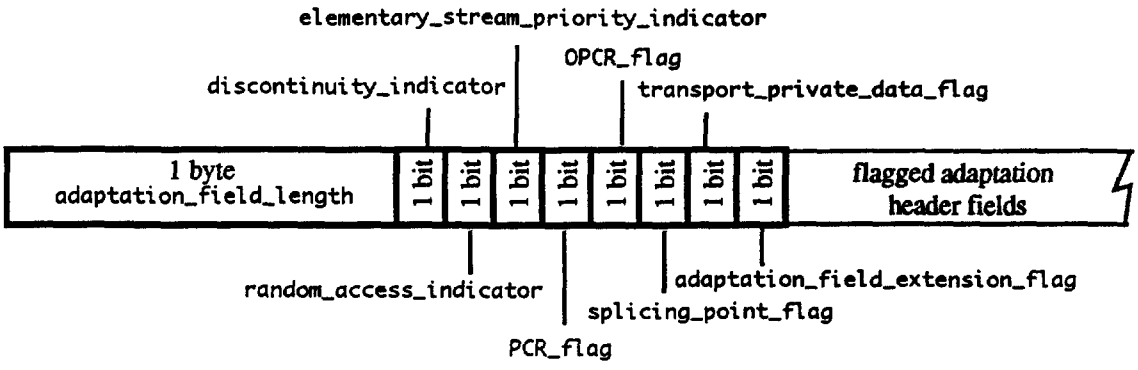

Fig. 5.5.2. Format for the fixed length component of the adaptation header.

The adaptation_field_length specifies the number of bytes that follow it in the adaptation header. The adaptation header could include stuffing bytes after the last adaptation header component field. (Stuffing bytes have a value of 0xff and are not interpreted at the decoder.) In this case the adaptation_field_length also reflects the stuffing bytes. The value in the adaptation_field_length field can also be used by the decoder to skip over the entire adaptation header, and to directly advance to the data payload in the packet if desired.

The presence of additional adaptation header fields is indicated by the state of the last five single bit flags shown in Fig. 5.5.2. (with a value of '1' indicating that a particular field is present). The three flags at the beginning do not result in extensions to the adaptation header and are described in the table below.

field	General function	GA usage
discontinuity_indicator	Indicates if there is a discontinuity in the PCR values that will be received from this packet onwards. This happens when bit streams are spliced. This flag should be used at the receiver to change the phase of the local clock.	As defined for local program insertion. See section 2.2.3.
random_access_indicator	Indicates that the packet contains data that can serve as random access point into the bit stream. As an example, these can correspond to the start of sequence header information in the GA video bit stream.	As defined. See section 2.2.2.
elementary_stream_ priority_indicator	Logical indication of priority of the data being transmitted in the packet.	Not used. If set it is ignored at a GA decoder.

As mentioned earlier, the other components of the adaptation header appear based on the state of the flags shown in Fig. 5.5. The order in which these components appear in the bit stream is the same as the order of the flags. Based on the type of adaptation header information being

conveyed, the data in these fields may be either fixed length or variable length. These fields are described in detail next.

5.5.2.1. The PCR and OPCR fields

The use of the PCR has been described in detail in section 5.2.2.1. This section deals mainly with the format for transmission.

Overall Format

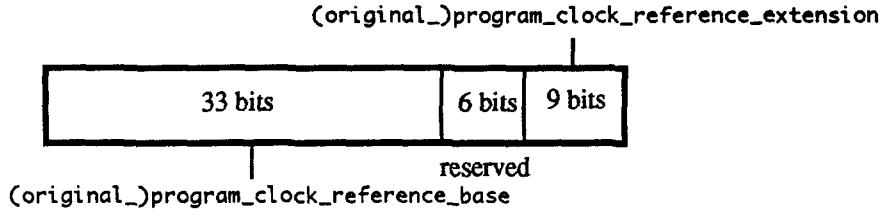

Fig. 5.5.3. The (O)PCR header format.

Functionality

field	General function	GA usage
PCR	Indicates intended time of arrival of last byte of the program_clock_ reference_extension at target decoder. Used for synchronization of the system decoding process. This field can be modified during the transmission process.	As defined. The PCR will be transmitted at least once every 100 milliseconds.
OPCR	Indicates intended time of arrival of last byte of the original_program_ clock_reference_extension at target decoder for a single program. This field is not modified during transmission.	May be used for recording and playback of single programs. Not used in the GA receiver in the decoding process.

Functional details of the (O)PCR format

The total PCR value is based on the state of a 27 MHz clock. The 9 bit extension field cycles from 0 to 299 at 27 MHz, at which point the value in the 33 bit field is incremented by one. (This results in the 33 bit field being compatible with the 33 bit field that are used for the 90 KHz clock of MPEG-1. Backward compatibility was a concern in the MPEG-2 system design.) The cycle time of the PCR value is approximately 26 hours.

5.5.2.2. The transport_private_data and adaptation_field_extension fields

Format

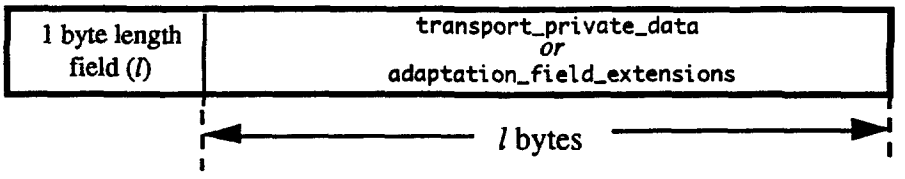

Fig. 5.5.4. The transport_private_data and adaptation_field_extension header format.

Functionality

field	General function	GA usage
transport_private_data	For private data not recognizable by the general MPEG decoders. Meant for short bursts of control information.	To be decided
adaptation_header_ extensions	For future extensions of the adaptation header which may have not been thought of yet.	Not used currently.

5.5.2.3. The splice_countdown field

This field is useful for local program insertion as described in section 5.2.2.3.

Format: This is a one byte field that is present if the splicing_point_flag is set.

Functionality

General Function	GA Usage
Indicates number of packets present in the bit stream, with the same PID as current packet, until a splicing point packet. The splicing point packet is defined as the packet containing a point in the elementary bit stream from which point onwards data can be removed and replaced by another bit stream, so that the resulting transport bit stream is valid according to MPEG-2 rules. Transmitted as a 2s-complement value.	Used for supporting insertion of local programming.

5.5.3. PSIs and the pointer_field

As mentioned in chapter 5.3, the program_association_table and the program_map_tables that describe the organization of a multiplexed GA bit stream are a part of the PSI layer of the GA system. PSI tables, in general, are transmitted in the appropriate bit stream sequentially, without any gap between the tables. This implies that tables do not necessarily start at the beginning of a transport packet. This also implies that in order to decode specific tables, there needs to be some indication of where these begin in the bit stream. This functionality is achieved with the pointer_field. The pointer_field, if present, is the byte of the payload of a packet (after the link and adaptation headers). The pointer_field is present in the packet if a PSI table begins in the packet, an event which is

signaled at the link level, by setting the payload_unit_start_indicator (described in section 5.5.1) to '1'. The pointer_field indicates the number of bytes that follow it before the start of a PSI table. As an example a pointer_field value of 0x00 indicates that a new PSI table begins immediately following it.

