

CEA Standard

Digital Television (DTV) Closed
Captioning

CEA-708-B

December 1999



CEA

Consumer Electronics Association

www.CE.org

NOTICE

CEA Standards, Bulletins and other technical publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards, Bulletins and other technical publications shall not in any respect preclude any member or nonmember of CEA from manufacturing or selling products not conforming to such Standards, Bulletins or other technical publications, nor shall the existence of such Standards, Bulletins and other technical publications preclude their voluntary use by those other than CEA members, whether the standard is to be used either domestically or internationally.

Standards, Bulletins and other technical publications are adopted by CEA in accordance with the American National Standards Institute (ANSI) patent policy. By such action, CEA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard, Bulletin or other technical publication.

This CEA Standard is considered to have International Standardization implication, but the International Electrotechnical Commission activity has not progressed to the point where a valid comparison between the CEA Standard and the IEC document can be made.

This Standard does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this Standard to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

(From Project Number 4749, formulated under the cognizance of the CEA R4.3 Television Data Systems Subcommittee.)

Published by

©CONSUMER ELECTRONICS ASSOCIATION 2002
Technology & Standards Department
2500 Wilson Boulevard
Arlington, VA 22201

**PRICE: Please call Global Engineering Documents, USA and Canada (1-800-854-7179)
International (303-397-7956), or
<http://global.ihs.com>**

All rights reserved
Printed in U.S.A.

PLEASE!

DON'T VIOLATE
THE
LAW!

This document is copyrighted by CEA and may not be reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies through entering into a license agreement. For information, contact:

Global Engineering Documents
15 Inverness Way East
Englewood, CO 80112-5704 or call
U.S.A. and Canada 1-800-854-7179, International (303) 397-7956
See <http://global.ihs.com> or email global@ihs.com

July 22, 2002

CLARIFICATION

TO: Recipients of EIA-708-B, Digital Television (DTV) Closed Captioning

NOTE:

- 1. Please replace page 28 with the page 28 enclosed. In the original publication of EIA-708-B, the maximum number of virtual rows and columns was unclearly defined. That is clarified in the attached page 28.**
- 2. Please replace page 41 with the page 41 enclosed. In the original publication of EIA-708-B, row count and column count were unclearly defined. That is clarified in the attached page 41.**

We are sorry for any inconvenience this may have caused.

Sincerely,

**Technology & Standards Staff
Consumer Electronics Association**

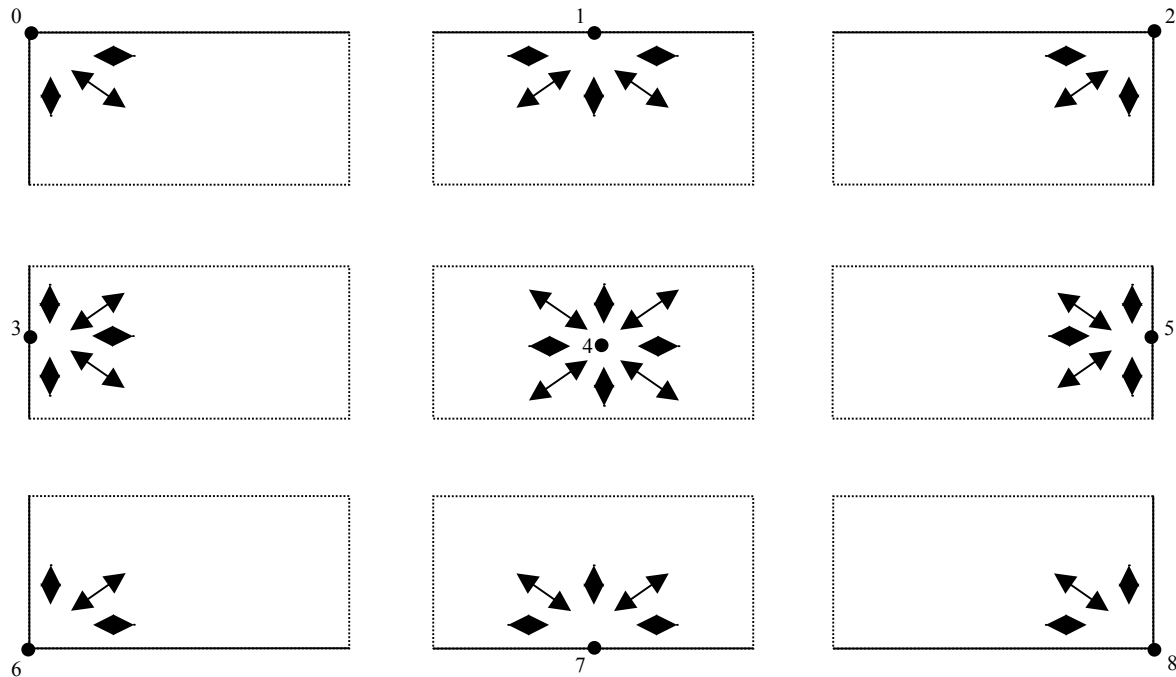


Figure 13 Implied Caption Text Expansion Based on Anchor Points

8.4.5 Anchor Location

The anchor location specifies where (in grid coordinates) the window's anchor point is to be physically located, and thus, the window itself. These grid coordinates are described in Section 8.2.

8.4.6 Window Size

The window size is specified in numbers of character rows and character columns for all display formats (16:9, 4:3, etc.). For all display formats, 15 is the maximum number of virtual rows and 32 is the maximum number of virtual columns. The DefineWindow parameter values are zero-based. See Section 8.10.5 Command Definitions, Define Window.

NOTE--A 16:9 format could handle longer rows (i.e., up to 42 characters), but caption providers should keep the window column size at 32 characters, or less, in order to leave room on the display when the user selects the LARGE font size (see Section 8.4.7).

As for the physical size of the window on the screen, the receiver scales the window based on the “effective font size”. The effective font size is based on a combination of the pen size chosen by the caption provider and the font size chosen by the receiver user. The height of the window is calculated as the number of rows multiplied by the physical height of the tallest character in the effective font size. The width of the window is calculated as the number of columns multiplied by the physical width of the widest character in the effective font size.

8.4.7 Window Row and Column Locking

The “lock rows” and “lock columns” window parameters fix the maximum number of rows and columns of caption text that a window may have. If a row or column parameter is “unlocked”, the receiver may automatically add or adjust columns or rows to a window under certain circumstances.

This means that in a text-mode or roll-up type caption service (with no embedded carriage returns) where the user has specified a font size smaller than intended by the caption provider, more rows and columns could fit into the physical window (as it appears on the screen) than specified in the original window size parameters (see examples below).

DEFINE WINDOW - (DF0 ... DF7)

Name:	DefineWindow - Create window and set initial parameters
Command Type:	Window
Format:	DefineWindow (<i>window ID</i> , <i>priority</i> , <i>anchor point</i> , <i>relative positioning</i> , <i>anchor vertical</i> , <i>anchor horizontal</i> , <i>row count</i> , <i>column count</i> , <i>row lock</i> , <i>column lock</i> , <i>visible</i> , <i>window style ID</i> , <i>pen style ID</i>)
Parameters:	<ul style="list-style-type: none"> ■ <u><i>window ID</i></u> (id) is the unique window identifier (0 - 7) ■ <u><i>priority</i></u> (p) is the window display priority (0 - 7) ■ <u><i>anchor point</i></u> (ap) is the window anchor position number to use for the window's position on the screen (0 - 8). ■ <u><i>relative positioning</i></u> (rp) is a flag that, when set to 1, indicates that the <i>anchor vertical</i> (av) and <i>anchor horizontal</i> (ah) coordinates specify “relative coordinates” (i.e., percentages) instead of physical screen coordinates. ■ <u><i>anchor vertical</i></u> (av) is the vertical position of the window's anchor point on the viewing screen when the window is displayed (0 – 74 for 16:9 and 4:3 systems and the <i>relative positioning</i> (rp) parameter is set to 0, or 0-99 when the ‘rp’ parameter is set to 1). ■ <u><i>anchor horizontal</i></u> (ah) is the horizontal position of the window's anchor point on the viewing screen when the window is displayed (0 - 209 for 16:9 systems, 0 - 159 for 4:3 systems and the <i>relative positioning</i> (rp) parameter is set to 0, or 0-99 when the ‘rp’ parameter is set to 1). ■ <u><i>row count</i></u> (rc) is the number of virtual rows of text (assuming the STANDARD <i>pen size</i>; see SetPenAttributes) the window body will hold minus one (0-11). For example, a window with rc=2 will have three virtual rows of text. ■ <u><i>column count</i></u> (cc) is the number of virtual columns of text (assuming the STANDARD <i>pen size</i>; see SetPenAttributes) the window body will hold minus one (0 - 31 for 4x3 formats, and 0 - 41 for 16x9 formats). For example, a window with cc=31 will have thirty-two virtual columns of text. ■ <u><i>row lock</i></u> (rl), when set to YES, fixes the absolute number of rows of caption text the window will contain. When NO, <i>row lock</i> permits a receiver to add more rows to a window if the user selects a smaller size font other than intended by the caption provider. [YES, NO] = [1, 0]. ■ <u><i>column lock</i></u> (cl), when set to YES, fixes the absolute number of columns of caption text the window will contain. When NO, <i>column lock</i> permits a receiver to add more columns to a window if the user selects a smaller size font other than intended by the caption provider. [YES, NO] = [1, 0]. ■ <u><i>visible</i></u> (v), when set to YES, causes the window to be viewed (i.e., displayed) on the screen immediately after it is created. When set to NO, the window is not displayed (i.e., hidden) after it is created. [YES, NO] = [1, 0]. ■ <u><i>window style ID</i></u> (ws), when non-zero, specifies 1 of 7 static preset window attribute styles to use for the window when it is created (0 - 7). When zero during a window create, the window style is automatically set to window style #1. When zero during a window update, no window attribute parameters are changed. See SetWindowAttributes command. ■ <u><i>pen style ID</i></u> (ps), when non-zero, specifies 1 of 7 static preset pen attribute styles to use for the window when it is created (0 - 7). When zero during a window create, the pen style is automatically set to pen style #1. When zero during a window update, no pen attribute parameters are changed. See SetPenAttributes command.

Contents

1 Scope	1
1.1 Overview	1
2 Normative References.....	1
2.1 Normative Reference List	2
2.2 Normative Reference Acquisition	2
3 Caption Channel Layered Protocol.....	2
4 DTVCC Transport Layer.....	5
4.1 9,600 Bits Per Second DTVCC Transport Channel.....	6
4.2 Pre-Allocated Bandwidth.....	6
4.3 NTSC Caption Data	6
4.4 DTV/MPEG-2 Picture User Data (User Bits)	6
4.4.1 Picture User Data Bitstream Construct.....	6
4.4.2 Frame Rates	7
4.4.3 Typical Video Signals	9
4.4.4 Latency	9
4.5 Caption Channel Service Directory in the PMT and EIT	10
4.5.1 PMT and EIT Constraints	10
4.5.2 PMT Bandwidth Requirements.....	10
4.5.3 EIT Bandwidth Requirements.....	10
4.5.4 Decoder Processing of the PMT, EIT and User Data	10
5 DTVCC Packet Layer.....	10
6 Caption Service Layer	11
6.1 Services.....	11
6.2 Caption Channel Service Blocks	12
6.2.1 Standard Service Block Header.....	13
6.2.2 Extended Service Block Header	13
6.2.3 Null Service Block Header	13
6.2.4 Service Block Data.....	13
6.2.5 Service Blocks in Caption Channel Packet.....	13
7 DTVCC Coding Layer - Caption Data Services (Services 1 - 63).....	14
7.1 Code Space Organization	14
7.2 Extending the Code Space	17
7.3 Unused Codes.....	17
7.4 Numerical Organization of Codes.....	17
7.4.1 C0 Code Set - Miscellaneous Control Codes	17
7.4.2 C1 Code Set - Captioning Command Control Codes	18
7.4.3 G0 Code Set - ASCII Printable Characters.....	19
7.4.4 G1 Code Set - ISO 8859-1 LATIN-1 Character Set.....	20
7.4.5 G2 Code Set - Extended Miscellaneous Characters.....	21
7.4.6 G3 Code Set - Future Expansion	22

7.4.7 C2 Code Set - Extended Control Code Set 1	23
7.4.8 C3 Code Set - Extended Control Code Set 2	24
8 DTVCC Interpretation Layer	25
8.1 DTVCC Caption Components	25
8.2 Screen Coordinates.....	25
8.3 User Options	26
8.4 Caption Windows	26
8.4.1 Window Identifier.....	27
8.4.2 Window Priority	27
8.4.3 Anchor Points.....	27
8.4.4 Anchor ID	27
8.4.5 Anchor Location	28
8.4.6 Window Size	28
8.4.7 Window Row and Column Locking.....	28
8.4.7.1 Effects When Choosing Smaller Font	29
8.4.7.2 Effects When Choosing Larger Font	30
8.4.8 Word Wrapping.....	30
8.4.9 Window Text Painting.....	31
8.4.9.1 Justification	31
8.4.9.2 Print Direction	31
8.4.9.3 Scroll Direction	32
8.4.9.4 Combining Text Painting Attributes.....	32
8.4.10 Window Display	32
8.4.11 Window Colors and Borders	33
8.4.12 Predefined Window and Pen Styles	33
8.5 Caption Pen.....	33
8.5.1 Pen Size	33
8.5.2 Pen Spacing	33
8.5.3 Font Styles	33
8.5.4 Character Offsetting.....	34
8.5.5 Pen Styles.....	34
8.5.6 Foreground Color and Opacity	34
8.5.7 Background Color and Opacity	34
8.5.8 Character Edges.....	34
8.5.9 Caption Text Function Tags	34
8.6 Caption Text	35
8.7 Caption Positioning	35
8.8 Color Representation	35
8.9 Service Synchronization.....	35
8.9.1 Delay Command.....	35
8.9.2 DelayCancel Command.....	36
8.9.3 Reset Command	36
8.9.4 Reset and DelayCancel Command Recognition.....	36
8.9.5 Service Reset Conditions	37
8.10 DTVCC Command Set	37
8.10.1 Window Commands	37
8.10.2 Pen Commands	38
8.10.3 Synchronization Commands.....	38
8.10.4 Caption Text.....	38

8.10.5 Command Descriptions	39
9 DTVCC Decoder Manufacturer Recommendations.....	57
9.1 DTVCC Section 4.2 - Pre-Allocated Bandwidth.....	57
9.2 DTVCC Section 6.1 - Services	57
9.3 DTVCC Section 6.2 - Caption Channel Service Blocks	57
9.4 DTVCC Section 7.1 - Code Space Organization.....	57
9.5 DTVCC Section 8.2 - Screen Coordinates.....	58
9.6 DTVCC Section 8.4 - Caption Windows	59
9.7 DTVCC Section 8.4.2 - Window Priority	59
9.8 DTVCC Section 8.4.6 - Window Size.....	59
9.9 DTVCC Section 8.4.8 - Word Wrapping.....	59
9.10 DTVCC Section 8.4.9 - Window Text Painting	59
9.10.1 Justification	59
9.10.2 Print Direction	59
9.10.3 Scroll Direction	59
9.10.4 Scroll Rate	60
9.10.5 Smooth Scrolling.....	60
9.10.6 Display Effects.....	60
9.11 DTVCC Section 8.4.11 - Window Colors and Borders	60
9.12 DTVCC Section 8.4.12 - Predefined Window and Pen Styles	60
9.13 DTVCC Section 8.5.1 - Pen Size.....	63
9.14 DTVCC Section 8.5.3 - Font Styles	63
9.15 DTVCC Section 8.5.4 - Character Offsetting	63
9.16 DTVCC Section 8.5.5 - Pen Styles.....	63
9.17 DTVCC Section 8.5.6 - Foreground Color and Opacity	63
9.18 DTVCC Section 8.5.7 - Background Color and Opacity	63
9.19 DTVCC Section 8.5.8 - Character Edges	63
9.20 DTVCC Section 8.8 - Color Representation	63
9.21 Character Rendition Considerations.....	65
9.22 DTVCC Section 8.9 - Service Synchronization.....	66
9.23 DTV to NTSC Transcoders	66
10 DTVCC Authoring and Encoding for Transmission.....	67
10.1 Caption Authoring and Encoding.....	67
10.2 Monitoring Captions	68
10.3 Encoder Interfacing	68
11 DTV Closed Captioning Content Package.....	69
11.1 Introduction	69
11.1.1 Frame Rates	70
11.1.2 Time Code.....	70
11.1.3 Caption Data	70
11.1.4 Caption Service Information	70
11.1.5 Interface data rates.....	71
11.2 CDP Detailed Specification.....	71
11.2.1 General Construct	71
11.2.2 cdp_header	71
11.2.3 time_code_section	73
11.2.4 ccdata_section	74

11.2.5 ccsvcinfo_section	75
11.2.5.1 Service information signalling	77
11.2.6 cdp_footer	77
11.2.7 future_section()	77
11.3 Serial Interface	78
11.3.1 Physical Interface.....	78
11.3.2 Operation of the CDP Serial Interface.....	79
Annex A (Informative)	81

Figures

Figure 1 DTV Closed-Captioning Protocol Model.....	4
Figure 2 DTVCC Caption Data in the DTV Bitstream	5
Figure 3 Caption Channel Packet.....	11
Figure 4 Caption Channel Packet Syntax.....	11
Figure 5 Service Block	12
Figure 6 Service Block Syntax	12
Figure 7 Standard Service Block Header	13
Figure 8 Extended Service Block Header	13
Figure 9 Null Service Block Header	13
Figure 10 Service Blocks in Caption Channel Packets (Example)	14
Figure 11 DTV 16:9 Screen and DTVCC Window Positioning Grid.....	26
Figure 12 Anchor Points.....	27
Figure 13 Implied Caption Text Expansion Based on Anchor Points.....	28
Figure 14 Examples of Caption Window Shrinking when User Picks Smaller Font.....	29
Figure 15 Examples of Caption Window Growing when Going to Larger Font	30
Figure 16 Examples of Various Justifications, Print Directions and Scroll Directions	32
Figure 17 Reset & DelayCancel Command Detector(s) and Service Input Buffers.....	36
Figure 18 Reset & DelayCancel Command Detector(s) Detail	37
Figure 19 Caption Authoring and Encoding into Caption Channel Packets	67
Figure 20 Relationship Between Caption Data and Picture Frames.....	68
Figure 21 Interface of Caption Data to ATSC Emission Encoding Equipment.....	69
Figure 22 DTVCC Transport Stream Decoder	81

Tables

Table 1 DTVCC Protocol Stack.....	3
Table 2 Closed-Caption Type (cc_type) Coding.....	7
Table 3 DTVCC Transport Channel Transmit Rate Parameters	8
Table 4 Aligned User Data and DTVCC Channel Packet Example.....	9
Table 5 Unaligned User Data and DTVCC Channel Packet Example.....	9
Table 6 DTVCC Code Space Organization	15
Table 7 DTVCC Code Set Mapping.....	16
Table 8 C0 Code Set.....	18
Table 9 C1 Code Set.....	18
Table 10 G0 Code Set	19
Table 11 G1 Code Set	20

Table 12 G2 Code Set	21
Table 13 G3 Code Set	22
Table 14 C2 Code Set.....	23
Table 15 C3 Code Set.....	24
Table 16 Cursor Movement After Drawing Characters.....	31
Table 17 G2 Character Substitution Table.....	58
Table 18 Screen Coordinate Resolutions & Limits.....	58
Table 19 Predefined Window Style ID's	61
Table 20 Predefined Pen Style ID's	62
Table 21 Minimum Color List Table.....	64
Table 22 Alternative Minimum Color List Table	65
Table 23 CDP General Construct.....	71
Table 24 CDP Header Syntax	72
Table 25 CDP Frame Rate	72
Table 26 CDP Time Code Section Syntax.....	74
Table 27 CDP CC Data Section Syntax	75
Table 28 CC Service Information Syntax	76
Table 29 CDP Footer Syntax	77
Table 30 future_section syntax	78
Table 31 CDP Source Connector Pinout (DTE Male)	78
Table 32 CDP Receiver Connector Pinout (DCE Female)	79
Table 33 TIA/EIA-574 Interface Parameters for CDP Stream	79

(This page intentionally left blank.)

FOREWORD

This document specifies the standards for Closed Captioning in Digital Television (DTV) technology. This standard was developed under the auspices of the Consumer Electronics Association (CEA, formerly CEMA) Technology & Standards R4.3 Television Data Systems Subcommittee in parallel with the U.S. Advanced Television Systems Committee's (ATSC) and the Advanced Television Grand Alliance's definition, design, and development of the audio, video and ancillary data processing standard for Advanced Television. The DTV standard developed by the Grand Alliance and other industry members is represented in ATSC A/53, and the informative document, ATSC A/54.

Users of this standard should note that, at some future point, it is expected that provisions necessary to accommodate NTSC line 21 data transmission will be established.

(This page intentionally left blank.)

DIGITAL TELEVISION (DTV) CLOSED CAPTIONING

1 Scope

This document is intended as a definition of DTV Closed Captioning (DTVCC) and provides specifications and guidelines for caption service providers, DTVCC decoder and encoder manufacturers, DTV receiver manufacturers, and DTV signal processing equipment manufacturers. This specification includes the following:

- a description of the transport method of DTVCC data in the DTV signal
- a description of DTVCC specific data packets and structures
- a specification of how DTVCC information is to be processed
- a list of minimum implementation recommendations for DTVCC receiver manufacturers
- a set of recommended practices for DTV encoder and decoder manufacturers

The use of the term “DTV” (Digital Television) throughout is intended to include, and apply to, HDTV (High Definition Digital Television) and SDTV (Standard Digital Television) which use the digital data stream specified in ATSC A/53 and related ATSC A/54.

1.1 Overview

DTV Closed Captioning is a migration of the closed-captioning concepts and capabilities developed in the 1970’s for NTSC television video signals to the high-definition television environment defined by the ATV (Advanced Television) Grand Alliance and standardized by the Advanced Television Systems Committee (ATSC). This new television environment provides for larger screens and higher screen resolutions, and higher data rates for transmission of closed-captioning data.

NTSC Closed Captioning consists of an analog waveform inserted into Line 21 of the NTSC Vertical Blanking Interval (VBI). This waveform provides a transport channel which can deliver 2 bytes of data on every field of video. This translates to 120 bytes per second (Bps), or 960 bits per second (bps). In contrast, DTV Closed Captioning is transported as a logical data channel in the DTV digital bit stream. Of the DTV bitstream bit rate (which is 19.4 Mbps for terrestrial broadcast, and 38.4 Mbps for cable), DTV-specific closed captioning is allocated 9600 bps. This increased capacity opens the possibilities for the simultaneous transmission of captions in multiple languages and at multiple reading levels.

The DTV standard accommodates a variety of increased vertical and horizontal screen resolutions (e.g., 480 x 704, 720 x 1280 and 1080 x 1920), versus the single 525 vertical scan line format for NTSC. These added resolutions provide for more defined representations of character fonts and other on-screen objects.

The heart of DTVCC caption display is the caption “window” which is identical to the *window* concept found in all computer Graphical User Interfaces (GUI). Windows are placed within the DTV screen, and caption text is placed within windows. Windows and text have a variety of color, size and other attributes.

This document describes the above issues in a reverse-hierarchical (i.e., low-to-high level) fashion. It follows an “Open Systems Interconnect (OSI) Reference Model”-type protocol stack for layered protocols. DTVCC consists of 5 protocol layers: the Transport Layer, the Packet Layer, the Service Layer, the Coding Layer, and the Interpretation Layer. The discussion of the first 3 layers is a detailed presentation of data transport and organization issues. The discussion of the last 2 layers provides a more informative presentation of the unique aspects of closed captioning. Some readers may wish to start with these last 2 layers first, beginning in section 7.

Throughout EIA-708-B, in concert with ATSC A/53, the lowest numbered bit in a multibit numbered value is considered to be least significant (uimbsf).

2 Normative References

The following references contain provisions, which, through reference in this text, constitute normative provisions of this standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below.

