Final draft ETSI EN 300 743 V1.2.1 (2002-06)

European Standard (Telecommunications series)

Digital Video Broadcasting (DVB); Subtitling systems

European Broadcasting Union Union Européenne de Radio-Télévision

EBU·UER

DVB Digital Video Broadcasting

ETSI

Reference

REN/JTC-DVB-126

Keywords

broadcasting, digital, DVB, TV, video

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.fr

Copyright Notification

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This European Standard (Telecommunications series) has been produced by Joint Technical Committee (JTC) of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI), and is now submitted for the ETSI standards One-step Approval Procedure.

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

Proposed national transposition dates	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

1 Scope

The present document specifies the method by which subtitles, logos and other graphical elements may be coded and carried in DVB bitstreams. The system applies Colour Look-Up Tables (CLUTs) to define the colours of the graphical elements. The transport of the coded graphical elements is based on the MPEG-2 system described in ISO/IEC 13818-1 [1].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] ISO/IEC 13818-1: "Information technology - Generic coding of moving pictures and associated audio information: Systems ".

[2] ETSI EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[3] ITU-R Recommendation BT.601-3: "Encoding parameters of digital television for studios".

[4] ITU-R Recommendation BT.656-4: "Interfaces for digital component video signals in 525-line and 625-line television systems operating at the 4:2:2 level of ITU-R Recommendation BT.601 (Part A)".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

ancillary page: means of conveying subtitle elements that may be shared by multiple **subtitle services** within a **subtitle stream**

NOTE: For example, an ancillary page can be used to carry logos or character glyphs.

Colour Look-Up Table (CLUT): look-up table applied in each region for translating the objects' pseudo-colours into the correct colours to be displayed

CLUT-family: Family of **CLUT**s which may consist of:

- one CLUT with 4 entries;
- one CLUT with 16 entries;
- one CLUT with 256 entries.

A CLUT-family is used in a region to define colours for decoders with different rendering capabilities.

NOTE: Three CLUTs are defined to allow flexibility in the decoder design. Not all decoders may support a CLUT with 256 entries, some may provide sixteen or even only four entries. A palette of four colours might be enough for graphics that are basically monochrome, like very simple subtitles, while a palette of sixteen colours allows for cartoon-like coloured objects or coloured subtitles with antialiassed edges.

composition page: means of conveying subtitle elements for one specific **subtitle service**

display set: set of **subtitle segments** of a specific **subtitle service** to which the same **PTS** value is associated

epoch: period of time for which the decoder maintains an invariant memory layout

NOTE: This layout may be altered by resets to the decoder state caused by receiving page composition segments with page state = "mode change". The end of an epoch therefore signals the "death" of a **page**. The epoch may, if so desired, be considered to be the highest level data structure in DVB subtitling.

object: graphical unit that can be positioned within a **region**; examples of an object include a character glyph, a logo, a map, etc.

NOTE: Each object has its own object_id.

Packet IDentifier (PID): Transport packet identifier

NOTE: See ISO/IEC 13818-1 [1].

page: Set of subtitles for a **subtitle service** during a certain period. A page consists of one or more **page instances**. Each page update or refresh will result in a new page instance. A page contains a number of **regions**, and in each region there may be a number of **objects**.

page composition: composition (use and positioning) of **regions** that may be displayed within the **page**

NOTE: At any new **page instance** the page composition may change; for example, some regions may not yet or no longer be displayed. At any one time, only one page composition can be active for displaying.

page instance: period of time during which that **page** does not change i.e. there is no change to the **page composition**, to any **region composition**, to any **object** within a **region** or any applicable **CLUT**

NOTE: Typically, a new page instance is defined by the **PTS** of a **display set**.

PES packet: See ISO/IEC 13818-1 [1].

pixel-data: string of data bytes that contains, in coded form, the representation of a graphical object

Presentation Time Stamp (PTS): See ISO/IEC 13818-1 [1].

region: rectangular area on the **page** in which **objects** can be positioned

NOTE: Regions may be shared by multiple **subtitling services** within the same **subtitle stream**. Objects that share one or more horizontal scan lines on the screen are included in the same region.

region composition: composition (use and positioning) of **objects** within a **region**

subtitle element: subtitle data used within a **page composition** and contained within a **subtitle segment**

NOTE: **Regions**, **region composition**s, **CLUT**s and **object** data are examples of subtitle elements.

subtitle segment: basic syntactical element of a **subtitle stream**

subtitle service: Service that provides subtitling for a program for a certain purpose, such as subtitles in a specific language or for the hard of hearing. A subtitle service is displayed as a series of one or more **pages.**

NOTE: Typically, a subtitle service meets a single communication requirement (e.g. the graphics to provide subtitles in one language for one program).

subtitle stream: Stream of **subtitling segment**s carried in **transport packet**s identified by the same **PID**. A subtitle stream contains one or more **subtitle services**.

transport packet: See ISO/IEC 13818-1 [1].

transport stream: A stream of **transport packet**s carrying one or more MPEG programs

NOTE: See ISO/IEC 13818-1 [1].

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

bslbf	bit string, left bit first
Cb	Chrominance value, as defined in ITU-R Recommendation BT.601-3 [3], clause 7.2.3
CLUT	Colour Look-Up Table
Cr	Chrominance value, as defined in ITU-R Recommendation BT.601-3 [3], see clause 7.2.3
DVB	Digital Video Broadcasting
IRD	Integrated Receiver Decoder
MPEG	Moving Pictures Experts Group (WG11 in SC 29 of JTC1 of ISO/IEC)
PCR	Programme Clock Reference
PCS	Page Composition Segment
PES	Packetized Elementary Stream, as defined in ISO/IEC 13818-1 [1]
PID	Packet IDentifier, as defined in ISO/IEC 13818-1 [1]
PID	transport packet identifier, as defined in ISO/IEC 13818-1 [1]
PMT	Program Map Table, as defined in ISO/IEC 13818-1 [1]
PTS	Presentation Time Stamp, as defined in ISO/IEC 13818-1 [1]
RCS	Region Composition Segment
ROM	Read-Only Memory
T	Transparency value
TS	Transport Stream, as defined in ISO/IEC 13818-1 [1]
uimsbf	unsigned integer, most significant bit first
Y	luminance value, as defined in ITU-R Recommendation BT.601-3 [3], see clause 7.2.3

4 Introduction to DVB subtitling system

The present document specifies the DVB subtitling system for the transport and coding of subtitles.

4.1 Overview

The DVB subtitling system defined in the present document provides a syntax for decoding **subtitle streams**. A subtitle stream conveys one or more **subtitle services**; each service containing the textual and/or graphical information needed to provide subtitles or glyphs for a particular purpose. Separate subtitle services may be used, for example, to convey subtitles in several languages.

Each subtitle service displays its information in a sequence of so-called **pages** that are intended to be overlayed on the associated video image. A subtitle page contains one or more **regions**, each region being a rectangular area with a specified set of attributes. These attributes include a region identifier, the horizontal and vertical size, pixel depth and background colour. A region is used as the background structure into which graphical **objects** are placed. An object may represents a character, a word, a line of text or an entire sentence; it might also define a logo or icon.

The use and positioning of objects within a region is defined by the **region composition segment**.

The use and positioning of regions within a page is defined by the **page composition segment**, in which a list of displayed regions is provided, each with their own spatial position. A page composition need not change when objects are added to or removed from a region. Furthermore regions may be declared but not used. By way of example one region can be used to display multiple subtitle fragments, as depicted in figure 1. First the text "Shall we?" is displayed in the region; subsequently this text is removed and the new text "Said the fly on the mirror" is displayed. It is possible to use more than one region at the same time; for example one region could be used to display subtitles on the bottom of the screen, while another one might be used to display a logo somewhere else on the screen.

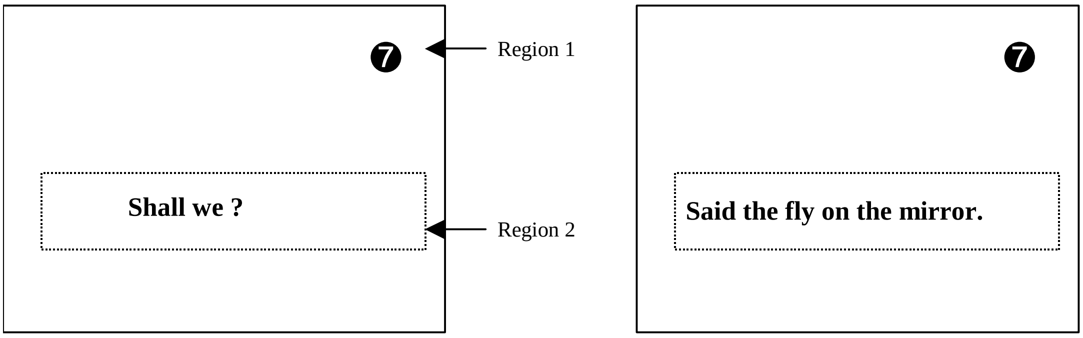

Figure 1: Two regions overlayed on top of video; one with a logo and another one with subtitles. The subtitles are positioned within the same region.

A DVB subtitle stream is carried in **PES packets** and the timing of their presentation is defined by the **PTS** in the PES header. Upon reception and decoding of the subtitle data for a page (such as the page composition, the region composition, the objects to be used and any other associated data) the page contents are displayed at the time indicated by the associated PTS. When objects are to be added, the decoder receives region composition updates and the data for the new objects, and will display the updated page at the time indicated by the new PTS. At the page update only page differences need be provided. To improve random access to DVB subtitling, a page refresh is also possible. At page refresh all the subtitling data needed to display a page is provided. Each page update or refresh will result in a new page instance. A page ceases to exist after the time-out of the page, or when a new page is defined.

To provide efficient use of display memory in the decoder the DVB subtitling system uses region based graphics with indexed pixel colours. Pixel depths of 2, 4 and 8-bits are supported allowing up to 4, 16 or 256 different pixel codes to be used in each region. Each region is associated with a single **CLUT** family to define the colour and transparency for each of the pixel codes. In most cases, one CLUT is sufficient to present correctly the colours of all objects in a region, but if it is not enough, the objects can be split horizontally into smaller objects across separate vertically adjacent regions with one CLUT each.

The use of CLUTs allows colour schemes to be dynamic. The colours that correspond to the entries within the region can be redefined at any suitable time, for instance in case of a CLUT with four entries from a black-grey-white scheme to a blue-grey-yellow scheme. Furthermore, a graphical unit may be divided into several regions each using a different CLUT, i.e. a different colour scheme may be applied in each of the regions. At the discretion of the encoder, objects designed for displays supporting 16 or 256 colours can be decoded into displays supporting fewer colours. A quantization algorithm is defined to ensure that the result of this process can be predicted by the originator. Use of this feature allows a single data stream to be decoded by a population of decoders with mixed, and possibly evolving, capabilities.

A subtitle stream may transport multiple subtitle service components. In this case the pages of one particular subtitle service are all identified by the same page-id value. This value is used when transporting the subtitling data so as to provide a mechanism to retrieve the data that is specific to a service from a subtitle stream. The subtitling system allows sharing of subtitling data between services within the same subtitle stream. A frequent *and often preferred* method is to convey the distinct services in different streams on separate **PID**s. In either case the appropriate PID, language and page-ids will be signalled in the program map table (**PMT)** for the television service of interest (language and page-id in the subtitling descriptor defined in DVB-SI [2]). These two approaches are illustrated in figure 2.

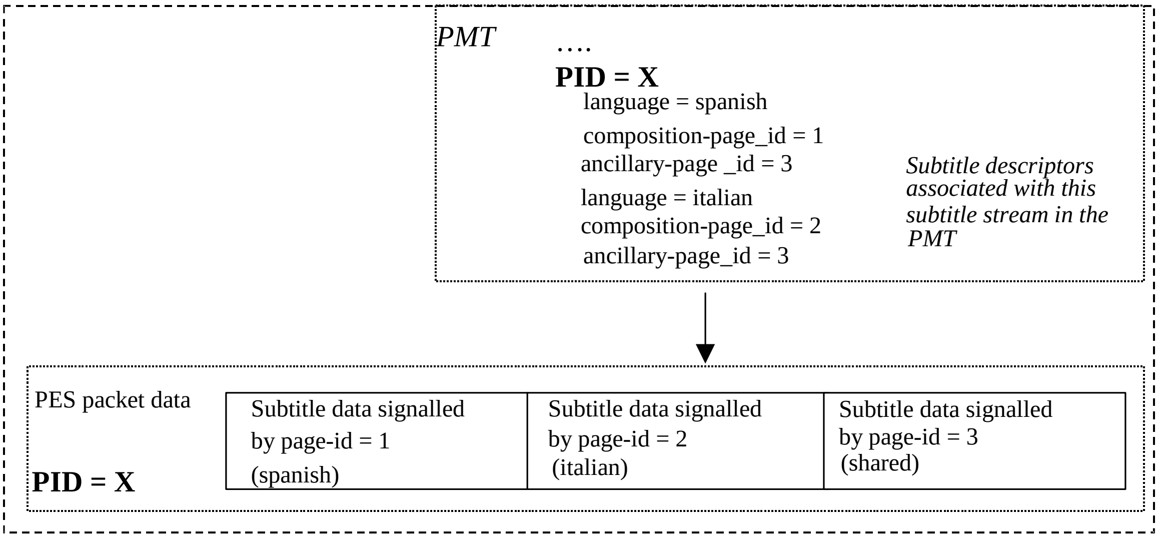
PMT
….
PID = X
language = spanish
composition-page_id = 1
ancillary-page _id = 3
language = italian
composition-page_id = 2
ancillary-page_id = 3
Subtitle descriptors associated with this subtitle stream in the PMT
PES packet data
PID = X
Subtitle data signalled by page-id = 1 (spanish)
Subtitle data signalled by page-id = 2 (italian)
Subtitle data signalled by page-id = 3 (shared)