5.5.4. The program_association_table

As discussed in section 5.3.2, the program_association_table is transmitted as the payload of the bit stream with PID = 0 and describes how program numbers associated with programs, (e.g., HBO™, ESPN™, etc.) map on to bit streams containing the program_map_tables for these programs. This section discusses the syntax of the table in some depth. The program_association_table may be transmitted as multiple program_association_segments with each segment having a maximum length of 1024 bytes. The transport decoder can extract individual table segments from the bit stream in whatever order it desires. As shown in Fig. 5.5.5, each table segment has a fixed length 10 byte header component for table segment identification, a variable length component that depends on the number of entries contained, and a 4 byte CRC-32 field.

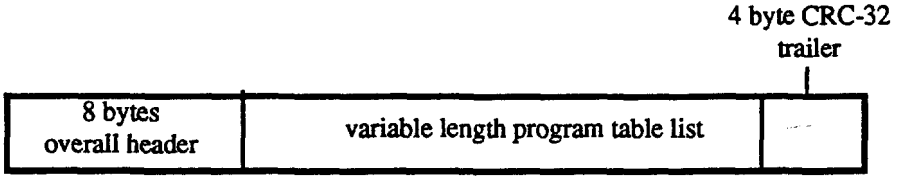

Fig. 5.5.5. High level overall picture of the program_association_segment.

i. The fixed length overall header component is shown below in Fig. 5.5.6 and is described in the table below.

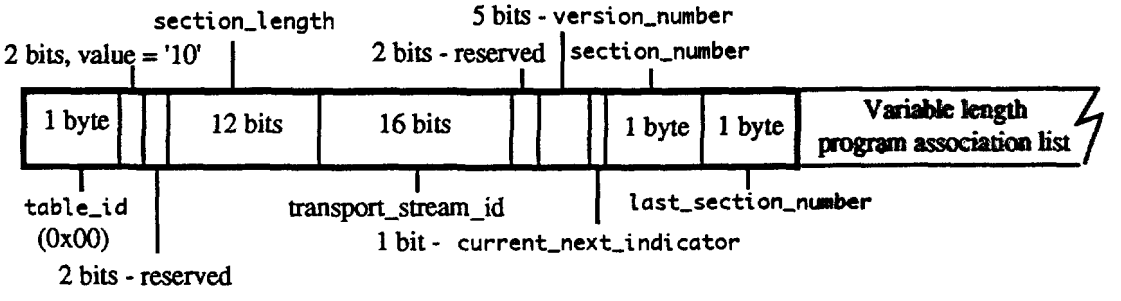

Fig. 5.5.6. Fixed length header component of the program_association_table.

field	General function	GA usage
table_id	Indicates the nature of the table. 0x00 indicates a program_association_table	As defined.

section_length	Length of the section of the program_association_table. This length includes all bytes following this field, up to and including the CRC. The two most significant bits of this field are set to 0, i.e., maximum value is 1024. This field allows the transport decoder to skip sections when reading from the bit stream if desired.	As defined.
transport_stream_id	Identication of a particular multiplex from several in the network.	*Should correspond to a channel number for the terrestrial application.*
version_number	Incremented each time there is a change in the program_association_ table being transmitted.	As defined
current_next_indicator	Value of 1 indicates that the map is currently valid. Value of 0 indicates that the map is not currently being used and will be used next.	As defined
section_number	Identifies the particular section being transmitted.	As defined
last_section_number	Section_number for the last section in the program_association_ table. Needed to confirm when an entireprogram_association_table has been received at the decoder.	As defined

The value of the reserved bits is undefined and the GA system should not interpret these. On the other hand the 2 bit '10' value following the table_id needs to be received correctly.

ii. The variable length component of the table consists of program_count number of fixed length entries corresponding to each program, and stuffing_bytes (to make up the program_association_segment_length). The format for each fixed length entry is shown in Fig. 5.5.7.

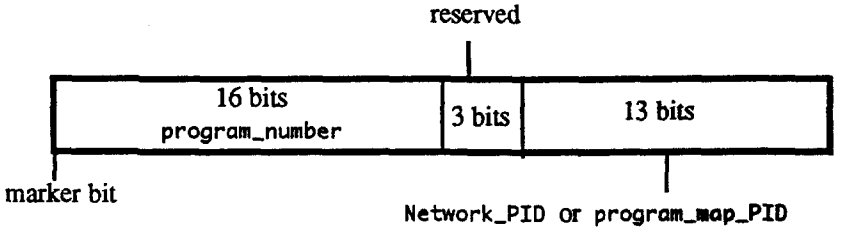

Fig.5. 5.7. Format of each entry in the program_association_table.

The program identity '0' is reserved for the network_PID, i.e., the PID of the bit stream carrying information about the configuration of the overall system. The nature of this bit stream is yet to be defined for the GA system. The format for this bit stream is completely open since it is meant to be a private bit stream. For all other program identities, the program_map_PID is the PID of the bit stream containing the program_map_table for the particular program.

iii. The program_association_table ends with a four byte CRC field that contains results of a CRC done over the entire program map segment, starting at the segment_start_code_prefix. The CRC is based on the polynomial $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

5.5.5. The program_map_table

As discussed in the previous section, the program_map_table is transmitted as the payload of the bit stream with PID = program_map_PID (as indicated in the program_association_table). The program_map_table carries information about the applications that make up programs. Basically each program_map_table is transmitted as a single TS_program_map_section. The format for a TS_program_map_section can be described as a combination of an overall header field, fields that describe each program within the table and a trailer CRC field, as shown in Fig. 5.5.8. The CRC used is the same as for the program_association_table. In general, each program_map_PID may contain more than one TS_program_map_section, each describing a different program.

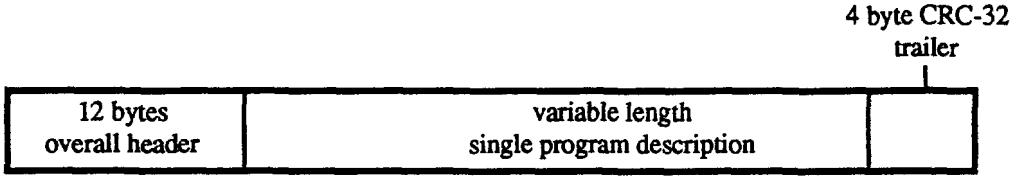

Fig. 5.5.8. High level overall view of the TS_program_map_section.

5.5.5.1. The overall TS_program_map_segment header format

The header format for a TS_program_map_section is shown in Fig. 5.5.9. The format of the first 8 bytes is the same and has similar functionality as that for the program_association_table. The similarity in format is intended to facilitate simple software decoding of the headers. Note that the table_id for TS_program_map_section is different from that for the program_association_table.

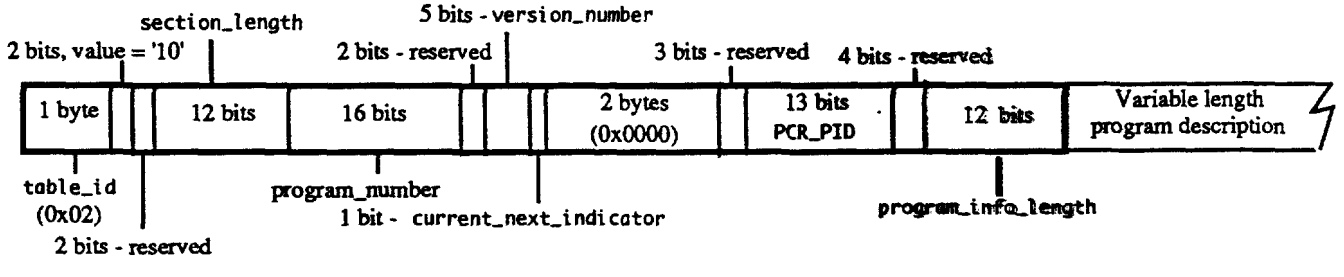

Fig. 5.5.9. Fixed length header for the TS_program_map_section.