2.1 Normative Reference List

ANSI X3.4, Information Systems - Coded Character Sets - 7-bit American National Standard Code (1986 (R1997))

ANSI X3.41, Code Extension Techniques for Use with the 7-Byte Coded Character Set of ASCII (1990)

ATSC A/53, ATSC Digital Television Standard (1995)

ATSC A/65, Program and System Information Protocol for Terrestrial Broadcast and Cable (1997)

EIA-608-A, Line 21 Data Service (1999)

ISO/IEC 2022, Information technology - Character code structure and extension techniques (1994)

ISO/IEC 8859-1, Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1 (1998)

ISO/IEC 13818-1, Information technology - Generic coding of moving pictures and associated audio information: Systems (1997)

SMPTE Standard 12M, Television, Audio and File – Time and Control Code (1999)

TIA/EIA-574, 9 Position Non-Synchronous Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange (1998)

2.2 Normative Reference Acquisition

ANSI or EIA or TIA/EIA Standards:

- Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO USA 80112-5776; Phone 800-854-7179; Fax 303-397-2740; Internet <http://global.ihs.com>; Email global@ihs.com

ATSC Standards:

- Advanced Television Systems Committee (ATSC), 1750 K Street N.W., Suite 1200, Washington, DC 20006; Phone 202-828-3130; Fax 202-828-3131; Internet <http://www.atsc.org/standards.html>

ISO/IEC Standards:

- Global Engineering Documents, World Headquarters, 15 Inverness Way East, Englewood, CO USA 80112-5776; Phone 800-854-7179; Fax 303-397-2740; Internet <http://global.ihs.com>; Email global@ihs.com
- IEC Central Office, 3, rue de Varembe, PO Box 131, CH-1211 Geneva 20, Switzerland; Phone +41 22 919 02 11; Fax +41 22 919 03 00; Internet <http://www.iec.ch>; Email pubinfo@iec.ch
- ISO Central Secretariat, 1, rue de Varembe, Case postale 56, CH-1211 Geneve 20, Switzerland; Phone +41 22 749 01 11; Fax +41 22 749 01 55; Internet <http://www.iso.ch>; Email mbinfo@iso.ch

SMPTE Standards:

- Society of Motion Picture & Television Engineers, 595 W. Hartsdale Ave., White Plains, NY 10607-1824 USA Phone: 914-761-1100 Fax: 914-761-3115, Email: eng@smpte.org; Web: <http://www.smpte.org>

3 Caption Channel Layered Protocol

A formal data communications channel protocol has been established for the DTVCC caption data channel. This formalization provides a framework for describing the caption communications hierarchy. Grouping the structures, concepts, and features of this environment into the following hierarchical layers aids in the understanding of the organizational aspects of the DTVCC system.

There are 5 layers in the Caption Channel data framework: Transport Layer, Packet Layer, Service Layer, Coding Layer, and Interpretation Layer (see Figure 1). These layers map to the top-most layers of the OSI Reference Model as shown in Table 1.

OSI Protocol Reference Model	DTVCC Protocol Model
Application	Interpretation Layer
Presentation	Coding Layer
Session	Service Layer
-	Packet Layer
Transport	Transport Layer
Network	-
Link	-
Physical	-

Table 1 DTVCC Protocol Stack

The DTVCC Transport Layer maps to the OSI Transport Layer. This layer marks where DTVCC data leave the DTV Video subsystem and are introduced to the DTV Closed-Caption decoder in the receiving equipment. Within the DTVCC decoder, DTVCC data are further processed up through the remaining layers of the DTVCC Protocol Model.

The DTVCC Packet Layer marks where DTVCC data enter the DTVCC decoder. This is a protocol data reassembly layer which buffers incremental bitstream data into a byte-aligned, multi-byte packet. There is no specific counterpart in the OSI model for this layer.

Processing of the Caption Channel packet data begins in the DTVCC Service Layer. Caption Channel packets are broken up into the encapsulated sub-blocks of data to be routed to the separate caption service processing routines within the decoder. Services define separate caption data streams. TV viewers may choose to view the processed data for one or more services at a time. For example, a caption channel may contain an English language service and a Spanish language service.

The DTVCC Coding Layer breaks out the individual caption commands and caption text sequences from the service data blocks. The DTVCC Coding Layer maps to the OSI Presentation Layer.

The Interpretation Layer processes the caption elements presented by the DTVCC Coding Layer. The Interpretation Layer maps to the OSI Application Layer.

This layered framework for DTVCC is further detailed in the remaining sections of EIA-708-B.

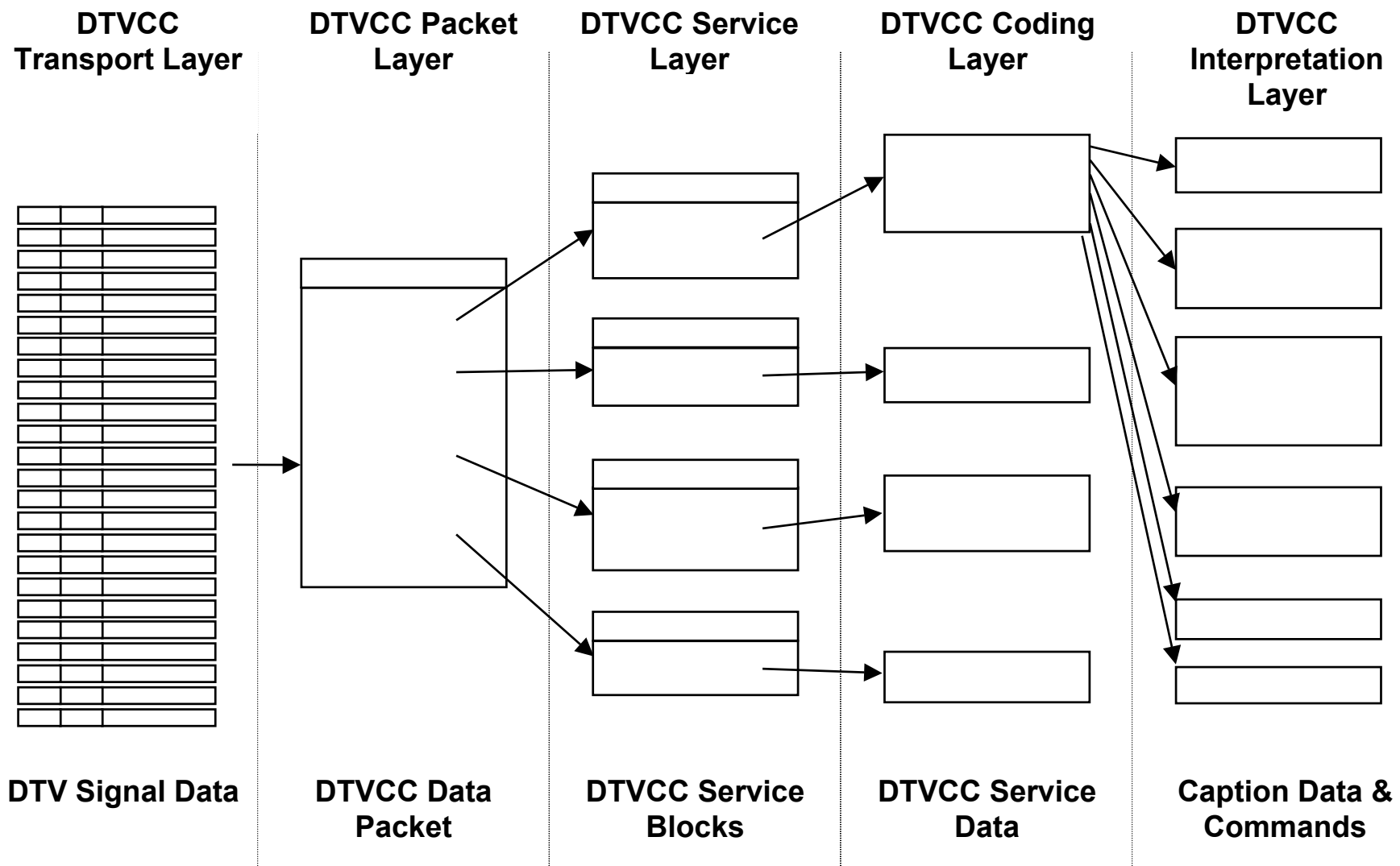


Figure 1 DTV Closed-Captioning Protocol Model

4 DTVCC Transport Layer

The transport of the Caption Channel is defined in ATSC A/53 and ISO/IEC 13818. The DTVCC Transport Layer consists of the mechanisms for transporting caption data from the encoder at the caption-encoding head-end to the decoding hardware in the TV receiver. DTVCC related data, when present, is transported in three separate portions of the DTV stream: the Picture User Data, the Program Mapping Table (PMT) and the Event Information Table (EIT). DTVCC Service Data (caption text, window commands, etc) are carried as MPEG-2 Picture User Data, and the DTVCC Caption Channel Service Directory is carried as descriptor information in the PMT and, when present, the EIT.

The DTV video bitstream, the PMT and the EIT are multiplexed with the other audio, data, control and synchronization bit streams comprising the DTV system signal as depicted in Figure 2.

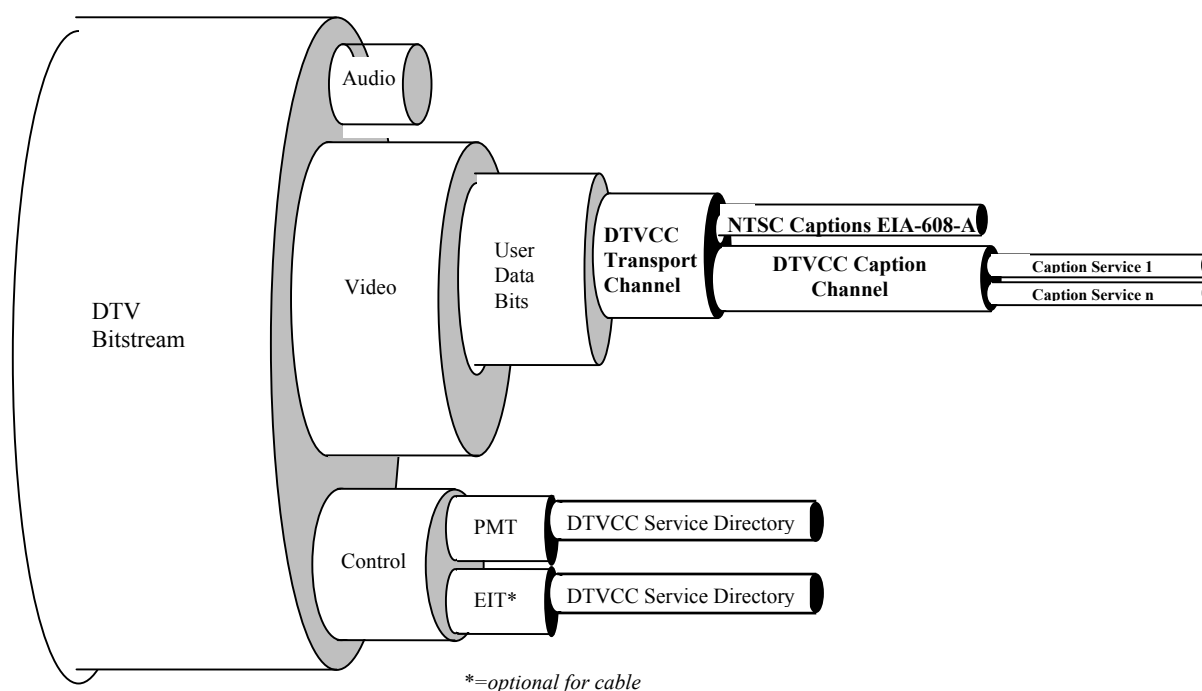


Figure 2 DTVCC Caption Data in the DTV Bitstream

4.1 9,600 Bits Per Second DTVCC Transport Channel

The DTVCC design allows for caption data to be transmitted at various data rates within a DTV signal or any similar MPEG-2 variation. However, for DTV and DTVCC specific (i.e., ATSC A/53) caption encoding, the DTV Closed-Captioning channel is a continuous 9600 bps stream allocated from the DTV signal capacity. In order to provide this continuous stream and to ensure that specific caption data will reach receivers as intended (in relationship to the audio and video), this data channel is allocated on a frame-by-frame basis such that 1200 bytes of data are transported per second. For example, for an interleaved DTV video signal with a 60 Hz frame rate, 20 bytes of DTVCC caption channel data are allocated in each frame.

It is very important to note that the allocation for NTSC closed-caption encoding within the DTVCC Transport Channel (i.e., Picture User Data) is included in the DTVCC Caption Channel bandwidth. That is, the total DTVCC Transport Channel (consisting of the DTVCC Caption Channel and the NTSC Caption Channel) within the Picture User Data is 9600 bps. On average, NTSC (EIA-608-A) captions are allocated 960 bps, and DTVCC captions (EIA-708-B) are allocated 8640 bps.

4.2 Pre-Allocated Bandwidth

The DTVCC Transport Channel is a fixed, pre-allocated stream which exists in all DTV-system bit streams, even though captions may not be present. This ever-present bandwidth (which includes NTSC and DTVCC caption data) allows encoders to easily insert caption data into the DTV bit-stream at the point of origin and at multiple downstream encoding points without having to perform complex picture data processing and bandwidth re-allocation. In addition, it allows for easier decoding of the closed-caption data by the receiver since the data will always be at known locations.

4.3 NTSC Caption Data

The consideration for NTSC captioning (EIA-608-A) exists to facilitate the transcoding of the DTV video to NTSC video, and to preserve all line 21 data during the transcode process. The NTSC closed-captioning allocation is detailed further in this section. Section 9.23 provides a further discussion of the transcoding issue.

The NTSC closed caption data bytes are not embedded within the DTV closed-caption protocol stack. That is, they are not passed onto the DTVCC Packet Layer; rather, they are extracted at the Transport Layer and routed to a separate NTSC caption decoder (if present). This allows for simpler closed-caption decoder implementations (e.g., DTV-to-NTSC settop transcoders) since the entire DTVCC Caption Channel data stream does not have to be parsed to find a few bytes of NTSC data.

This is especially true for devices (e.g., set-top boxes) which are to convert DTV signals into NTSC to accommodate NTSC television receivers. These devices ignore the DTVCC closed-caption data in the DTV Picture User Data bits and transcode the NTSC closed-caption data into the NTSC Line-21 data stream. This allows for easier insertion of NTSC closed-caption data by the encoder since the NTSC data bytes do not have to be buried within the DTVCC Coding Layer.

4.4 DTV/MPEG-2 Picture User Data (User Bits)

While Video User Bits may be inserted at any of three levels (i.e., Sequence level, the Group of Pictures (GOP) level, and the Picture Data level) within the video bitstream sequence, ATSC A/53 specifies that the DTVCC Transport Channel is only encoded at the Picture Data level (i.e., user data level 2).

The following is a technical discussion of the DTV Picture User Data format. This discussion is intended for the technical audience, such as encoder and decoder manufacturers. The bitstream description syntax convention used in the two referenced documents is also used here to describe the DTV user data. Please refer to these documents for a normative explanation of this syntax.

4.4.1 Picture User Data Bitstream Construct

The `extension_and_user_data(2)` and `user_data()` constructs defined in ATSC A/53 carry the DTVCC data within the MPEG Video Picture User Data Level. This section augments the definition of the `cc_type` and `cc_valid` elements in the `user_data(2)` construct.

`cc_valid` - This flag indicates whether the following two closed-caption data bytes are valid. If `cc_valid = 1`, then the following two bytes of closed-caption data are valid. Otherwise, the two closed caption bytes are invalid (i.e., null).

filled). For NTSC Line 21, Field 1 and Field 2 captions, if *cc_valid* = 0 then the run-in clock and start bit should not be generated for the NTSC data stream when transcoding from DTV to NTSC.

cc_type - Denotes the type of the following two bytes of closed caption data as indicated in Table 2.

cc_type	Contents
00	NTSC line 21 field 1 closed captions
01	NTSC line 21 field 2 closed captions
10	DTVCC Channel Packet Data
11	DTVCC Channel Packet Start

Table 2 Closed-Caption Type (*cc_type*) Coding

DTV Closed-Caption data is transmitted (encoded) in variable-rate, variable-sized DTVCC Caption Channel Packets (see Section 5). To allow for simple extraction and insertion of these packets without the need to fully parse their contents, the first byte-pair of each DTVCC Caption Channel Packet shall be marked using the *cc_type* = 11 syntax flag (i.e., DTVCC Caption Channel Packet Start). The remaining byte-pairs of the DTVCC packet shall be marked with the *cc_type* = 10 flag (i.e., DTVCC Caption Channel Packet Data).

A DTVCC Caption Channel Packet may continue from one picture user_data extension to the next. The end of the packet is indicated by either: (1) receipt of the header (*cc_valid* = 1, *cc_type* = 11) of the next DTVCC packet, or (2) receipt of a byte-pair where *cc_valid* = 0 and *cc_type* = 10 or *cc_type* = 11.

NOTE--In the DTV video compression standard, coded pictures are transmitted in a different order (“Transport” order) than they are displayed (“Display” order). Since the DTVCC captioning data extensions are part of the video coded picture constructs and follow the MPEG Video coded-picture reordering process, the order in which the captioning data extensions are transmitted is not the same order in which they will be processed by the DTVCC Decoder. The captioning data extensions must be reordered (by the “MPEG Video Decoder”) along with the pictures to which they correspond prior to the DTVCC Caption Channel Packet location and extraction method described previously. Once these captioning data are reordered, they will be ready for processing by the DTVCC Decoder.

4.4.2 Frame Rates

As stated above, the value of *cc_count* will vary depending on the frame rate. In any given video service, the frame rate may change (e.g. to “film-mode” during extended still picture periods). The *user_data()* construct shall be transmitted a number of times per second, and *cc_count* shall be adjusted for each such construct so that the 9600 bps DTV Transport Channel throughput is sustained no matter what the instantaneous frame rate is.

The frame rate for a DTV video stream is dependent upon a combination of the values for the following MPEG-2 parameters (i.e., video bitstream elements): *frame_rate_code*, *progressive_sequence*, *top_field_first*, *repeat_first_field* and *picture_structure*. These parameters are defined in ISO/IEC 13818. When defining the frame rate, these parameters also govern the *user_data()* construct transmit rate and the value of *cc_count*.

Table 3 shows the values derived for the *user_data(2)* transmit rate, and the *cc_count* values per each frame such that the DTVCC Transport Channel data rate of 9,600 bps is maintained.

Table 3 also shows the relationship between the number of DTV data bytes vs. NTSC bytes contained in the DTVCC user data.

NTSC captioning information shall not exceed the data rate of an NTSC display (30/1.001 * 2 bytes per second). The number of *cc_count* of each NTSC *cc_type* shall be no greater than 30 /1.001 per second while the stream is active. This data rate may be maintained by averaging the number of each *cc_type* over up to 1 001/30 seconds.

This requirement must be met across splices. For example, when two sequences with *frame_rate_value* of either 30 or 60 are spliced, the splicer could delete NTSC caption information from either the 1/30 second of the ending sequence or the first 1/30 second of the new sequence.

NTSC captioning information should maintain the order of an NTSC display. If both `cc_type 00` and `cc_type 01` data are present in the picture user data constructs of a sequence, they should alternate when the pictures are restored to display order.

For video sequences with `progressive_sequence == 0` for pictures with `top_field_first == 0`, `cc_type 00` shall precede `cc_type 01`.

For video sequences with `progressive_sequence == 0` for pictures with `top_field_first == 1`, `cc_type 01` shall precede `cc_type 00`.

For picture_structure of Top Field, `cc_type 00` shall not be present.

For picture_structure of Bottom Field, `cc_type 01` shall not be present.

For video sequences with `progressive_sequence == 1`, the ordering of `cc_type 00` and `cc_type 01` is arbitrary. Either 4 or 6 NTSC `cc_data` bytes may be included in each `user_data()` structure, subject to the data rate restriction above. DTV `cc_data` bytes may also be present to produce a total data rate of 9120 bps, subject to the constraint on `cc_count`.

When the picture display time is extended from the nominal value of `frame_rate` stated in the video sequence header through the repeat mechanisms in the picture_coding_extension, the values of `cc_count` and number of NTSC `cc_data_bytes` in `user_data()` shall be modified. Values for `cc_count` and `cc_data_bytes` are shown in Table 3.

Frame_rate_value	picture_structure	progressive_frame	repeat_first_field	top_field_first	cc_count	NTSC cc_data_bytes	DTV cc_data_bytes
60 or 59.94	11	1	0	0	10	2	18
60 or 59.94	11	1	1	0	20	4	36
60 or 59.94	11	1	1	1	30	6	54
30 or 29.97	01 or 10	0	0	0	10	2	18
30 or 29.97	11	x	0	x	20	4	36
30 or 29.97 (note)	11	1	1	x	30	6	54
24 or 23.97	11	1	0 (required)	0 (required)	25	4 or 6	44 or 46
(x = may be 0 or 1) Note: For sequences with <code>frame_rate_value = 30</code> , or <code>29.97</code> , <code>progressive_frame == 1</code> and <code>repeat_first_field == 1</code> implies <code>progressive_sequence == 0</code> .							

Table 3 DTVCC Transport Channel Transmit Rate Parameters

For ease of inserting and retrieving the NTSC closed captions, NTSC Line 21, Field 1 and Field 2 captions must always be placed in the User Data stream before any DTVCC caption data. The NTSC closed-caption data order (Field 1 vs. Field 2) will depend on the field display order. With this scheme, once the decoder sees `cc_type = 10` or `cc_type = 11`, indicating DTVCC caption data, it will no longer continue searching the User Data for NTSC captions. All NTSC closed-caption data will follow the format and protocol as specified in EIA-608-A.

In the case of a 60 Hz frame rate, 2 bytes of NTSC captions are allocated per `user_data` extension. Two (2) bytes of Field 1 NTSC captions are to be encoded every other frame, with 2 bytes of Field 2 NTSC captions encoded in the intervening frames. In all other cases (i.e., lower frame rates), Field 1 and Field 2 captions exist within the same `user_data` extension.

4.4.3 Typical Video Signals

Table 4 illustrates an example of the arrangement of the closed-caption data stream in the Picture User Data for a typical 30 frames per second video signal frame where *cc_count* = 20. A Caption Channel Packet starts immediately after the NTSC caption data.

	Marker_bits	cc_valid	cc_type	cc_data1	cc_data2
1	1111 1	1	00	NTSC Line 21 field 1	NTSC Line 21 field 1
2	1111 1	1	01	NTSC Line 21 field 2	NTSC Line 21 field 2
3	1111 1	1	11	byte #1: DTVCC Pkt Header	byte #2: DTVCC Pkt Data
4	1111 1	1	10	byte #3: DTVCC Pkt Data	byte #4: DTVCC Pkt Data
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•
20	1111 1	1	10	byte #35: DTVCC Pkt Data	byte #36: DTVCC Pkt Data

Table 4 Aligned User Data and DTVCC Channel Packet Example

Table 5 shows the User Data frame encoding of unaligned Caption Channel Packets. The end of a previous Caption Channel Packet (that was 128 bytes in length) is in the first 4 bytes of the User Data after the NTSC caption data. The start of the first 32 bytes of a new Caption Channel Packet fills the remainder of the 40 bytes of the User Data for a 30 frames per second video signal.

	Marker_bits	cc_valid	cc_type	cc_data1	cc_data2
1	1111 1	1	00	NTSC Line 21 field 1	NTSC Line 21 field 1
2	1111 1	1	01	NTSC Line 21 field 2	NTSC Line 21 field 2
3	1111 1	1	10	byte #125: DTV Pkt Data	byte #126: DTV Pkt Data
4	1111 1	1	10	byte #127: DTV Pkt Data	byte #128: DTV Pkt Data
5	1111 1	1	11	byte #1: DTVCC Pkt Header	byte #2: DTVCC Pkt Data
6	1111 1	1	10	byte #3: DTVCC Pkt Data	byte #4: DTVCC Pkt Data
7	1111 1	1	10	byte #5: DTVCC Pkt Data	byte #6: DTVCC Pkt Data
8	1111 1	1	10	byte #7: DTVCC Pkt Data	byte #8: DTVCC Pkt Data
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•
20	1111 1	1	10	byte #31: DTVCC Pkt Data	byte #32: DTVCC Pkt Data

Table 5 Unaligned User Data and DTVCC Channel Packet Example

The *marker_bits*, *cc_type* and *cc_valid* fields DO NOT use any part of the 9,600 bps bandwidth allocated for closed-caption data. Therefore, 9,600 bps will be maintained for the transport of closed-caption information, including NTSC and DTVCC captions, within the DTV stream.

