_rate_flag=='1')~~ {		
elementary_stream_rate Atlanta "should be readily available ~~or not~~" to mux	~~22~~	uimsbf
~~reserved~~	~~2~~	bslbf
}		
if(Private_flag==1) for(i=0; i<N; i++) { in order		
private_segment_length	8	uimsbf
private_data_bytes		
}		
}		

is PES unscrambled or not

Semantics:

discontinuity_flag, if set to '1', means the PCR in this packet or in some packet which follows closely is discontinuous with respect to previous PCR values, and the PTS and DTS values in the streams which are part of this same program also have discontinuities in this or shortly following packets. Details of "shortly following" remain to be provided.

random_access_point_flag, is set '1', means that this packet or some packet or a shortly following packet contains coded information which is convenient for accessing the stream at random. Details remain to be provided.

2.4.4.4 ~~System header segment~~ Prog. Specific. Info

4 pages >

NOTE: This information is intended to be inlcuded in ~~a global information table, and it is intended to be optional. The syntax that follows is currently the same as in MPEG 1 and requires editing.~~ the program table

table, some fields may be optional) (to be included in program

Syntax	No. of bits	Identifier
system_header_segment() {		
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~~ PID	8	uimsbf
'11'	2	bslbf
STD_buffer_bound_scale	1	bslbf
STD_buffer_size_bound	13	uimsbf
STD_buffer_size_increase	8	uimsbf
Note: STD_buffer_size_increase denotes additional STD buffer size due to the placement of PES packets in a Transport Stream		
}		
}		

2.4.5 Semantic Definition of Fields in Syntax

NOTE: this section needs to be revised to include the fields which are new to MPEG-2. The following section currently is taken from MPEG-1.

2.4.5.1 MPEG 2 Program Layer

MPEG_program_end_code -- The MPEG_program_end_code is the bit string "0000 0000 0000 0000 0000 0001 1011 1001" (000001B9 in hexadecimal). It terminates the MPEG 2 program stream.

2.4.4.2 Pack Layer

Pack

pack_start_code -- The pack_start_code is the bit string "0000 0000 0000 0000 0000 0001 1011 1010" (000001BA in hexadecimal). It identifies the beginning of a pack.

system_clock_reference -- The system_clock_reference (SCR) is a 33-bit number coded in three separate fields. It indicates the intended time of arrival of the last byte of the system_clock_reference field at the input of the system target decoder. The value of the SCR is measured in the number of periods of a 90kHz

system clock with a tolerance specified in (Clause 2.4.2). Using the notation of (Clause 2.4.2) the value encoded in the system_clock_reference is:

$$SCR(i) = NINT\ (system\_clock\_frequency * (tm(i))\ )\ \%\ 2^{33}$$

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

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

mux_rate -- This is a positive integer specifying the rate at which the system target decoder receives the MPEG 2 program stream during the pack in which it is included. The value of mux_rate is measured in units of 50 bytes/second rounded upwards. The value zero is forbidden. The value represented in mux_rate is used to define the time of arrival of bytes at the input to the system target decoder in (Clause 2.4.2) of this Working Draft. The value encoded in the mux_rate field may vary from pack to pack in an MPEg 2 program multiplexed stream.

System Header

system_header_start_code -- The system_header_start_code is the bit string "0000 0000 0000 0000 0000 0001 1011 1011" (000001BB in hexadecimal). It identifies the beginning of a system header.

header_length -- The header_length shall be equal to the number of bytes in the system header following the header_length field. Note that future extensions of this Working Draft 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 MPEG 2 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 MPEG 2 program stream of which the decoding processes are simultaneously active. For the purpose of this Clause, the decoding process of an MPEG 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 one-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 11172 stream shall adhere to the following linear equation:

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

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 11172 stream of the final byte of any system_clock_reference field in the stream.

CSPS_flag -- The CSPS_flag is a one-bit flag. If its value is set to "1" the MPEG 2 program stream meets the constraints defined in Part 1 (Clause 2.4.6) of this Working Draft.

system_audio_lock_flag -- The system_audio_lock_flag is a one-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 (2.4.2) defines system_clock_frequency and the audio sampling rate is specified in Part 3 of this Working Draft. The system_audio_lock_flag may only be set to "1" if, for all presentation units in all audio elementary streams in the MPEG 2 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.

$$SCASR = \frac{\text{system\_clock\_frequency}}{\text{audio sample rate in the STD}}$$