The two bytes that were used to identify the transport_stream_id in the association table are now used to identify the program_number of the program whose description follows. The section identification

functions are not required for this table since the description of each program is defined to have to fit into one section. Hence these fields are set to '0' as shown in the figure. The other common header fields between the program_association_table and the TS_program_map_section have functionality as desecribed in the table following Fig. 5.5.8. The additional header fields in a TS_program_map_section are the 13 bit PCR_PID field that identifies the PID of the particular packetized elementary bit stream in the program that contains the PCR values for the program, and the program_info_length field, that indicates the number of bytes of program_descriptors that follow this header field.

The overall program description that follows the header described above consists of the optional, variable length, program_descriptor field (whose length was indicated by the program_info_length field shown in Fig. 5.5.9), followed by a descriptions of each of the individual elementary bit streams that make up the program, i.e., there are one or more elementary bit stream descriptions for each TS_program_map_section.

5.5.5.2 An elementary stream description

Each elementary stream description for has a 5 byte fixed length component and a variable length elementary_stream_descriptor component as shown in Fig. 5.5.10.

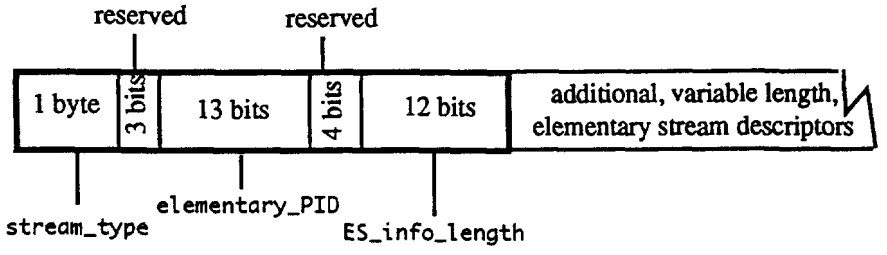

Fig. 5.5.10. The fixed length component of the elementary stream description.

The functionality of the different fields is as follows.

field	General function	GA usage
stream_type	Indicates the application being considered in this elementary stream.	??.
elementary_pid	Indicates the PID of the transport bit stream containing the elementary bit stream.	As defined
ES_info_length	Indicates the length of a variable length elementary_stream_descriptor field that immediately follows.	As defined.

As before reserved bits are ignored.

5.5.6. Descriptors

Descriptors are transmitted in the program_descriptor and the elementary_stream_descriptor fields to describe certain charaacterisitics of the program or the elementary bit stream. In general, each program_descriptor and the elementary_stream_descriptor can consist of number of individual descriptor field elements transmitted sequentially.

Two factors need to be considered in order to use descriptors. In the first place, there has to be an mechanism for indicating the presence of the descriptors. In the PSI tables that have been described, this functionality is achieved by the length field that preceeds the descriptor, with a value of zero indicating that no descriptor are present. A second function is the identification of the descriptor itself. This is achieved within the descriptor header itself, which consists of a one byte descriptor_tag field followed by a one byte descriptor_length field that specificies the number of bytes in the descriptor following the descriptor_length field. The set of valid descriptor_tags in the GA systems is the same as that defined for MPEG-2. The table of tags and the format for each descriptor are not described in this document.

5.6. The PES packet format

As described before, some elementary bit streams, including the GA video and compressed audio, will go through a PES layer packetization prior to the GA transport layer. The PES header carries various rate, timing, and data descriptive information, as set by the encoder. The PES packetization interval is application dependent. The resulting PES packets are of variable length with a maximum size of 2^{16} bytes, when the PES packet length field is set to its maximum value. This value is set to zero for the GA video stream, implying that the packet size is unconstrained and that the header information cannot be used to skip over the particular PES packet. This value of Note also that the PES packet format has been defined to also be of use as an input bit stream for Digital Storage Media (DSM) applications. Although the DSM format will not be used for GA broadcast application, some of the PES header fields related to the DSM functions are also described in this chapter. Note that the ability to handle input bit streams in the DSM format is not essential for a GA receiver, but may be useful for VCR applications.

The constraints on the length of the PES packets for GA video have to be settled upon. Among the decisions to be reached are whether PES packets should only start on GOP boundaries, whether all GOP boundaries should start a new PES packet, whether PES packets should be defined for each access unit (corresponding to a picture), etc.. Note that the format for carrying the PES packet within the GA transport is a subset of the general definition in MPEG. This choice was made to simplify the implementation of the GA receiver. Essentially, in the GA transport system, all data for a PES packet, including the header, are transmitted contiguously as the payload of transport packets. New PES packet data always starts a new transport packet, and PES packets that end in the middle of a transport packet are followed by stuffing bytes for the remaining length of the transport packet.

A PES packet consists of a PES_packet_start_code, PES header flags, PES packet header fields, and a payload (or data block), as shown in Fig. 5.6.1. It is created by the application encoder. The packet payload is a stream of contiguous bytes of a single elementary stream. For video and audio packets, the payload is a sequence of access units from the encoder. The access units correspond to the video pictures and audio frames.

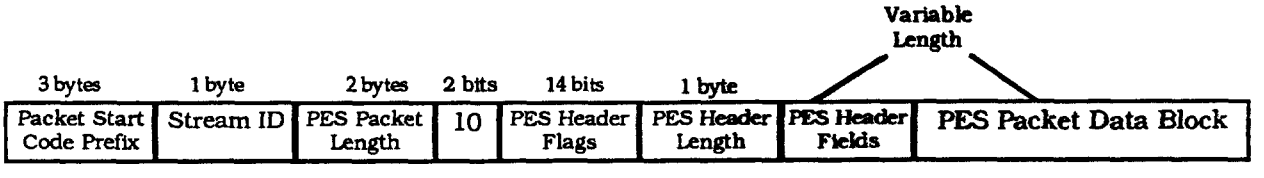

Figure 5.6.1. Structural Overview of a PES Packet

Each elementary stream is identified by a unique stream_id. The PES packets from each encoder carry the corresponding stream_id. PES packets carrying various types of elementary streams can be multiplexed to form a program or transport stream in accordance with Part 1 of the MPEG-2 standard. This chapter deals with the organization of PES packets for MPEG-2 compliant streams only. Specifically, the stream_id field can take on a number of values, indicating the type of data in the payload (See Table for valid values). This section does not define the PES packet structure for 'private data', i.e. stream_id must be 'Reserved', 'Padding', or 'MPEG Audio' or 'MPEG Video'. For 'private data', the PES packet structure is user defined.

The preliminary fields, the packet_start_code_prefix, stream_id, and PES_packet_length, are described in the table below.