Figure 2a: Example of use of different page_ids to dictinguish between different subtitle languages for the same service (shown with a shared ancillary page)

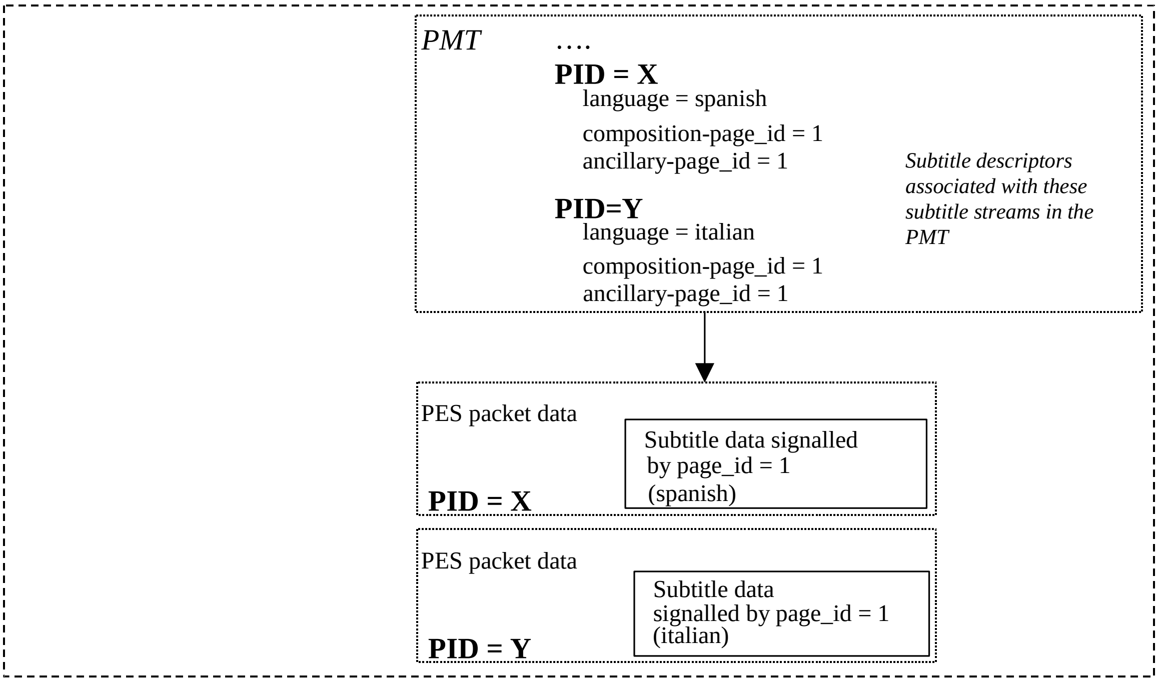

Figure 2b: Example of use of PIDs to distinguish between different subtitle languages for the same service (shown with no ancillary page)

Figure 2: Example of two ways of conveying dual language subtitles (one using shared data)

In summary, the DVB subtitling system provides a number of techniques that allow efficient transmission of the subtitling data:

- objects that occur more than once within a region need only be transmitted once, and then positioned multiple times within the region;
- objects used in more than one subtitle service need only be transmitted once;
- pixel data within objects are compressed using run-length coding;

- where the gamut of colours required for part of a graphical object is suitably limited, that part can be coded using a smaller number of bits per pixel and a map table. For example, an 8-bit per pixel graphical object may contain areas coded as 4 or 2-bits per pixel each preceded by a map table to map the 16 or 4 colours used onto the 256 colour set of the region. Similarly, a 4-bit per pixel object may contain areas coded as 2-bits per pixel;

- colour definitions can be coded using either 16 or 32-bits per CLUT entry. This provides a trade-off between colour accuracy and transmission bandwidth;

- only those CLUT values to be used need be transmitted.

The above features are fully supported within the DVB subtitling system.

In addition, functionality is provided to allow more efficient operation where there are private agreements between the data provider and the manufacturer of the decoder:

- objects resident in ROM in the decoder can be referenced;

- character codes, or strings of character codes, can be used instead of objects with the graphical representation of the character(s). This requires the decoder to be able to generate glyphs for these codes.

The private agreements required to enable these features are beyond the scope the present document.

4.2 Data hierarchy and terminology

The basic "building block" of a DVB subtitle stream is the **subtitling segment**. These segments are carried in **PES packets**, which are in turn carried by **transport packets**. The number of segments carried in a PES packet is only limited by the maximum length of a PES packet, as defined by ISO/IEC 13818-1 [1].

A subtitle stream shall be carried in transport packets identified by the same PID. A single subtitle stream can carry several different subtitle services. All the subtitling data required for a subtitle service shall be carried by a single subtitle stream. The different subtitle services can be subtitles in different languages for a common program. Alternatively, they could in principle be for different programs (provided that the programs share a common **PCR**).

Different subtitle services can also be supplied to address different display characteristics or to address special needs. For instance:

- different subtitle services might be provided for 4:3 and 16:9 aspect ratio displays;

- subtitle services might be provided specifically for viewers with impaired hearing. These may include graphical representations of sounds.

Within a subtitle stream, a page id value is assigned to each segment. Segments can either contain data specific for one subtitle service, or data that is to be shared by more than one subtitle service. The data for a subtitle service shall be carried in segments identified by at most two different page id values:

- one page id value signalling segments with data specific for that subtitle service; the use of this type of data is mandatory;

- one page id value signalling segments with data that may be shared by multiple subtitle services; the use of this type of data is optional.

For each subtitle service a subtitling_descriptor as defined in EN 300 468 [2] signals the page id values of the segments needed to decode that subtitle service. The subtitling descriptor shall be included in the PMT of the program and shall be associated to the PID that conveys the subtitle stream. In the subtitling descriptor the page id of segments with data specific to that service is referred to as the **composition page id**, while the page id of segments with shared data is referred to as the **ancillary page id**. For example, the ancillary page id might signal segments carrying a logo that is common to subtitles in several different languages.

The **PTS** in the PES packet header provides presentation timing information for the subtitling data, and is associated with the subtitle data in all segments carried in that PES packet. The PTS defines the time at which the associated decoded segments should be presented. This may include removal of subtitles, for example when an entire region is removed or when all objects in a region are removed. There may be two or more PES packets with the same PTS value, for example when it is not possible or desirable to include all segments associated to the same PTS in one PES packet.

The complete set of segments of a subtitle service that are associated to the same PTS is referred to as a **display set**. The last segment of a display set shall be followed by an "end_of-display-set segment", which signals that no more subtitling data associated to a certain PTS is needed for that service before decoding can commence. The display sets shall be delivered in their correct presentation-order, and the PTSs of subsequent display sets shall differ by more than one video frame period.

For carriage of multiple types of subtitling data, several segment types are defined, in particular:

- page composition segment; the decoding of a subtitle service will typically result in the display of subsequent pages, each consisting of one or more regions; the page composition segment carries information on the page composition, such as the list of included regions, the spatial position of each region, some time-out information for the page and the state of the page;
- region composition segment; in each region typically one or more objects are positioned, while using one specific CLUT, identified by a CLUT-id; the region composition segment carries information on the region composition and on region attributes, such as the horizontal and vertical size, the background colour, the pixel depth of the region, which CLUT is used and a list of included objects with their position within the region;
- CLUT definition segment; the CLUT definition segment contains information on a specific CLUT, identified by a CLUT-id, such as the colours used for a CLUT entry;
- object data segment; the object data segment carries information on a specific object; there are two types of objects, graphical objects and text objects. An object data segment with a graphical object contains run-length encoded bitmap colours, while a text object carries a string of one character codes;
- end of display set segment; the end of display set segment contains no internal information, but is used to signal explicitly that no more segments need to be received before the decoding of the current display set can commence.

The page id value of a segment containing data for a subtitle service shall be equal either to the value of the composition_page_id or the ancillary_page_id provided in the subtitle descriptor. Page compositions are not shared by multiple subtitle services; consequently, the page id of each page composition segment shall be equal to the composition_page_id value.

In summary, the data hierarchy is:

- Transport Stream (TS);
- transport packets with the same PID;
- PES packets, with PTSs providing timing information;
- subtitle service;
- segments signalled by the composition page id and optionally the ancillary page id;
- subtitle data, containing information on page composition, region composition, CLUTs, objects and end of display set.

4.3 Temporal hierarchy and terminology

At the segment level in the data hierarchy there is also a temporal hierarchy. The highest level is the epoch; in an epoch the page composition and the region composition may change - for example objects and regions may be added or removed. The concept of an epoch is analogous to that of an MPEG video sequence. No decoder state is preserved from one epoch to the next.

An epoch is a sequence of one or more page instances. Each page instance is a completed screen of graphics. Consecutive page instances may differ little (e.g. by a single word when stenographic subtitling is being used) or may be completely different. The set of segments needed to decode a new page instance is called a display set.

Within a display set the sequence of segments (when present) is:

- page composition;
- region composition;

- CLUT definition;
- object data;
- end of display set segment.

All segments signalled by the composition page id value shall be delivered before any segment signalled by the ancillary page id value. The ancillary page id value shall not signal page composition segments and region composition segments

5 Subtitle decoder model

The subtitle decoder model is an abstraction of the processing required for the decoding of a subtitle service within a subtitle stream. The main purpose of this model is to define requirements for compliant subtitling streams. The following figure shows the prototypical model of a subtitling decoder.

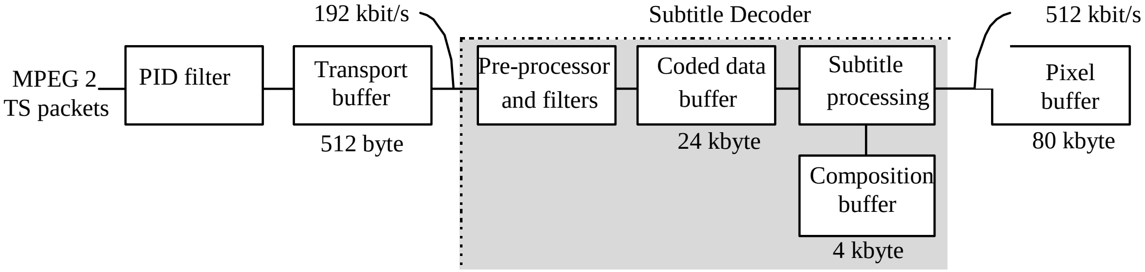

Figure 3: Subtitle decoder model

The input to the subtitle decoder model is an MPEG-2 Transport Stream (TS). After a selection process based on PID value, complete MPEG-2 Transport Stream packets containing the subtitle stream enter a transport buffer with a size of 512 byte. When there is data in the transport buffer, data is removed from this buffer at a rate of 192 kbit/s. When no data is present, this data rate equals zero.

The transport packets from the transport buffer are processed by stripping off the headers of the transport packets and of the PES packets. The Presentation Time Stamp (PTS) values are passed on to the next stages of the subtitling processing. In the pre-processor, the segments required for the selected subtitle service are filtered from the subtitle stream. Hence, the output of the pre-processor is a stream of subtitling segments which are filtered based on the page_id values signalled in the subtitling descriptor.

The selected segments enter into a coded data buffer which has a size of 24 kbyte. Only complete segments are removed from this buffer by the subtitle decoder. The removal and decoding of the segments is instantaneous (i.e. it takes zero time). If a segment produces pixel data, the subtitle decoder stops removing segments from the coded data buffer until all pixels have been transferred to the pixel buffer. The pixel data of objects that are used more than once, is transferred separately for each use. The data rate for the transport of pixel data into the pixel buffer is 512 kbit/s.

The data needed for the composition of the subtitles, such as the page composition, the region composition and the CLUTs are stored in the composition buffer, which has a size of 4 kbyte.

5.1 Decoder temporal model

The requirements for memory usage in the subtitle decoder model depends on the size and colour depth of the applied regions in the page. A complete description of the memory usage of the decoder shall be delivered at the start of each epoch. Hence, epoch boundaries provide guaranteed service acquisition points. Epoch boundaries are signalled by page composition segments with a page state of type "mode change".

The pixel buffer and the composition buffer hold the state of the subtitling decoder. The epoch for which this state is defined is between Page Composition Segments (PCSs) with page state of "mode change". When a PCS with page state of type "mode change" is received by a decoder all memory allocations implied by previous segments are discarded i.e. the decoder state is reset.

All the regions to be used in an epoch shall be introduced by the Region Composition Segments (RCSs) in the display set that accompanies the PCS with page state of "mode change" (i.e. the first display set of the epoch). This requirement allows a decoder to plan all of its pixel buffer allocations before any object data is written to the buffers. Similarly, all of the CLUT entries to be used during the epoch shall be introduced in this first display set. Subsequent segments can modify the values held in the pixel buffer and composition buffer but may not alter the quantity of memory required.

5.1.1 Service acquisition

The other allowed values of page state are "acquisition point" and "normal case". Each such "acquisition point" and "normal case" results in a new page instance. The "acquisition point" state (like the "mode change" state) indicates that a complete description of the memory use of the decoder is being broadcast. However, the memory usage shall not change. Decoders that have already acquired the service are only required to look for development of the page (e.g. new objects to be displayed). Re-decoding of previously received segments is optional. Decoders trying to acquire the service can treat a page state of "acquisition point" as if it were "mode change".

Use of the page state of "mode change" may require the decoder to remove the graphic display for a short period while the decoder reallocates its memory use. The "acquisition point" state should not cause any disruption of the display. Hence it is expected that the "mode change" state will be used infrequently (e.g. at the start of a program, or when there are significant changes in the graphic display). The "acquisition point" state will be used every few seconds to enable rapid service acquisition by decoders.