4.4.4 Latency

As previously discussed in Section 4.4.1, the DTV picture data are transmitted in “Transport Order”. This transmission path has an inherent latency associated with it, in that there can be a significant delay from the time the picture data enter a DTV encoder to the time these same data exit a DTV decoder and are available for display processing. The DTVCC data experience this same delay since they are transmitted within the Picture User Data bits, even though the actual DTVCC data are introduced to the encoder in “Display Order”, as is the video. ATSC A/53 specifies that this latency may not exceed 0.5 seconds.

This requirement imposes a constraint on the *vbv_delay* parameter of the MPEG-2 Picture Header bitstream construct. This value is the Video Buffering Verifier (VBV) Delay. The VBV is a hypothetical decoder that is conceptually connected to the output of the DTV encoder. Its purpose is to provide a constraint on the variability of the data that an encoder or editing process may produce.

vbv_delay is a 16-bit unsigned integer in the Picture Header. For constant bit rate operation, *vbv_delay* is used to set the initial occupancy of the decoder's video buffer at the start of play so that the decoder's buffer does not overflow or underflow. *vbv_delay* measures the time needed to fill the VBV buffer from an initially empty state at the target bit rate, *R*, to the correct level immediately before the current picture is removed from the buffer.

The value of *vbv_delay* is the number of periods of the 90 kHz system clock that the VBV shall wait after receiving the final byte of the picture start code. For a 0.5 second delay, the *vbv_delay* value is 45,000 (i.e., 45,000 cycles / 90,000 cycles per second = 0.5 seconds). If *vbv_delay* ≤ 45,000 for a picture start code, it means that the closed captions in that picture data will experience a delay of less than 0.5 second in the decoder buffer.

4.5 Caption Channel Service Directory in the PMT and EIT

The Caption Channel Service Directory describes the types and attributes of the Caption Services (see Section 6) encoded in the Picture User Data bits. The Caption Channel Service Directory is contained in the *caption_service_descriptor()*. The *caption_service_descriptor()* is defined by ATSC A/65, and is carried in the PMT and, when present, the EIT of the MPEG-2 transport stream. Reference ATSC A/65 for a complete description of the *caption_service_descriptor*.

4.5.1 PMT and EIT Constraints

The following constraints shall be placed on the use of the caption service descriptor:

- 1) One caption service descriptor shall be present in the PMT and, when present, the EIT to describe each caption service present within the video Picture User Data.
- 2) There shall be no more than 16 simultaneous caption services present.

4.5.2 PMT Bandwidth Requirements

For the case where the PMT is repeated at the minimum rate of once per 400 msec, the bandwidth needed to deliver a full set of 16 caption service descriptions is 1.98 kbps (i.e., 3 bytes overhead + (6 bytes/description * 16 descriptions) = 99 bytes. 99*8=792 bits. 792 * 2.5 reps/second = 1980 bps). Packetization overhead is not included in this rate.

4.5.3 EIT Bandwidth Requirements

According to ATSC A/65, the recommended cycle time for the currently applicable EIT is 500 msec. Using the same assumptions as the previous paragraph (full set of services for a total of 99 bytes), the bandwidth required is 1.584 kbps.

4.5.4 Decoder Processing of the PMT, EIT and User Data

Even though the Caption Service Descriptors are encoded in the PMT and, when present, the EIT, the decoder only needs to acquire this information from one location. For terrestrial broadcasts, decoders may acquire service descriptors from the EIT only. The EIT carries all relevant event information for a particular program (program titles, duration, ratings, caption directory, etc.). Acquiring all program information from a single, concise table facilitates fast tuning.

In other transmission environments, it may be preferable to use the PMT for acquisition of the Caption Service Descriptor. See Annex A (informative).

5 DTVCC Packet Layer

The DTVCC Caption Channel data within the DTVCC Transport Channel are framed within data “packets” prior to encoding. The DTVCC Caption Channel Packet is totally defined within EIA-708-B. Error correction, error detection, compression, and other low-level transport overhead issues are handled in the transmission DTV layers, and thus fall into the purview of the ATSC and MPEG-2 standards.

The DTVCC Packet Layer is defined by a Caption Channel Packet of *n* bytes of closed captioning data, where *n* ≤ 128 and *n* is an even number. Packets are coded in the *user_data* extensions of the coded pictures in the DTV stream. The beginning and ending of each DTVCC packet in the User Data is indicated in the syntax defined in Section 4.4.1.

The Caption Channel Packet consists of a 1 byte header and $n-1$ bytes of packet data, where n is the total packet size. See Figure 3. The Caption Channel Packet Header contains the packet_size code and a sequence_number. The packet_size_code is in the lower 6 bits ($b_0 - b_5$) of the header; its value is “0” when the packet size (n) is 128 bytes and “ $n/2$ ” when the packet size is less than or equal to 126 bytes.

Sequence_number is a 2-bit ($b_6 - b_7$) rolling sequence number (0 - 3) which is used by receivers to determine lost Caption Channel Packets. When a lost packet is detected, any partially accumulated data from the previously received packet is to be discarded, and the processing associated with the Reset command should be performed for each existing service. See section 8.9.5. Decoding shall resume with the first Service Block encountered in the new packet.

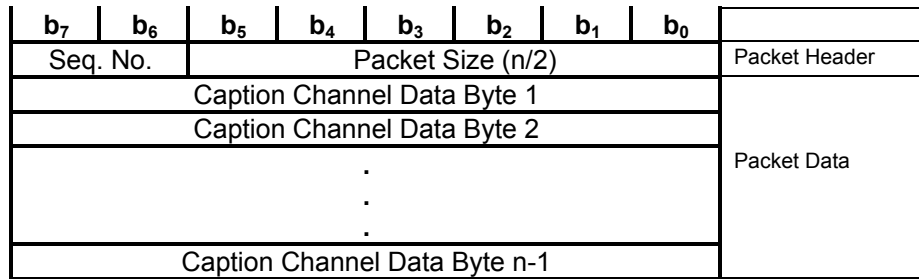


Figure 3 Caption Channel Packet

The syntax for the caption_channel_packet is shown in Figure 4.

```
caption_channel_packet {
    /* packet header */
    sequence_number: 2 bits
    packet_size: 6 bits

    /* packet data sequence */
    /*if (packet_size_code == 0)*/
        /*packet_data_size = 127 */
    /*else*/
        /*packet_data_size = (packet_size_code * 2) - 1 */
    for (i = 0; i < packet_data_size; i++)
    {
        Packet_data[i]    : 8 bits;
    }
}
```

Figure 4 Caption Channel Packet Syntax

6 Caption Service Layer

The DTVCC Caption Channel is divided into a set of logical sub-channels; i.e., “services”. Service data are inserted in the Caption Channel when required and where it fits in the caption channel data stream (i.e., Time Division Multiplexed).

The Service Layer defines the headers for the caption data channel service numbers, service types, and service attributes. Receivers use the information in this layer to route the caption packets to the appropriate internal processing modules.

6.1 Services

Caption Channel services are virtual sub-channels in the Caption Channel stream. There are 6 standard services and up to 57 additional extended services allowing for 63 total services.

Service #1 is designated as the Primary Caption Service. This service contains the verbatim, or near-verbatim captions for the primary language being spoken in the accompanying program audio.

Service #2 is designated as the Secondary Language Service. This service contains captions in a secondary language which are translations of the captions in the Primary Caption Service.

The other service sub-channels are not pre-assigned. It is up to the discretion of the individual caption provider to utilize the remaining service channels.

6.2 Caption Channel Service Blocks

Caption Channel Service Blocks provide the structures for the asynchronous time division multiplexing of service data within the DTV Closed-Caption Channel. Caption providers and encoding equipment algorithms govern the frequency, priority, and bandwidth consumption for each individual service by these service blocks, as needed.

A Caption Channel Service Block consists of a Service Block Header followed by from 1 to 31 bytes of service data. See Figure 5. As described in 6.2.1, a Service Block Header may consist of 1 or 2 bytes.

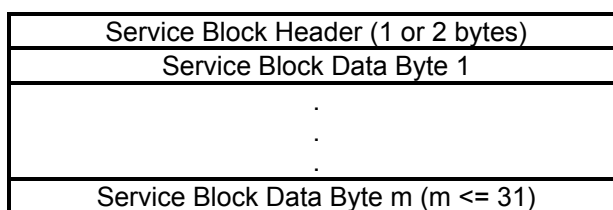


Figure 5 Service Block

NOTE--A Service Block cannot span a Caption Channel Packet. Service Blocks must begin and end within the same Caption Channel Packet.

The syntax for the Service Block is shown in Figure 6.

```

service_block {
    /* service block header */
    block_size      : 5 bits /*  >= 0 && <= 31      */
    service_number  : 3 bits

    /* extended service block header extension */
    if (service_number == b'111' && block_size != 0)
    {
        extended_service_number : 6 bits
        null_fill                : 2 bits
    }

    /* block data sequence */
    if (service_number != 0)
    {
        for (i = 0; i < block_size; i++)
        {
            Block_data[i]      : 8 bits
        }
    }
}

```

Figure 6 Service Block Syntax

6.2.1 Standard Service Block Header

The one-byte Standard Service Block Header consists of 2 parts: the Service Number (sn), and the Service Block Size (bs). See Figure 7. The Service Number (sn₀ - sn₂) is defined in the 3 high-order bits of the block header byte, and the Block Size (bs₀ - bs₄) is defined in the 5 low-order bits of the block header byte.

“Standard” service numbers range from 1 to 6 (service #0 is reserved), with Service #1 being reserved for the Primary Caption Service, and Service #2 reserved for the Secondary Language Service. A service number value of 7 in bits sn₀ - sn₂ indicates an “Extended” Service Block (Section 6.2.2).

The service Block Size value ranges from 1 to 31 and indicates the number of bytes following the header. A Standard Service Block Size of 0 is meaningless.

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
sn ₂	sn ₁	sn ₀	bs ₄	bs ₃	bs ₂	bs ₁	bs ₀

Figure 7 Standard Service Block Header

6.2.2 Extended Service Block Header

In the event that a Caption Channel requires more than 6 simultaneous services, a 2-byte Extended Service Block Header is used. See Figure 8. The first byte of this extended header has the same format as the Standard Service Block Header with the exception that the high order 3 bits are fixed to a value of all 1's. This standard service number value of “7” identifies the Extended Service Block and indicates that the 2nd byte of the header contains an extended service number (sn₀ - sn₅) whose value may range from 7 to 63. Extended service numbers less than 7 are illegal (since they can already be specified with the Standard Service Block Header).

The Block Size (bs₀ - bs₄) value in the lower 5 bits of the first byte of the Extended Service Block Header is the number of bytes following the Extended Service Block Header.

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
1	1	1	bs ₄	bs ₃	bs ₂	bs ₁	bs ₀
0	0	sn ₅	sn ₄	sn ₃	sn ₂	sn ₁	sn ₀

Figure 8 Extended Service Block Header

6.2.3 Null Service Block Header

A Null Service Block Header must be inserted as the last Service Block in the Caption Channel Packet, space permitting. This header type indicates that there are no more Service Blocks in the packet for the DTVCC decoding hardware to process. See Figure 9. Encoding equipment should null fill (i.e., zero out) the Caption Channel Packet buffer before Service Blocks are inserted, thus insuring that a null header is always present in non-full Caption Channel Packets.

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀
0	0	0	0	0	0	0	0

Figure 9 Null Service Block Header

6.2.4 Service Block Data

Up to 31 bytes of Service Data follow the Service Block Header within the Service Block. The contents of the data portion of the service block consists of caption data coding and interpretation (Sections 7 and 8). Service Block Data are separated from the Service Block and are routed to the appropriate service processors in the DTV receiver decoder. This separation process creates the individual byte streams for each of the services which are handed off to the DTVCC Coding Layer.

6.2.5 Service Blocks in Caption Channel Packet

Service Blocks are time division multiplexed and inserted sequentially in the Caption Channel Packet path (Section 4.5).

Service Blocks may not cross Caption Channel Packet boundaries; i.e., if a service requires more data than the current packet allows, then the service block is truncated to fit the current packet, and a new Service Block with a new Service Block Header and the remaining data bytes are placed at the start of, or elsewhere, within the next packet.

Figure 10 shows an example of a Caption Channel Data Packet with 3 Standard Service Blocks and 1 Extended Service Block. The Caption Channel Packet size is 20 bytes and the Packet Sequence Number is 2.

		Packet Byte							
Pkt Size: 20/2, Seq# 2		1	0	0	0	1	0	1	0
SN: 1, BS: 3		0	0	1	0	0	0	1	1
SN: 6, BS: 4		1	1	0	0	0	1	0	0
ESN: 21, BS: 8		1	1	1	0	1	0	0	0
		0	0	0	1	0	1	0	1
Null Service Header		0	0	0	0	0	0	0	0

Figure 10 Service Blocks in Caption Channel Packets (Example)

7 DTVCC Coding Layer - Caption Data Services (Services 1 - 63)

The DTVCC Coding Layer describes how data are coded for the caption channel services. The Caption Data Coding Layer defines the assignment of numerical codes for code space control, caption commands, and caption characters and symbols.

7.1 Code Space Organization

Consistent with ANSI X3.41 and ISO 2022, the 256 position code space is divided into four code groups: the CL, GL, CR and GR. See Table 6 and Table 7. Each group contains a standard code set and an extended code set.

- The CL group contains the 32 addressable codes from 00h to 1Fh. The C0 (a subset of the ASCII, ANSI X3.4 Miscellaneous Control Codes) and C2 (Extended Miscellaneous Control Codes) code sets are mapped to this space.
- The GL group contains the 96 addressable codes from 20h to 7Fh. The G0 (a slightly modified version of the ANSI X3.4 ASCII Printable Character set) and G2 (Extended Control Code Set 1) code sets are mapped to this space.
- The CR group contains the 32 addressable codes from 80h to 9Fh. The C1 (Caption Control Codes) and C3 (Extended Control Code Set 2) code sets are mapped to this space.
- The GR group contains the 96 addressable codes from A0h to FFh. The G1 (ISO 8859-1 Latin 1 Characters) and G3 (future character and icon expansion) code sets are mapped to this space.

b7-b4:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	b3-b0															
	0	CL (C0, C2) Set	GL (G0, G2) Set						CR (C1, C3) Set		GR (G1, G3) Set					
	1															
	2															
	3															
	4															
	5															
	6															
	7															
	8															
	9															
	A															
	B															
	C															
	D															
	E															
	F															

Table 6 DTVCC Code Space Organization

		C 0		G 0					C 1		G 1						
b7-b4		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
b3-b0	0	NUL	EXT1	SP	0	@	P	`	p	CW0	SPA	NBS	°	À	Ð	à	ð
	1			!	1	A	Q	a	q	CW1	SPC	¡	±	Á	Ñ	á	ñ
	2			"	2	B	R	b	r	CW2	SPL	¢	²	Â	Ò	â	ò
	3	ETX		#	3	C	S	c	s	CW3		£	³	Ã	Ó	ã	ó
	4			\$	4	D	T	d	t	CW4		¤	´	Ä	Ô	ä	ô
	5			%	5	E	U	e	u	CW5		¥	µ	Å	Õ	å	õ
	6			&	6	F	V	f	v	CW6		¦	¶	Æ	Ö	æ	ö
	7			'	7	G	W	g	w	CW7	SWA	§	·	Ç	×	ç	÷
	8	BS	P16	(8	H	X	h	x	CLW	DF0	¨	¸	È	Ø	è	ø
	9)	9	I	Y	i	y	DSW	DF1	©	¹	É	Ù	é	ù
	A			*	:	J	Z	j	z	HDW	DF2	ª	º	Ê	Ú	ê	ú
	B			+	;	K	[k	{	TGW	DF3	«	»	Ë	Û	ë	û
	C	FF		,	<	L	\	l		DLW	DF4	¬	¼	Ì	Ü	ì	ü
	D	CR		-	=	M]	m	}	DLY	DF5	-	½	Í	Ý	í	ý
	E	HCR		.	>	N	^	n	~	DLC	DF6	®	¾	Î	Þ	î	þ
	F			/	?	O	_	o	¸	RST	DF7	¯	¿	Ï	ß	ï	ÿ
				TSP	■							☐					
				NBTSP	‘												
					’												
					“												
					”												
				...	•												
									1/8								
									3/8								
									5/8								
					™				7/8								
				Š	š												
]								
				Œ	œ				L								
					™				—								
									┘								
					Ÿ				┘								
		C 2		G 2					C 3		G 3						

Table 7 DTVCC Code Set Mapping

NOTE--This section lists all required and optional caption codes. Refer to Section 9.4 for a list of required codes for a minimum DTVCC receiver implementation.

7.2 Extending the Code Space

The characters in the extended code sets of the CL, CR, GL, and GR code groups are accessed using the **EXT1** code (10h) in the C0 code set (Miscellaneous Control Codes). Normally (i.e., without extending the code space) the base codes in the four code groups (CL, GL, CR, and GR) represent the characters, control codes, and commands in the C0, C1, G0, and G1 code sets. By prefixing the codes in the code groups with the **EXT1** code (and thus forming the start of a two-byte code), the symbols in the extended C2, C3, G2, and G3 code sets are referenced; that is, each character in these code sets require the transmission of two codes (i.e., **EXT1** plus a base code) in order to be referenced. **EXT1** must start every 2-byte extended code sequence.

When **EXT1** is not encountered, then reference to the base code sets (C0, C1, G0, and G1) is assumed. That is, **EXT1** is only active for the two-byte extended code sequence in which it exists.

For example, to generate the closed-caption symbol (**CC**) in the G3 code set, the following sequence must be specified: 10h, A0h (**EXT1**, **NBS**).

The code space may be extended further (for future use) to handle other character sets not included in the language sets listed in section 7.4.4. This extension capability anticipates code sets which require 16-bit character code addressing, such as characters to implement Chinese and Japanese. The **P16** code in the C0 Miscellaneous Control Code set is used for this purpose. When a decoder encounters the **P16** code, it uses the succeeding two bytes to address characters in a 16-bit code set.

7.3 Unused Codes

Any codes not specified in the following sections are reserved for future standardization. Decoders encountering any undefined codes should adhere to the size (e.g., 1 byte, 2 byte, 3 byte and variable length) characteristics of these codes, as indicated in the following sections. This assures that future extensions to the coding scheme can be ignored by decoders which do not support them.

7.4 Numerical Organization of Codes

The following subsections detail the mapping of the individual code sets (C0, C1, C2, C3, G0, G1, G2, and G3) of the DTVCC code space.

7.4.1 C0 Code Set - Miscellaneous Control Codes

The C0 code set contains the 32 addressable codes from 00h to 1Fh. See Table 8.

Codes 00h through 0Fh are single byte codes.

Codes 10h through 17h are two byte codes.

Codes 18h through 1Fh are three byte codes.

The **NUL**, **BS**, **FF**, and **CR** codes are preserved from the standard ASCII control code set. The **ETX** code is also from the ASCII control code set, but it has a special use in that it is required at the end of a caption text segment to terminate the segment when the segment is not immediately followed by another caption command (see Section 8.10.4). The **EXT1** code is used to extend the DTVCC code space (see Section 7.2). The **P16** code is used as a further code space extension for 16-bit character sets.

		C 0	
		b7-b4	
b3-b0		0	1
	0	NUL	EXT1
	1		
	2		
	3	ETX	
	4		
	5		
	6		
	7		
	8	BS	P16
	9		
	A		
	B		
	C	FF	
	D	CR	
	E	HCR	
	F		

Table 8 C0 Code Set**7.4.2 C1 Code Set - Captioning Command Control Codes**

The C1 code set contains the 32 addressable codes from 80 to 9F. See Table 9. This set contains the captioning command control codes (window creation commands, character attributes, etc.). The use of these codes and the DTVCC captioning commands are detailed in Section 8.10.

		C 1	
		b7-b4	
b3-b0		8	9
	0	CW0	SPA
	1	CW1	SPC
	2	CW2	SPL
	3	CW3	
	4	CW4	
	5	CW5	
	6	CW6	
	7	CW7	SWA
	8	CLW	DF0
	9	DSW	DF1
	A	HDW	DF2
	B	TGW	DF3
	C	DLW	DF4
	D	DLY	DF5
	E	DLC	DF6
	F	RST	DF7

Table 9 C1 Code Set

7.4.3 G0 Code Set - ASCII Printable Characters

The G0 code set contains the 96 addressable codes from 20h to 7Fh. See Table 10. This set provides the ANSI X3.4 ASCII printable characters with the substitution of the music note character for the ASCII **DEL** character.

		G 0					
b3-b0	b7-b4	2	3	4	5	6	7
	0	SP	0	@	P	`	p
1	!	1	A	Q	a	q	
2	"	2	B	R	b	r	
3	#	3	C	S	c	s	
4	\$	4	D	T	d	t	
5	%	5	E	U	e	u	
6	&	6	F	V	f	v	
7	'	7	G	W	g	w	
8	(8	H	X	h	x	
9)	9	I	Y	i	y	
A	*	:	J	Z	j	z	
B	+	;	K	[k	{	
C	,	<	L	\	l		
D	-	=	M]	m	}	
E	.	>	N	^	n	~	
F	/	?	O	-	o		♪

Table 10 G0 Code Set

7.4.4 G1 Code Set - ISO 8859-1 LATIN-1 Character Set

The G1 code space contains the 96 addressable codes from A0h to FFh. See Table 11. This set consists of the ISO 8859-1 Latin-1 character set, also known as the Windows/ANSI character set. This set, when used in conjunction with the ASCII set, provides all the characters needed to encode text in Danish, Dutch, Faeroese, Finnish, French, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish. Many other languages can be written with this set of letters, including Hawaiian, Indonesian/Malay, and Swahili. ISO 8859-1 also extends the ASCII set with additional miscellaneous punctuation and mathematical signs.

Code **NBS** (A0h) represents a non-breaking space. This code is to be used (instead of a space character) between words that should not be split when word-wrapping is in effect.

		G 1					
b3-b0	b7-b4	A	B	C	D	E	F
	0	NBS	°	À	Ď	à	ð
1		í	±	Á	Ñ	á	ñ
2		ç	²	Â	Ò	â	ò
3		£	³	Ã	Ó	ã	ó
4		¤	´	Ä	Ô	ä	ô
5		¥	µ	Å	Õ	å	õ
6		¦	¶	Æ	Ö	æ	ö
7		§	·	Ç	×	ç	÷
8		¨	¸	È	Ø	è	ø
9		©	¹	É	Ù	é	ù
A		ª	º	Ê	Ú	ê	ú
B		«	»	Ë	Û	ë	û
C		¬	¼	Ì	Ü	ì	ü
D		-	½	Í	Ý	í	ý
E		®	¾	Î	Þ	î	þ
F		¯	¿	Ï	ß	ï	ÿ

Table 11 G1 Code Set


7.4.5 G2 Code Set - Extended Miscellaneous Characters

The G2 code set contains the extended miscellaneous characters. See Table 12. Characters in the G2 space are transmitted by preceding them with the control character **EXT1** (10h) from the C0 character set. Following the **EXT1** extender prefix, the G2 characters are addressed with the base in the range 20h - 7Fh.

The unshaded characters in the G2 table below are those which exist in the Microsoft Windows character map. They are positioned in this code set to reflect their relative positioning in the MS Windows map. In Windows, these characters are mapped to the code set in the range 80h to 9Fh.

The **TSP** character (20h) represents a transparent space. This character has no text foreground or background color; i.e., it passes through the fill color of the window containing it.

The **NBTSP** character (21h) represents a non-breaking transparent space. This character is the same as a transparent space (TSP) with the exception that it should be used between words that should not be split when word-wrapping is in affect.

The  character (30h) is a solid block which fills the entire character position with the text foreground color.