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

Ratio SCASR	90 000 / 32 000	90 000 / 44 100	90 000 / 48 000

Nominal audio sampling frequency (kHz)	32	44.1	48

system_video_lock_flag -- The system_video_lock_flag is a one-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 (2.4.2) defines system_clock_frequency and the video picture rate is specified in Part 2 of this Working Draft. The system_video_lock_flag may only be set to "1" if, for all presentation units in all video elementary streams in the MPEG 2 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.

$$SCPR = \frac{\text{system\_clock\_frequency}}{\text{picture rate in the STD}}$$

Nominal picture rate (Hz)	23.976	24	25	29.97	30	50	59.94	60
Ratio SCPR	15 015 / - 4	3 750	3 600	3 003	3 000	1 800	3 003 / - 2	1 500

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 MPEG 2 video streams in the MPEG 2 program stream of which the decoding processes are simultaneously active. For the purpose of this Clause, the decoding process of an MPEg 2 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. Until otherwise specified by ISO it shall have the value "1111 1111".

stream_id -- The stream_id indicates the type and number of the stream to which the following STD_buffer_bound_scale and STD_buffer_size_bound fields refer.

If stream_id equals "1011 1000" the STD_buffer_bound_scale and STD_buffer_size_bound fields following the stream_id refer to all audio streams in the MPEg 2 program stream.

If stream_id equals "1011 1001" the STD_buffer_bound_scale and STD_buffer_size_bound fields following the stream_id refer to all video streams in the MPEg 2 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 type and number according to the following table. This table is used also to identify the stream type and number indicated by the stream_id defined in (Clause 2.4.4.3).

stream_id table

stream_id	stream type
1011 1100	reserved stream
1011 1101	private_stream_1
1011 1110	padding stream
1011 1111	private_stream_2
110x xxxx	MPEG audio stream - number xxxxx
1110 xxxx	MPEG video stream - number xxxx
1111 xxxx	reserved data stream - number xxxx
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.	

Each elementary stream present in the MPEG 2 program stream shall have its STD_buffer_bound_scale and STD_buffer_size_bound specified exactly once by this mechanism in each system header.

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

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 MPEg 2 program stream. If STD_buffer_bound_scale has the value "0" then STD_buffer_size_bound measures the buffer size bound in units of 128 bytes. If STD_buffer_bound_scale has the value "1" then STD_buffer_size_bound measures the buffer size bound in units of 1024 bytes. Thus:

```
if (STD_buffer_bound_scale == 0)
        BSn <= STD_buffer_size_bound * 128;
else
        BSn <= STD_buffer_size_bound * 1024;
```

2.4.5.3 Packet Layer

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" (000001 in hexadecimal).

stream_id -- The stream_id specifies the type and number of the elementary stream as defined by the stream_id table in Clause 2.4.4.2.

packet_length -- The packet_length specifies the number of bytes remaining in the packet after the packet_length field.

stuffing_byte -- This is a fixed 8-bit value equal to "1111 1111" that can be inserted by the encoder for example to meet the requirements of the digital storage medium. It is discarded by the decoder. No more than sixteen stuffing bytes shall be present in one packet header.

STD_buffer_scale -- The STD_buffer_scale is a one-bit field that indicates the scaling factor used to interpret the subsequent STD_buffer_size field. If the preceding stream_id indicates an audio stream, STD_buffer_scale shall have the value "0". If the preceding stream_id indicates a video stream, STD_buffer_scale shall have the value "1". For all other stream types, the value may be either "1" or "0".

STD_buffer_size -- The STD_buffer_size is a 13-bit unsigned integer defining the size of the input buffer, BSn, in the system target decoder. If STD_buffer_scale has the value "0" then the STD_buffer_size measures the buffer size in units of 128 bytes. If STD_buffer_scale has the value "1" then the STD_buffer_size measures the buffer size in units of 1024 bytes. Thus:

```
if (STD_buffer_scale == 0)
        BSn = STD_buffer_size * 128;
else
        BSn = STD_buffer_size * 1024;
```

The encoded value of the STD buffer size takes effect immediately when the STD_buffer_size field is received by the MPEG System Target Decoder.

presentation_time_stamp -- The presentation_time_stamp (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). Using the notation of Clause (2.4.2) the value encoded in the presentation_time_stamp is:

$$PTS = NINT\ (system\_clock\_frequency * (tp_n(k))) \ \%\ 2^{33}$$

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 a video picture start code or the first byte of the synchronization word of an audio frame (see Parts 2 and 3 of this International Standard) is present in the packet data.