Field	Description	GA Usage
packet_start_code_prefix	Indicates the start of a new packet. Together with the stream_id, it forms the packet_start_code. Takes on the value 0 x 00 00 01.	As defined
stream_id	Specifies the type and number of the stream, to which the packet belongs. 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	As defined
PES_packet_length	Specifies the number of bytes remaining in the packet after the this Field.	0 x 00 00 - Only allowed value for video. Details for audio yet to be determined

5.6.1. PES header Flags

A breakdown of the PES header flags is shown in Figure 5.6.2. These flags are a combination of indicators of the properties of the bit stream and indicators of the existence of additional fields in the PES header. The following table describes the flags present the header. The flags not supported by the GA system are set to '0' and form the basis of some of the "constraints" discussed earlier. (These entries are shaded in the table.) Note that the abbreviations in the tables are from the flag names in Fig. 5.6.2.

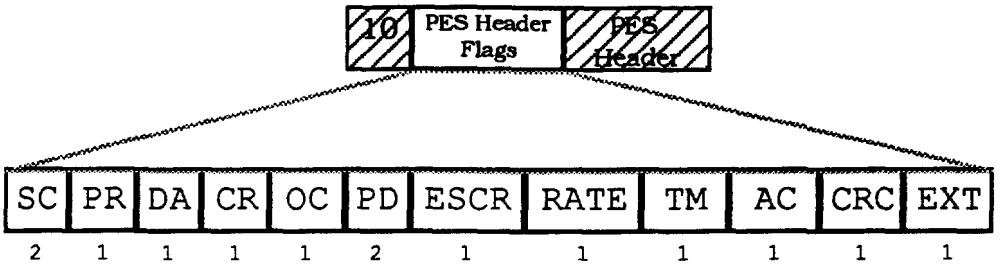

Figure 5.6.2. PES header flags in their relative positions (all sizes in bits)

Flag	Description	GA Usage
PES_scrambling_control (SC)	Indicates the scrambling of the PES packet payload. 00 -- Not Scrambled 01 -- User Defined 10 -- User Defined 11 -- User Defined	Set to 00.
PES_priority (PR)	Indicates the priority of this packet with respect to other packets whose PR field is not set. 1 -- Higher priority 0 -- Same priority	GA does not care how this field is set. (It may be used by applications where necessary.)
data_alignment_indicator (DA)	Indicates the nature of alignment of the first start code occurring in the payload. The type of data in the payload is indicated by the data_stream_alignment_descriptor. 1 - Aligned 0 - No indication of alignment	Must be aligned for video. To be decided for Audio.
copyright (CR)	Indicates the copyright nature of the associated PES packet payload. 1 -- Copyrighted 0 -- Not copyrighted	The GA has yet to define its use of this field.
original_or_copy (OC)	Indicates whether the associated PES packet payload is the original program or a copy. 1 -- Original 0 -- Copy	The GA has yet to define its use of this field.
PTS_DTS_flags (PD)	This flag indicates whether the PTS or PTS and DTS are in the PES header. 00 -- Neither PTS nor DTS is present in header. 1x -- PTS field present 11 -- PTS and DTS field present in header.	The PTS flag is set when video data alignment indicator is set. The DTS may be included to signal the decoder of any special decoding requirements. The PTS transmissions should be spaced less than 700 msecs apart.
ESCR_flag (ESCR)	Indicates whether the Elementary Stream Clock Reference field is present in the PES Header.	Set to '0'.
ES_rate_flag (RATE)	Indicates whether the Elementary Stream Rate field is present in the PES Header.	Set to '0'.
DSM_trick_mode_flag (TM)	Indicates the presence of an 8 bit field describing the mode of operation of the DSM (Digital Storage Media). 1 -- Field present 0 -- Field not present	Set to '0' for the broadcast transmission. May be used for trick modes when using bit streams specifically generated for VCR type operations.
additional_copy_info_flag (AC)	Indicates the presence of the additional_copy_info field. 1 -- Field Present 0 -- Field not present	The GA has yet to define its use of this field.
PES_CRC_flag (CRC)	Indicates whether a CRC field is present in the PES packet.	Set to '0'.
PES_extension_flag (EXT)	This flag is set as necessary to indicate that extension flags are set in the PES header. Its use includes support of private data. 1 -- Field present 0 -- Field not present	As defined.

5.6.2. The PES header

The PES header immediately follows the field PES_header_length, which indicates the header size in bytes. The size of the header includes all the header fields, any extension fields, and stuffing_bytes. The flags described in the previous section indicate the organization of the PES header, i.e. which fields it does and does not contain. In essence, all the fields of the PES header are optional. Certain applications require particular fields to be set appropriately. For example, GA transport of video PES packets requires that the data_alignment_indicator be set. The trick mode flag is not set in this case. For DSM retrieval of video, the opposite is true. It is the application encoder's function to set the appropriate flags, and encode the corresponding fields. The fields are further described in the following sections. The association between the flags and the corresponding fields is obvious.

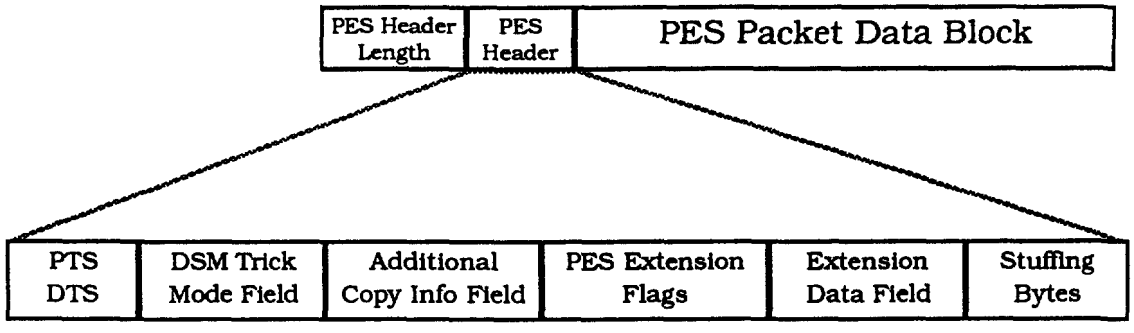

Figure 5.6.3. Organization of PES header.

The PES header Fields are organized according to Fig. 5.6.3 for the GA PES packets for video elementary streams. Most fields require marker bits to be inserted, as described later, in order to avoid the occurrence of long strings of 0's which could resemble a start code.

PTS and DTS

The presentation_time_stamp (PTS) informs the decoder of the intended time of presentation of a presentation unit, and the decoding_time_stamp (PTS) is the intended time of decoding of an access unit. An access unit is an encoded presentation unit. When it is encoded, the PTS refers to the presentation unit corresponding to the first access unit occurring in the packet. If an access unit does not occur in a PES packet, it shall not contain a PTS. Here, a video access unit is said to occur if the first byte of the picture start code is present in the PES packet payload, and an audio access unit occurs if the first byte of the synchronization word of an audio frame is present. Under

normal conditions, the DTS may be derived from the PTS. So, it is not required to encode the DTS. Consequently, encoding the DTS may indicate special decoding requirements to the decoder. Under no circumstance does the DTS occur by itself; it must occur along with the PTS although the converse is not true. The PTS field is organized as shown, if it present without the DTS.