Also included in this set are the block drawing characters, opening and closing single and double quote marks, the trade mark symbol, the service mark symbol, and the remaining Latin-1 characters.


		G 2					
		2	3	4	5	6	7
b3-b0	b7-b4	0	TSP				
	1	NBTSP	‘				
	2		’				
	3		“				
	4		”				
	5	...	•				
	6						¹ / ₈
	7						³ / ₈
	8						⁵ / ₈
	9		™				⁷ / ₈
	A	Š	š				
	B]
	C	Œ	œ				L
	D		SM				—
	E]
	F		ÿ]

Table 12 G2 Code Set

7.4.6 G3 Code Set - Future Expansion

The G3 code set is reserved for future expansion. It currently contains the single caption icon . See Table 13.

Characters in the G3 space are transmitted by preceding them with the control character **EXT1** (10h) from the C0 code set. Following the **EXT1** extender prefix, the G3 characters are addressed with the base codes in the range A0h - FFh.


		G 3					
		A	B	C	D	E	F
b7-b4	b3-b0						
	0						
	1						
	2						
	3						
	4						
	5						
	6						
	7						
	8						
	9						
	A						
	B						
	C						
	D						
	E						
	F						

Table 13 G3 Code Set

7.4.7 C2 Code Set - Extended Control Code Set 1

The C2 code set is reserved for future extended miscellaneous control and caption command codes. See Table 14. Codes in the C2 space are transmitted by preceding them with the control character **EXT1** (10h) from the C0 code set. Following the **EXT1** extender prefix, the C2 characters are addressed with the base codes in the range 00h - 1Fh.

These codes can be succeeded by additional data bytes per the following:

Codes 00h through 07h are single-byte control codes (0 - additional bytes).

Codes 08h through 0Fh are two-byte control codes (1 - additional byte).

Codes 10h through 17h are three-byte control codes (2 - additional bytes).

Codes 18h through 1Fh are four-byte control codes (3 - additional bytes).

Example: The total sequence for a four-byte control code would be:

EXT1, 18h, <data1>, <data2>, <data3>

DTVCC decoders which do not implement these commands must use their implied sizes to skip over them in the Service Blocks.

		C	2
		0	1
b3-b0	b7-b4		
	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	A		
	B		
	C		
	D		
	E		
	F		

Table 14 C2 Code Set

7.4.8 C3 Code Set - Extended Control Code Set 2

The C3 code set is an additional set reserved for future extended miscellaneous control and caption command codes. See Table 15. Codes in the C3 space are transmitted by preceding them with the control character **EXT1** (10h) from the C0 character set. Following the **EXT1** extender prefix, the C3 command codes are addressed with the base codes in the range 80h - 9Fh.

Codes 80h through 8Fh are reserved for fixed-sized commands. These codes can be succeeded by additional data bytes per the following:

Codes 80h through 87h are five-byte control codes (4 - additional bytes).

Codes 88h through 8Fh are six-byte control codes (5 - additional bytes)

Example: The total sequence for a six-byte control code would be:

EXT1, 88h, <data1>, <data2>, <data3>, <data4>, <data5>

DTVCC decoders which do not implement these commands must use their implied sizes to skip over them in the Service Blocks.

Codes 90 through 9Fh are reserved for variable length caption commands. Variable-length caption commands have a 1-byte header following the command code. This header contains a 2-bit Type field (b7 - b6) and a 6-bit Length field (b5 - b0). The Type field allows a command to be broken up into several segments in order to be transmitted across multiple Service Blocks. The Type field has the following possible values: 00 - Beginning of Command (BOC), 01 - Continuation of Command (COC), 11 - End of Command (EOC), and All of Command (AOC). The Length field ranges from 0 - 63 and indicates the number of data bytes following the header. Data bytes can have any 8-bit value. Only one variable-length caption command per service can be transmitted at a time, and the command segments must be transmitted in order. This variable-length command capability is intended for commands which require the downloading of large units of data (e.g., fonts and graphics). DTVCC decoders which do not implement these commands must use the Length field to skip over them in the Service Blocks.

		C	3
b7-b4		8	9
b3-b0	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	A		
	B		
	C		
	D		
	E		
	F		

Table 15 C3 Code Set

8 DTVCC Interpretation Layer

The DTVCC Interpretation Layer defines the DTVCC Graphical User Interface. This discussion includes how the caption data coding is to be formatted when encoded and how it is to be interpreted when decoded. While the Caption Data Services Coding Layer (Section 7) identifies how service data bytes are represented, the Interpretation Layer describes how these bytes of data are to be processed. The data bytes for each caption service are interpreted as a unique data stream, independent from the other services.

This section defines all required and optional DTV closed-captioning features. Refer to Section 9 for a list of features required within a minimum DTVCC receiver implementation.

8.1 DTVCC Caption Components

The five major components of DTVCC captioning are Caption Screen, Caption Windows, Caption Pens, Caption Text, and Caption Display Synchronization.

- **Caption Screen:** The Caption Screen is the canvas on which caption windows are displayed.
- **Caption Windows:** The heart of caption definition consists of Caption Windows within which caption text is displayed.
- **Caption Pens:** The Caption Pens component defines styles and attributes for the appearance of the text within the caption windows.
- **Caption Text:** The Caption Text component defines how text is encoded and directed to specific windows.
- **Caption Synchronization:** The Caption Synchronization component controls the flow of interpretation of commands and caption text within the independent service data streams.

The following subsections provide a general overview of these components. These subsections are then followed by a detailed presentation of the DTVCC caption command set.

8.2 Screen Coordinates

A set of coordinates is defined to map a rectangular grid onto the “safe-title” area of the screen. This grid is used as a reference for superimposing captions. This reference is used to specify the position of caption windows.

Receivers which decode the DTV bit stream may have either a 16:9, 4:3 or other display screen aspect ratio. The coordinate-system grid size for a 16:9 receiver is 210 horizontal cells by 75 vertical cells. The 4:3 coordinate-system grid size is 160 horizontal cells by 75 vertical cells. For all other aspect ratio formats, a percentage or relative positioning coordinate system may be used such that a grid is mapped to the screen with 100 horizontal cells and 100 vertical cells.

The grid coordinates are specified as a pair of values in the form: (horizontal, vertical). The origin is the point in the upper left-most corner of the safe-title area, and is assigned the coordinate (0, 0). For the 16:9 format, the upper-right corner is (209, 0), the lower-left corner is (0, 74), and the lower right corner is (209, 74). A similar set of reference points is defined for the 4:3 format’s 160 x 75 coordinate system: respectively, (0,0), (159, 0), (0, 74), and (159, 74). It is important to remember that these grid cells are not intended for text positioning, but for window positioning.

Once the window is positioned, the starting positioning of the text rows and columns follow, depending upon the size of the displayed font. The ending of the text rows depends upon the spacing (monospace vs. proportional space) of the chosen font.

Figure 11 shows an exaggerated relationship between a 16:9 screen, the 210 x 75 grid, the overscan area, the viewable display area, the safe-title area, and the caption windows.

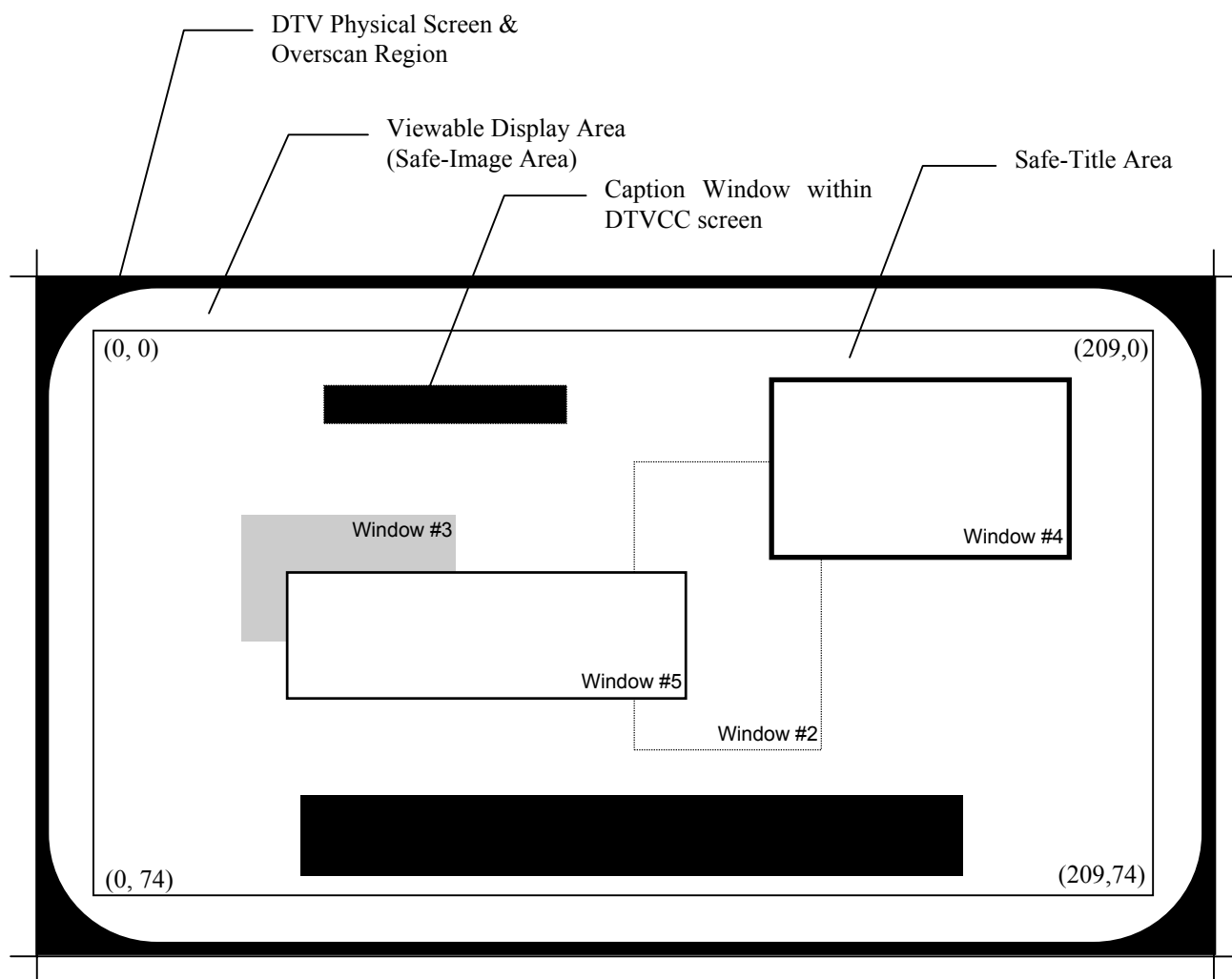


Figure 11 DTV 16:9 Screen and DTVCC Window Positioning Grid

8.3 User Options

Receiver manufacturers have the option to provide controls which may allow users to override styles and attributes specified in the service channel caption streams. Optional user controls might consist of caption font size, caption color and caption intensity (e.g., brightness) overrides. For further discussion, see the minimum DTVCC receiver decoder manufacturer recommendations in Section 9.

8.4 Caption Windows

All caption text is displayed and manipulated in receivers in the context of caption windows. There are 8 possible windows per service in which caption providers may write caption text. These windows may be implemented as buffers within receivers where any, none or all can be displayed at the same time. All 8 windows are available for the service currently selected by the user.

Manufacturers have the option of maintaining multiple sets of window buffers (i.e., instead of a single set of 8 buffers). Each set would be assigned to a service so that window processing of multiple services could occur simultaneously. This feature would have the effect that when a user switches services, the previously acquired and processed service data would be presented immediately. If only one set is used, the window buffers would have to be deallocated during service switching, and the new service would not appear until the buffers are reallocated and sufficient new service data are received.

The dimensions of a unique window specify an area on the screen which may contain caption text. The size of the window is based on the font size (SMALL, STANDARD, or LARGE) that the user has selected. Caption text designated for the window may not exceed the boundaries of the window, regardless of the font size the caption provider has specified or the font size the user has chosen.

A window's size may change on screen when a user changes the font size at the receiver. The effects of this window sizing are described further below.

8.4.1 Window Identifier

Each of the eight windows (and window buffers) is uniquely addressed by its window ID. Window ID numbers range from 0 to 7.

8.4.2 Window Priority

Each window has an associated priority which affects how it is displayed in conjunction with other windows that may be displayed at the same time. A higher priority (with 0 being the highest and 7 being the lowest) displayed window will overlap lower priority displayed windows on the screen.

8.4.3 Anchor Points

There are 9 locations within a window which serve as "anchors". An anchor specifies the reference point for positioning the window on the screen, and the "shrink and grow" directions (see Figure 12) of the window and caption text within the window when a user changes the font size.

8.4.4 Anchor ID

The 9 window anchor points are addressed by an Anchor ID which ranges from 0 to 8. Anchor ID 0 refers to the top-left corner of a window. Anchor 8 specifies the bottom-right corner of a window. Anchor 4 specifies the middle in the window.

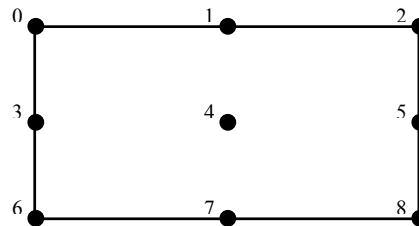


Figure 12 Anchor Points

Anchor points specify bounded and unbounded areas of caption text expansion and compression when the user overrides the standard font size for caption text display. Figure 13 shows the directions of expansion and compression of caption text for each anchor point. Solid lines indicate the bounded edges of the caption window. Dashed lines indicate the unbounded directions in which the caption text may shrink or grow.

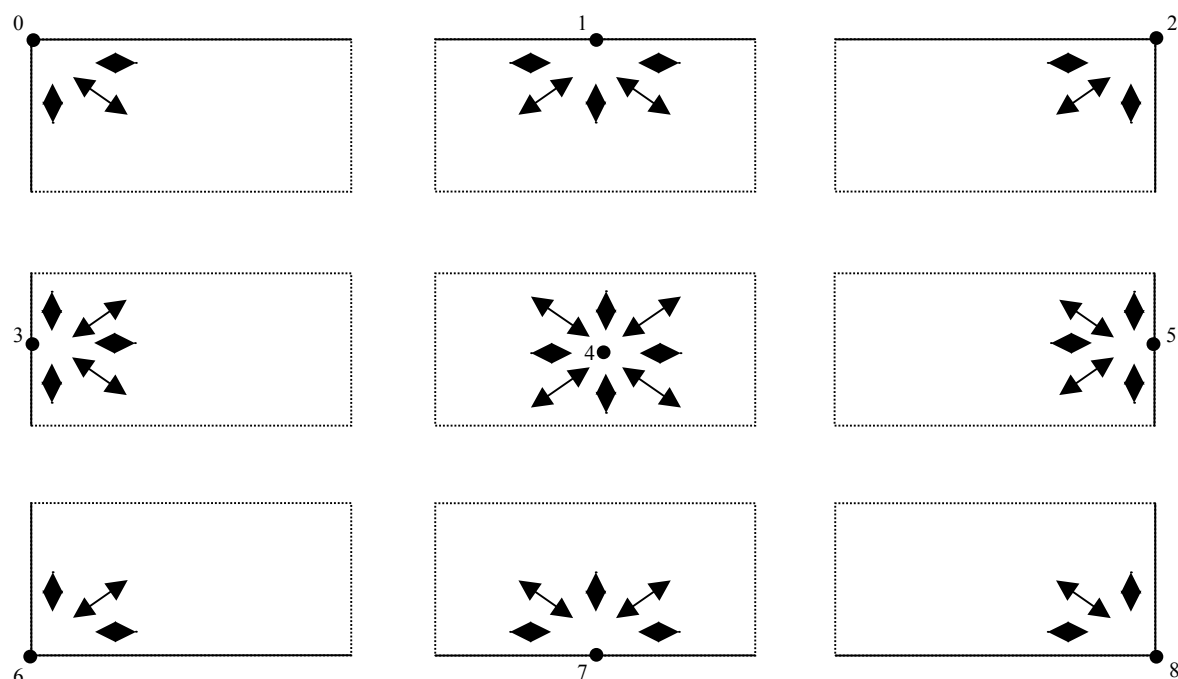


Figure 13 Implied Caption Text Expansion Based on Anchor Points

8.4.5 Anchor Location

The anchor location specifies where (in grid coordinates) the window's anchor point is to be physically located, and thus, the window itself. These grid coordinates are described in Section 8.2.

8.4.6 Window Size

The window size is specified in numbers of character rows and character columns for all display formats (16:9, 4:3, etc.). For all display formats, 15 is the maximum character row count and 32 is the maximum character column count.

NOTE--A 16:9 format could handle longer rows (i.e., up to 42), but caption providers should keep the window column size at 32 characters, or less, in order to leave room on the display when the user selects the LARGE font size (see Section 8.4.7).

As for the physical size of the window on the screen, the receiver scales the window based on the “effective font size”. The effective font size is based on a combination of the pen size chosen by the caption provider and the font size chosen by the receiver user. The height of the window is calculated as the number of rows multiplied by the physical height of the tallest character in the effective font size. The width of the window is calculated as the number of columns multiplied by the physical width of the widest character in the effective font size.

8.4.7 Window Row and Column Locking

The “lock rows” and “lock columns” window parameters fix the maximum number of rows and columns of caption text that a window may have. If a row or column parameter is “unlocked”, the receiver may automatically add or adjust columns or rows to a window under certain circumstances.

This means that in a text-mode or roll-up type caption service (with no embedded carriage returns) where the user has specified a font size smaller than intended by the caption provider, more rows and columns could fit into the physical window (as it appears on the screen) than specified in the original window size parameters (see examples below).

When the user has specified a font size larger than intended by the caption provider, the window's width may grow larger if the columns are locked. In order to insure that there is enough room on the display for 16:9 formats to grow a caption row so that it will fit on the screen, caption providers should not create windows or caption rows with more than 32 character columns. Since 16:9 formats can display 42 characters per row and if the widest character in the large font is no more than 1.3 (i.e., $42/32$) times larger than the widest character in the standard font, the enlarged caption (and window) should fit on the screen without word wrapping.

8.4.7.1 Effects When Choosing Smaller Font

Figure 14 provides examples of the effects row and column locking have on a window and its STANDARD-sized caption text when a receiver user chooses the SMALL font.

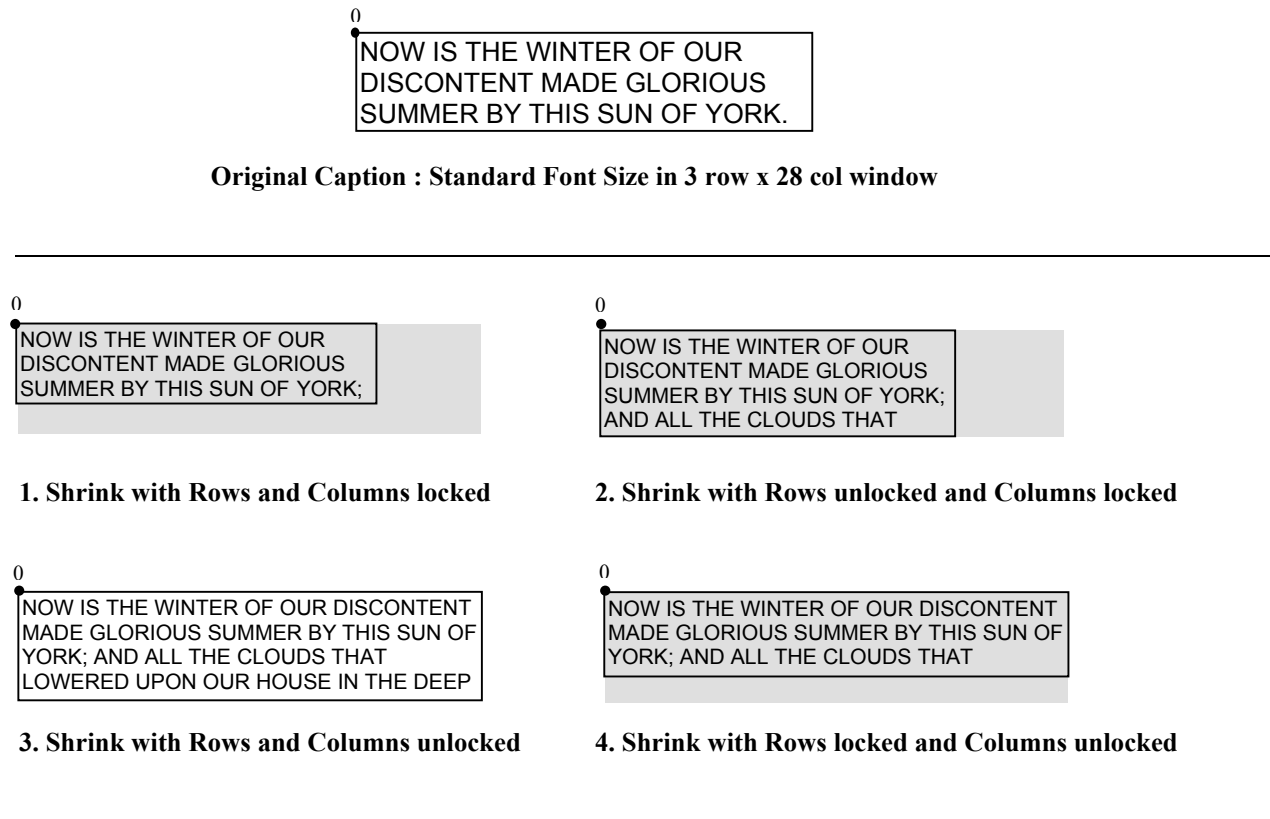


Figure 14 Examples of Caption Window Shrinking when User Picks Smaller Font

In the Figure 14 examples, the shaded rectangle indicates the size of the original window.

In example 1, the caption text and window shrink to a size in direct proportion to the original window. The caption contains the same number of lines and columns as the original, and the same words appear on the same line.

In example 2, the window indicates that there is room for another row of caption text. The window's height remains the same size.

In example 3, the window width and height remain the same; and with a smaller font, more characters may fit on a row. Thus, words are shifted up into the previous rows.

In example 4, the window shrinks in the vertical direction in order to maintain the same number of rows. The window height remains the same.

8.4.7.2 Effects When Choosing Larger Font

Figure 15 provides examples of the effects row and column locking have on a window and its STANDARD-sized caption text when a receiver user chooses the LARGE font.