A page state of "normal case" indicates that the set of RCS may not be complete (the set is only required to include the regions whose region data structures - bitmap or CLUT family - are to be modified in this display set). There is no requirement on decoders to attempt service acquisition at a "normal case" display set.

A display set is not required to contain a page composition segment. Within the same page composition for example a region composition may change. If no page composition segment is contained, the page state is not signalled; however, such display set will result in a new page instance equivalent to a "page update".

5.1.2 Presentation Time Stamps (PTS)

Subtitling segments are encapsulated in PES packets, partly because of their capability to carry a Presentation Time Stamp (PTS) for the subtitling data.

Unlike video pictures, subtitles have no innate refresh rate. Therefore all subtitle data are associated with a PTS to control when the decoded subtitle is displayed. Each PES header shall carry a PTS, associated with all the subtitle data contained within that PES packet. Consequently, for any subtitling service there can be at most one display set in each PES packet. However, the PES packet can contain concurrent display sets for a number of different subtitle services, all sharing the same presentation time. It is possible that segments sharing the same PTS have to be split over more than one PES packet (e.g. because of the 64 kbyte limit on PES packet length). In this case more than one PES packet will have the same PTS value.

In summary, all of the segments of a single display set shall be carried in one (or more) PES packets that have the same PTS value.

For each subtitling service all data of a display set shall be delivered within the constraints defined for the subtitle decoder model, so as to allow practical decoders sufficient headroom to present the decoded data by the time indicated by the PTS.

NOTE: There may be times when, due for example to slightly late arrival of a complete display set or to slow rendering in the decoder, the correct time to present a subtitle (i.e. when PTS= local system clock derived from the PCR) has passed. (Late arrival can result from injudicious throttling of the bit-rate assigned to a subtitling stream at some point in the distribution network.) Under such conditions decoder designers should recognize that it is almost always better to display a late subtitle than to discard it.

5.1.3 Page composition

The Page Composition Segment (PCS) carries a list of zero or more regions. This list defines the set of regions that will be displayed at the time defined by the associated PTS. In the subtitle decoder model, the display instantly switches from any previously existing set of visible regions to the newly defined set.

The PCS may be followed by zero or more Region Composition Segments (RCS). The region list in the PCS may be quite different from the set of RCSs that follow, in particular when some of the regions are initially not displayed.

The PCS provides the list of regions with their spatial positions on the screen. The vertical position of the regions shall be defined such that regions do not share any horizontal scan lines on the screen. A region therefore monopolizes any part of the scan lines that it occupies; no two regions can be presented horizontally next to each other.

5.1.4 Region composition

A complete set of Region Composition Segments (RCS) shall be present in the display set that follows a PCS with page state of "mode change" or "acquisition point" as this is the process that introduces regions and allocates memory for them. Display sets that represent a page update are only required to contain the data to be modified.

Once introduced the memory "foot print" of a region shall remain fixed for the remainder of the epoch. Therefore the following attributes of the region shall not change within an epoch:

- width;
- height;
- depth;
- region_level_of_compatibility;
- CLUT_id.

Other attributes of the region specified in the RCS are the region_fill_flag and the region_n-bit_pixel_code, specifying the background colour of the region. When the region_fill_flag is set the first graphics operation performed on a region should be to colour all pixels in the region with the colour indicated by the region_n-bit_pixel_code. The value of the region_n-bit_pixel_code shall not change in RCS where the region_fill_flag is not set. This allows decoders that have already acquired the subtitling service to ignore the region_n-bit_pixel_code when the region_fill_flag is not set. A decoder in the process of acquiring the service can rely on the region_n-bit_pixel_code being the current region fill colour regardless of the state of region_fill_flag.

There is no requirement for a region to be initialized by filling it with the background colour when the region is introduced at the start of the epoch. This allows the rendering load to be deferred until the region is included in the region list of the PCS, indicating that presentation of the region is required. In the limiting case, the region need never be filled with the background colour. For example, if the region is completely covered with objects.

5.1.5 Points to note

- At the start of an epoch the display set shall include a complete set of RCSs for all the regions that will be used during that epoch. The PCS shall only list the subset of those regions that presented at the start of the epoch. In the limiting case any PCS may list zero visible regions.
- An RCS shall be present in a display set if the region is to be modified. However, the RCS is not required to be in the PCS region list. This allows regions to be modified while they are not visible.
- An RCS may be present in a display set even if they are not being modified. For example, a broadcaster may choose to broadcast a complete list of RCSs in every display set.
- A decoder shall inspect every RCS in the display set to determine if the region is to be modified, for example, which pixel buffer modifications are required or where there is a modification to the associated CLUT family. It is sufficient for the decoder to inspect the RCS version number to determine if a region requires modification. There are three possible causes of modification, any or all of which may cause the modification:
    - region fill flag set;
    - CLUT contents modification;
    - a non-zero length object list.

5.2 Buffer memory model

A page composition segment with the page state of type "mode change" destroys all previous pixel buffer and composition buffer allocations by erasing the contents of the buffers.

Various processes, as detailed in the following clauses, allocate memory from the pixel and composition buffers. These allocations persist until the next page composition segment with page state of type "mode change".

There is no mechanism to partially re-allocate memory within an epoch. During an epoch, the memory allocation in the pixel buffer remains the same.

5.2.1 Pixel buffer memory

The pixel buffer in the subtitle decoder has a size of 80 kbyte. The pixel buffer shall never overflow. Up to 60 kbyte (i.e. 75 %) is assigned for active display. The remaining capacity is assigned for future display. The subtitle decoder model assumes that all regions used during an epoch are stored in the pixel buffer and defines the following memory allocation requirement for a region in the pixel buffer:

$$\text{region\_bits} = \text{region\_width} \times \text{region\_height} \times \text{region\_depth}$$

where region_depth is the region's pixel depth in bits specified in the RCS. A practical implementation of a subtitle decoder may require more memory to store each region. Any such implementation dependent overhead is not taken into account by the subtitle decoder model.

During an epoch, the occupancy of the pixel buffer is the sum of the region_bits of all regions used in that epoch.

5.2.2 Region memory

The pixel buffer memory for a region is allocated at the start of an epoch. This memory allocation is retained until a page composition segment with page state of "mode change" destroys all memory allocations.

5.2.3 Composition buffer memory

The composition buffer contains all information on page composition, region composition and CLUT definition.

The number of bytes defined by the subtitle decoder model for composition buffer memory allocation is given below:

Page composition except region list	4 bytes
per included region	6 bytes
Region composition except object list	12 bytes
per included object	8 bytes
CLUT definition excluding entries	4 bytes
per non full range entry	4 bytes
per full range entry	6 bytes

5.3 Cumulative display construction

During an epoch the region modifications defined in display sets accumulate in the pixel buffer, but without any impact on the memory allocation for each region

5.4 Decoder rendering bandwidth model

The rendering bandwidth into the pixel buffer is specified as 512 kbit/s. The subtitle decoder model assumes 100 % efficient memory operations. So, when 10 pixel × 10 pixel object is rendered in a region with a 4-bit pixel depth, 400-bit operations are consumed.

The 512 kbit/s budget comprises all modifications to the pixel buffer. Certain decoder architectures may require a different number of memory operations. For example, certain architectures may require read, modify, write operation on several bytes to modify a single pixel. These implementation dependent issues are beyond the scope of the subtitle decoder model and are to be compensated for by the decoder designer.

5.4.1 Page erasure

A page erasure occurs at a page time-out. Page erasure does not imply any modifications to the pixel buffer. So, page erasure does not impact rendering in the subtitle decoder model.

5.4.2 Region move or change in visibility

Regions can be repositioned by altering the specification of their position in the region list in the PCS. The computational load for doing this may vary greatly depending on the implementation of the graphics system. However, the subtitle decoder model is region based so the model assumes no rendering burden associated with a region move.

Similarly, the visibility of a region can be changed by including it in or excluding it from the PCS region list. As above, the subtitle decoder model assumes that no rendering is associated with modifying the PCS region list.

5.4.3 Region fill

Setting the region fill flag instructs that the region is to be completely re-drawn with the defined fill colour. For example, filling a 128 pixel × 100 pixel 4-bit deep region will consume 51 200 -bit operations, which will take 0,1 s with the rendering bandwidth of 512 kbit/s. Where the region fill flag is set, the region fill in the subtitle decoder model happens before any objects are rendered into the region.

Regions are only filled when the region fill flag is set. There is no fill operation when a region is introduced at the start of an epoch. This allows the encoder to defer the fill operation, and hence the rendering burden until later.

A decoder can optionally look at the intersection between the objects in the region's object list and the area to be filled and then only fill the area not covered by objects. Decoders should take into account that objects can have a ragged right hand edge and can contain transparent holes. Any such optimization is beyond the scope of the subtitle decoder model.

5.4.4 CLUT modification

Once introduced a region is always bound to a particular CLUT. However, new definitions of the CLUT may be broadcast, i.e. the mapping between pixel code and displayed colour can be redefined. No rendering burden is assumed when CLUT definitions change.

5.4.5 Graphic object decoding

Graphical objects shall be rendered into the pixel buffer as they are decoded. One object may be referenced several times, for example, a character used several times in a piece of text. Within a region the rendering burden for each object is derived from:

- the number of pixels enclosed within the smallest rectangle that can enclose the object;
- the pixel depth of the region where the object is positioned;
- the number of times the object is positioned in the region.

The "smallest enclosing rectangle" rule is used to simplify calculations and also to give some consideration for the read-modify-write nature of pixel rendering processes.

The object coding allows a ragged right edge to objects. No coded information is provided for the pixel positions between the "end of object line code" and the "smallest enclosing rectangle" and therefore these pixels should be left unmodified by the rendering process.

The same rendering burden is assumed, regardless of whether an object has the non_modifying_colour_ flag set to implement holes in the object. Again this gives some consideration for the read-modify-write nature of pixel rendering processes.

5.4.6 Character object decoding

The subtitling system allows character references to be delivered as an alternative to graphical objects. The information inside the subtitling stream is not sufficient to make such a character coded system work reliably.

A local agreement between broadcasters and equipment manufacturers may be an appropriate way to ensure reliable operation of character coded subtitles. A local agreement would probably define the characteristics of the font (character size and other metrics). It should also define a model for rendering of the characters.

6 PES packet format

For carriage of DVB subtitles the PES packet syntax and semantics as defined in ISO/IEC 13818-1 [1] are applied within the constraints in table 1.

Table 1

stream_id	Set to '1011 1101' indicating "private_stream_1".
PES_packet_length	Set to a value that specifies the length of the PES packet, as defined in ISO/IEC 13818-1 [1].
data_alignment_indicator	Set to '1' indicating that the subtitle segments are aligned with the PES packets.
Presentation_Time_Stamp of subtitle	The PTS, indicating the time at which the presentation begins of the display set carried by the PES packet(s) with this PTS. The PTSs of subsequent displays shall differ by more than one video frame.
PES_packet_data_byte	The PES_data_field specified in clause 7 of the present document.

7 The PES packet data for subtitling

7.1 Syntax and semantics of the PES data field for subtitling

When carrying a DVB subtitle stream, the PES_packet_data_bytes shall be encoded as the PES_data_field defined in the table below.

Syntax	Size	Type
PES_data_field() {		
data_identifier	8	bslbf
subtitle_stream_id	8	bslbf
while nextbits() == '0000 1111' {		
Subtitling_segment()		
}		
end_of_PES_data_field_marker	8	bslbf
}		

Semantics:

data_identifier: For DVB subtitle streams the data_identifier field shall be coded with the value 0x20.

subtitle_stream_id: This identifies the subtitle stream in this PES packet. A DVB subtitling stream shall be identified by the value 0x00.

end_of_PES_data_field_marker: An 8-bit field with fixed contents '1111 1111'.

7.2 Syntax and semantics of the subtitling segment

The basic syntactical element of the subtitling streams is the "segment". It forms the common format shared amongst all elements of this subtitling specification. A segment shall be encoded as described in the table below.

Syntax	Size	Type
Subtitling_segment() {		
sync_byte	8	bslbf
segment_type	8	bslbf
page_id	16	bslbf
segment_length	16	uimsbf
segment_data_field()		
}		

sync_byte: An 8-bit field that shall be coded with the value '0000 1111'. Inside a PES packet, decoders can use the sync_byte to verify synchronization when parsing segments based on the segment_length, so as to determine transport packet loss.

segment_type: This indicates the type of data contained in the segment data field. The following segment_type values are defined in this subtitling specification.

Table 2

0x10	page composition segment	defined in clause 7.2.1
0x11	region composition segment	defined in clause 7.2.2
0x12	CLUT definition segment	defined in clause 7.2.3
0x13	object data segment	defined in clause 7.2.4
0x40 - 0x7F	reserved for future use	
0x80	end of display set segment	defined in clause 7.2.5
0x81 - 0xEF	private data	
0xFF	stuffing (see note)	
All other values	reserved for future use	
NOTE: The present document does not define a syntax for stuffing within the PES. In applications where stuffing is deemed to be necessary (for example for monitoring or for network management reasons) implementers of DVB subtitle coding equipment are strongly advised to use the transport packet adaptation field for stuffing since that method will usually place no processing overhead on the subtitle encoder.		

page_id: The page_id identifies the subtitle service of the data contained in this subtitling_segment. Segments with a page_id value signalled in the subtitling descriptor as the composition page id, carry subtitling data specific for one subtitle service. Accordingly, segments with the page_id signalled in the subtitling descriptor as the ancillary page id, carry data that may be shared by multiple subtitle services.

segment_length: The segment_length shall specify the number of bytes contained in the immediately following segment_data_field.

segment_data_field: This is the payload of the segment. The syntax of this payload depends on the segment type, and is defined in clauses 7.2.1 to 7.2.5.