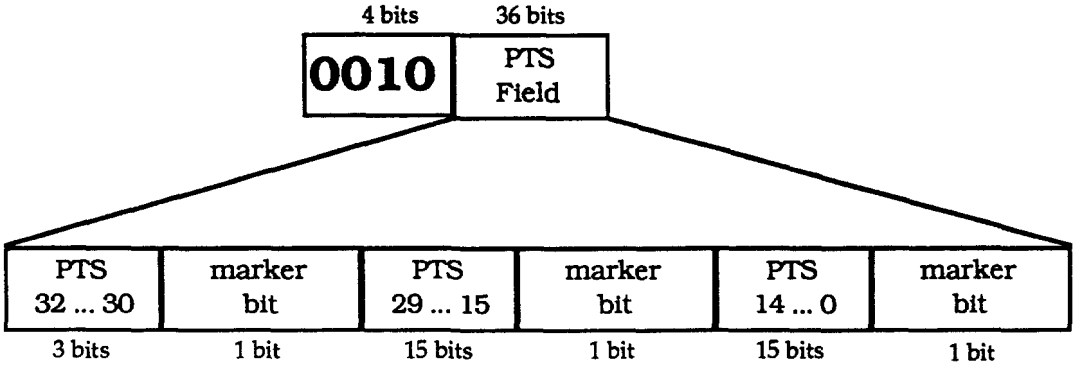

Figure 5.6.4. Organization of the PTS field when only the PTS is encoded.

If both the PTS and DTS are sent, the following organization is required.

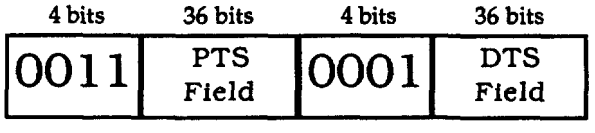

Figure 5.6.5. Organization of the PTS and DTS field when both PTS and DTS are encoded.

Here, the DTS field is defined in the same manner as the PTS field.

DSM Trick Mode Field[5]

The DSM trick mode field is an eight bit field, indicating the nature of the information encoded in the PES packet. The first three bits of this field form the identifier trick_mode_control, indicating the nature of the DSM mode. There are four modes to the DSM, as summarized in the table below.

[5] It is emphasized once again that the ability to deal with trick mode bit streams is not a basic requirement for a GA receiver and this description is here only for completeness. This field is not present in the broadcast GA bit stream. Since the objective in this document is not to describe in detail the DSM mode of operation, all that is presented is the DSM syntax with some description of the respective fields. For more detailed information the reader is directed to the MPEG-2 Systems document.

Value	Description
'000'	Fast Forward
'001'	Slow Motion
'010'	Freeze Frame
'011'	Fast Reverse
'1xx'	Reserved

DSM Trick Modes

Depending on the value of trick_mode_control, a combination of four identifiers are encoded as described in the table below.

Identifier	Description
field_id	This identifier is valid for interlaced pictures only. It is a 2 bit field which identifies how the current frame is to be displayed. '00' -- Display field 1 only '01' -- Display field 2 only '10' -- Display complete frame '11' -- Reserved
frequency_truncation	This 2 bit field indicates the selection of coefficients from the DSM. '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 This field is for informational purposes only. i.e. the DSM may at times send more than the specified number of coefficients and at other times less. However, that information is not normative.
intra_slice_refresh	This 1 bit field indicates that each picture is composed of intra slices with possible gaps between them. The decoder should replace the missing slices by repeating the colocated sites from the previous decoded picture.
field_rep_cntrl	This field indicates how many times the decoder should repeat field #1 as both top and bottom fields alternatively. After field #1 has been displayed, the decoder should repeat field #2 the same number of times. This identifier being set to 0 is equivalent to a freeze frame with field_id being set to '10'.

Fast Forward and Fast Reverse Modes: The format in this case is shown in Fig. 5.6.6. The decoder is told how many coefficients are encoded, how the fields are to be displayed, and how to replace any missing slices in the access units.

trick_mode_control	field_id	intra_slice_refresh	frequency_truncation
3 bits	2 bits	1 bit	2 bits

Fig. 5.6.6. Trick Mode Field in Fast Forward and Fast Reverse Modes

Slow Motion Mode: In this mode, the DSM informs the decoder how many times a particular field is to be repeated. The identifier field_rep_cntrl is encoded as shown.

trick_mode_ control	field_rep_ control
3 bits	5 bits

Fig. 5.6.7. Trick Mode Field in Slow Motion Mode

Freeze Frame Mode: Only the identifier is encoded in this mode. It indicates to the decoder how to display the frozen picture. The field is organized as shown in Figure 5.6.8.

trick_mode_ control	field_id	Reserved
3 bits	5 bits	5 bits

Fig. 5.6.8. Organization of Trick Mode Field for Freeze Frame Mode.

Additional Copy Info

This is a one byte field with a marker bit up front and 7 bits of information. The use of these seven bits is yet to be determined.

PES Extension Flags

The header could contain additional flags if the EXT flag (shown in Fig. 5.6.2) is set. These flags are transmitted in a one byte data field as shown in Fig. 5.6.9.

PES private data flag	pack header field flag	program packet sequence counter flag	STD Buffer Flag	Reserved	PES extension field flag
1 bit	1 bit	1 bit	1 bit	3 bits	1 bit

Figure 5.6.9. Organization of the PES Extension Flags Field.

The flags indicate whether further extensions to the PES header exist. The table below describes the nature of this additional data. As with the flags defined previously, the flag is set to '1' if the header field is present.

Field	General Description
PES_private_data_flag	Indicates whether the PES packet contains private data.
pack_header_field_flag	Indicates whether an MPEG-1 systems pack header or an MPEG-2 program stream pack header is present in the PES Header.
program_packet_sequence_counter_flag	Indicates the presence of a PES packet counter, which allows the decoder to retrieve the packets in the correct order, from the PES/program multiplex.
STD_buffer_flag	Indicates whether the STD_buffer_scale and the STD_buffer_size flags are encoded in the PES header.

PES_extension_field_flag	Indicates the presence of additional data in PES header.

For the GA system, as indicated in the constraint document submitted to MPEG, the flags that are shaded in the table are always set to '0'. It is likely that the EXT flag will set to '0' in the header flags field unless there is information to be transmitted in the PES_extension_field. This field is meant for functions that may have been missed in the initial design specification.

PES Extension Field

This field is shown in Fig. 5.6.10. The length of the PES_extension_field data is given by PES_extension_field_length.

marker bit	PES Extension Field Length	Reserved
1 bit	7 bits	

Figure 5.6.10. Organization of the Extension Field

5.7. Conditional Access

The transport protocol implements functions useful for supporting conditional access. The functionality that is available is flexible and complete in the sense of supporting all transmission aspects of applicable key encryption and descrambling approaches that may be used. Conditional access is also flexible in the sense that it can be exercised on a elementary stream by stream basis, including the ability to selectively scramble bit streams in a program if desired.