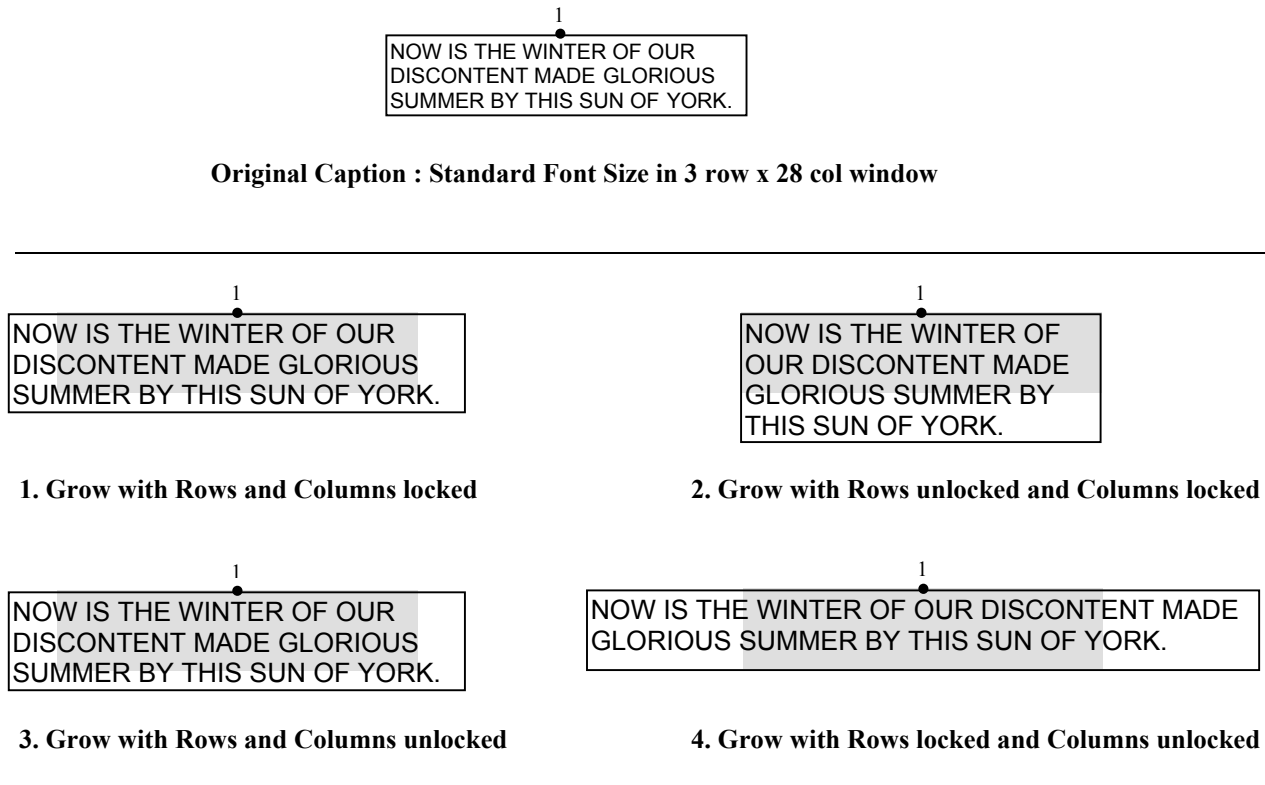


Figure 15 Examples of Caption Window Growing when Going to Larger Font

In the Figure 15 examples above, the shaded rectangle indicates the size of the original window.

In example 1, the caption text and window grow to a size in direct proportion to the original window. The caption contains the same number of lines and columns as the original, and the same words appear on the same line.

In example 2, the window width remains the same, but enough rows are automatically added to accommodate all of the caption text.

In example 3, the window is unrestricted in growth either in width or height. The decoder manufacturer is free to format the caption in anyway desired. However, since this allows for much ambiguity, it is suggested that the caption be formatted as in Example 1.

In example 4, the window height remains the same, but enough columns are automatically added to accommodate all of the caption text.

8.4.8 Word Wrapping

Whenever possible, the receiver should render embedded carriage returns as line breaks, since these carriage returns indicate an important aspect of the caption's formatting as determined by the service provider. However, it may sometimes be necessary for the receiver to ignore embedded line breaks. For example, if a caption is to appear in a larger font, and if its window's rows and/or columns are unlocked, the rows of text may need to become longer or shorter to fit within the allocated space. Such automatic reformatting of a caption is known as "word wrap."

The receiver should follow standard typographic practice when implementing word wrap. Potential breaking points (word-wrapping points) are indicated by the space character (20h) and by the hyphen character (2Dh).

If a row is to be broken at a space, the receiver should remove the space from the caption display. If a row is to be broken after a hyphen, the hyphen should be retained.

If an embedded return is to be removed, it should usually be replaced with a space. However, if the character to the left of the embedded return is a hyphen, the embedded return should be removed but NOT replaced with a space.

This specification does not include optional hyphens, nor does it provide for any form of automatic hyphenation. No non-breaking hyphen is defined. The non-breaking space (A0h in the G1 code set) and the non-breaking transparent space (21h in the G2 code set) should not be considered as potential line breaks.

If a single word exceeds the length of a row, the word should be placed at the start of a new row, broken at the character following the last character that fits on the row, and continued with further breaks if needed.

8.4.9 Window Text Painting

This section defines window text painting parameters and the interactions between them.

8.4.9.1 Justification

Caption text shall be formatted within the window based upon the justification type. The nomenclature for the justification types is based on standard English left-to-right Print Direction (see Section 8.4.9.2), and should be interpreted as follows:

Left Justification -- Text is justified to the start of the row, e.g., the left side for left-to-right print direction, the bottom for bottom-to-top print direction, and so on.

Right Justification -- The opposite of left justification. The text is aligned to the end of the row.

Centered Justification -- The text is centered horizontally when the print direction is left-to-right or right-to-left, and centered vertically when the print direction is top-to-bottom or bottom-to-top.

Full Justification -- The text is aligned to the left and right margins when the print direction is left-to-right or right-to-left, and to the top and bottom margins when the print direction is top-to-bottom or bottom-to-top.

8.4.9.2 Print Direction

The Print Direction parameter specifies in which direction characters will be written on a (horizontal or vertical) caption row (left-to-right, right-to-left, top-to-bottom, or bottom-to-top).

Table 16 defines how the cursor should move after drawing a character (when Justification is “Left” -- see Section 8.4.9.1) or after receiving a carriage return for each combination of Print Direction and Scroll Direction (see Section 8.4.9.3). Combinations of Print Direction and Scroll Direction that are not listed in this table are not permitted.

Print Direction	Scroll Direction	Cursor Movement	Carriage Return Behavior
Left -> Right	Top-> Bottom	increment column	decrement row, column=0
Left -> Right	Bottom->Top	increment column	increment row, column=0
Right->Left	Top-> Bottom	decrement column	decrement row, column=max
Right->Left	Bottom->Top	decrement column	increment row, column=max
Top-> Bottom	Left -> Right	increment row	decrement column, row=0
Top-> Bottom	Right->Left	increment row	increment column, row=0
Bottom->Top	Left -> Right	decrement row	decrement column, row=max
Bottom->Top	Right->Left	decrement row	increment column, row=max

Table 16 Cursor Movement After Drawing Characters

Figure 16 shows how a 3-row caption would appear using various justifications (see Section 8.4.9.1), Print Directions and Scroll Directions (see Section 8.4.9.3).

Left justified Left->Right print Bottom ->Top scroll ROW ONE TWO AND THREE	Left justified Right->Left print Bottom ->Top scroll ENO WOR OWT EERHT DNA	Right justified Left->Right print Top ->Bottom scroll AND THREE TWO ROW ONE	Right justified Right->Left print Top ->Bottom scroll EERHT DNA OWT ENO WOR
Left justified Top->Bottom print Right->Left scroll RTA OWN WOD O T N H E R E E	Right justified Bottom ->Top print Left->Right scroll EOE EWN RTO H T W O D R N A	Center justified Left->Right print Bottom ->Top scroll ROW ONE TWO AND THREE	Center justified Top->Bottom print Left->Right scroll A N R D O TW TW HOO R N E E E

Figure 16 Examples of Various Justifications, Print Directions and Scroll Directions

8.4.9.3 Scroll Direction

The Scroll Direction parameter specifies in which direction the text will scroll (left-to-right, right-to-left, top-to-bottom, or bottom-to-top) when carriage returns and word wrapping is encountered. For example, NTSC style roll-up captions are achieved by specifying left-to-right printing with bottom-to-top scrolling.

Horizontal scrolling is allowed only with vertical print directions, and vertical scrolling with horizontal print directions (see Section 8.4.9.2). For example, left-to-right scrolling may be used with top-to-bottom or bottom-to-top printing, but not with left-to-right or right-to-left printing.

8.4.9.4 Combining Text Painting Attributes

This section describes one of a multitude of text painting effects using the various window attributes.

A 2-line ticker-tape effect can be accomplished with the following parameter settings: print direction set to "top-to-bottom" with scroll direction set to "right-to-left", and window size set to 2 rows and X columns. The initial pen location is set to the upper right corner of the window. One character from row 1 is sent, followed by 1 character from row 2, and a carriage return, and so on. The first character for row 1 will appear on the screen by itself. With top-to-bottom print direction, the 1st character for row 2 will appear beneath the first row 1 character. The carriage return is sent, creating a new column and causing the existing 2 characters to scroll to the left and the next row 1 character to appear to the right of the existing row 1 character.

8.4.10 Window Display

Caption text can be written to a window whether it is currently displayed (i.e., visible) or not. Writing a whole caption to a non-displayed window and then sending a **DisplayWindows** command will cause the caption to "pop-on" (if the SNAP display effect has been set for the window). To achieve successive "pop-on" type captions, two windows are required (just as displayed and non-displayed memory buffers are used in NTSC captioning), with the required sequencing of **DisplayWindows** and **HideWindows** commands to maintain the effect.

When a window is displayed or hidden, a Display Effect parameter can be specified which controls whether the window snaps on/off ("pop-on"), fades on/off, or wipes on/off. The rate of fade on/off may be a manufacturer's option, but an effect speed can be specified with the **SetWindowAttributes** command. The direction (left-to-right wipe, right-to-left wipe, top-to-bottom wipe, and bottom-to-top wipe) in which a window wipes on/off can also be specified with this command.

8.4.11 Window Colors and Borders

The area within a window can have different characteristics regarding color and opacity. Windows can be clear, or filled with a specified color. A window can be filled with an opaque color from a color palette (see Section 8.8). The colors within a window can have various effects associated with them. The window can be transparent (i.e., clear - that is, letting the underlying video show through), translucent (i.e., the underlying video is passed through a fixed level of color background filtering so that underlying video may partially show through the window), solid (opaque with a nominal level of saturation), or flashing (alternating transparency and opacity at the nominal, or solid, level of saturation).

Windows may have no borders, or be bounded by an enclosing border. Borders may be raised, depressed, uniform, or drop-shadowed (left or right). Manufacturers have a bit of latitude in how these border effects are achieved. A uniform border may appear as a single line surrounding the window. Raised, depressed and shadowed borders can be used to achieve 3-dimensional effects.

8.4.12 Predefined Window and Pen Styles

Colors, text painting, effects, and border types can be customized with the **SetWindowAttributes** and **SetPenAttributes** commands. However, the caption provider may wish to use predefined standard windows styles. A set of predefined styles will be hard stored in receivers. This set will anticipate the most widely used types of caption windows in order to conserve caption channel bandwidth by eliminating the need to transmit superfluous **SetWindowAttributes** and **SetPenAttributes** commands.

Predefined window and pen styles can be specified by the *window style* and *pen style* ID parameters in the **DefineWindow** command. See Section 9.12 for the defined “predefined window and pen styles”.

8.5 Caption Pen

The caption text within a window is written with a *pen*. A pen governs the size, font, colors, and styles of the text within a window. Pen attributes are specified separately from window attributes through the use of the **SetPenAttributes** command. A window may contain text with more than one combination of pen attributes (i.e., different fonts, colors, etc.). Pen characteristics remain constant for a window unless specifically changed (via a subsequent **SetPenAttributes** command) by the caption provider. Pen attribute changes only affect text written after the change (i.e., the **SetPenAttributes** command does not change existing text within a window) and until a new **SetPenAttributes** command is encountered.

8.5.1 Pen Size

TV receivers may implement different sized characters (e.g., small, standard, and large) for supported fonts. Caption providers specify the pen size to be displayed, but users may override the size of the characters (i.e., which font size is to be viewed on the receiver) as desired.

8.5.2 Pen Spacing

Monospacing and proportional spacing are inherent attributes of the font chosen to write caption text. They cannot otherwise be controlled by either the caption provider or the user.

8.5.3 Font Styles

Caption providers may specify 1 of 8 different font styles to be used to write caption text. The styles specified in the “font style” parameter of the **SetPenAttributes** command are numbered from 0 through 7.

The following is a list of the 8 recommended font styles. For information purposes only, each font style references one or more popular fonts which embody the characteristics of the style:

- 0 - Default (undefined)
- 1 - Monospaced with serifs (similar to Courier)
- 2 - Proportionally spaced with serifs (similar to Times New Roman)
- 3 - Monospaced without serifs (similar to Helvetica Monospaced)
- 4 - Proportionally spaced without serifs (similar to Arial and Swiss)
- 5 - Casual font type (similar to Dom and Impress)
- 6 - Cursive font type (similar to Coronet and Marigold)
- 7 - Small capitals (similar to Engravers Gothic)

Implementation of these fonts styles by decoder manufacturers is optional. Font styles which are not supported in a decoder should be displayed in an available font which is most similar to the requested font style in the **SetPenAttributes** command.

8.5.4 Character Offsetting

Characters can be positioned relative to the row baseline in one of three ways: subscript (offset vertically downward), superscript (offset vertically upward), or normal (no offset).

8.5.5 Pen Styles

Pen styles can be italicized and/or underlined.

8.5.6 Foreground Color and Opacity

The foreground color and opacity of the caption characters can be specified by the caption provider. The character foreground opacity can be set to transparent (i.e., showing underlying video), translucent (i.e., showing a filtered level of underlying video), solid, or flashing.

8.5.7 Background Color and Opacity

Characters are individually contained within a small, rectangular background box. These background boxes have color and opacity attributes which may be specified separately from character foreground attributes.

8.5.8 Character Edges

The color attributes of the edges (or outlines) of the character foregrounds may be specified separately from the character foreground and background. Edge opacities have the same attribute as the character foreground opacities. The type of edge surrounding the body of the character may be NONE, RAISED, DEPRESSED, UNIFORM, or DROP_SHADOWED.

8.5.9 Caption Text Function Tags

Caption text can be tagged with a marker indicating the type of text content that is being encoded. This tagging is performed by caption providers via the **SetPenAttributes** command. With this command, caption text can be tagged with any one of the following qualities:

- 0 - Dialog** (normal words being spoken by characters in the programming)
- 1 - Source or speaker ID** (name of the speaker, or a description of the source of a sound)
- 2 - Electronically reproduced voice** (spoken audio heard by the characters in the drama coming from a phone, radio, PA, etc.)
- 3 - Dialog in a language other than the drama's primary language**
- 4 - Voiceover** (narration or other disembodied voice NOT heard by the characters in the drama)
- 5 - Audible Translation** (voice of a disembodied translator NOT heard by the characters in the drama)
- 6 - Subtitle Translation** (text showing a translation into the primary language of the drama)
- 7 - Voice quality description** (description of a voice quality)
- 8 - Song Lyrics** (words being sung)
- 9 - Sound effect description** (a description of a nonverbal sound or music heard by the characters in the drama)
- 10 - Musical score description** (a description of background music NOT heard by the characters in the drama)
- 11 - Expletive** (an interjectory word or expression, possible profane or harsh)
- 12 to 14 - (undefined)**
- 15 - Text not to be displayed** (reserved for future use by a text-based control and information channel within the caption text stream; e.g., hypertext, related non-caption program information)

In the previous list, all of the tagged text is to be displayed in the current caption window as indicated, except for tag 15 ("Text not to be displayed"), and optionally, tag 11 ("Expletive").

Decoder manufacturers have the flexibility to display tagged text in an number of ways desired in order to enhance the caption viewing experience. For example, "Source or speaker ID" text may always be automatically displayed in italics by the decoder. This may be a feature which the viewer can enable and disable.

If no other information is available, decoders are to default caption text to the “Dialog” tag (0). That is, if caption text is received for a window when no **SetPenAttributes** command has been received, the text is to be treated as “Dialog”.

8.6 Caption Text

Caption text is always written to the current caption window with the attributes set with the preceding **SetPenColor** and **SetPenAttributes** commands. There is no specific Text Write caption command; any displayable characters or codes from the G0, G1, G2, or G3 code sets in Service Blocks which are not part of any caption command are considered text to be written to the current window.

Caption text sequences must be terminated by either the start of a new DTVCC Command, or with an ASCII **ETX** (0x03) character when no other DTVCC Commands follow. This requirement aids decoders in processing text sequences when text spans multiple service blocks.

8.7 Caption Positioning

The starting row and column position of an individual character or set of characters may be specified at any time via the **SetPenLocation** command. The position of subsequent characters written after a location is specified depends upon the Print Direction and Scroll Direction specified for a window.

Character positions specify memory locations within the internal buffer holding the window text, and thus, do not specify physical locations on the screen. Screen locations are dependent on a multitude of pen and window attributes (e.g., character size, character spacing, justification, and print direction), and whether or not the font is proportionally spaced.

8.8 Color Representation

Foreground and background colors are specified in the Caption Commands as combinations of the red-green-blue color triad. Two bits are specified for each red, green, and blue color value which defines the intensity of each individual color component. Colors are specified as a group; i.e., (<red>, <green>, <blue>). The range of color specification is (0, 0, 0) [for black] through (3, 3, 3) [for bright white]. Bright red has a color value of (3, 0, 0); bright green has a color value of (0, 3, 0); and bright blue has a color value of (0, 0, 3). This coding scheme provides 64 different colors.

8.9 Service Synchronization

For the most part, caption providers insert caption commands and text into the DTVCC Caption Channel stream in a real-time manner. That is, captions are transmitted shortly before they are to be displayed. This is the technique used in NTSC captioning.

DTVCC provides for an additional synchronization capability by allowing DTVCC data to be pre-sent and processed at a later time, all under the control of the decoder. The **Delay** command provides this added functionality.

8.9.1 Delay Command

Decoders must maintain a Service Input Buffer for each of the services which can be processed simultaneously. This input buffer has a minimum size of 128 bytes. All DTVCC data for a service pass through the Service Input Buffer. Most of the time, the data falls through the buffer instantaneously, and the data are processed by the DTVCC decoder as they are received.

The **Delay** command is used to instruct the decoder to suspend processing of the Service Input Buffer data for a specified time period. This command has a time-out period specified as its parameter. This period is specified in tenths of seconds.

When a **Delay** command is encountered, the decoder waits for the specified delay time to expire before processing any other service data. During the delay interval, incoming data for the active service is buffered in the Service Input Buffer. When the delay interval expires, interpretation of the incoming data is resumed.

The **Delay** command may be used in instances where the caption provider knows that it is about to lose access to the DTVCC encoder-to-decoder stream. Prior to losing access, the caption provider pre-sends a set of DTVCC

commands with one or more embedded **Delay** commands. When the caption provider loses access, the decoders process the sequencing of the buffered data according to the providers' instructions.

Any active delay interval is automatically canceled when the Service Input Buffer becomes full. The decoder begins interpreting buffered data in order not to lose any incoming data.

8.9.2 DelayCancel Command

Caption providers may override an active **Delay** command via the **DelayCancel** command. The **DelayCancel** command terminates any currently pending Delay command processing, and causes the decoder to begin processing any data in the Service Input Buffer.

The **DelayCancel** command must be detected (recognized) prior to the input of Service Input Buffer. That is, the **DelayCommand** is not buffered. If it were, it would not be processed until the delay interval expires.

The **Delay** and **DelayCancel** commands are analogous to Service Input Buffer processing “suspend” and “resume”. These two commands allow caption providers to preload a set of commands and with a **Delay** command (suspend) to delay their output (using the maximum delay interval), and then issue a **DelayCancel** command (resume) at a desired instance to have the commands executed en mass.

8.9.3 Reset Command

The **Reset** command reinitializes a DTVCC service. The effects of re-initialization are described in Section 8.9.5. The **Reset** command is encoded by a caption provider to reset caption service processing within decoders.

It is recommended that a **Reset** command be issued at the beginning of a captioned program, or program segment. This ensures that decoders are initialized and ready for a new program, and erases any left-over windows and captions that were not deleted as a result of such events as video up-cutting during transitions from one program source to another.

8.9.4 Reset and DelayCancel Command Recognition

In order for the **DelayCancel** and service **Reset** commands to be effective, they must be recognized after they are retrieved from the Caption Channel Data Stream and prior to their routing to (insertion into) the Service Input Buffer. All other commands are interpreted when they are pulled from the Service Input Buffer.

Figure 17 shows this point of “pre-processing” for a decoder which simultaneously processes multiple service streams.

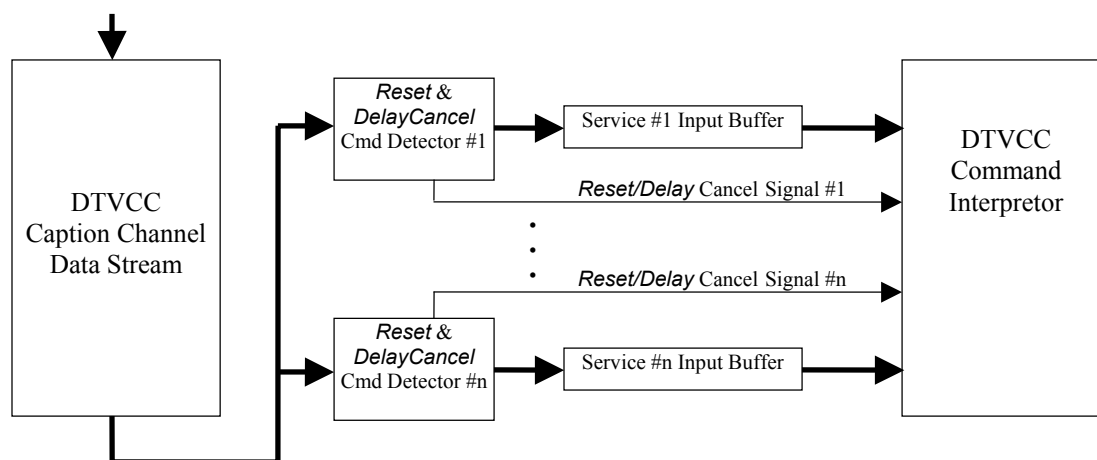


Figure 17 Reset & DelayCancel Command Detector(s) and Service Input Buffers

For descriptive purposes, Figure 18 shows a “hardware” implementation for the “*Reset & DelayCancel* Command Detector(s)”; a software implementation should be straight-forward. The Detectors may be implemented as a pair of 8-bit comparators. Since each of these commands is only a single byte in length (**Reset** = 0x8F, **DelayCancel** = 0x8E), this comparison may be simple, as is shown in Figure 18.

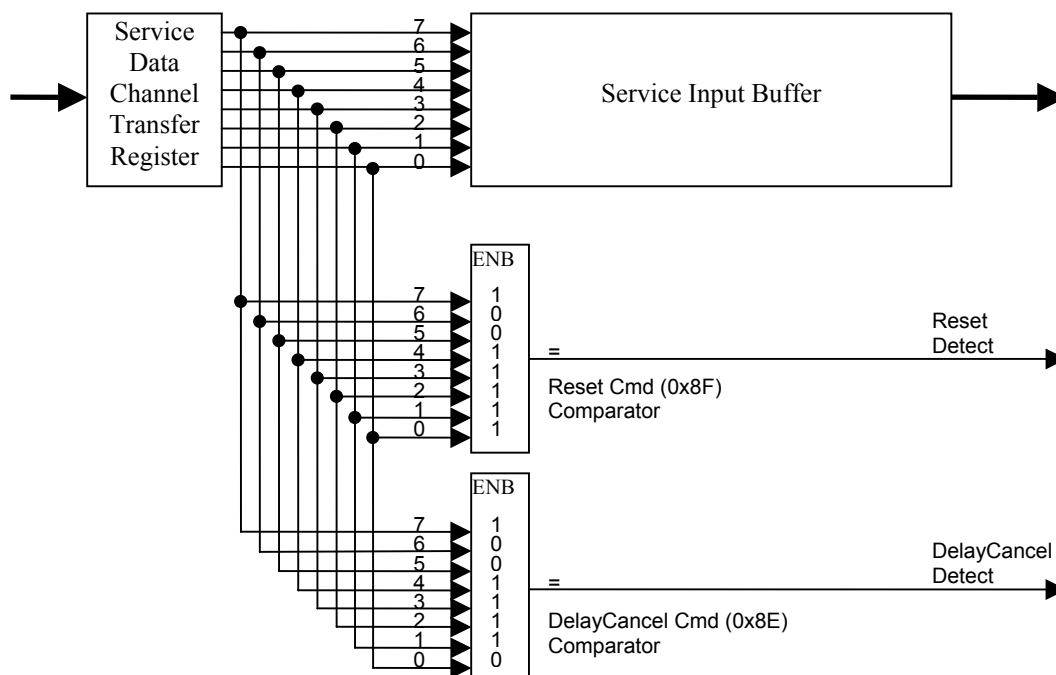


Figure 18 Reset & DelayCancel Command Detector(s) Detail

8.9.5 Service Reset Conditions

The “resetting” or “re-initialization” of a service means:

- that the service’s windows are removed from the display,
- all defined windows for the service are deleted,
- all window and pen attributes for the service are deleted, and
- the Service Input Buffer is cleared.