7.2.1 Page composition segment

The page composition for a subtitle service is carried in page_composition_segments. The page_id of each page_composition_segment shall be equal to the composition_page_id value provided by the subtitling descriptor.

Syntax	Size	Type
page_composition_segment() {		
sync_byte	8	bslbf
segment_type	8	bslbf
page_id	16	bslbf
segment_length	16	uimsbf
page_time_out	8	uimsbf
page_version_number	4	uimsbf
page_state	2	bslbf
reserved	2	bslbf
while (processed_length < segment_length) {		
region_id	8	bslbf
reserved	8	bslbf
region_horizontal_address	16	uimsbf
region_vertical_address	16	uimsbf
}		
}		

Semantics

page_time_out: The period, expressed in seconds, after which a page instance is no longer valid and consequently shall be erased from the screen, should it not have been redefined before that. The time-out period starts when the page instance is first displayed. The page_time_out value applies to each page instance until its value is redefined. The purpose of the time-out period is to avoid a page instance remaining on the screen "for ever" if the IRD happens to have missed the redefinition or deletion of the page instance. The time-out period does not need to be counted very accurately by the IRD: a reaction accuracy of -0/+5 ss is accurate enough.

page_version_number: The version of this page composition segment. When any of the contents of this page composition segment change, this version number is incremented (modulo 16).

page_state: This field signals the status of the subtitling page instance described in this page composition segment. The values of the page_state are defined in table 3.

Table 3

Value	Page state	Effect on page	Comments
00	normal case	page update	The display set contains only the subtitle elements that are changed from the previous page instance.
01	acquisition point	page refresh	The display set contains all subtitle elements needed to display the next page instance.
10	mode change	new page	The display set contains all subtitle elements needed to display the new page.
11	reserved		Reserved for future use.

If the page state is "mode change" or "acquisition point", then the display set shall contain a region page composition segment for each region used in this epoch.

processed_length: The number of bytes from the field(s) within the while-loop that have been processed by the decoder.

region_id: This uniquely identifies a region within a page. Each identified region is displayed in the page instance defined in this page composition. Regions shall be listed in the page_composition_segment in the order of ascending region_vertical_address field values.

region_horizontal_address: This specifies the horizontal address of the top left pixel of this region. The left-most pixel of the 720 active pixels has horizontal address zero, and the pixel address increases from left to right.

region_vertical_address: This specifies the vertical address of the top line of this region. The top line of the 720 × 576 frame is line zero, and the line address increases by one within the frame from top to bottom.

NOTE: All addressing of pixels is based on a frame of 720 pixels horizontally by 576 scan lines vertically. These numbers are independent of the aspect ratio of the picture; on a 16:9 display a pixel looks a bit wider than on a 4:3 display. In some cases, for instance a logo, this may lead to unacceptable distortion. Separate data may be provided for presentation on each of the different aspect ratios. The subtitle_descriptor signals whether the associated subtitle data can be presented on any display or on displays of specific aspect ratio only.

7.2.2 Region composition segment

The region composition for a specific region is carried in region_composition_segments. The region composition contains a list of objects; the listed objects shall be positioned in such a way that they do not overlap.

If an object is added to a region in case of a page update, new pixel data will overwrite either the background colour of the region or "old objects". The programme provider shall take care that the new pixel data overwrites only information that needs to be replaced, but also that it overwrites all pixels in the region that are not to be preserved. Note that a pixel is either defined by the background colour, or by an "old" object or by a "new" object; if a pixel is overwritten none of its previous definition is retained.

Syntax	Size	Type
region_composition_segment() {		
sync_byte	8	bslbf
segment_type	8	bslbf
page_id	16	bslbf
segment_length	16	uimsbf
region_id	8	uimsbf
region_version_number	4	uimsbf
region_fill_flag	1	bslbf
reserved	3	bslbf
region_width	16	uimsbf
region_height	16	uimsbf
region_level_of_compatibility	3	bsblf
region_depth	3	bsblf
reserved	2	bsblf
CLUT_id	8	bslbf
region_8-bit_pixel_code	8	bslbf
region_4-bit_pixel-code	4	bsblf
region_2-bit_pixel-code	2	bslbf
reserved	2	bslbf
while (processed_length < segment_length) {		
object_id	16	bslbf
object_type	2	bslbf
object_provider_flag	2	bslbf
object_horizontal_position	12	uimsbf
reserved	4	bslbf
object_vertical_position	12	uimsbf
if (object_type ==0x01 or object_type == 0x02){		
foreground_pixel_code	8	bslbf
background_pixel_code	8	bslbf
}		
}		
}		

Semantics

region_id: This 8-bit field uniquely identifies the region for which information is contained in this region_composition_segment.

region_version_number: This indicates the version of this region. The version number is incremented (modulo 16) if one or more of the following conditions is true:

- the region_fill_flag is set;
- the regions CLUT family has been modified;
- the region has a non-zero length object list.

region_fill_flag: If set to '1', signals that the region is to be filled with the background colour defined in the region_n-bit_pixel_code fields in this segment.

region_width: Specifies the horizontal length of this region, expressed in number of pixels. The value in this field shall be within the range 1 to 720, and the sum of the region_width and the region_horizontal_address (see clause 7.2.1) shall not exceed 720.

region_height: Specifies the vertical length of the region, expressed in number of pixels . The value in this field shall be within the inclusive range 1 to 576, and the sum of the region_height and the region_vertical_address (see clause 7.2.1) shall not exceed 576.

region_level_of_compatibility: This indicates the minimum type of CLUT that is necessary in the decoder to decode this region as defined in table 4.

Table 4

0x00	reserved
0x01	2-bit/entry CLUT required
0x02	4-bit/entry CLUT required
0x03	8-bit/entry CLUT required
0x04 - 0x07	reserved

If the decoder does not support the specified minimum requirement for the type of CLUT, then this region shall not be displayed, even though some other regions, requiring a lesser type of CLUT, may be presented.

region_depth: Identifies the intended pixel depth for this region as defined in table 5.

Table 5

0x00	reserved
0x01	2 bit
0x02	4 bit
0x03	8 bit
0x04 - 0x07	reserved

CLUT_id: Identifies the family of CLUTs that applies to this region.

region_8-bit_pixel-code: Specifies the entry of the applied 8-bit CLUT as background colour for the region when the region_fill_flag is set, but only if the region depth is 8 bit. The value of this field is undefined if a region depth of 2 or 4 bit applies.

region_4-bit_pixel-code: Specifies the entry of the applied 4-bit CLUT as background colour for the region when the region_fill_flag is set, if the region depth is 4 bit, or if the region depth is 8 bit while the region_level_of_compatibility specifies that a 4-bit CLUT is within the minimum requirements. In any other case the value of this field is undefined.

region_2-bit_pixel-code: Specifies the entry of the applied 2-bit CLUT as background colour for the region when the region_fill_flag is set, if the region depth is 2 bit, or if the region depth is 4 or 8 bit while the region_level_of_compatibility specifies that a 2-bit CLUT is within the minimum requirements. In any other case the value of this field is undefined.

processed_length: The number of bytes from the field(s) within the while-loop that have been processed by the decoder.

object_id: Identifies an object that is shown in the region.

object_type: Identifies the type of object as defined in table 6.

Table 6

0x00	basic_object, bitmap
0x01	basic_object, character
0x02	composite_object, string of characters
0x03	reserved

object_provider_flag: A 2-bit flag indicating how this object is provided, as defined in table 7.

Table 7

0x00	provided in the subtitling stream
0x01	provided by a ROM in the IRD
0x02	Reserved
0x03	reserved

object_horizontal_position: Specifies the horizontal position of the top left pixel of this object, expressed in number of horizontal pixels, relative to the left-hand edge of the associated region. The specified horizontal position shall be within the region, hence its value shall be in the range between 0 and region_width - 1.

object_vertical_position: Specifies the vertical position of the top left pixel of this object, expressed in number of lines, relative to the top of the associated region. The specified vertical position shall be within the region, hence its value shall be in the range between 0 and region_height - 1.

foreground_pixel_code: Specifies the entry in the applied 8-bit CLUT that has been selected as the foreground colour of the character(s).

background_pixel_code: Specifies the entry in the applied 8-bit CLUT that has been selected as the background colour of the character(s).

NOTE: IRDs with CLUT of four or sixteen entries find the foreground and background colours through the reduction schemes described in clause 9.

7.2.3 CLUT definition segment

Colours to be applied in a CLUT family are carried in CLUT_definition_segments as shown in the table below:

Syntax	Size	Type
CLUT_definition_segment() {		
sync_byte	8	bslbf
segment_type	8	bslbf
page_id	16	bslbf
segment_length	16	uimsbf
CLUT-id	8	bslbf
CLUT_version_number	4	uimsbf
reserved	4	bslbf
while (processed_length < segment_length) {		
CLUT_entry_id	8	bslbf
2-bit/entry_CLUT_flag	1	bslbf
4-bit/entry_CLUT_flag	1	bslbf
8-bit/entry_CLUT_flag	1	bslbf
reserved	4	bslbf
full_range_flag	1	bslbf
if full_range_flag =='1' {		
Y-value	8	bslbf
Cr-value	8	bslbf
Cb-value	8	bslbf
T-value	8	bslbf
} else {		
Y-value	6	bslbf
Cr-value	4	bslbf
Cb-value	4	bslbf
T-value	2	bslbf
}		
}		
}		

Semantics

CLUT-id: Uniquely identifies within a page the CLUT family whose data is contained in this CLUT_definition_segment field.

CLUT_version_number: Indicates the version of this segment data. When any of the contents of this segment change this version number is incremented (modulo 16).

processed_length: The number of bytes from the field(s) within the while-loop that have been processed by the decoder.

CLUT_entry_id: Specifies the entry number of the CLUT. The first entry of the CLUT has entry number zero.

2-bit/entry_CLUT_flag: If set to '1', this indicates that this CLUT value is to be loaded into the identified entry of the 2-bit/entry CLUT.

4-bit/entry_CLUT_flag: If set to '1', this indicates that this CLUT value is to be loaded into the identified entry of the 4-bit/entry CLUT.

8-bit/entry_CLUT_flag: If set to '1', this indicates that this CLUT value is to be loaded into the identified entry of the 8-bit/entry CLUT.

NOTE 1: Only one N-bit/entry_CLUT_flag shall be set to 1 per CLUT_entry_id and its associated Y-, Cr-, Cb- and T-values.

full_range_flag: If set to '1', this indicates that the Y_value, Cr_value, Cb_value and T_value fields have the full 8-bit resolution. If set to '0', then these fields contain only the most significant bits.

Y_value: The Y output value of the CLUT for this entry. A value of zero in the Y_value field signals full transparency. In that case the values in the Cr_value, Cb_value and T_value fields are irrelevant and shall be set to zero.

NOTE 2: Implementers should note that Y=0 is disallowed in ITU-R BT601-3. This condition should be recognized and mapped to a legal value (e.g. Y=16d) before conversion to RGB values in a decoder.

Cr_value: The Cr output value of the CLUT for this entry.

Cb_value: The Cb output value of the CLUT for this entry.

NOTE 3: Y, Cr and Cb have meanings as defined in ITU-R Recommendation BT.601-3 [3] and in ITU-R Recommendation BT.656-4 [4].

NOTE 4: Note that, whilst this subtitling specification defines CLUT entries in terms of Y, Cr, Cb and T values, the standard interface definition of digital television (ITU-R BT656-4) presents co-sited sample values in the order Cb,Y,Cr. Failure correctly to interpret the rendered bitmap image in terms of ITU-R BT656-4 may result in incorrect colours and chrominance mistiming.

T_value: The Transparency output value of the CLUT for this entry. A value of zero identifies no transparency. The maximum value plus one would correspond to full transparency. For all other values the level of transparency is defined by linear interpolation.

Full transparency is acquired through a value of zero in the Y_value field.

NOTE 5: Decoder models for the translation of pixel-codes into Y, Cr, Cb and T values are depicted in clause 9. Default contents of the CLUT are specified in clause 10.

NOTE 6: The colour for each CLUT entry can be redefined. There is no need for CLUTs with fixed contents as every CLUT has default contents, see clause 10.

7.2.4 Object data segment

The object_data_segment contains the data of an object. For graphical objects (i.e. the object_coding_method indicates coding as pixels) the following applies:

- an object is assumed to be interlaced, with a top field and a bottom field;
- the first pixel of the first line of the top field is the top left pixel of the object;
- the first pixel of the first line of the bottom field is the most left pixel on the second line of the object;
- the same object_data_segment shall carry a pixel-data_sub-block for both the top field and the bottom field;
- if a segment carries no data for the bottom field, i.e. the bottom_field_data_block_length contains the value '0x0000', then the pixel-data_sub-block for the top field shall apply for the bottom field also.

An object_data_segment is defined as shown in the table below:

Syntax	Size	Type
object_data_segment() {		
sync_byte	8	bslbf
segment_type	8	bslbf
page_id	16	bslbf
segment_length	16	uimsbf
object_id	16	bslbf
object_version_number	4	uimsbf
object_coding_method	2	bslbf
non_modifying_colour_flag	1	bslbf
reserved	1	bslbf
if (object_coding_method == '00'){		
top_field_data_block_length	16	uimsbf
bottom_field_data_block_length	16	uimsbf
while(processed_length<top_field_data_block_length)		
pixel-data_sub-block()		
while (processed_length<bottom_field_data_block_length)		
pixel-data_sub-block()		
if (!wordaligned())		
8_stuff_bits	8	bslbf
}		
if (object_coding_method == '01') {		
number of codes	8	uimsbf
for (i == 1, i <= number of codes, i ++)		
character_code	16	bslbf
}		
}		

Semantics

object_id: Uniquely identifies within the page the object for which data is contained in this object_data_segment field.

object_version_number: Indicates the version of this segment data. When any of the contents of this segment change, this version number is incremented (modulo 16).

object_coding_method: Specifies the method used to code the object:

Table 8

0x00	coding of pixels
0x01	coded as a string of characters
0x02	reserved
0x03	reserved

non_modifying_colour_flag: If set to '1' this indicates that the CLUT entry value '1' is a non modifying colour. When the non modifying colour is assigned to an object pixel, then the pixel of the underlying region background or object shall not be modified. This can be used to create "transparent holes" in objects.

top_field_data_block_length: Specifies the number of bytes contained in the pixel-data_sub-blocks for the top field.

bottom_field_data_block_length: Specifies the number of bytes contained in the data_sub-block for the bottom field.