A conditional access system operates on the principle of randomizing the transmitted data so that unauthorized decoders cannot decode the signal. Authorized decoders are delivered a "key" which initializes the circuit which inverts the bit randomization. In subsequent discussion, we use the term scrambling to mean the pseudo-random inversion of data bits based on a "key" which is valid for a short time. We use the term encryption to mean the process of transforming the "key" into an encrypted key by a means which protects the key from unauthorized users. From a cryptographic point of the view, this transformation of the key is the only part of the system which protects the data from a highly motivated pirate. The scrambling portion of the process alone, in the absence of key encryption, can be defeated. Conditional Access (CA) is a blanket term for the system which implements the key encryption and distribution. The primary requirements which a scrambling and CA subsystem must meet for digital TV delivery are:

- Protection of programmer's revenues
  - robust against piracy.
- Private encryption system for each program provider.
- Standard consumer instruments
  - no secrets in consumer equipment
- Mobility of consumer equipment
- Consumer equipment should be cost effective

5.7.1. General Description

There are two features of the GA Transport system which support conditional access. The first feature is the two bit transport_scrambling_control field which signals the decoder whether the transport packet was scrambled or not. In the case that it was scrambled, the field identifies which scrambling key was used. As will be shown shortly, the use of two bits in the transport_scrambling_control to define the descrambling process is a necessary and sufficient bound for the key distribution function. The second feature is the ability to insert "private" data at several places in the GA Transport stream. These include entirely private streams and private fields in the adaptation header of the transport bit stream being scrambled. These private fields can be used to transmit the encrypted scrambling key to the decoding device.

The key distribution and usage process is clarified in Fig. 5.7.1. Basically, when the bit stream is scrambled, one descrambling key needs to be in use while the other is being received and decrypted. Two keys are transmitted at any time, with the keys being linked to a transport_scrambling_control value as shown in the figure. The transmission of a key should begin well before it is going to be used, to allow time to decrypt it. Note that this function does not bound the total number of keys that may be used during an entire transmission session.

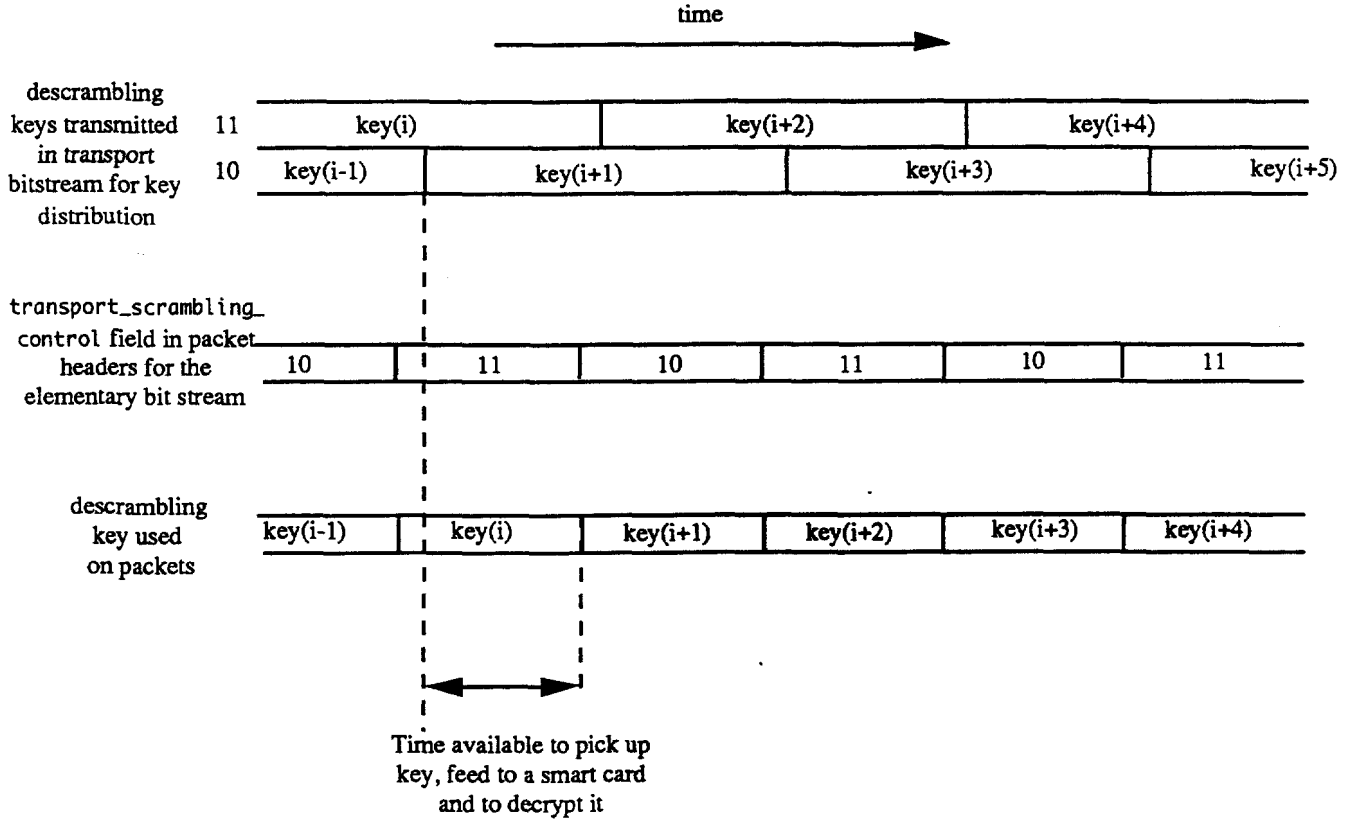

Fig. 5.7.1. Illustration of key distribution and usage process.

As stated previously, the amount of data to be scrambled in a packet is variable depending on the length of the adaptation header. It should be noted that some padding of the adaptation field might be necessary for certain block mode algorithms.

5.7.2. Example of Conditional Access Implementation

In this section, we go through a simple example of a conditional access implementation. Consider the receiver architecture shown in Figure 5.7.2. The high speed manipulations required to implement the descrambling are embedded in the transport demultiplexer, where they are shown as a DES block. Note that other scrambling schemes, such as stream ciphers based on a Pseudo Random Binary Sequence (PRBS), could be employed. The PRBS uses a shift register implementation, where it the initial register value is reset periodically for error robustness. The data security is achieved by the "key" which properly configures the descrambler. This element is delivered to the decoder through an ancillary data service, and is encrypted by the conditional access administrator. In the equipment at the customer's premises, the key is decrypted within the outboard Smart-Card. The Smart-Card interface will conform to ISO standard ISO-7816, which permits a variety of implementations and conditional access solutions.

The Smart-Card maintains a short list of two key's, commonly denoted as the "odd" key and the "even" key. The proper key to be used to descramble is signaled in the transport prefix in the transport_scrambling_control field. The transport_scrambling_control takes on one of the following 4 states:

transport_scrambling_control	*Description*
00	Not Scrambled
01	Reserved
10	"even" key
11	"odd" key

The scrambling in this example is a block cipher called the "Electronic Code Book" mode of the Digital Encryption Standard (DES). For DES, the key is a 56 bit binary sequence. The television electronics are "standard", while the Smart-Card implementation is proprietary to the service provider.