A service shall be reset when any one of the following events occur:

1. a **Reset** command is received for a service.
2. a channel change occurs.
3. the Service Input Buffer overflows.
4. A loss of continuity in the Caption Channel Packet sequence_number.

8.10 DTVCC Command Set

This section presents the commands which may be encoded by caption providers. The commands are grouped into the following command types: Window Commands, Pen Commands, Caption Text Commands, and Synchronization Commands.

8.10.1 Window Commands

These commands create, delete, modify, and display windows, and specify the current caption window for a caption service.

<u>Command Code</u>	<u>Command Name</u>	<u>Parameters</u>
CW0, ... , CW7	SetCurrentWindow	window ID
DF0, ... , DF7	DefineWindow	window ID, priority, anchor point, relative positioning, anchor vertical, anchor horizontal, row count, column count, row lock, column lock, visible, window style ID, pen style ID
DLW	DeleteWindows	window map
DSW	DisplayWindows	window map
HDW	HideWindows	window map
TGW	ToggleWindows	window map
SWA	SetWindowAttributes	justify, print direction, scroll direction, wordwrap, display effect, effect direction, effect speed, fill color, fill opacity, border type, border color

8.10.2 Pen Commands

These commands define pen attributes and colors.

<u>Command Code</u>	<u>Command Name</u>	<u>Parameters</u>
SPA	SetPenAttributes	pen size, font, text tag, offset, italics, underline, edge type
SPC	SetPenColor	fg color, fg opacity, bg color, bg opacity, edge color
SPL	SetPenLocation	row, <column

8.10.3 Synchronization Commands

These commands control the rate of service data interpretation.

<u>Command Code</u>	<u>Command Name</u>	<u>Parameters</u>
DLY	Delay	tenths of seconds
DLC	DelayCancel	
RST	Reset	

8.10.4 Caption Text

There is no specific Text Write caption command. That is, any characters or codes from the G0, G1, G2, or G3 code sets in Service Blocks which are not part of any caption command are considered text to be written to the current window. Any such encountered text is written to the current window starting at the window's current cursor position. The window's current cursor is adjusted automatically to the row and column following the last character in the text string.

In order to assist decoders in determining the end of caption text segments for dangling segments at the end of a series of caption commands and text, an ETX code (0x03) from the C0 Code Space is to be inserted at the end of the text segment to terminate the segment. Text which is immediately followed by a caption command code (from the C1 Code Space) does not require the ETX code. That is, the decoder determines the end of a text segment when it encounters a caption command code, or an ETX code.

Without the ETX assist, decoding software may find it difficult to know if it should wait until the next service block to see if there is more caption text for a caption row when a service block terminates in text. If the last used byte in the block is an ETX, then the decoder knows there is no more text for this segment in a subsequent service block, and the decoder can process it as a single entity (this is especially helpful when doing Center justification of a row and word wrap processing, for example). If the last used byte in the block is a text character, then the decoder must wait until the next service block to see if there is any more text.

8.10.5 Command Descriptions

Each command is described in detail within this section.

SET CURRENT WINDOW - (CW_x)
--

Name: **SetCurrentWindow** - Specify current window ID

Command Type: Window

Format: **SetCurrentWindow** (*window ID*)

Parameters: ■ *window ID* (id) is the unique window identifier (0 - 7).

Command Coding: CW0, ... , CW7 = 80h, ... , 87h (10000000b, ... , 10000111b)

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	0	0	id ₂	id ₁	id ₀	command

Description: **SetCurrentWindow** specifies the window to which all subsequent window style and pen/text commands are directed. The *window ID* must address a window which has already been created by the **DefineWindow** command. This command initializes a “current window ID” variable internal to the decoder. **SetCurrentWindow** directs the following commands to the specified window: **SetWindowAttributes**, **SetPenAttributes**, **SetPenColor**, **SetPenLocation**, **WriteText**.

DEFINE WINDOW - (DF0 ... DF7)

Name:	DefineWindow - Create window and set initial parameters
Command Type:	Window
Format:	DefineWindow (<i>window ID, priority, anchor point, relative positioning, anchor vertical, anchor horizontal, row count, column count, row lock, column lock, visible, window style ID, pen style ID</i>)
Parameters:	<ul style="list-style-type: none"> ■ <u>window ID</u> (id) is the unique window identifier (0 - 7) ■ <u>priority</u> (p) is the window display priority (0 - 7) ■ <u>anchor point</u> (ap) is the window anchor position number to use for the window's position on the screen (0 - 8). ■ <u>relative positioning</u> (rp) is a flag that, when set to 1, indicates that the <i>anchor vertical</i> (av) and <i>anchor horizontal</i> (ah) coordinates specify “relative coordinates” (i.e., percentages) instead of physical screen coordinates. ■ <u>anchor vertical</u> (av) is the vertical position of the window's anchor point on the viewing screen when the window is displayed (0 – 74 for 16:9 and 4:3 systems and the <i>relative positioning</i> (rp) parameter is set to 0, or 0-99 when the ‘rp’ parameter is set to 1). ■ <u>anchor horizontal</u> (ah) is the horizontal position of the window's anchor point on the viewing screen when the window is displayed (0 - 209 for 16:9 systems, 0 - 159 for 4:3 systems and the <i>relative positioning</i> (rp) parameter is set to 0, or 0-99 when the ‘rp’ parameter is set to 1). ■ <u>row count</u> (rc) is the number of virtual rows of text (assuming the STANDARD <i>pen size</i>; see SetPenAttributes) the window body will hold (0 - 11). ■ <u>column count</u> (cc) is the number of virtual columns of text (assuming the STANDARD <i>pen size</i>; see SetPenAttributes) the window body will hold (0 - 31 for 4x3 formats, and 0 - 41 for 16x9 formats). ■ <u>row lock</u> (rl), when set to YES, fixes the absolute number of rows of caption text the window will contain. When NO, <i>row lock</i> permits a receiver to add more rows to a window if the user selects a smaller size font other than intended by the caption provider. [YES, NO] = [1, 0]. ■ <u>column lock</u> (cl), when set to YES, fixes the absolute number of columns of caption text the window will contain. When NO, <i>column lock</i> permits a receiver to add more columns to a window if the user selects a smaller size font other than intended by the caption provider. [YES, NO] = [1, 0]. ■ <u>visible</u> (v), when set to YES, causes the window to be viewed (i.e., displayed) on the screen immediately after it is created. When set to NO, the window is not displayed (i.e., hidden) after it is created. [YES, NO] = [1, 0]. ■ <u>window style ID</u> (ws), when non-zero, specifies 1 of 7 static preset window attribute styles to use for the window when it is created (0 - 7). When zero during a window create, the window style is automatically set to window style #1. When zero during a window update, no window attribute parameters are changed. See SetWindowAttributes command. ■ <u>pen style ID</u> (ps), when non-zero, specifies 1 of 7 static preset pen attribute styles to use for the window when it is created (0 - 7). When zero during a window create, the pen style is automatically set to pen style #1. When zero during a window update, no pen attribute parameters are changed. See SetPenAttributes command.

Command Coding: **DF0, ... , DF7 = 98h, ... , 9Fh (10011000b, ... , 10011111b)**

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	1	1	id ₂	id ₁	id ₀	command
0	0	v	rl	cl	p ₂	p ₁	p ₀	parm1
rp	av ₇	av ₆	av ₅	av ₄	av ₃	av ₁	av ₀	parm2
ah ₇	ah ₆	ah ₅	ah ₄	ah ₃	ah ₂	ah ₁	ah ₀	parm3
ap ₃	ap ₂	ap ₁	ap ₀	rc ₃	rc ₂	rc ₁	rc ₀	parm4
0	0	cc ₅	cc ₄	cc ₃	cc ₂	cc ₁	cc ₀	parm5
0	0	ws ₂	ws ₁	ws ₀	ps ₂	ps ₁	ps ₀	parm6

Description:

DefineWindow creates a window for the specified window identifier (*window ID*) and initializes the window with the non-style parameters listed in the command and with the available static style presets specified with the *window style ID* and *pen style ID* parameters. If the window is not already defined when the **DefineWindow** command is received, the window is created; otherwise it is simply updated. If the window is being created, all character positions in the window are set to the window fill color and the pen location is set to (0, 0). The **DefineWindow** command also makes the defined window the current window (see **SetCurrentWindow**).

When a window is created, a specific or automatic *window style ID* is assigned in order to preload a set of known window attribute values. These attributes can be subsequently modified with the **SetWindowAttributes** command. A *pen style ID* is assigned in the same fashion. Pen style attributes can be subsequently modified with the **SetPenAttributes** command.

When a decoder receives a **DefineWindow** command for an existing window, the command is to be ignored if the command parameters are unchanged from the previous window definition. Encoders or caption providers may periodically repeat window definitions in order for receivers to acquire a service and begin decoding and displaying captions with little delay. It is suggested that all window and pen definition and attribute commands be repeated in this way.

When an existing window is being updated (e.g., resized or moved) with the **DefineWindow** command, the pen location and pen attributes are unaffected.

CLEAR WINDOWS - (CLW)

Name: **ClearWindows** - Clears Text from a set of windows

Command Type: Window

Format: **ClearWindows** (*window map*)

Parameters: ■ *window map* (*w*) is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents a window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A bit value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **CLW = 88h (10001000b)**

b₇	b₆	b₅	b₄	b₃	b₂	b₁	b₀	
1	0	0	0	1	0	0	0	command
<i>w₇</i>	<i>w₆</i>	<i>w₅</i>	<i>w₄</i>	<i>w₃</i>	<i>w₂</i>	<i>w₁</i>	<i>w₀</i>	parm1

Description: **ClearWindows** removes any existing text from the specified window(s) in the window map. When a window is cleared, the entire window is filled with the window fill color.

DELETE WINDOWS - (DLW)

Name: **DeleteWindows** - Deletes window definitions for a set of windows

Command Type: Window

Format: **DeleteWindows** (*window map*)

Parameters: ■ *window map* (*w*) is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents a window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A bit value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **DLW = 8Ch (10001100b)**

b₇	b₆	b₅	b₄	b₃	b₂	b₁	b₀	
1	0	0	0	1	1	0	0	command
w ₇	w ₆	w ₅	w ₄	w ₃	w ₂	w ₁	w ₀	parm1

Description: **DeleteWindows** removes all specified windows from the receiver. For example, a *window map* value of 64 (hex) deletes windows 6 (w₆), 5 (w₅), and 2 (w₂). A *window map* value of FF (hex) deletes all defined windows in the receiver. If the current window is deleted, then the decoder's current window ID is unknown and must be reinitialized with either the **SetCurrentWindow** or **DefineWindow** command.

DISPLAY WINDOWS - (DSW)

Name: **DisplayWindows** – Causes a set of windows to become visible

Command Type: Window

Format: **DisplayWindows** (*window map*)

Parameters: ■ *window map* (*w*) is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents a window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A value of 0 indicates that the associated window is unaffected by the command..

Command Coding: **DSW = 89h (10001001b)**

b₇	b₆	b₅	b₄	b₃	b₂	b₁	b₀	
1	0	0	0	1	0	0	1	command
w ₇	w ₆	w ₅	w ₄	w ₃	w ₂	w ₁	w ₀	parm1

Description: **DisplayWindows** causes the specified, existing windows to be visible on the receiver display screen. For example, a *window map* value of 96 (hex) displays windows 7 (w₇), 4 (w₄), 2 (w₂), and 1 (w₁). A *window map* value of FF (hex) causes all existing windows in the receiver to be displayed. This command does not affect the current window ID.

HIDE WINDOWS - (HDW)

Name: **HideWindows** – Causes a set of windows to become invisible

Command Type: Window

Format: **HideWindows** (*window map*)

Parameters: ■ *window map* is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents the window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **HDW = 8Ah (10001010b)**

b₇	b₆	b₅	b₄	b₃	b₂	b₁	b₀	
1	0	0	0	1	0	1	0	command
w ₇	w ₆	w ₅	w ₄	w ₃	w ₂	w ₁	w ₀	parm1

Description: **HideWindows** causes all specified and currently defined windows to be removed from the receiver display screen. For example, a *window map* value of 72 (hex) hides windows 6 (w₆), 5 (w₅), 4 (w₄), and 1 (w₁). A *window map* value of FF (hex) causes all existing windows in the receiver to be hidden.

TOGGLE WINDOWS - (TGW)

Name: **Toggle Windows** - Toggles Display/Hide status of a set of windows

Command Type: Window

Format: **Toggle Windows** (*window map*)

Parameters: ■ *window map* is an 8-bit bitmap which specifies the window(s) affected by the command. Each bit position represents the window (i.e., *window ID*) to be affected (e.g., bit position 4 addresses the window with the *window ID* of 4). A value of 1 in a bit position specifies that the associated window is to be processed by the command. A value of 0 indicates that the associated window is unaffected by the command.

Command Coding: **TGW = 8Bh (10001011b)**

b₇	b₆	b₅	b₄	b₃	b₂	b₁	b₀	
1	0	0	0	1	0	1	1	command
w ₇	w ₆	w ₅	w ₄	w ₃	w ₂	w ₁	w ₀	parm1

Description: **Toggle Windows** causes all specified and currently defined windows to toggle their display/hide status. That is, the specified windows in the *window map* which are currently displayed will be hidden, and the hidden ones will be displayed. For example, a *window map* value of 83 (hex) toggles windows 7 (w₇), 1 (w₁), and 0 (w₀). A *window map* value of FF (hex) causes all existing windows in the receiver to be toggled.

SET WINDOW ATTRIBUTES - (SWA)

Name: **SetWindowAttributes** - Defines the window styles for the current window.

Command Type: Window

Format: **SetWindowAttributes** (*justify, print direction, scroll direction, wordwrap, display effect, effect direction, effect speed, fill color, fill opacity, border type, border color*)

Parameters:

- *justify* (j) specifies how the text to be written in the window will be justified. [LEFT, RIGHT, CENTER, FULL] == [0, 1, 2, 3].
- *print direction* (pd) specifies in which direction text will be written in the window. [LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP] == [0, 1, 2, 3].
- *scroll direction* (sd) specifies which direction text will scroll when the end of a caption "line" is reached. [LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP] == [0, 1, 2, 3].
- *wordwrap* (ww), when set to YES, word wrapping is enabled. When set to NO, wordwrapping is disabled. [YES, NO] == [1, 0].
- *display effect* (de) specifies the effect that is to take place when the window is displayed and when it is hidden. When the SNAP effect is chosen, the window will pop-on the screen when the window is displayed and pop-off when the window is hidden. The FADE effect causes the window to fade onto and off of the screen at the specified *effect rate*. The WIPE effect causes the window to swipe onto and off of the screen at the specified *effect rate* and *effect direction*. [SNAP, FADE, WIPE] == [0, 1, 2].
- *effect direction* (ed) specifies which direction a WIPE window will appear on the screen. Note that a WIPE window will wipe-off of the screen in the opposite direction from which it wiped-on. [LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM, BOTTOM_TO_TOP] == [0, 1, 2, 3].
- *effect speed* (es) specifies, in .5 second units, how fast windows with WIPE and FADE effects will appear and disappear from the screen when they are displayed and hidden. The effect speed value may range from 1 to 15, approximating .5 (1 x .5) to 7.5 (15 x .5) seconds of effect speed variation.
- *fill color* (fr, fg, fb) is the color of the windows interior (see Section 8.4.11).
- *fill opacity* (fo) is the characteristic of the fill color of the window and the window border. [SOLID, FLASH, TRANSLUCENT, TRANSPARENT] == [0, 1, 2, 3].
- *border type* (bt) defines the type of outer edge surrounding the window. [NONE, RAISED, DEPRESSED, UNIFORM, SHADOW_LEFT, SHADOW_RIGHT] == [0, 1, 2, 3, 4, 5].
- *border color* (br, bg, bb) is the color of the windows outer edge(see Section 8.4.11).

Command Coding: **SWA = 97h (10010111b)**

b₇	b₆	b₅	b₄	b₃	b₂	b₁	b₀	
1	0	0	1	0	1	1	1	command
fo ₁	fo ₀	fr ₁	fr ₀	fg ₁	fg ₀	fb ₁	fb ₀	parm1
bt ₁	bt ₀	br ₁	br ₀	bg ₁	bg ₀	bb ₁	bb ₀	parm2
bt ₂	ww	pd ₁	pd ₀	sd ₁	sd ₀	j ₁	j ₀	parm3
es ₃	es ₂	es ₁	es ₀	ed ₁	ed ₀	de ₁	de ₀	parm4

Description:

SetWindowAttributes assigns the specified style attributes to the current window. This command can be issued any number of times to an existing window. The style attributes will overwrite any existing attributes assigned to the window.

SET PEN ATTRIBUTES - (SPA)

Name:	SetPenAttributes - Assign pen style attributes for the current window.
Command Type:	Pen
Format:	SetPenAttributes (<i>pen size, font, text tag, offset, italics, underline, edge type</i>)
Parameters:	<ul style="list-style-type: none"> ■ <i>pen size</i> (s) defines which of three pen sizes is to be used for text written to the current window, as specified by the current window ID. Note that the pen size displayed on the screen can be overridden by the user. [SMALL, STANDARD, LARGE] == [0, 1, 2]. ■ <i>font style</i> (fs) specifies which one of 8 different predefined font styles (see Section 8.5.3) to be used for text written to the current window (0 - 7). <ul style="list-style-type: none"> 0 - Default (undefined) 1 - Monospaced with serifs 2 - Proportionally spaced with serifs 3 - Monospaced without serifs 4 - Proportionally spaced without serifs 5 - Casual font type 6 - Cursive font type 7 - Small capitals ■ <i>text tag</i> (tt) specifies which one of 16 different predefined caption text function tags (see Section 8.5.9) is to be associated with the following caption text to be written to the current window (0 - 15). <ul style="list-style-type: none"> 0 - Dialog 1 - Source or speaker ID 2 - Electronically reproduced voice 3 - Dialog in language other than primary 4 - Voiceover 5 - Audible Translation 6 - Subtitle Translation 7 - Voice quality description 8 - Song Lyrics 9 - Sound effect description 10 - Musical score description 11 - Expletive 12 to 14 - (undefined) 15 - Text not to be displayed ■ <i>offset</i> (o) specifies sub-scripting and super-scripting attributes for text written to the current window. [SUBSCRIPT, NORMAL, SUPERScript] == [0, 1, 2]. ■ <i>italics</i> (i) specifies if text written to the current window is italicized. [YES, NO] == [1, 0]. ■ <i>underline</i> (u) specifies if text written to the current window is underlined. [YES, NO] == [1, 0]. ■ <i>edge type</i> (et) is the type of outlined edge of the text. [NONE, RAISED, DEPRESSED, UNIFORM, LEFT_DROP_SHADOW, RIGHT_DROP_SHADOW] == [0, 1, 2, 3, 4, 5].

Command Coding: SPA = 90h (10010000b)

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	1	0	0	0	0	command
tt ₃	tt ₂	tt ₁	tt ₀	o ₁	o ₀	s ₁	s ₀	parm1
i	u	et ₂	et ₁	et ₀	fs ₂	ft ₁	ft ₀	

Description:

SetPenAttributes assigns pen style attributes for the currently defined window. Text written to the current window will have the attributes specified by the most recent **SetPenAttributes** command written to the window. Pen attributes for a window can be changed as often as desired. These attributes will remain in effect for the window during its entire existence.

SET PEN COLOR - (SPC)

- Name:** **SetPenColor** - Assign styles to a dynamic preset style number.
- Command Type:** Pen
- Format:** **Set Pen Color** (*fg color, fg opacity, bg color, bg opacity, edge color*)
- Parameters:**
- *fg color* (fr, fg, fb) is the color of the text foreground body (see Section 8.5.6).
 - *fg opacity* (fo) is the characteristic of the text foreground body. [SOLID, FLASH, TRANSLUCENT, TRANSPARENT] == [0, 1, 2, 3].
 - *bg color* (br, bg, bb) is the color of the background box surrounding the window text (see Section 8.5.7).
 - *bg opacity* (bo) is the characteristic of the text background. [SOLID, FLASH, TRANSLUCENT, TRANSPARENT] == [0, 1, 2, 3].
 - *edge color* (er, eg, eb) is the color of the outlined edges of the text. The text character edges have the same opacity value as *fg opacity* (see Section 8.5.6).
- Command Coding:** **SPC = 91h (10010001b)**

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	1	0	0	0	1	command
fo ₁	fo ₀	fr ₁	fr ₀	fg ₁	fg ₀	fb ₁	fb ₀	parm1
bo ₁	bo ₀	br ₁	br ₀	bg ₁	bg ₀	bb ₁	bb ₀	parm2
0	0	er ₁	er ₀	eg ₁	eg ₀	eb ₁	eb ₀	parm3

- Description:** **SetPenColor** assigns the pen color attributes for the current window, as specified by the current window ID. Text written to the current window will have the color attributes specified by the most recent **SetPenColor** command written to the window. Pen color attributes for a window can be changed as often as desired. These attributes will remain in effect for the window during its entire existence.

SET PEN LOCATION - (SPL)

- Name:** **SetPenLocation** - Specifies the pen cursor location within a window.
- Command Type:** Pen
- Format:** **SetPenLocation** (*row, column*)
- Parameters:**
- row (r) is the text row within the current window's text buffer (0-14).
 - column (c) is the text column within the current window's text buffer (0-31 for 4:3 formats, 0-41 for 16:9 formats).
- Command Coding:** **SPL = 92h (10010010b)**

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	1	0	0	1	0	command
0	0	0	0	r ₃	r ₂	r ₁	r ₀	parm1
0	0	c ₅	c ₄	c ₃	c ₂	c ₁	c ₀	parm2

- Description:** **SetPenLocation** repositions the pen cursor for the current window, as specified by the current window ID. . When the window justification type is “left,” The next group of text written to the current window will start at the specified row and column, and justification will be ignored. When the window justification type is not left and the print direction is left-to-right or right-to-left, the column parameter shall be ignored. When the window justification type is not left and the print direction is top-to-bottom or bottom-to-top, the row parameter shall be ignored. When the window justification type is not left, text shall be formatted based upon the current window justification type. Note that if a window is not *locked* (see **DefineWindow**) and the SMALL *pen size* (see **SetPenAttributes**) is in effect, more than 12 rows and 36 columns could possibly be addressed.

DELAY - (DLY)

Name: Delay - Delays service data interpretation

Command Type: Synchronization

Format: Delay (*tenths of seconds*)

Parameters: ■ *tenths of seconds* (t) is the number of tenths of seconds to delay before recommencing service data interpretation.

Command Coding: DLY = 8Dh (10001101b)

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	0	1	1	0	1	command
t ₇	t ₆	t ₅	t ₄	t ₃	t ₂	t ₁	t ₀	parm1

Description: Delay instructs receivers to suspend interpretation of the current service's command input buffer. The delay is specified in tenths of seconds. Once the delay time expires, interpretation of caption commands recommences.

The delay value may range from 1 to 255 -- which specifies an effective delay time from 1/10 to 25.5 (255/10) seconds.

A delay for a service will remaining in effect until one of the following occurrences:

- the specified delay time expires
- a DelayCancel command is received
- the service's input buffer becomes full
- a service Reset command is received

DELAY CANCEL - (DLC)

Name: **DelayCancel** - Cancels an Active Delay Command

Command Type: Synchronization

Format: **DelayCancel**

Parameters: none

Command Coding: **DLC = 8Eh (10001110b)**

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	
1	0	0	0	1	1	1	0	command

Description: **DelayCancel** command terminates any active **Delay** command processing within the decoder.

RESET - (RST)

Name: **Reset** - Resets the Caption Channel Service

Command Type: Synchronization

Format: **Reset**

Parameters: none

Command Coding: **RST = 8Fh (10001111b)**

b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	command
1	0	0	0	1	1	1	1	

Description: **Reset** command reinitializes the service for which it is received.