For each object the pixel-data sub-block for the top field and the pixel-data sub-block for the bottom field shall be carried in the same object_data_segment. If this segment carries no data for the bottom field, i.e. the bottom_field_data_block_length contains the value '0x0000', then the data for the top field shall be valid for the bottom field also.

NOTE: This effectively forbids an object from having a height of only one TV picture line. Isolated objects of this height would be liable to suffer unpleasant flicker effects at the TV display frame rate when displayed on an interlaced display.

processed_length: The number of bytes from the field(s) within the while-loop that have been processed by the decoder.

8_stuff_bits: Eight stuffing bits that shall be coded as '0000 0000'.

number_of_codes: Specifies the number of character codes in the string.

character_code: Specifies a character through its index number in the character table identified in the subtitle_descriptor. Each reference to the character table is counted as a separate character code, even if the resulting character is non spacing. For instance floating accents are counted as separate character codes.

7.2.4.1 Pixel-data sub-block

Syntax	Size	Type
pixel-data_sub-block() {		
data_type	8	bslbf
if data_type =='0x10' {		
repeat {		
2-bit/pixel_code_string()		
} until (end of 2-bit/pixel_code_string)		
while (!bytealigned())		
2_stuff_bits	2	bslbf
if data_type =='0x11' {		
repeat {		
4-bit/pixel_code_string()		
} until (end of 4-bit/pixel_code_string)		
if (!bytealigned())		
4_stuff_bits	4	bslbf
}		
}		
if data_type =='0x12' {		
repeat {		
8-bit/pixel_code_string()		
} until (end of 8-bit/pixel_code_string)		
}		
if data_type =='0x20'		
2_to_4-bit_map-table	16	bslbf
if data_type =='0x21'		
2_to_8-bit_map-table	32	bslbf
if data_type =='0x22'		
4_to_8-bit_map-table	128	bslbf
}		

Semantics

data_type: Identifies the type of information contained in the pixel-data_sub-block according to table 9.

Table 9

0x10	2-bit/pixel code string
0x11	4-bit/pixel code string
0x12	8-bit/pixel code string
0x20	2_to_4-bit_map-table data
0x21	2_to_8-bit_map-table data
0x22	4_to_8-bit_map-table data
0xF0	end of object line code
NOTE: All other values are reserved.	

A code '0xF0' = "end of object line code" shall be included after every series of code strings that together represent one line of the object.

2_to_4-bit_map-table: Specifies how to map the 2-bit/pixel codes on a 4-bit/entry CLUT by listing the 4 entry numbers of 4-bits each; entry number 0 first, entry number 3 last.

2_to_8-bit_map-table: Specifies how to map the 2-bit/pixel codes on an 8-bit/entry CLUT by listing the 4 entry numbers of 8-bits each; entry number 0 first, entry number 3 last.

4_to_8-bit_map-table: Specifies how to map the 4-bit/pixel codes on an 8-bit/entry CLUT by listing the 16 entry numbers of 8-bits each; entry number 0 first, entry number 15 last.

2_stuff_bits: Two stuffing bits that shall be coded as '00'.

4_stuff_bits: Four stuffing bits that shall be coded as '0000'.

7.2.4.2 Syntax and semantics of the pixel code strings

Syntax	Size	Type
2-bit/pixel_code_string() {		
if (nextbits() != '00') {		
2-bit_pixel-code	2	bslbf
} else {		
2-bit_zero	2	bslbf
switch_1	1	bslbf
if (switch_1 == '1') {		
run_length_3-10	3	uimsbf
2-bit_pixel-code	2	bslbf
} else {		
switch_2	1	bslbf
if (switch_2 == '0') {		
switch_3	2	bslbf
if (switch_3 == '10') {		
run_length_12-27	4	uimsbf
2-bit_pixel-code	2	bslbf
}		
if (switch_3 == '11') {		
run_length_29-284	8	uimsbf
2-bit_pixel-code	2	bslbf
}		
}		
}		
}		
}		

Semantics

2-bit_pixel-code: A 2-bit code, specifying the pseudo-colour of a pixel as either an entry number of a CLUT with four entries or an entry number of a map-table.

2-bit_zero: A 2-bit field filled with '00'.

switch_1: A 1-bit switch that identifies the meaning of the following fields.

run_length_3-10: Number of pixels minus 3 that shall be set to the pseudo-colour defined next.

switch_2: A 1-bit switch. If set to '1', it signals that one pixel shall be set to pseudo-colour (entry) '00', else it indicates the presence of the following fields.

switch_3: A 2-bit switch that may signal the following:

Table 10

00	end of 2-bit/pixel_code_string
01	two pixels shall be set to pseudo colour (entry) '00'
10	the following 6 bits contain run length coded pixel data
11	the following 10 bits contain run length coded pixel data

run_length_12-27: Number of pixels minus 12 that shall be set to the pseudo-colour defined next.

run_length_29-284: Number of pixels minus 29 that shall be set to the pseudo-colour defined next.

Syntax	Size	Type
4-bit/pixel_code_string() {		
if (nextbits() != '0000') {		
4-bit_pixel-code	4	bslbf
} else {		
4-bit_zero	4	bslbf
switch_1	1	bslbf
if (switch_1 == '0') {		
if (nextbits() != '000')		
run_length_3-9	3	uimsbf
else		
end_of_string_signal	3	bslbf
} else {		
switch_2	1	bslbf
if (switch_2 == '0') {		
run_length_4-7	2	bslbf
4-bit_pixel-code	4	bslbf
} else {		
switch_3	2	bslbf
if (switch_3 == '10') {		
run_length_9-24	4	uimsbf
4-bit_pixel-code	4	bslbf
}		
if (switch_3 == '11') {		
run_length_25-280	8	uimsbf
4-bit_pixel-code	4	bslbf
}		
}		
}		
}		
}		

Semantics

4-bit_pixel-code: A 4-bit code, specifying the pseudo-colour of a pixel as either an entry number of a CLUT with sixteen entries or an entry number of a map-table.

4-bit_zero: A 4-bit field filled with '0000'.

switch_1: A 1-bit switch that identifies the meaning of the following fields.

run_length_3-9: Number of pixels minus 2 that shall be set to pseudo-colour (entry) '0000'.

end_of_string_signal: A 3-bit field filled with '000'. The presence of this field, i.e. nextbits() == '000', signals the end of the 4-bit/pixel_code_string.

switch_2: A 1-bit switch. If set to '0', it signals that that the following 6-bits contain run-length coded pixel-data, else it indicates the presence of the following fields.

switch_3: A 2-bit switch that may signal the following:

Table 11

00	1 pixel shall be set to pseudo-colour (entry) '0000'
01	2 pixels shall be set to pseudo-colour (entry) '0000'
10	the following 8 bits contain run-length coded pixel-data
11	the following 12 bits contain run-length coded pixel-data

run_length_4-7: Number of pixels minus 4 that shall be set to the pseudo-colour defined next.

run_length_9-24: Number of pixels minus 9 that shall be set to the pseudo-colour defined next.

run_length_25-280: Number of pixels minus 25 that shall be set to the pseudo-colour defined next.

Syntax	Size	Type
8-bit/pixel_code_string() {		
if (nextbits() != '0000 0000') {		
8-bit_pixel-code	8	bslbf
} else {		
8-bit_zero	8	bslbf
switch_1	1	bslbf
if switch_1 == '0' {		
if nextbits() != '000 0000'		
run_length_1-127	7	uimsbf
else		
end_of_string_signal	7	bslbf
} else {		
run_length_3-127	7	uimsbf
8-bit_pixel-code	8	bslbf
}		
}		
}		

Semantics

8-bit_pixel-code: An 8-bit code, specifying the pseudo-colour of a pixel as an entry number of a CLUT with 256 entries.

8-bit_zero: An 8-bit field filled with '0000 0000'.

switch_1: A 1-bit switch that identifies the meaning of the following fields.

run_length_1-127: Number of pixels that shall be set to pseudo-colour (entry) '0x00'.

end_of_string_signal: A 7-bit field filled with '000 0000'. The presence of this field, i.e. nextbits() == '000 0000', signals the end of the 8-bit/pixel_code_string.

run_length_3-127: Number of pixels that shall be set to the pseudo-colour defined next. This field shall not have a value of less than three.

7.2.5 End of display set segment

The end_of_display_set_segment provides an explicit indication to the decoder that transmission of a display set is complete. The end_of_display_set_segment shall be inserted into the stream immediately after the last object_data_segment for each display set. It shall be present for each subtitle service in a subtitle stream, although decoders need not take advantage of this segment and may apply other strategies to determine when they have sufficient information from a display set to commence decoding.

Syntax	Size	Type
end_of_display_set_segment() {		
sync_byte	8	bslbf
segment_type	8	bslbf
page_id	16	bslbf
segment_length	16	uimsbf
}		

Semantics

page_id: If the subtitle service uses shared data, then the page_id shall be coded with the ancillary page id value signalled in the subtitle descriptor. Otherwise the page_id shall have the value of the composition page id.

8 Requirements for the subtitling data

Unless stated otherwise, all requirements apply at any particular point in time but they do not relate to situations at different points in time. In this clause the following terminology is used. If a segment is signalled by the composition page id value, then the segment is said to be "in" the composition page and the composition page is said to "contain" that segment. Similarly, a segment signalled by the ancillary page id value is said to be "in" the ancillary page and the ancillary page is said to "contain" such segment.

8.1 Scope of Identifiers

All identifiers (region_id, CLUT_id, object_id) are unique within a page.

8.2 Scope of dependencies

8.2.1 Composition page

A segment in the composition page may reference segments in that composition page as well as segments in the ancillary page.

8.2.2 Ancillary page

The ancillary page may contain only CLUT definition segments and object data segments. Neither page composition segments, nor region composition segments shall be carried in the ancillary page. Segments in an ancillary page can be referenced by segments in any (composition) page.

NOTE: From clauses 8.2.1 and 8.2.2 it follows that segments in a composition page can be referenced only by segments in the same composition page.

8.3 Order of delivery

8.3.1 PTS field

The PTS field in successive PES packets shall either remain the same or proceed monotonically. Thus PES packets are delivered in their presentation time-order.

Discontinuities in the PTS sequence may occur if there are discontinuities in the PCR time base.

8.4 Positioning of regions and objects

8.4.1 Regions

A region monopolizes the scan lines on which it is shown; no two regions can be presented horizontally next to each other.

8.4.2 Objects sharing a PTS

Objects that are referenced by the same PTS (i.e. they are part of the same display set) shall not overlap on the screen.

8.4.3 Objects added to a region

If an object is added to a region, the new pixel data will overwrite the present information in the region. Thus a new object may (partly) cover old objects. The programme provider shall take care that the new pixel data overwrites only information that needs to be replaced, but also that it overwrites all information on the screen that is not to be preserved.

NOTE: A pixel is either defined by an "old" object or by the background colour or by the "new" object; if a pixel is overwritten none of its previous definition is retained.

9 Translation to colour components

An IRD can present only a limited number of different colours simultaneously within a single region. The colours themselves may be chosen from a larger palette, but the number of choices from the palette that can be used per region is limited. The subtitling system directly supports IRDs that can present four colours, sixteen colours and 256 colours, respectively. Three cases are distinguished:

- 4 colour IRDs. Pixel codes that use a 2-bit CLUT can be decoded into Y, Cr, Cb and T directly; pixel codes that use a 4-bit or 8-bit CLUT can be decoded also, but only if the region allows for decoding on a 2-bit CLUT. If such decoding is allowed, a reduction scheme is provided for translating the original 16 or 256 colours to the available 4 colours.

- 16 colour IRDs. Pixel codes that use a 2-bit or 4-bit CLUT can be decoded into Y, Cr, Cb and T directly; pixel codes that use an 8-bit CLUT can be decoded if the region allows for decoding on a 4-bit CLUT. If such decoding is allowed, a reduction scheme is provided for translating the original 256 colours to the available 16 colours. When pixel codes use a 4-bit CLUT, it is possible to switch to a 2-bit coding scheme within certain areas where at most 4 out of the 16 available colours are used. This requires a map table specifying which 4 CLUT entries are addressed with the 2-bit codes.

- 256 colour IRDs. All pixel codes can be decoded into Y, Cr, Cb and T directly, irrespective whether they use a 2-bit or 4-bit or an 8-bit CLUT. When a pixel code uses a 4-bit or an 8-bit CLUT, it is possible to switch to a 2-bit or a 4-bit coding scheme within a certain area where at most 4 or 16 out of the 256 available colours are used. This requires a map table specifying which 4 or 16 CLUT entries are addressed with the 2-bit or 4-bit codes, respectively.