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.

decoding_time_stamp -- The decoding_time_stamp (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). Using the notation of Clause (2.4.2) the value encoded in the decoding_time_stamp is:

$$DTS = NINT (system\_clock\_frequency * (td_n(j))) \ \% \ 2^{33}$$

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 a video picture start code or the first byte of the synchronization word of an audio frame (see Parts 2 and 3 of this Working Draft) is present in the packet data.

packet_data_byte -- packet_data_bytes shall be contiguous bytes of data from the elementary stream indicated by the packet's stream_id. The byte-order of the elementary stream shall be preserved. The number of packet_data_bytes, N, may be calculated from the packet_length field. N is equal to the value indicated in the packet_length minus the number of bytes between the last byte of the packet_length field and the first packet_data_byte.

In the case of a video stream, packet_data_bytes are coded video data as defined in Part 2 of this International Standard. In the case of an audio stream, packet_data_bytes are coded audio data as defined in Part 3 of this International Standard. In the case of a padding stream, packet_data_bytes consist of padding bytes. Each padding byte is a fixed bit-string with the value "1111 1111". In the case of a private stream (type 1 or type 2), packet_data_bytes are user definable and will not be defined by ISO in the future. The contents of packet_data_bytes in reserved streams may be specified in the future by ISO.

2.4.6 Restrictions on the Multiplexed Stream Semantics

2.4.6.1 Buffer Management

The MPEg 2 program stream, M(i) in the notation described in Clause (2.4.2), shall be constructed and tm(i) shall be chosen so that the input buffers of size BS_1

through BSn neither overflow nor underflow in the system target decoder. That is:

$0 <= F_n(t) <= BS_n$ for all t and n

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

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

For all MPEg 2 program streams the delay caused by system target decoder input buffering shall be less than or equal to one second. The input buffering delay is the difference in time between a byte entering the input buffer and when it is decoded.

Specifically:

$td_n(j) - tm(i) <= 1$

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

2.4.6.2 Frequency of Coding the system_clock_reference

The MPEG 2 program stream, M(i), shall be constructed so that the time interval between the final bytes of system_clock_reference fields in successive packs shall be less than or equal to 0.7 seconds. Thus:

$|t_m(i) - t_m(i')| <= 0.7$ seconds

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

2.4.6.3 Frequency of presentation_time_stamp coding

The MPEG 2 program stream M(i) shall be constructed so that the maximum difference between coded presentation_time_stamps is 0.7 seconds. Thus:

$|tp_n(k) - tp_n(k'')| <= 0.7$ seconds

for all n and all k, 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 (as defined in Clause (2.4.5.4)) exists in elementary stream n between $P_n(k)$ and $P_n(k'')$.

2.4.6.4 Conditional Coding of Time Stamps

For each elementary stream of an MPEg 2 program stream, the presentation_time_stamp shall be encoded in the packet in which the first access unit of that elementary stream commences. For the purposes of this Clause a video access unit commences in a packet if the first byte of the picture_start_code is present in the packet data (see Part 2 of this Working Draft). An audio access unit commences in a packet if the first byte of the synchronization word of the audio frame is present in the packet data (see Part 3 of this Working Draft).

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 MPEG 2 program 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 packet header with the following exception. If no access unit commences in the packet data, the presentation_time_stamp shall not be present in the packet header. If a presentation_time_stamp is present in a packet header it shall refer to the presentation unit corresponding to the first access unit that commences in the packet data.

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

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

2.4.6.5 Frequency of Coding STD_buffer_size in Packet Headers

The STD_buffer_scale and STD_buffer_size fields shall occur in the first packet of each elementary stream and again whenever the value changes. They may also occur in any other packet.

2.4.6.6 Coding of System Header

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 MPEG 2 program stream. The values encoded in all the system headers in the MPEg 2 program stream shall be identical.

2.4.7 Constrained System Parameter Stream

An MPEg 2 program stream is a "constrained system parameters stream" (CSPS) if it conforms to the bounds specified in this Clause. MPEG 2 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 stream header (Clause 2.4.3.2). The CSPS is a subset of all possible MPEg 2 program streams.

Packet Rate

In the CSPS, the maximum rate at which packets shall arrive at the input to the system target decoder is 300 packets per second if the value encoded in the mux_rate field is less than or equal to 5 000 000 bits/second. For higher bit-rates the CSPS packet rate is bounded by a linear relation to the value encoded in the mux_rate field.