9 DTVCC Decoder Manufacturer Recommendations

The following are the recommendations for DTV Closed Captioning decoder implementation including DTV, SDTV and HDTV. These recommendations are directed to the least common denominator of all of the DTVCC features described in the previous sections. Although voluntary, these recommendations should be considered as requirements for a realistic minimal implementation of the DTVCC capabilities. These minimal recommendations provide a bridge from NTSC (EIA-608-A) captioning implementation to the eventual full-feature implementation of this DTVCC specification.

It should be emphasized that these minimum recommendations are not intended to, and should not, restrict caption providers from using the whole suite of DTVCC commands and their extensive capabilities. The following sections address the minimum recommendations that have been anticipated, but may not cover all conditions and manifestations. It is up to the manufacturer to consider all situations that are not explicitly presented herein.

NOTE--The section numbers in the following headings refer to the corresponding sections in the current DTVCC Specification, EIA-708-B, to which the minimum recommendations apply.

9.1 DTVCC Section 4.2 - Pre-Allocated Bandwidth

While the DTVCC Caption Channel provides a continuous 9600 bps bit stream within the DTVCC Transport Channel, the individual bandwidth allocated to any single service shall not exceed 25% of the total bandwidth averaged over any 1 second time interval. This limit permits a maximum, average captioning data rate of 300 Bps per service.

That is, decoders need only implement enough buffering and processing power to handle a maximum of 2400 bps for each service. In effect, when this limit is exceeded for a service, the input storage buffer allocated for the service will overflow and data not already buffered will be lost.

NOTE-- In contrast, the per-service limitation addressed above still provides a five-fold enhancement over the maximum possible NTSC Closed-Caption service data rate of 60 Bps.

9.2 DTVCC Section 6.1 - Services

Decoders should be capable of decoding and processing data for at least one (1) service. Decoders shall be capable of decoding and processing the Caption Service Directory data.

9.3 DTVCC Section 6.2 - Caption Channel Service Blocks

Decoders should be capable of decoding all Caption Channel Block Headers consisting of Standard Service Headers, Extended Service Block Headers, and Null Block headers. However, decoding of the data is required only for Standard Service Blocks (Service IDs ≤ 6), and then only if the characters for the corresponding language are supported.

Decoders should be able to display the directory for services 1 through 6. Service decoding and directory display for services numbered 7 or greater are optional.

9.4 DTVCC Section 7.1 - Code Space Organization

Decoders must support Code Space C0, G0, C1, and G1 in their entirety.

The following characters within code space G2 must be supported:

- transparent space ($_{TSP}$)
- non-breaking transparent space ($_{NBTSP}$)
- solid block (■)
- trademark symbol ($^{\text{TM}}$)
- Latin-1 characters (Š, Œ, š, œ, Ÿ)

The substitutions in Table 17 are to be made if a decoder does not support the remaining G2 characters.

G2 Character	Substitute With
open single quote (‘), G2 char code 0x31	G0 single quote (‘), char code 0x27
close single quote (’), G2 char code 0x32	G0 single quote (‘), char code 0x27
open double quote (“), G2 char code 0x33	G0 double quote (“), char code 0x22
close double quote (”), G2 char code 0x34	G0 double quote (“), char code 0x22
bold bullet (•), G2 char code 0x35	G1 bullet (•), char code 0xB7
ellipsis(...), G2 char code 0x25	G0 underscore (_), char code 0x5F
one-eighth ($\frac{1}{8}$), G2 char code 0x76	G0 percent sign (%), char code 0x25
three-eighths ($\frac{3}{8}$), G2 char code 0x77	G0 percent sign (%), char code 0x25
five-eighths ($\frac{5}{8}$), G2 char code 0x78	G0 percent sign (%), char code 0x25
seven-eighths ($\frac{7}{8}$), G2 char code 0x79	G0 percent sign (%), char code 0x25
vertical border (), G2 char code 0x7A	G0 stroke (), char code 0x7C
upper-right border (┐), G2 char code 0x7B	G0 dash (-), char code 0x2D
lower-left border (└), G2 char code 0x7C	G0 dash (-), char code 0x2D
horizontal border (—), G2 char code 0x7D	G0 dash (-), char code 0x2D
lower-right border (┘), G2 char code 0x7E	G0 dash (-), char code 0x2D
upper-left border (┌), G2 char code 0x7F	G0 dash (-), char code 0x2D

Table 17 G2 Character Substitution Table

Support for code spaces C2, C3, and G3 is optional.

All unsupported graphic symbols in the G3 code space are to be substituted with the G0 underscore character (_), char code 0x5F.

9.5 DTVCC Section 8.2 - Screen Coordinates

Table 18 specifies the screen coordinate resolutions and limits for anchor point positioning in 4:3 and 16:9 display formats, and the number of characters per row.

Screen Aspect Ratio	Maximum Anchor Position Resolution	Minimum Anchor Position Resolution	Maximum Displayed Rows	Maximum Characters per Row
4:3	75v x 160h	15v x 32h	4	32
16:9	75v x 210h	15v x 42h	4	42
other	75v x (5 x H)	15v x H*	4	*

Table 18 Screen Coordinate Resolutions & Limits

*H = 32 x (the width of the screen in relation to a 4:3 display). For example, the 16:9 format is 1/3 wider than a 4:3 display; thus, H = 32 * 4/3 = 42.667, or 42.

This means that the minimum grid resolution for a 4:3 aspect ratio instrument is 15 vertical positions x 32 horizontal positions. This minimum grid resolution for 16:9 ratio instrument is 15 vertical positions x 42 horizontal positions. These minimum grid sizes are to cover the entire safe-title area of the corresponding screen.

The minimum coordinates equate to a 1/5 reduction in the maximum horizontal and vertical grid resolution coordinates. Caption providers are to use the maximum coordinate system values when specifying anchor point positions. Decoders using the minimum resolution are to divide the provided horizontal and vertical screen coordinates by 5 to derive the equivalent minimum coordinates.

Any caption targeted for both 4:3 and 16:9 instruments is limited to 32 contiguous characters per row. If a caption is received by a 4:3 instrument that is targeted for a 16:9 display only, or requires a window width greater than 32

characters, then the caption may be completely disregarded by the decoder. 16:9 instruments should be able to process and display captions intended for 4:3 displays, providing all other minimum recommendations are met.

If the resulting size of any window is larger than the safe title area for the corresponding display's aspect ratio, then this window will be completely disregarded.

9.6 DTVCC Section 8.4 - Caption Windows

Decoders need to display no more than 4 rows of captions on the screen at any given time, regardless of the number of windows displayed. This implies that no more than 4 windows can be displayed at any given time (with each having only one caption row).

However, decoders should maintain storage to support a minimum total of 8 rows of captions. This storage is needed for the worst-case support of a displayed window with 4 rows of captioning and a non-displayed window which is buffering the incoming rows for the next 4-row caption.

As implied above, the maximum number of windows that may be displayed at any one time by a minimum decoder implementation is 4. If more than 4 windows are defined in the caption stream, the decoder may disregard the youngest and lowest priority window definition(s). Caption providers must be aware of this limitation, and either restrict the total number of windows used or accept that some windows will not be displayed.

9.7 DTVCC Section 8.4.2 - Window Priority

Decoders do not need to support overlapped windows. If a window overlaps another window, the overlapped window need not be displayed by the decoder. Decoders may support overlapped windows as an option.

9.8 DTVCC Section 8.4.6 - Window Size

At a minimum, decoders will assume that all windows have rows and columns "locked". This implies that if a decoder implements the optional SMALL pen-size, then word-"un"wrapping, when shrinking captions, need not be implemented. Also, if a decoder implements the optional LARGE pen size, then word wrapping (when enlarging captions) need not be implemented.

9.9 DTVCC Section 8.4.8 - Word Wrapping

Decoders may support word wrapping as an option.

9.10 DTVCC Section 8.4.9 - Window Text Painting

9.10.1 Justification

All decoders should implement "left", "right", and "center" caption-text justification. Implementation of "full" justification is optional. . If "full" justification is not implemented, fully justified captions should be treated as though they are "left" justified.

For "left" justification, decoders should display any portion of a received row of text when it is received. For "center", "right", and "full" justification, decoders may display any portion of a received row of text when it is received, or may delay display of a received row of text until reception of a row completion indicator. A row completion indicator is defined as receipt of a CR, ETX or any other command , except SetPenColor, SetPenAttributes, or SetPenLocation where the pen relocation is within the same row.

Receipt of a character for a displayed row which already contains text with "center", "right" or "full" justification will cause the row to be cleared prior to the display of the newly received character and any subsequent characters. Receipt of a justification command which changes the last received justification for a given window will cause the window to be cleared.

9.10.2 Print Direction

At a minimum, decoders must support LEFT_TO_RIGHT printing.

9.10.3 Scroll Direction

At a minimum, decoders must support BOTTOM_TO_TOP scrolling.

For windows sharing the same horizontal scan lines on the display, scrolling may be disabled.

9.10.4 Scroll Rate

At a minimum, decoders must support the same recommended practices for scroll rate as is provided for NTSC closed-captioning.

9.10.5 Smooth Scrolling

At a minimum, decoders must support the same recommended practices for smooth scrolling as is provided for NTSC closed-captioning.

9.10.6 Display Effects

At a minimum, decoders must implement the “snap” window display effect. If the window “fade” and “wipe” effects are not implemented, then the decoder will “snap” all windows when they are to be displayed, and the “effect speed” parameter is ignored.

9.11 DTVCC Section 8.4.11 - Window Colors and Borders

At a minimum, decoders need only to implement borderless windows with solid, black backgrounds (i.e., border type = NONE, fill color = (0,0,0), fill opacity = SOLID), and borderless transparent windows (i.e., border type = NONE, fill opacity = TRANSPARENT).

9.12 DTVCC Section 8.4.12 - Predefined Window and Pen Styles

Predefined Window Style and Pen Style ID's may be provided in the DefineWindow command. At a minimum, decoders should implement Predefined Window Attribute Style 1 and Predefined Pen Attribute Style 1, as shown in Table 19 and Table 20.

Style ID #	Justify	Print Direction	Scroll Direction	Word Wrap	Display Effect	Effect Direction	Effect Speed	Fill Color	Fill Opacity	Border Type	Border Color	Usage
1	LEFT	LEFT-TO-RIGHT	BOTTOM-TO-TOP	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	<i>NTSC Style PopUp Captions</i>
2	LEFT	LEFT-TO-RIGHT	BOTTOM-TO-TOP	NO	SNAP	n/a	n/a	n/a	TRANSPARENT	NONE	n/a	<i>PopUp Captions w/o Black Background</i>
3	CNTR	LEFT-TO-RIGHT	BOTTOM-TO-TOP	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	<i>NTSC Style Centered PopUp Captions</i>
4	LEFT	LEFT-TO-RIGHT	BOTTOM-TO-TOP	YES	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	<i>NTSC Style RollUp Captions</i>
5	LEFT	LEFT-TO-RIGHT	BOTTOM-TO-TOP	YES	SNAP	n/a	n/a	n/a	TRANSPARENT	NONE	n/a	<i>RollUp Captions w/o Black Background</i>
6	CNTR	LEFT-TO-RIGHT	BOTTOM-TO-TOP	YES	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	<i>NTSC Style Centered RollUp Captions</i>
7	LEFT	TOP-TO-BOTTOM	RIGHT-TO-LEFT	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	<i>Ticker Tape</i>

Table 19 Predefined Window Style ID's

Predefined Style ID	Pen Size	Font Style	Offset	Italics	Underline	Edge Type	Foregrnd Color	Foregrnd Opacity	Backgrnd Color	Backgrnd Opacity	Edge Color	Usage
1	STNDR	0	NORMA L	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	<i>Default NTSC Style*</i>
2	STNDR	1	NORMA L	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	<i>NTSC Style* Mono w/ Serif</i>
3	STNDR	2	NORMA L	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	<i>NTSC Style* Prop w/ Serif</i>
4	STNDR	3	NORMA L	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	<i>NTSC Style* Mono w/o Serif</i>
5	STNDR	4	NORMA L	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	<i>NTSC Style* Prop w/o Serif</i>
6	STNDR	3	NORMA L	NO	NO	UNIFRM	(2,2,2) White	SOLID	n/a	TRANS- PARENT	(0,0,0) Black	<i>Mono w/o Serif, Bordered Text, No BG</i>
7	STNDR	4	NORMA L	NO	NO	UNIFRM	(2,2,2) White	SOLID	n/a	TRANS- PARENT	(0,0,0) Black	<i>Prop. w/o Serif, Bordered Text, No BG</i>

Table 20 Predefined Pen Style ID's

* "NTSC Style" - White Text on Black Background

9.13 DTVCC Section 8.5.1 - Pen Size

At a minimum, decoders must support the STANDARD pen size, with the implementation of the LARGE and SMALL pen sizes being optional.

The STANDARD pen size should be implemented such that the height of the tallest character in any implemented font is no taller than 1/15 of the height of the safe-title area, and the width of the widest character is no wider than 1/32 of the width of the safe-title area for 4:3 displays and 1/42 of the safe-title area width for 16:9 displays.

The LARGE pen size should be implemented such that the width of the widest character in any implemented font is no wider than 1/32 of the safe-title area for 16:9 displays. This recommendation allows for captions to grow to a LARGE pen size without having to reformat the caption since no caption will have more than 32 characters per row (see Section 8.4.6).

9.14 DTVCC Section 8.5.3 - Font Styles

Although a caption service provider may specify any one of 8 font styles using the **SetPenAttributes** command, decoders need only to implement a single font for caption text display.

Decoders that implement more than one font but do not support a font style specified in the **SetPenAttributes** command should instead display the caption text in the most similar font available. In decoders with only one font (i.e., font style 0, the default), all caption text, regardless of the specified font style, will be displayed in the default font.

In decoders with more than one but less than eight fonts, unsupported font styles should be displayed using an alternate font, giving precedence to the spacing attribute of the indicated font style, if possible. For example, if the specified but unsupported font style is “monospaced with serifs”, the best substitute would be another monospaced font, and the second-best alternative would be a proportionally spaced font with serifs. If the Cursive font style is not supported, an acceptable substitution is an italicized version of an available font.

All supported font styles may be implemented in any typeface which the decoder manufacturer deems to be a readable rendition of the font style, and need not be in the exact typefaces given as examples in Section 8.5.3.

9.15 DTVCC Section 8.5.4 - Character Offsetting

Decoders need not to implement the character offsetting (i.e., subscript and superscript) pen attributes.

9.16 DTVCC Section 8.5.5 - Pen Styles

At a minimum, decoders must implement normal, italic, and underline pen styles.

9.17 DTVCC Section 8.5.6 - Foreground Color and Opacity

At a minimum, decoders must implement solid and flashing character foreground type attributes.

At a minimum, decoders must implement the following character foreground colors: white, black, red, green, blue, yellow, magenta and cyan.

9.18 DTVCC Section 8.5.7 - Background Color and Opacity

Decoders need only implement solid black character backgrounds. It is recommended that this background is extended beyond the character foreground to a degree that the foreground is separated from the underlying video by a sufficient number of background pixels to insure the foreground is separated from the background.

9.19 DTVCC Section 8.5.8 - Character Edges

Decoders need not to implement separate character edge color, opacity, and type attribute control. In this case, there is no separately controlled edge surrounding the body of characters.

9.20 DTVCC Section 8.8 - Color Representation

At a minimum, decoders must support the 8 colors described in Table 21.

Color	Red	Green	Blue
Black	0	0	0
White	2	2	2
Red	2	0	0
Green	0	2	0
Blue	0	0	2
Yellow	2	2	0
Magenta	2	0	2
Cyan	0	2	2

Table 21 Minimum Color List Table

When a decoder supporting this Minimum Color List receives an RGB value not in the list, it will map the received value to one of the values in the list via the following algorithm:

- All one (1) values are to be changed to 0
- All two (2) values are to remain unchanged
- All three (3) values are to be changed to 2

For example, the RGB value (1,2,3) will be mapped to (0,2,2), (3,3,3) will be mapped to (2,2,2) and (1,1,1) will be mapped to (0,0,0).

Table 22 is an alternative minimum color list table supporting 22 colors.

Color	Red	Green	Blue
Black	0	0	0
Gray	1	1	1
White	2	2	2
Bright White	3	3	3
Dark Red	1	0	0
Red	2	0	0
Bright Red	3	0	0
Dark Green	0	1	0
Green	0	2	0
Bright Green	0	3	0
Dark Blue	0	0	1
Blue	0	0	2
Bright Blue	0	0	3
Dark Yellow	1	1	0
Yellow	2	2	0
Bright Yellow	3	3	0
Dark Magenta	1	0	1
Magenta	2	0	2
Bright Magenta	3	0	3
Dark Cyan	0	1	1
Cyan	0	2	2
Bright Cyan	0	3	3

Table 22 Alternative Minimum Color List Table

When a decoder supporting the Alternative Minimum Color List in Table 22 receives an RGB value not in the list (i.e., an RGB value whose non-zero elements are not the same value), it will map the received value to one of the values in the list via the following algorithm:

- For RGB values with all elements non-zero and different - e.g., (1,2,3), (3,2,1), and (2,1,3), the 1 value will be changed to 0, the 2 value will remain unchanged, and the 3 value will be changed to 2.
- For RGB values with all elements non-zero and with two common elements - e.g. (3,1,3), (2,1,2), and (2,2,3), if the common elements are 3 and the uncommon one is 1, then the 1 elements is changed to 0; e.g. (3,1,3) -> (3,0,3). If the common elements are 1 and the uncommon element is 3, then the 1 elements are changed to 0, and the 3 element is changed to 2; e.g. (1,3,1) -> (0,2,0). In all other cases, the uncommon element is changed to the common value; e.g., (2,2,3) -> (2,2,2), (1,2,1) -> (1,1,1), and (3,2,3) -> (3,3,3).

All decoders not supporting either one of the two color lists described above, must support the full 64 possible RGB color value combinations.

9.21 Character Rendition Considerations

In NTSC Closed Captioning, decoders were required to insert leading and trailing spaces on each caption row. There were two reasons for this requirement:

1. to provide a buffer so that the first and last characters of a caption row do not fall outside the safe title area, and

2. to provide a black border on each side of a character so that the “white” leading pixels of the first character on a row and the trailing “white” pixels of the last character on a row do not bleed into the underlying video.

Since caption windows are required to reside in the safe title area of the DTV screen, reason number 1 (above) is not applicable to DTVCC captions.

The attributes available in the **SetPenAttributes** command for character rendition (e.g., character background and edge attributes) provide unlimited flexibility to the caption provider when describing caption text in an ideal decoder implementation. However, manufacturers need only implement a minimum of pen attributes and font styles. Thus it is recommended that no matter what the level of implementation, decoder manufacturers should take into account the readability of all caption text against a variety of all video backgrounds, and should implement some automatic character delineation when the individual control of character foreground, background and edge is not supported; and when only a minimum number of font styles are implemented.

9.22 DTVCC Section 8.9 - Service Synchronization

Service Input Buffers must be at least 128 bytes in size. Caption providers must keep this lower limit in mind when following Delay commands with other commands and window text. In other words, no more than 128 bytes of DTVCC commands and text should be transmitted (encoded) before a pending Delay command’s delay interval expires.

9.23 DTV to NTSC Transcoders

It is anticipated that receiver (decoder) manufacturers will develop devices (e.g., settop boxes) which process an DTV stream and transcode it for display on NTSC monitors. The DTVCC command set is not necessarily transcodable to NTSC captions; i.e., there are DTVCC captions which have no NTSC equivalent.

Although receiver manufacturers are free to attempt an automatic transcode of the captions, there is no guarantee that the captions will appear as the caption provider intended. Caption providers apply many techniques to make the captions easy to read and as unobtrusive as possible over the underlying video. To maintain caption quality during an automated transcode process, a set of conversion rules would have to be defined which cover all possible window, pen and text attribute combinations.

Therefore, a separate NTSC caption channel was added to the Picture User Data (see Section 4.3). This channel allows caption providers to encode dual caption streams within the same programming. NTSC captions are under the complete control of the caption provider; and thus, no automated transcoding of captions is necessary.

10 DTVCC Authoring and Encoding for Transmission

This section describes a DTV captioning "food chain". This is the path the captioning information takes from initial authoring intentions to being multiplexed into the ATSC emission bit stream. The content of this section is an example for information.

10.1 Caption Authoring and Encoding

High quality captioning starts with the creation of the captioning intentions. This is a high level, generally editable, representation of how and when the captions should appear when rendered on the consumer receiver. SMPTE 12M time code is generally used for synchronization with picture. The output of the initial authoring process is generally a computer file that contains a list of time codes and the intention as to what the receiver should render when the picture, with the corresponding time code, appears on the display device. This computer file is typically editable. The file may be stored on a hard disc or floppy disc, and distributed by either computer networking techniques, or via floppy net. This process is illustrated in Figure 19.

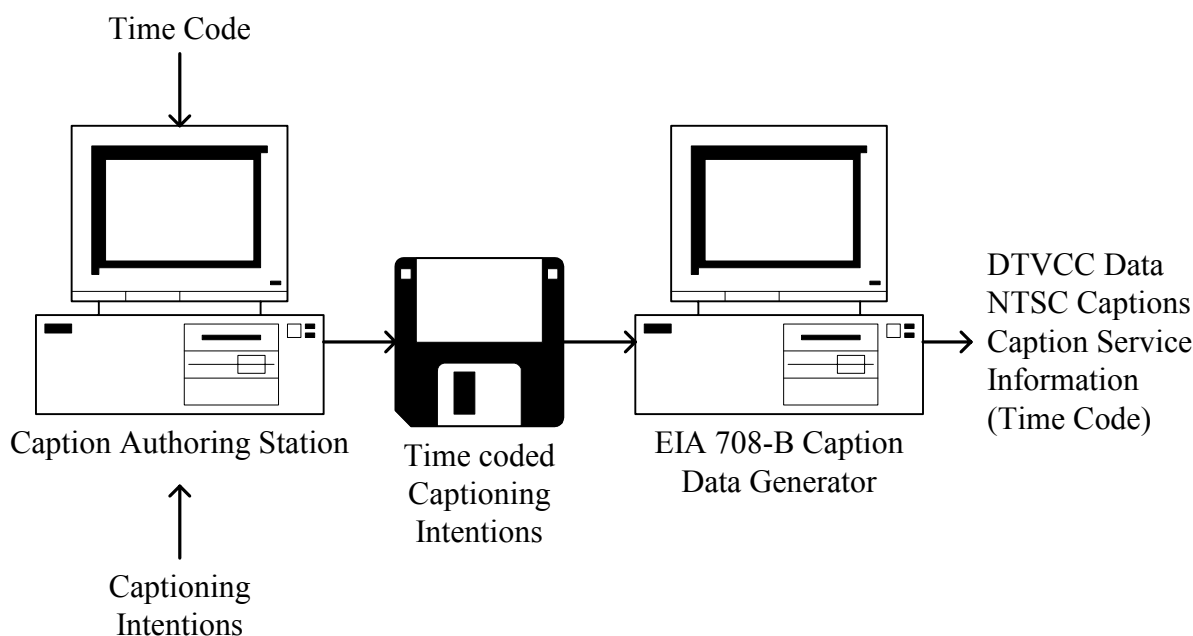


Figure 19 Caption Authoring and Encoding into Caption Channel Packets

The caption intentions are rendered into variable length *DTV Caption Channel Packets* (see Section 5). This rendering process is performed by an EIA-708-B caption data generator. This generator may also create a EIA-608-A (NTSC) captioning stream. This rendering process should consider the latency of the caption decoding process in the receiver, and thus should generate the packets pre-timed for transmission slightly early. Besides rendering the intentions into DTV Caption Channel Packets, the generator should create the caption service information that will be used to create the caption service descriptors that are carried in both the MPEG-2 PMT and in the ATSC PSIP EIT.

When delivered to the consumer receiver by the ATSC DTV system, the NTSC and DTV caption data is carried in the MPEG-2 picture user data, where a fixed number of bytes is allocated for each frame. The allocation provides 9,600 bits per second of data capacity. The number of bytes carried with each picture frame is dependent on the picture frame rate (see table 3 in section 4.4.2). In the case of 29.97 or 30 fps pictures, there are 4 bytes of NTSC

caption data and 36 bytes of DTV caption data carried per frame. The data rate of DTV Caption Channel Packets is equal to or lower than 9600 bps and so some zero padding is generally required. After the variable length DTV Caption Channel Packets are formed, it is necessary to parse these packets into the fixed length blocks that will be included with individual picture frames in the final emission multiplex. Many of these blocks will include zero padding so that the occupied bit rate is padded out to the full 9600 bps rate. This process is illustrated in figure 10-2.