The IRD shall translate a pixel's pseudo-colours into Y, Cr, Cb and T components according to the following model:

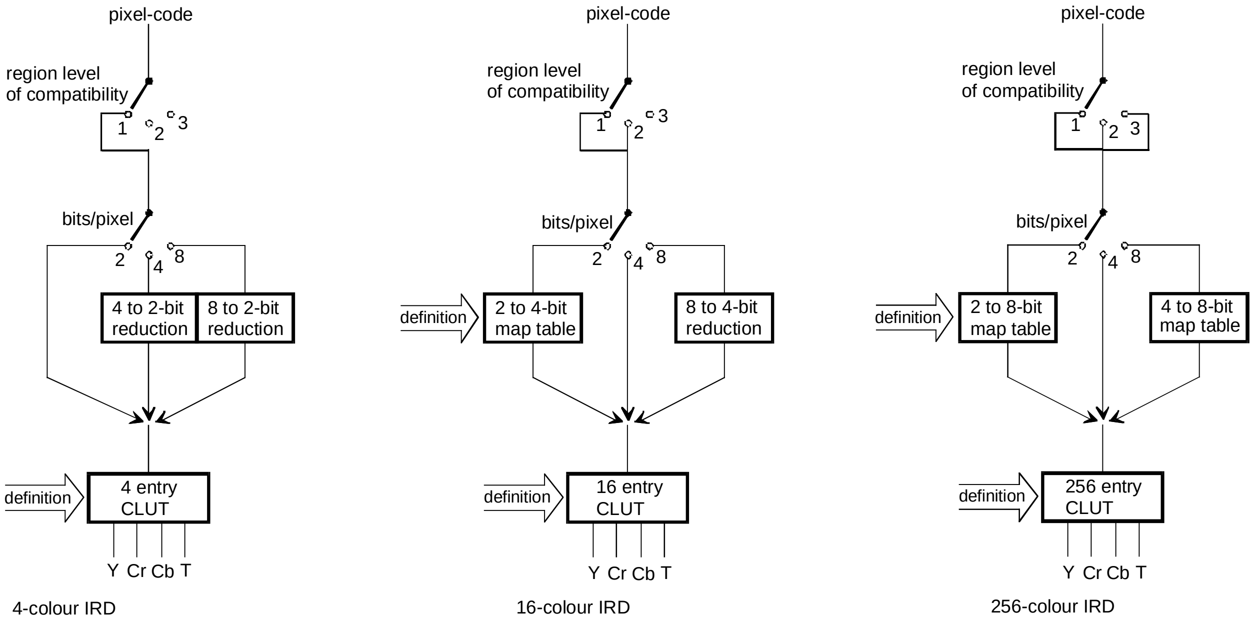

Figure 4

9.1 4- to 2-bit reduction

Let the input value be represented by a 4-bit field, the individual bits of which are called b_{i1}, b_{i2}, b_{i3} and b_{i4} where b_{i1} is received first and b_{i4} is received last. Let the output value be represented by a 2-bit field b_{o1}, b_{o2}.

The relation between output and input bits is:

$b_{o1} = b_{i1}$

$b_{o2} = b_{i2} \mid b_{i3} \mid b_{i4}$

9.2 8- to 2-bit reduction

Let the input value be represented by an 8-bit field, the individual bits of which are called bi1, bi2, bi3, bi4, bi5, bi6, bi7 and bi8 where bi1 is received first and bi8 is received last. Let the output value be represented by a 2-bit field bo1, bo2.

The relation between output and input bits is:

$b_{o1} = b_{i1}$

$b_{o2} = b_{i2} \mid b_{i3} \mid b_{i4}$

9.3 8- to 4-bit reduction

Let the input value be represented by a 8-bit field, the individual bits of which are called bi1, bi2, bi3, bi4, bi5, bi6, bi7 and bi8 where bi1 is received first and bi8 is received last. Let the output value be represented by a 4-bit field bo1 to bo4.

The relation between output and input bits is:

$b_{o1} = b_{i1}$ $b_{o2} = b_{i2}$

$b_{o3} = b_{i3}$ $b_{o4} = b_{i4}$

10 Default CLUTs and map-tables contents

This clause specifies the default contents of the CLUTs and map-tables for every CLUT family. Every entry for every CLUT can be redefined in a CLUT_definition_segment and every map-table can be redefined in an object_data_segment, but before such redefinitions the contents of CLUTs and map-tables shall correspond to the values specified here.

NOTE: CLUTs may be redefined partially. Entries that have not been redefined retain their default contents.

10.1 256-entry CLUT default contents

NOTE: The CLUT is divided in six sections: 64 colours of reduced intensity 0 to 50 %, 56 colours of higher intensity 0 to 100 %, 7 colours with 75 % transparency, 1 "colour" with 100 % transparency, 64 colours with 50 % transparency and 64 light colours (50 % white + colour 0 to 50 %).

Let the CLUT-entry number be represented by an 8-bit field, the individual bits of which are called b_1, b_2, b_3, b_4, b_5, b_6, b_7 and b_8 where b_1 is received first and b_8 is received last. The value in a bit is regarded as unsigned integer that can take the values zero and one.

The resulting colours are described here in terms of Red, Green and Blue contributions. To find the CLUT contents in terms of Y, Cr and Cb components, see ITU-R Recommendation BT.601-3 [3].

```
if $b_1$ == '0' andand $b_5$ == '0' {
    if $b_2$ == '0' andand $b_3$ == '0' andand $b_4$ == '0' {
        if $b_6$ == '0' andand $b_7$ == '0' andand $b_8$ == '0'
            $T = 100\ \%$
        else {
            $R = 100\ \% \times b_8$
            $G = 100\ \% \times b_7$
            $B = 100\ \% \times b_6$
            $T = 75\ \%$
        }
    }
    else {
        $R = 33{,}3\ \% \times b_8 + 66{,}7\ \% \times b_4$
        $G = 33{,}3\ \% \times b_7 + 66{,}7\ \% \times b_3$
        $B = 33{,}3\ \% \times b_6 + 66{,}7\ \% \times b_2$
        $T = 0\ \%$
    }
}
if $b_1$ == '0' andand $b_5$ == '1' {
    $R = 33{,}3\ \% \times b_8 + 66{,}7\ \% \times b_4$
    $G = 33{,}3\ \% \times b_7 + 66{,}7\ \% \times b_3$
    $B = 33{,}3\ \% \times b_6 + 66{,}7\ \% \times b_2$
    $T = 50\ \%$
}
if $b_1$ == '1' andand $b_5$ == '0' {
    $R = 16{,}7\ \% \times b_8 + 33{,}3\ \% \times b_4 + 50\ \%$
    $G = 16{,}7\ \% \times b_7 + 33{,}3\ \% \times b_3 + 50\ \%$
    $B = 16{,}7\ \% \times b_6 + 33{,}3\ \% \times b_2 + 50\ \%$
    $T = 0\ \%$
}
if $b_1$ == '1' andand $b_5$ == '1' {
    $R = 16{,}7\ \% \times b_8 + 33{,}3\ \% \times b_4$
    $G = 16{,}7\ \% \times b_7 + 33{,}3\ \% \times b_3$
    $B = 16{,}7\ \% \times b_6 + 33{,}3\ \% \times b_2$
    $T = 0\ \%$
}
```

10.2 16-entry CLUT default contents

Let the CLUT-entry number be represented by a 4-bit field, the individual bits of which are called b_1, b_2, b_3 and b_4 where b_1 is received first and b_4 is received last. The value in a bit is regarded as unsigned integer that can take the values zero and one.

The resulting colours are described here in terms of Red, Green and Blue contributions. To find the CLUT contents in terms of Y, Cr and Cb components, please see ITU-R Recommendation BT.601-3 [3].

```
if $b_1$ == '0' {
    if $b_2$ == '0' andand $b_3$ == '0' andand $b_4$ == '0' {
        T = 100 %
    }
    else {
        R = 100 % × $b_4$
        G = 100 % × $b_3$
        B = 100 % × $b_2$
        T = 0 %
    }
}
if $b_1$ == '1' {
    R = 50 % × $b_4$
    G = 50 % × $b_3$
    B = 50 % × $b_2$
    T = 0 %
}
```

10.3 4-entry CLUT default contents

Let the CLUT-entry number be represented by a 2-bit field, the individual bits of which are called b_1 and b_2 where b_1 is received first and b_2 is received last.

The resulting colours are described here in terms of Red, Green and Blue contributions. To find the CLUT contents in terms of Y, Cr and Cb components, please see ITU-R Recommendation BT.601-3 [3].

```
if $b_1$ == '0' andand $b_2$ == '0' {
    T = 100 %
}
if $b_1$ == '0' andand $b_2$ == '1' {
    R = G = B = 100 %
    T = 0 %
}
if $b_1$ == '1' andand $b_2$ == '0' {
    R = G = B = 0 %
    T = 0 %
}
if $b_1$ == '1' andand $b_2$ == '1' {
    R = G = B = 50 %
    T = 0 %
}
```

10.4 2_to_4-bit_map-table default contents

Table 11a

input value	output value
00	0000
01	0111
10	1000
11	1111

Input and output values are listed with their first bit left.

10.5 2_to_8-bit_map-table default contents

Table 12

Input value	Output value
00	0000 0000
01	0111 0111
10	1000 1000
11	1111 1111

Input and output values are listed with their first bit left.

10.6 4_to_8-bit_map-table default contents

Table 13

Input value	Output value
0000	0000 0000
0001	0001 0001
0010	0010 0010
0011	0011 0011
0100	0100 0100
0101	0101 0101
0110	0110 0110
0111	0111 0111
1000	1000 1000
1001	1001 1001
1010	1010 1010
1011	1011 1011
1100	1100 1100
1101	1101 1101
1110	1110 1110
1111	1111 1111

Input and output values are listed with their first bit left.

11 Structure of the pixel code strings (informative)

Table 14: 2-bit/pixel_code_string()

01	one pixel in colour 1
10	one pixel in colour 2
11	one pixel in colour 3
00 01	one pixel in colour 0
00 00 01	two pixels in colour 0
00 1L LL CC	L pixels (3-10) in colour C
00 00 10 LL LL CC	L pixels (12-27) in colour C
00 00 11 LL LL LL LL CC	L pixels (29-284) in colour C
00 00 00	end of 2-bit/pixel_code_string
NOTE: Runs of 11 pixels and 28 pixels can be coded as one pixel plus a run of 10 pixels and 27 pixels, respectively.	

Table 15: 4-bit/pixel_code_string()

0001 to 1111	one pixel in colour 1 to one pixel in colour 15
0000 1100	one pixel in colour 0
0000 1101	two pixels in colour 0
0000 0LLL	L pixels (3-9) in colour 0
0000 10LL CCCC	L pixels (4-7) in colour C
0000 1110 LLLL CCCC	L pixels (9-24) in colour C
0000 1111 LLLL LLLL CCCC	L pixels (25-280) in colour C
0000 0000	end of 4-bit/pixel_code_string
NOTE 1: Runs of 8 pixels in a colour not equal to '0' can be coded as one pixel plus a run of 7 pixels. NOTE 2: L>0	

Table 16: 8-bit/pixel_code_string()

00000001 to 11111111	one pixel in colour 1 to one pixel in colour 255
00000000 0LLLLLLL	L pixels (1-127) in colour 0 (L > 0)
00000000 1LLLLLLL CCCCCCCC	L pixels (3-127) in colour C (L > 2)
00000000 00000000	end of 8-bit/pixel_code_string

Annex A (informative): How the DVB subtitling system works

There are several possible ways to make the DVB subtitling system work. Aspects of several, incompatible approaches are described in the normative part of the present document.

Epoch boundaries (where page_state = "mode change") provide convenient service acquisition points. Short epochs will lead to quick service acquisition times. However, it is difficult to maintain smooth decoding across epoch boundaries and this is also likely to require more data to be broadcast. This is very similar to the issue of short GOPs in MPEG video.

The main issue is to allow the decoder to keep the last valid subtitle on the display until there is a new subtitle to replace it. This requires both subtitles being in the display memory at the same time. If each display takes up less than half the pixel buffer memory it should be possible for the decoder to switch between displays smoothly. However, there is a danger of the memory becoming fragmented over several epochs. If the decoder has to perform garbage collection it may be difficult to maintain its performance.

In practice the memory plan is likely to be identical for long periods. So, it would be useful if the broadcast data could differentiate new memory plans (justifying complete destruction of state) from repeat broadcasts of old memory plans (to provide service acquisition points).

It is expected that the screen may go blank for a short period when a new memory plan is issued. At service acquisition points practical decoders will continue decoding (building on the content of the regions that they have already decoded). Decoders newly acquiring the service are recommended to erase the regions to the defined background colour and then start decoding objects into them. Clearly after acquisition the display may be incomplete until sufficient objects have been received. Its up to the broadcaster to decide how rapidly to refresh the display.

A.1 Data hierarchy and terminology

The text of clause A.1, as present in earlier releases of the DVB Subtitling Specification, has been moved into the corresponding normative clause of the present document.

A.2 Temporal hierarchy and terminology

The text of clause A.2, as present in earlier releases of the DVB Subtitling Specification, has been moved into the corresponding normative clause of the present document.

A.3 Decoder temporal model

The text of clause A.3, as present in earlier releases of the DVB Subtitling Specification, has been moved into the corresponding normative clause of the present document.

A.4 Decoder display technology model

A.4.1 Region based with indexed colours

The DVB subtitling system is a region based, indexed colour, graphics system. This well matches the region-based on-screen displays being implemented at the time of writing. Such systems allow displays to be constructed using small amounts of memory. They also permit a number of apparently rapid graphical effects to be performed.

The display system can be implemented in other ways.

However, some effects that are simple when implemented in region based/indexed colour systems, may cause much greater demands when implemented in other ways. For example, in a region based system regions can be repositioned, or made visible/invisible with very little processing burden. In a simple bit mapped system such operations will require the pixel data to be moved within the display store or between the display store and some non-displayed storage. Similarly, in indexed colour systems certain effects can be implemented by redefining the contents of the CLUT associated with a particular region. In a system where there is one global CLUT for the complete display, or where pixels are not indexed before output (i.e. true colour) a CLUT redefinition may require the region to be redrawn.