Specifically, for all packs p in the ISO 11172 multiplexed stream,

$$NP <= (tm(i') - tm(i)) * 300 * \max\left[1, \frac{R_{max}}{5*10^6}\right]$$

where

R_{max} = 8 * 50 * rate_bound bits/second

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.

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

tm(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 MPEG 2 program stream, the time of arrival of the last byte of the MPEG_program_end_code.

System Target 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:

In Clause (2.4.3.2 of Part 2 of this Working Draft) the horizontal picture size, horizontal_size, and the vertical picture size, vertical_size, are defined. If the values encoded in horizontal_size and vertical_size meet the constraints on the picture size specified for the constrained_parameters_flag in Clause (2.4.3.2 of Part 2), then

$BS_n <= 46 * 1024$ bytes.

For all other video elementary streams in a CSPS,

$BS_n <= \max [46 * 1024, R_{vmax} * 46 * 1024 \ // \ (1{,}856*10^6)]$ bytes

where R_{vmax} is the greatest value of video bit_rate specified or used in the elementary video stream; reference Part 2 Clause (2.4.3.2).

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

BS_n <= 4096 bytes.

Annex

The following section has been proposed for inclusion but has not yet been fully discussed by the Systems group.

Transport stream multiplex map: A bitstream composed of the necessary and sufficient information to demultiplex and present programs carried within the transport multiplex.

Addition to 2.4.1

Some of the special information such an ECM and EMM streams do not have contents defined by this standard. The standard does however provide mechanisms for program service providers to transport and identify this data for decoder processing. The special stream which will be completely specied is the transport multiplex map. This map will have the necessary and sufficient information to demultiplex and present programs. This information includes:

- indentification of all transport packet streams associated with a program
- characteristics of the program elementary streams, including program identification table
- specific transport attributes of each transport stream corresponding to each program element
- assocations between program streams
- packet id of associated EMM stream
- packet id of associated ECM stream
- system mapping of multiple transport streams.

Addition following 2.4.4

2.4.x Specification of the Transport Stream Multiplex Map.

The following syntax describes a stream of bytes.

2.4..x.1 MPEG 2 Transport Stream Multiplex Map

Syntax	No. of bits	Identifier

```
MPEG_transport_muxmap()  {
        do {
                muxmap_packet()
        }  while  (nextbits()  ==  muxmap_start_code)

}
```

2.4.x.2 Multiplex Map Layer

Syntax	No. of Bits	Identifier
muxmap_packet() {		
muxmap_start_code	X	**bslbf**
transport_id	X	**uimsbf**
transport_segment	X	**uimsbf**
while(next_bits() == prgmap_start_code)		
prgmap()		
}		

2.4.x.3 Program Map Layer

Syntax	No. of Bits	Identifier
prgmap() {		
program_number	16	uimsbf
transport_number	X	uimsbf
PCR_PID	16	uimsbf
EMM_PID	16	uimsbf
CA_parameters	TBD	uimsbf
Event_label	TBD	UIMSBF
UK_Porn_code	8	uimsbf
program_content_type	TBD	UIMSBF
program_label_length	8	uimsbf
for(i=0 ; i<program_label_length; i++){		
ascii_char()		
}		
number_elementary_PID	8	uimsbf
for(i=0 ; i< number_of_associated_PID; i++)		
{		
element_PID	16	uimsbf
stream_type	8	uimsbf
ECM_PID	16	uimsbf
stream_attributes()		
}		
}		
coding layer priority	1	bslbf

2.4.x.4 Stream Attribute Layer

Syntax	No. of bits	Identifier
stream_attributes() {		
/* Angra Table + extensions */		
}		

2.4.x.3 Program Map Layer

Syntax	No. of Bits	Identifier
prgmap() {		
program_number	16	uimsbf
transport_number	X	uimsbf
SCR_PID	16	uimsbf
EMM_PID	16	uimsbf
CA_parameters	TBD	uimsbf
UK_Porn_code	8	uimsbf
program_label_length	8	uimsbf
for(i=0 ; i<program_label_length; i++){		
ascii_char()		
}		
number_elmentary_PID	8	uimsbf
for(i=0 ; i< number_of_associated_PID; i++)		
{		
element_PID	16	uimsbf
stream_type	8	uimsbf
ECM_PID	16	uimsbf
stream_attributes()		
}		
}		

2.4.x.4 Stream Attribute Layer

Syntax	No. of bits	Identifier
stream_attributes() {		
/* Angra Table + extensions */		
}		

2.4.X+1 Transport Stream Multiplex Map semantics.