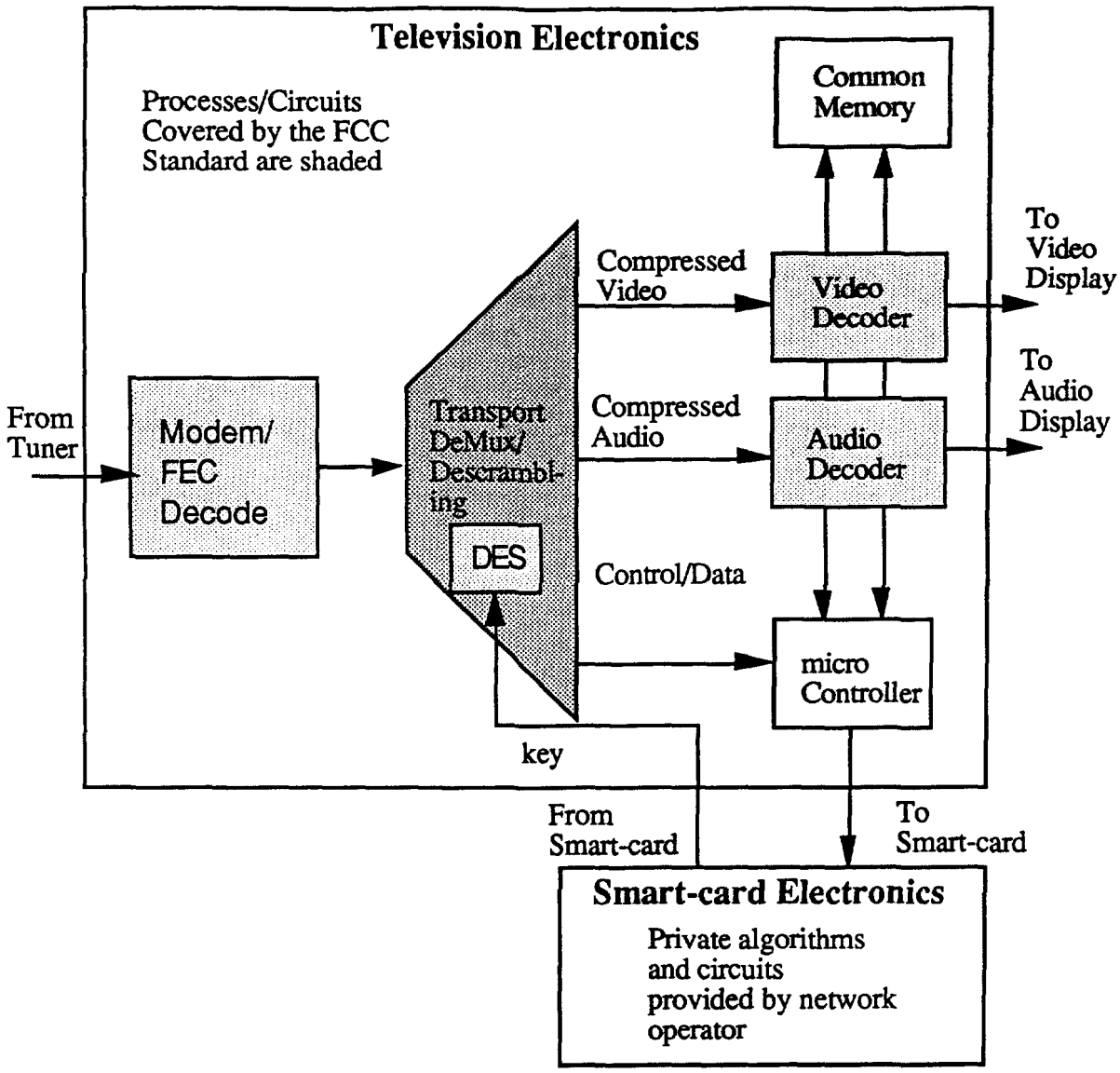

Fig. 5.7.2. Decoder with Example Conditional Access Implementation

There is significant flexibility for multihop encryption and nesting of encryption systems to ensure data security at every point in the transmission chain. Fig. 5.7.3 illustrates nested encryption systems, where the service provider provides authorization, keys and the scrambled data to the end user through Encryption System A. During transit, System B is employed by the carrier to protect the data while in the communications network. In fact, there are two implementation choices even for this segment of the delivery system. The provider can use both a second layer of scrambling in conjunction with a different authorization and key distribution. (This system need not comply with the transmission "Standard's" method.) Alternatively, System B could simply encrypt or scramble the encrypted keys distributed by System A, without the requirement of scrambling the actual

service data. The main appeal of this method is that it is a low bandwidth/complexity solution, while its drawback is that it requires some knowledge of the System A key distribution method.

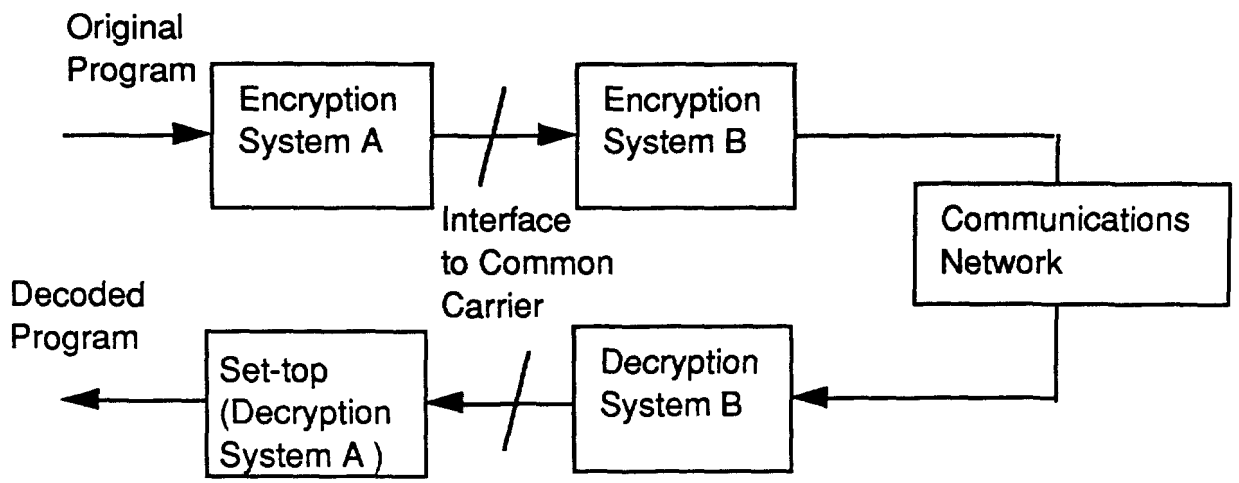

Figure 5.7.3. Nested encryption systems.

In Fig. 5.7.4, the two Encryption Systems are connected in series. System B is again used to protect the integrity of the data while in the communications network. System A is used by the local affiliate or cable company to authorize reception of the service within its own service area.

In fact, the two topologies discussed above can be combined, where either System A or System B could be a Nested or Series configuration. The number of nesting/series combinations can be arbitrarily large.