The specification makes demands which are assumed to be reasonable in a region based, indexed colour, graphics system. Implementers are free to implement the graphics system in other ways. However, it is their responsibility to compensate for the implications of using an architecture that is different from that envisaged in the subtitle decoder model.

A.4.2 Colour quantization

At the time of design it was felt that some applications of the subtitling system would benefit from a 256 colour (i.e. 8-bit pixel) display system. However, it was understood that initially many decoders would have only 4- or 16-colour graphics systems.

Accordingly, the DVB subtitling system allows 256 colour graphics to be broadcast but then provides a model by which the whole spectrum of 256 colours can be quantized to 16 or 4 colours. The intention is to offer broadcasters and equipment manufacturers both a route and an incentive to move to 256 colour systems while allowing introduction of subtitling services at a time when many systems will not be able to implement 256 colours.

A byproduct of this colour quantization model is that it may be possible to implement systems with less pixel buffer memory than the 60 kbyte specified in the decoder model while still giving useful functionality. The 60 kbyte pixel buffer memory can be partitioned into any mix of 8, 4 and 2 bit per pixel regions, covering between 60 k and 240 k pixels. If memory in the decoder is very limited it may be possible to implement regions using a reduced pixel depth. For example, a region could be implemented using 2- or 4-bit pixel depth where 8 bits is the intended pixel depth.

Quantizing the colour depth may also allow the subtitling system to work with slower processors as the number of bit operations may decrease with the shallower pixel depth.

Taking full advantage of these techniques will depend on certain implementation features in the decoder. For example, it may require that the pixel depth can be set per region.

There are also broadcaster requirements to make broadcast data suitable for this approach. For example, if the broadcaster sets the region_level_of_compatability equal to the region_depth the decoder is forbidden to quantize the pixel depth. Also, if the broadcaster uses a very large number of 2-bit pixels the decoder has no opportunity to quantize colours.

A.5 Decoder rendering bandwidth model

The text of clause A.5, as present in earlier releases of the DVB Subtitling Specification, has been moved into the corresponding normative clause of the present document.

A.6 Examples of the subtitling system in operation

A.6.1 Double buffering

Regions can be operated on while they are not visible. Also they can be made visible or invisible by modifying the region list in the page composition segment or by modifying the CLUT. These features allow a number of effects as follows.

A.6.1.1 Instant graphics

At the start of an epoch a display is defined as using 3 regions [A, B, C]. Region A is allocated to hold a station logo and so will be present in all PCS. Its content is delivered in the first display set and thereafter periodically repeated to refresh it.

Throughout the epoch PCSs will alternate between having regions A and B or A and C in their region list. When the currently active page instance uses regions A and B the decoder will be decoding the next display which will use regions A and C. As at this time region C is not visible the viewer will not see the graphics being rendered into region C. When the new display becomes valid the decoder (assuming that it has a linked list, region based, graphics system) need only modify its display list to switch from a display of regions A and B to one using regions A and C.

This approach allows the display presented to the viewer to change crisply. However, more object data may need to be broadcast (e.g. to update B to be like C).

Figures A.1 to A.5 illustrate this. The right hand side of each picture shows the display presented to the viewer. Data is always rendered into regions that are not in the display list of the currently active PCS. So, the viewer never sees data being decoded into the display.

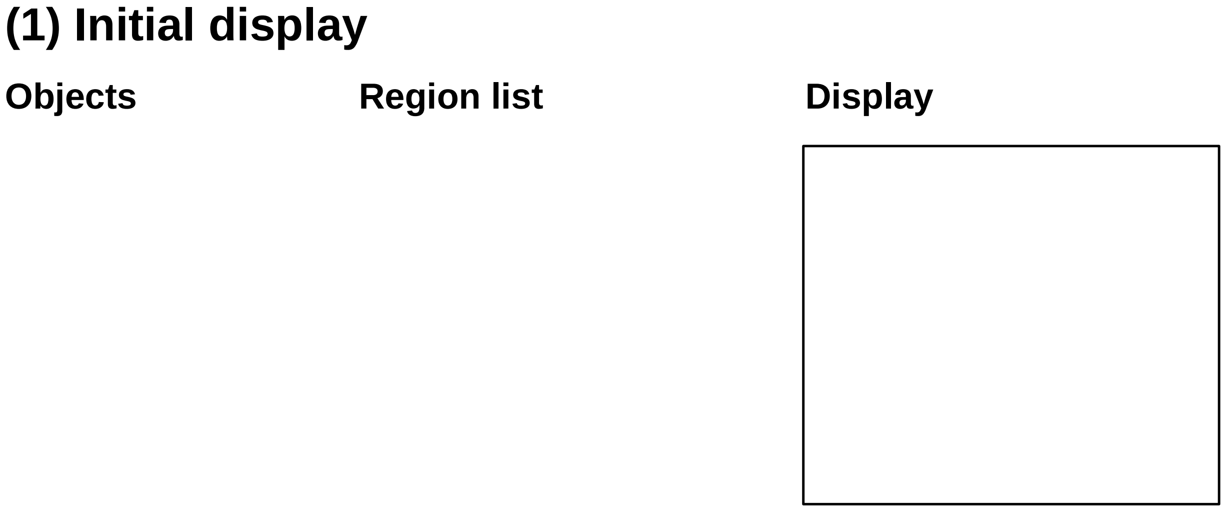

Figure A.1: Initial display

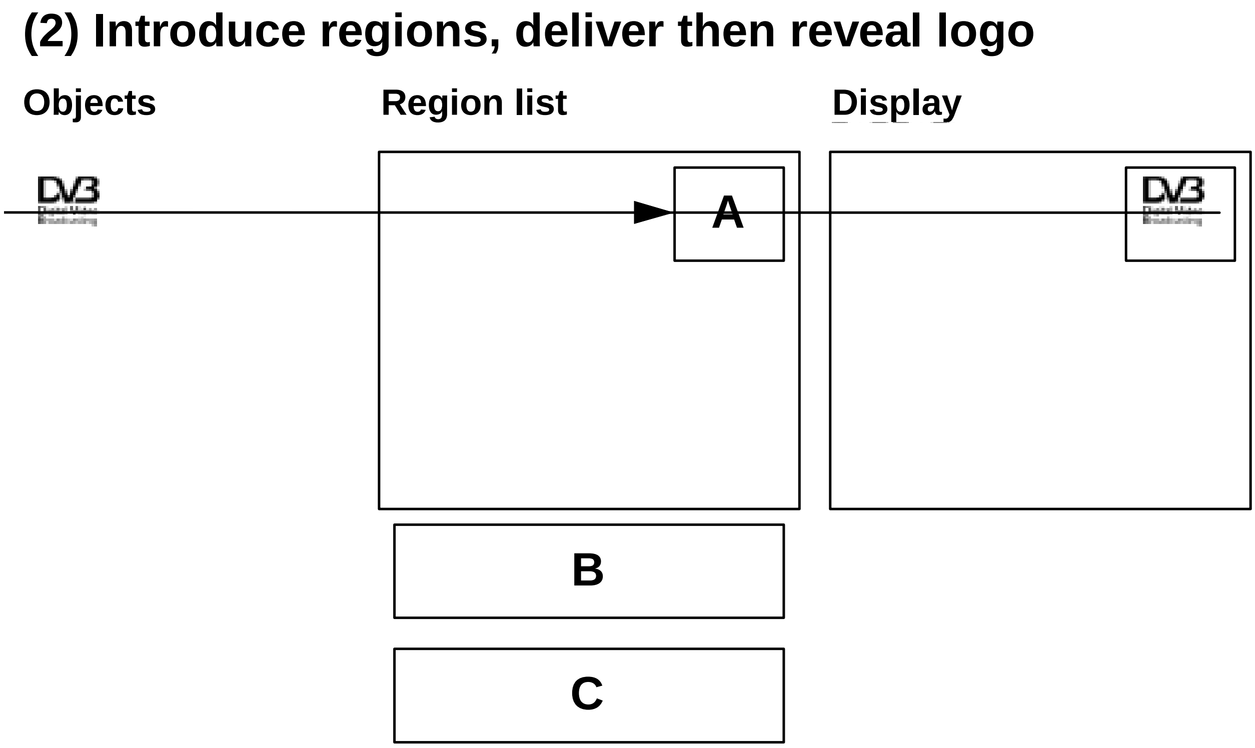

Figure A.2: Introduce regions, deliver then reveal logo

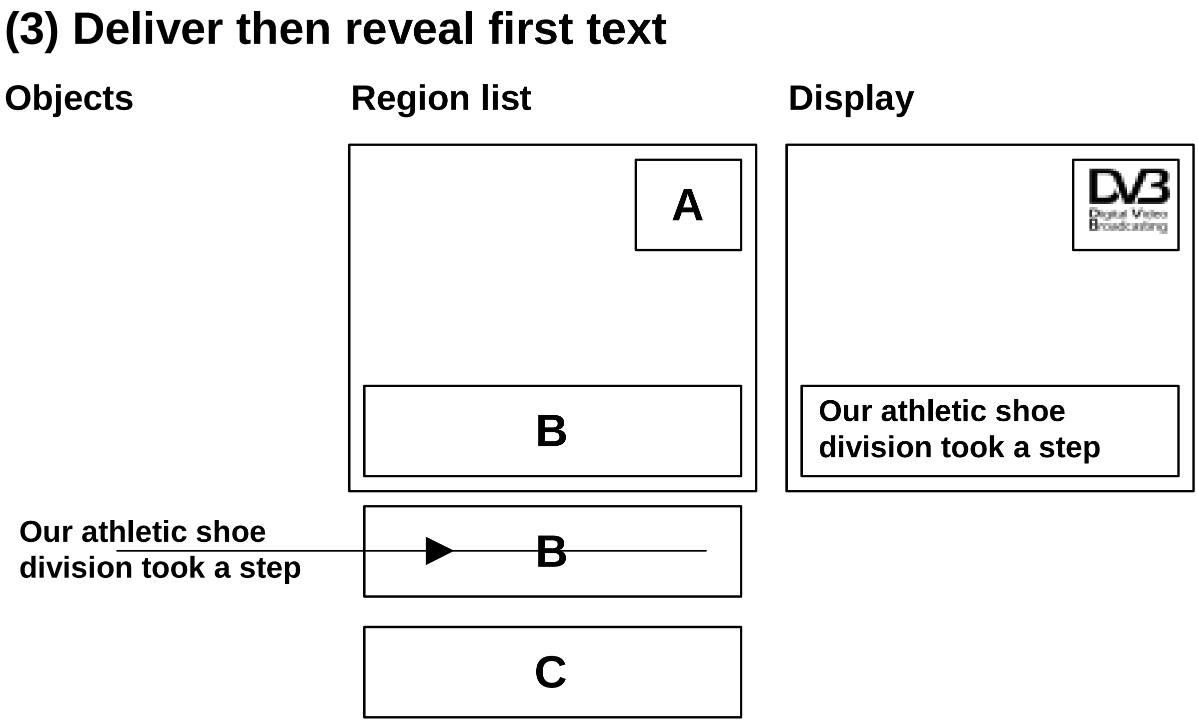
(3) Deliver then reveal first text
Objects
Region list
Display
A
DVB
B
Our athletic shoe division took a step
Our athletic shoe division took a step
B
C

Figure A.3: Deliver then reveal first text

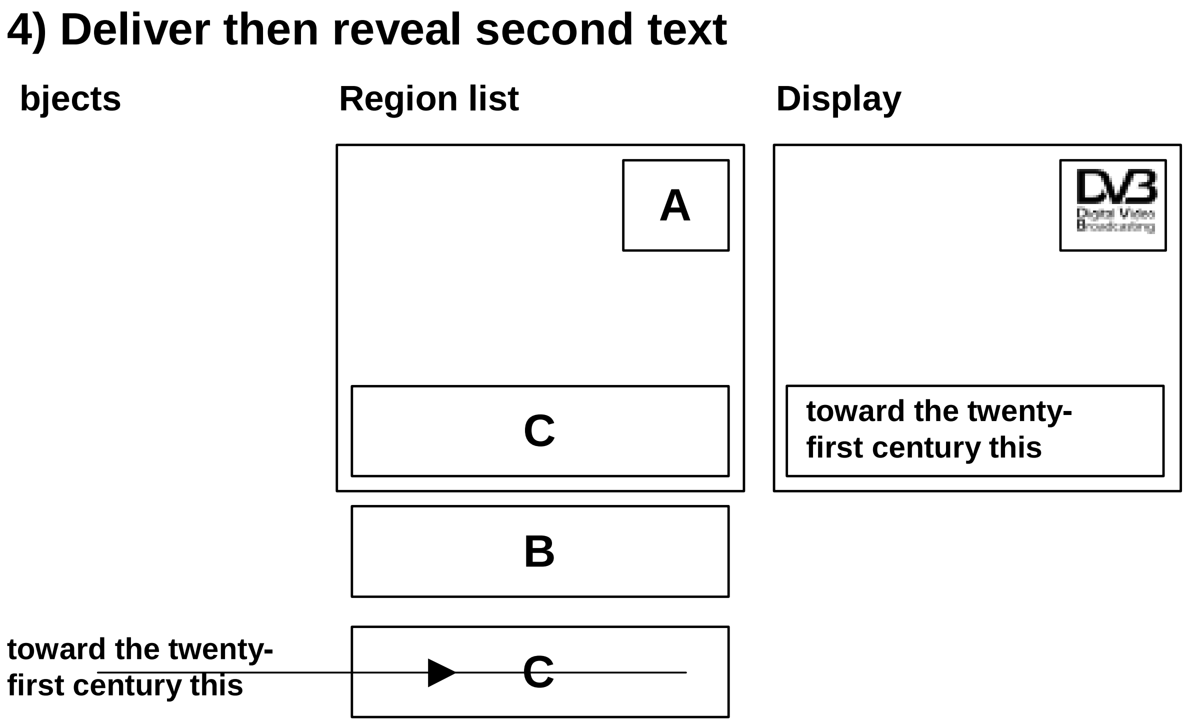

Figure A.4: Deliver then reveal second text

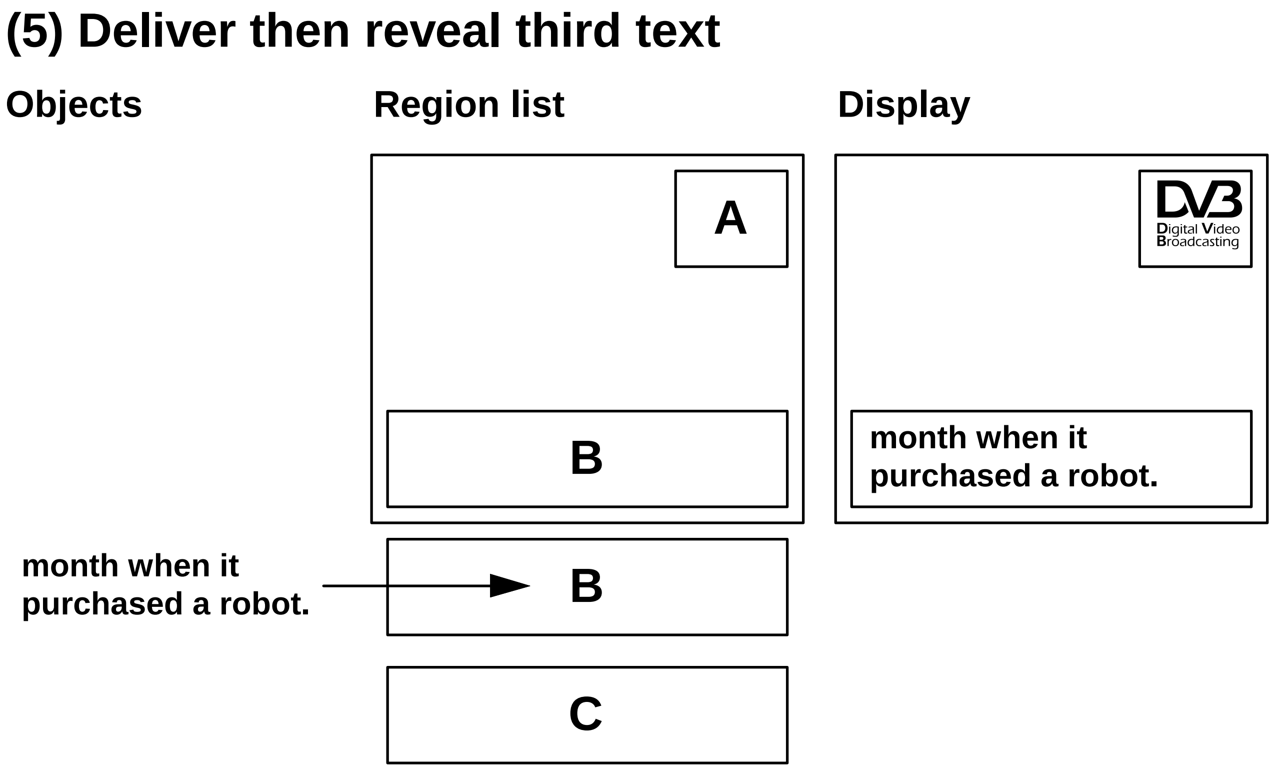

Figure A.5: Deliver then reveal third text

A.6.1.2 Stenographic subtitles

Four regions are defined (A, B, C, D). Regions A, B, C and D are identically sized rectangles sufficient to display a line of text each.

Initially the region list is A, B and C which are presented adjacent to each other to provide a 3-line text console. This region list is used for several page instances as new words are broadcast progressively filling A then B and finally C. When region C has been filled the region list for subsequent page instances uses B, C and D. In effect the text console has been scrolled-up by one line to provide an empty region E for new text. This process can continue with every few page instances the region list being changed to scroll the console (e.g. A, B and C then B, C and D then C, D and A.

Objects
Region list
Display

Figure A.6

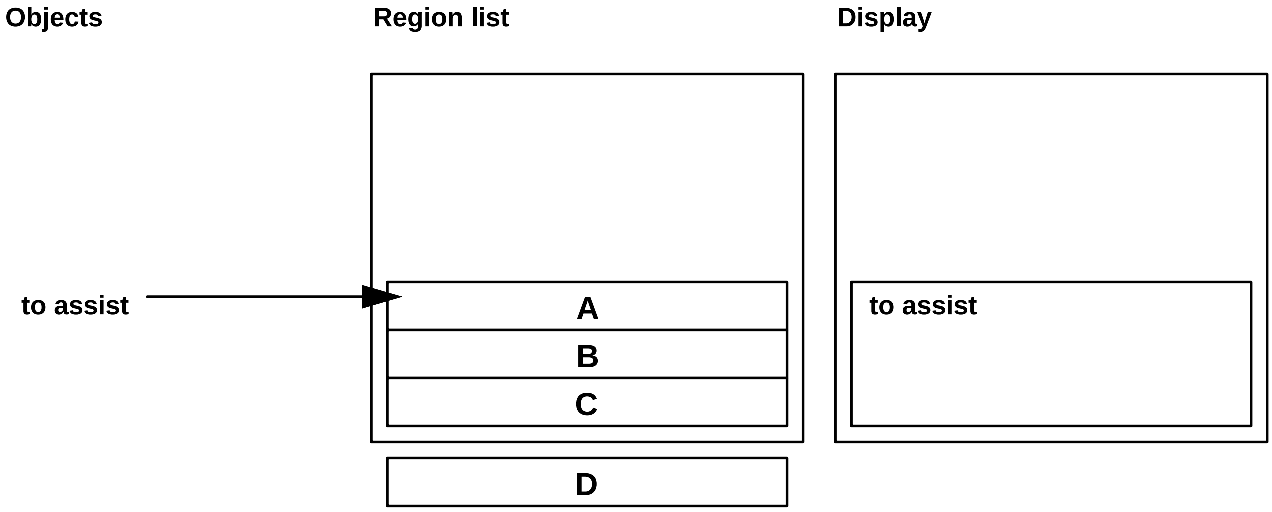

Figure A.7

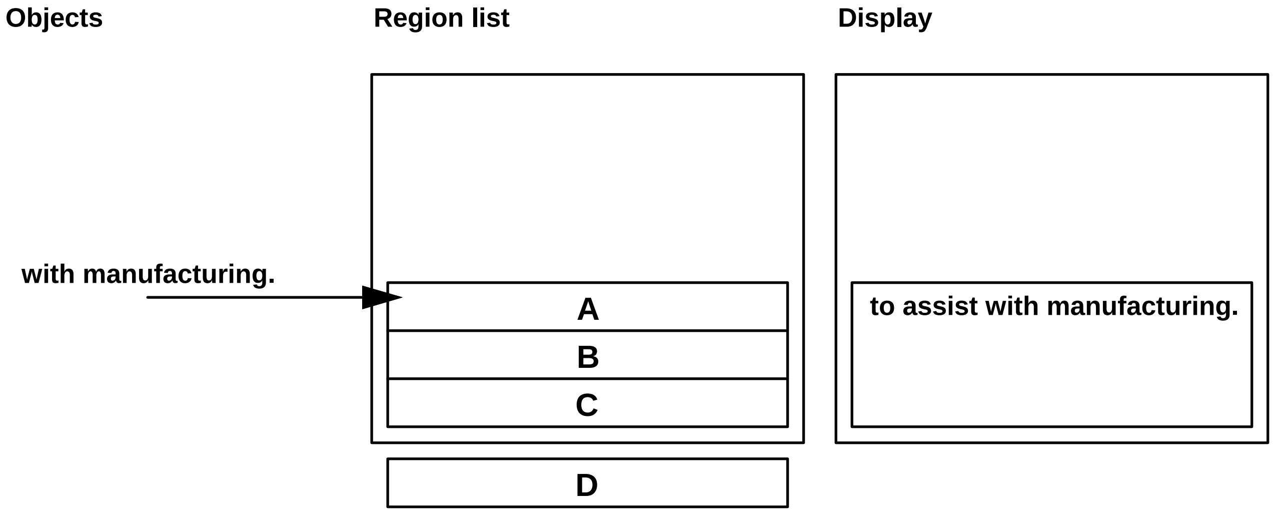

Figure A.8

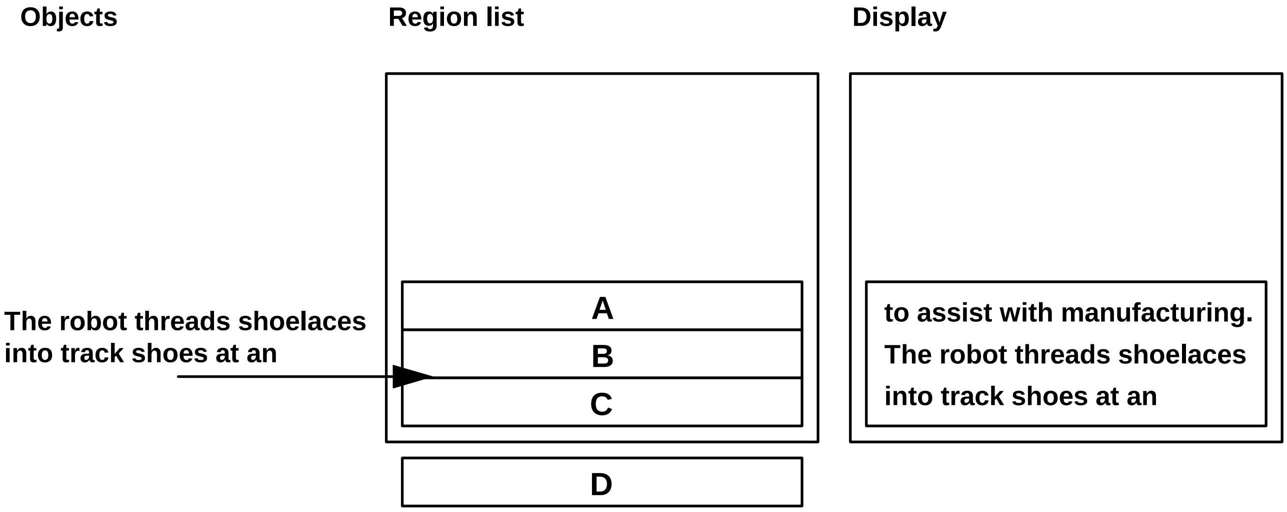
Objects
Region list
Display
The robot threads shoelaces
into track shoes at an
A
B
C
D
to assist with manufacturing.
The robot threads shoelaces
into track shoes at an

Figure A.9

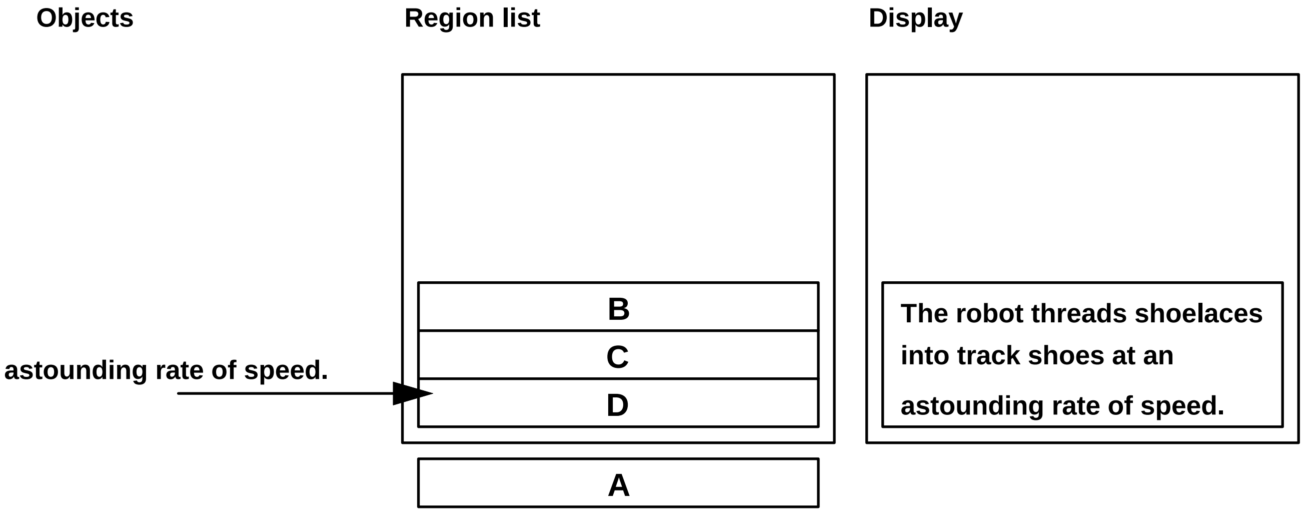
Objects
Region list
Display
astounding rate of speed.
B
C
D
A
The robot threads shoelaces
into track shoes at an
astounding rate of speed.

Figure A.10

A.7 Glossary

The text of clause A.7, as present in earlier releases of the DVB Subtitling Specification, has been integrated into the corresponding normative clause of the present document.

Annex B (informative): Bibliography

ISO/IEC 10646-1: "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane".

History

Document history		
Edition 1	September 1997	Publication as ETS 300 743
V1.2.1	June 2002	One-step Approval Procedure OAP 20021004: 2002-06-05 to 2002-10-04