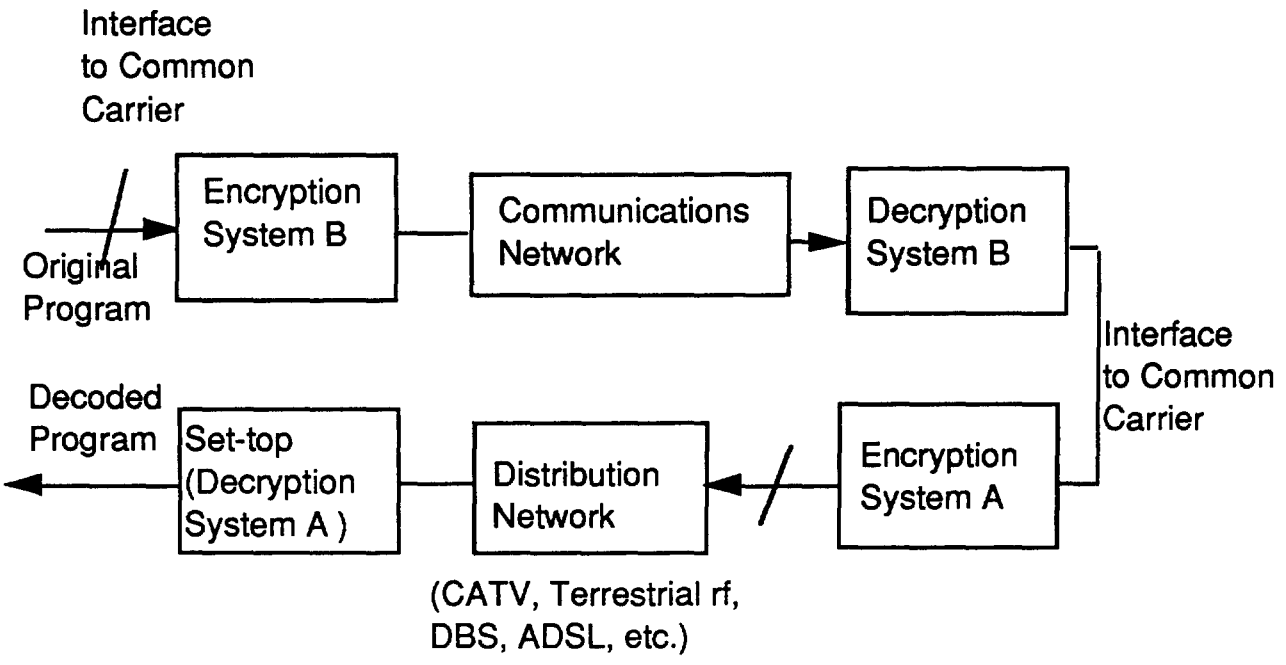

Figure 5.7.4. Series encryption systems.

5.8. Local Program Insertion

The Grand Alliance Transport supports insertion of programs and commercials, by use of flags and features dedicated to this purpose in the transport packets Adaptation Header. The use of these syntax elements will need to be within some imposed constraints to ensure proper operation of the video decoders. Furthermore, there will be some constraints on some of current common practices, imposed not by the GA transport, but rather by virtue of the compressed digital data format.

The functionality of program insertion and switching of channels at a broadcast head-end are quite similar, the difference being in the time constants involved in the splicing process, and also in the fact that in the program insertion the bit stream is switched back to the old program after insertion is complete, while in the channel switching case one most likely switches over to yet another program at the end of the splice. There are other detailed issues related to the hardware implementation that may differ for these two cases, including input source devices and buffering requirements. For example, if program insertion is to take place on a bit stream obtained directly from a network feed, and if the network feed does not include place-holders for program insertion, the input program transport stream will need to be buffered up for the duration of the program insertion. If the program is obtained from a local device, e.g., a video server or a tape machine, it may be possible to pause the input process for the duration of the program insertion. Neither of these is an issue for channel switching.

5.8.1. Systems level view

There are two layers of processing functionality to address when doing program insertion. The lower layer functionality is related to splicing of transport bit streams for the individual elements of the program. The higher level functionality is related to coordination of this process between the different elementary bit streams which make up the program transport stream. Fig. 5.8.1 illustrates the correct approach to implement program insertion.

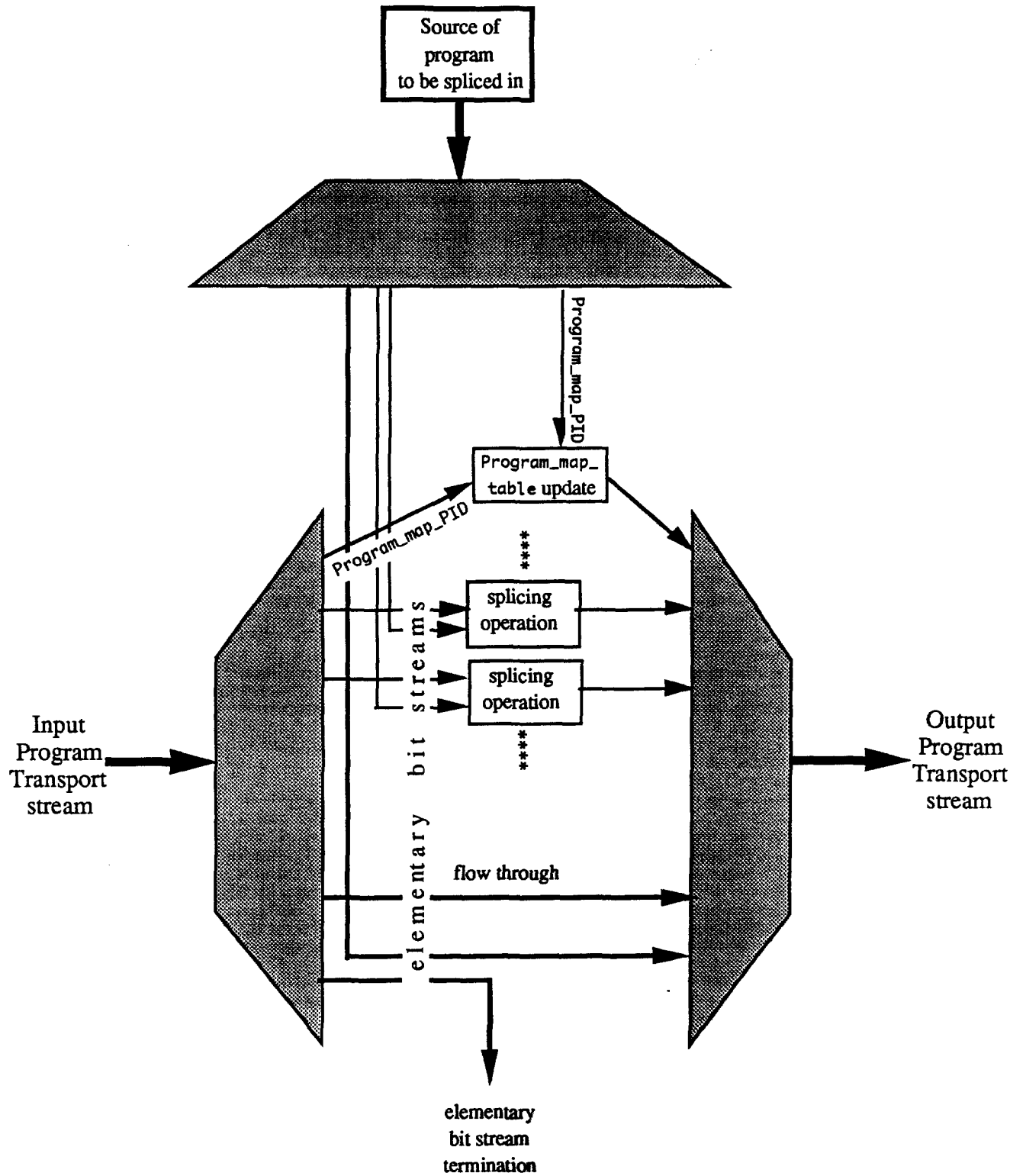

Figure 5.8.1 Example Program Insertion Architecture