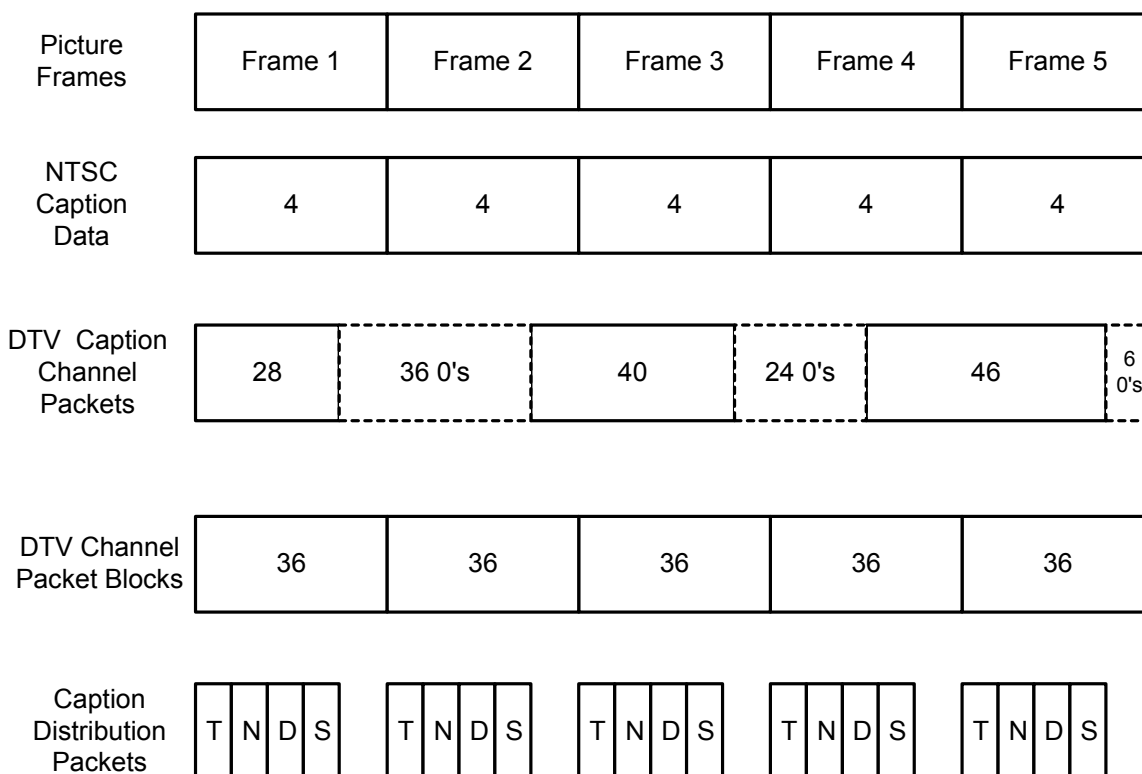


Figure 20 Relationship Between Caption Data and Picture Frames

Figure 20 illustrates five picture frame periods, at a 29.97/30 Hz frame rate. At this frame rate there are 4 bytes of NTSC caption data per frame. Three DTV Caption Channel Packets have been generated during the time of these 5 frames. These variable length packets have lengths of 28, 40, and 46 bytes. The DTV caption data is then formed into fixed length DTV Channel Packet Blocks that will be included in the corresponding MPEG-2 picture user data (see section 4) area. In this 29.97/30 Hz frame rate example, these blocks have a uniform length of 36 bytes. The first of these blocks contain the 28 bytes of the first DTV Caption Channel Packet plus 8 bytes of zero padding. The second block contains 28 bytes of zero padding followed by the first 8 bytes of the second DTV Caption Channel Packet, and so on. The result of this process is 36 bytes of DTV captioning data per picture frame. The final line in this figure shows the formation of all of the captioning data into *Caption Distribution Packets* that will be described in Section 11. These packets have a 1:1 relationship to video frames, and can include time code (T), the NTSC caption data (N), the blocked and zero padded DTV caption data (D), and the caption service information (S).

10.2 Monitoring Captions

Caption data may be monitored at various points during the distribution chain. To monitor the captions as they will be displayed on a consumer receiver, and to evaluate video/caption synchronization, it is necessary to decode the caption data after it has been rendered into a form (such as the Caption Distribution Packet) where there is an association of sets of caption data bytes with particular video frames.

10.3 Encoder Interfacing

DTVCC and NTSC captions are multiplexed into the Picture User Data in the MPEG-2 video elementary stream. This multiplexing is generally done within the MPEG-2 video encoder and so the caption data must be delivered to

this encoder. A Caption Service Descriptor is included in the PMT and in the EIT and so caption service information must be delivered to the PSIP generator, and to the MPEG-2 PMT generator. (It is possible that both PMT and EIT would be created by a common functional block.) It is likely that all of the caption information will arrive at the encoding system on a single interface connection, and be distributed to the various destinations by looping the interface connection to multiple functional blocks. See Figure 21.

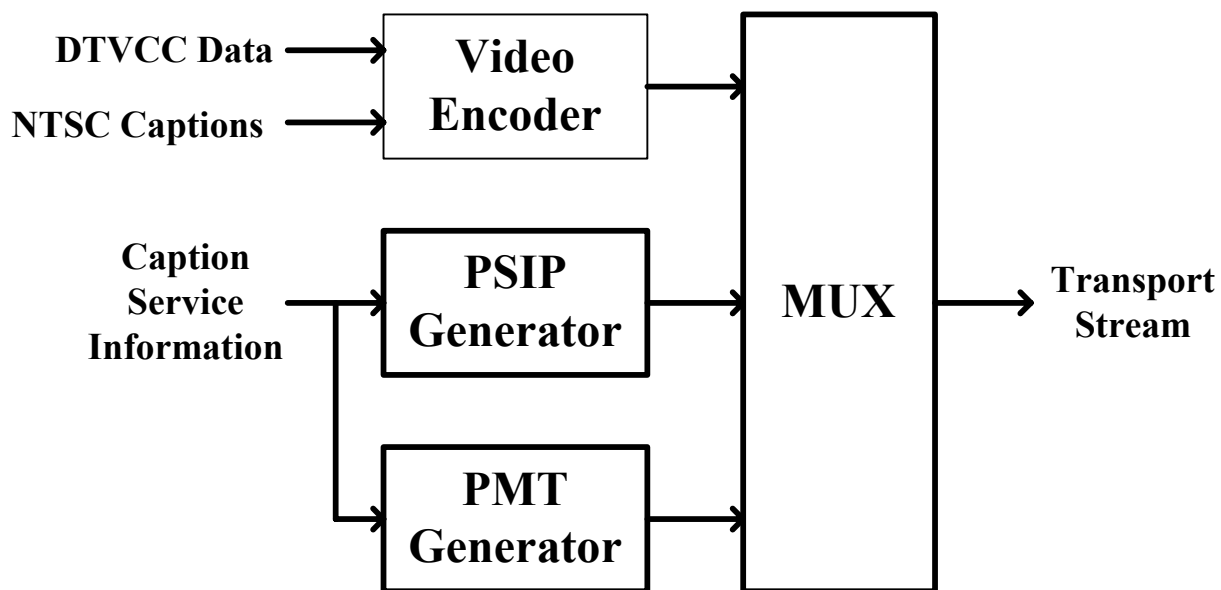


Figure 21 Interface of Caption Data to ATSC Emission Encoding Equipment

Several types of interfaces for the caption data may exist. These include a serial data format carried on an asynchronous RS232 like interface, embedding into an AES3 (digital audio interface) data stream, or embedding into SMPTE 259M or 292M serial digital video streams. For purposes of interoperability it is useful if there is as much commonality as possible to the data format on these different types of interfaces. The Caption Distribution Packet (CDP) described in Section 11 can carry DTVCC data, NTSC captions, caption service information, and time code. Section 11 defines an asynchronous serial interface to carry the CDP. SMPTE is standardizing carriage of the CDP over AES3, 259M, and 292M interfaces.

11 DTV Closed Captioning Content Package

This section describes how to create a closed captioning content package, designated the Caption Distribution Packet (CDP), which may contain time code, EIA-708-B closed captioning data, and ATSC closed caption descriptor information. The CDP is suitable for transport over a variety of streaming interfaces. The details of one specific serial interface are defined.

11.1 Introduction

The process of creating and delivering closed captions for ATSC DTV involves authoring captions into a representation which represents the frame accurate captioning intentions (e.g. SAMI high level representation), rendering the intentions into the EIA-708-B caption syntax, transport of this EIA-708-B data via storage and/or streaming media to the point of emission, and then packaging the EIA-708-B data into the MPEG-2 picture user data as specified in ATSC A/53. During this process the caption data must be kept properly synchronized to the picture and sound. It is also necessary to create, transport, and include the caption service information in order to create the Caption Service Descriptor that is carried in the PMT and EIT tables in the MPEG-2 emission transport stream. During the distribution chain, the EIA 708-B captioning data may be rendered and displayed for quality control purposes.

This section describes how to create a captioning content package, consisting of a defined sequence of bytes, which may carry the following:

- time code;
- EIA-708-B caption;
- EIA-608-A NTSC caption data;
- caption service information to form the caption service descriptor; and/or
- sequence counts to detect discontinuities in the stream of caption data packets.

The CDP may be employed in various types of transport of captioning data, including file formats, RS232, TIA/EIA-574, AES3, SMPTE 259M, SMPTE 292M, etc. The CDP may be further encapsulated into a SMPTE defined Key-Length_Value (KLV) construct. Details on the KLV construct may be obtained from SMPTE.

11.1.1 Frame Rates

The transport of closed captioning information over the DTV emission system involves packaging the CC data into the MPEG-2 picture user data area. In order to do this the captioning data must be packaged at the same frame rate as that used by the video encoder. Rendering the captioning data into a particular frame rate is done prior to or during creation of the CDP. If the video encoder encodes at a frame rate that differs from the frame rate of the CDP, the captioning data must be re-framed. This can be done by the video encoder, by a caption data server, or by other equipment upstream of the video encoder. However, if the video encoder is responsible for the determination of the encoded picture frame rate, the re-framing should be done in the video encoder or in a captioning server tightly coupled (with 2-way communication) to the video encoder.

From the point of view of captioning, frame rates which differ by 0.1% may be considered identical. For example, if picture and captions are rendered at a 30 Hz frame rate, they may both be played at 29.97 Hz without any reframing of the caption data. As long as captions are delivered at the same rate as the picture, and this rate does not change by more than 0.1 %, no reframing is needed. Also, captions may be generated at a 25 or nominal 30 Hz rate, and used with 50 or nominal 60 Hz video. In this case, a video encoder should place the first half of the caption data in the first picture user data area, and place the second half of the caption data in the second picture user data area.

11.1.2 Time Code

The CDP may carry a time code which may be derived from SMPTE 12M VITC or LTC. Carriage of a time code provides an important tool to allow captions to be kept properly synchronized with pictures. The picture, sound, and caption data elements may flow through differing paths to the emission encoding and multiplexing equipment. The inclusion of time code within each type of element makes it possible for the final multiplexer to provide buffering in order to deliver a properly synchronized multiplex of elements to the final receiver.

11.1.3 Caption Data

The caption data contained in the CDP is fully framed and formatted for direct inclusion within the ATSC video elementary stream picture user data, as defined in ATSC A/53, sections 5.2.2 and 5.2.3.

11.1.4 Caption Service Information

The caption service information in the CDP is fully framed and formatted for direct inclusion within the caption service descriptor as defined in ATSC A/65, section 6.7.3, table 6.17.

To reduce the data rate to carry the CDP stream (see section 11.1.5 below), the caption service information may be spread over a sequence of CDP packets. The receiver of the CDP stream must be able to collect service information from a sequence of CDP packets, and must be able to detect when the service information has changed.

Two general types of change may be envisioned. The first is a controlled change, where the generator of the CDP stream can insert an explicit indication that service information has changed, or that a service has been added or dropped. The second is an uncontrolled change that could be caused by a switch from one CDP stream to another CDP stream. In the case of an unsupervised switch, there can be no controlled signaling of the change, yet the receiver of the CDP stream must be able to easily detect that a change has occurred. In the event that a CDP stream is switched, the switch could result in a stream that has an incomplete, wrong, or damaged caption descriptor that

should be discarded (and not transmitted to a consumer DTV receiver). To provide the ability to detect stream switches, 16-bit sequence counts are included in the CDP header and footer. If the received sequence counts do not increment smoothly, a switch or error has occurred.

11.1.5 Interface data rates

The data rate required to convey a CDP stream is dependent on the frame rate of the stream and on the amount of service information included within each packet. The worst-case data rate would occur at a 60 Hz frame rate, with 16 services all fully described within each CDP packet. In this case, the CDP packet size would be 161 bytes, and require a transmission rate of 9,660 bytes per second, or 96,600 bps over a serial interface with 1 start bit and 1 stop bit. If the service information is limited to describing only 1 service per CDP, then the maximum data rate becomes 3,360 bytes per second, or 33.6 kbps over the serial interface. It is therefore practical to carry the CDP stream over an TIA/EIA-574 38.4 kbps serial interface, although in some cases it is necessary to limit the amount of service information included in each CDP packet.

11.2 CDP Detailed Specification

The CDP shall be as specified in Section 11.2.

11.2.1 General Construct

The general construct of the CDP shall be as defined in Table 23.

<u>Syntax</u>	<u>Comment</u>
cdp() {	Caption Distribution Packet (CDP)
cdp_header();	Required
time_code_section();	Optional
ccdata_section();	Optional
ccsvinfo_section();	Optional
future_section_1();	If defined, place here
future_section_2();	If defined, place here
cdp_footer();	Required
}	

Table 23 CDP General Construct

The CDP shall contain one header section and one footer section. The CDP may contain one time code section, one cc data section, and one cc service information section. The CDP shall not contain more than one of any of these sections. These sections shall be multiplexed in the order shown in Table 23. It is possible to extend the CDP to include additional sections. Any sections that are defined in the future shall be placed just prior to the cdp_footer. Any newly defined sections would begin with a unique identifier byte, and contain a length code. The syntax that a new section would follow is shown in Section 11.2.7. Equipment that receives the CDP shall skip over sections that begin with an unknown id byte, by means of the length code.

11.2.2 cdp_header

The CDP header is a required element, and shall be present in all CDP's. CDP header syntax shall be as indicated in Table 24. The length of the cdp_header is fixed at 6 bytes.

Syntax	Bits	Format	Comment
cdp_header() {			
cdp_identifier	16	uimsbf	0x9669
cdp_length	8	uimsbf	length of entire packet
cdp_frame_rate	4	uimsbf	frame rate of packets
Reserved	4	'1111'	
time_code_present	1	bit	'1' indicates time code section is included
ccdata_present	1	bit	'1' indicates cc data section is included
svcinfo_present	1	bit	'1' indicates svcinfo section is included
svc_info_start	1	bit	'1' indicates start of svc info data set
svc_info_change	1	bit	'1' indicates change in cc service information
svc_info_complete	1	bit	'1' indicates completion of svc info data set
caption_service_active	1	bit	'1' indicates caption service is active
Reserved	1	'1'	
cdp_hdr_sequence_cntr	16	uimsbf	
}			

Table 24 CDP Header Syntax

cdp_identifier – This is a 16-bit value set to 0x9669. All CDP packets begin with this value.

cdp_length – This 8-bit integer shall indicate the number of bytes of data in the entire CDP packet, from the first byte of the CDP_identifier, to the packet checksum, inclusive.

cdp_frame_rate – This field shall indicate the frame rate of the CDP stream. It shall be coded as indicated in Table 25. Also shown are the values of cc_count and the number of cc_data bytes that shall be included in each packet at each frame rate.

cdp_frame_rate	frame rate	cc_count	NTSC cc_data bytes	DTV cc_data bytes
0000	Forbidden			
0001	24000÷1001 (~23.976)	25	6	44
0010	24	25	6	44
0011	25	24	----	50
0100	30000÷1001 (~29.97)	20	4	36
0101	30	20	4	36
0110	50	12	----	25
0111	60000÷1001 (~59.94)	10	2	18
1000	60	10	2	18
...	Reserved			
1111	Reserved			

Table 25 CDP Frame Rate

NOTE--There is no practical difference between the pairs of frame rates which differ by 0.1%. Captions and pictures rendered at one rate may be played 0.1% fast or slow, with no impact on presentation, as the number of bytes per frame does not change.

time_code_present – This bit shall be set to '1' for CDP packets which include the time code section. Otherwise this bit shall be set to '0'.

cc_data_present – This bit shall be set to '1' for CDP packets which include the cc data section. Otherwise this bit shall be set to '0'.

svcinfo_present – This bit shall be set to '1' for CDP packets which include the service information section. Otherwise this bit shall be set to '0'.

svc_info_start – This bit shall be set to '1' to indicate that the current packet begins a complete set of service information. Otherwise this bit shall be set to '0'. This bit shall be set to '0' if this CDP packet does not contain a service information section. This bit is duplicated in the cc service information section. The value of this bit shall not be different from the value of the svc_info_start bit in the cc service information section.

svc_info_change – This bit shall be set to '1' during the packet which begins a complete set of service information to indicate that the service information in the following set of information has changed from the previously delivered set of information. Otherwise this bit shall be set to '0'. This bit shall be set to '0' if this CDP packet does not contain a service information section. This bit is duplicated in the cc service information section. The value of this bit shall not be different from the value of the svc_info_change bit in the cc service information section.

svc_info_complete – This bit shall be set to '1' to indicate that the current packet concludes a full set of service information. Otherwise this bit shall be set to '0'. This bit shall be set to '0' if this CDP packet does not contain a service information section. This bit is duplicated in the cc service information section. The value of this bit shall not be different from the value of the svc_info_complete bit in the cc service information section.

caption_active – This bit shall be set to '1' to indicate that the CDP stream is conveying an active caption service. This bit shall be set to '0' in the case that the CDP stream is not conveying an active caption service.

cdp_hdr_sequence_cntr – This is an unsigned 16-bit integer which shall be set to a value of 1 plus the value of CDP_hdr_sequence_cntr in the previous CDP. The value of this counter shall wrap from 65535 to 0. For the first CDP in a sequence of CDPs, dccp_hdr_sequence_cntr may be set to any 16-bit value.

11.2.3 time_code_section

The time code section is optional in a CDP. Time code syntax is indicated in Table 26. This section shall be composed of a section id byte and 4 bytes of time code information. The length of the time code section shall be 5 bytes. Inclusion of this section may help assure that synchronization between captions and pictures is maintained throughout the distribution chain and into the final emission transport stream.

Syntax	Bits	Format	Comment
time_code_section() {			
time_code_section_id	8	uimsbf	0x71
Reserved	2	'11'	
tc_10hrs	2	uimsbf	Tens of hours
tc_1hrs	4	uimsbf	Units of hours
Reserved	1	'1'	
tc_10min	3	uimsbf	Tens of minutes
tc_1min	4	uimsbf	Units of minutes
tc_field_flag	1	uimsbf	see text
tc_10sec	3	uimsbf	Tens of seconds
tc_1sec	4	uimsbf	Units of seconds
Reserved	1	'1'	
drop_frame_flag	1	uimsbf	Drop frame flag
tc_10fr	3	uimsbf	Tens of frames
tc_1fr	4	uimsbf	Units of frames
}			

Table 26 CDP Time Code Section Syntax

time_code_section_id – This 8-bit field shall have the value of 0x71.

tc_field_flag – For interlaced pictures, the value of this flag shall be '0' for interlace field 1, and shall be '1' for interlace field 2. In the case of frame rates equal to or greater than 50 Hz, the frame count shall be interpreted as follows. The frame count shall be doubled, and the tc_field_flag shall be interpreted as an adder to the indicated frame count. The frame count shall be interpreted as (2 * frame + flag). I.e. the frame:flag sequence shall be 0:0, 0:1, 1:0, 1:1, 2:0, 2:1, etc., and this frame:flag sequence shall be interpreted as progressive frame counts 0, 1, 2, 3, 4, 5, etc.

drop_frame_flag – This flag shall be set to '1' when the time code count is being drop-frame compensated.. When the count is not drop-frame compensated, this flag bit shall be set to '0'.

11.2.4 ccdata_section

The ccdata section should normally be present. If present, the ccdata section syntax shall be as indicated in Table 27.

This section need not be included if a CDP stream is intended to carry only caption service information to a PMT or PSIP generator. However, a CDP stream typically conveys both cc data and cc service information in parallel to both the emission encoder and to the PMT and PSIP generators, with each device extracting and using the appropriate information.

This section shall be composed of a section id byte, a count value cc_count, and cc_count groups of 3 bytes. The total length of this section is 2 + 3 * cc_count bytes. The value of cc_count shall be dependent on the frame rate that is indicated in the CDP_header.

The actual caption data is carried in the cc_data_1 and cc_data_2 fields. The value of cc_count is found in table 10-2, and provides sufficient space to carry both NTSC and DTV caption data. The DTV caption data may represent up to 16 caption services. If either the NTSC or DTV caption data is not present, the space is still allocated and filled with null (0x0) values. The NTSC caption data shall come first in this section, followed by the DTV caption data.

Syntax	Bits	Format	Comment
ccdata_section() {			
ccdata_id	8	0x72	Indicates ccdata section
marker_bits	3	'111'	
cc_count	5	uimbf	number of cc constructs in section
for (i = 0 ; i < cc_count ; i++)			
{			
marker_bits	5	'1111 1'	
cc_valid	1	bslbf	as defined in 4.4.1
cc_type	2	bslbf	as defined in 4.4.1
cc_data_1	8	bslbf	
cc_data_2	8	bslbf	
}			
}			

Table 27 CDP CC Data Section Syntax

ccdata_id – This 8-bit field shall have the value 0x72.

cc_count – This 5-bit field shall indicate the number of cc_[xxx] data byte triplets carried in this section, and shall have the value appropriate to the frame rate as indicated in Table 25.

cc_valid – This bit shall be as defined in section 4.4.1. This bit indicates whether the following two bytes of cc_data contain valid captioning data or zero padding.

cc_type – This 2-bit field shall be as defined in section 4.4.1. This field indicates whether the following two bytes of data represent NTSC field 1 or 2 captions, or DVC channel packet data or DTVCC channel packet start.

cc_data_1 – This byte shall be as defined in section 4.4.3.

cc_data_2 – This byte shall be as defined in section 4.4.3.

11.2.5 ccsvinfo_section

The ccsvinfo section carries information for the Caption Service Descriptor. This section shall be composed of a section id byte, indication of controlled changes in the service information, an indication of the number of services that are described in the current packet, and caption service information. The ccsvinfo_section syntax shall be as described in Table 28.

The complete set of caption service information may describe 1 to 16 different caption services. A complete set of service information may be included in the current packet, or may be distributed over a number of packets. The total length of this section is $2 + 7 * \text{svc_count}$ bytes.

Syntax	Bits	Format	Comment
ccsvinfo_section() {			
ccsvinfo_id	8	0x73	Indicates ccsvinfo section
marker_bit	1	'1'	
svc_info_start	1	bit	
svc_info_change	1	bit	
svc_info_complete	1	bit	
svc_count	4	uimbsf	number of svc constructs in section
for (i = 0 ; i < svc_count ; i++)			
{			
reserved	3	'111'	
caption service number	5	uimbsf	
svc_data_byte_1	8	bslbf	
svc_data_byte_2	8	bslbf	
svc_data_byte_3	8	bslbf	
svc_data_byte_4	8	bslbf	
svc_data_byte_5	8	bslbf	
svc_data_byte_6	8	bslbf	
}			
}			

Table 28 CC Service Information Syntax

svc_info_start – This bit shall be set to '1' to indicate that the current packet begins a complete set of service information. Otherwise this bit shall be set to '0'. This bit shall be duplicated in the CDP header section. The value of this bit shall not be different from the value of the svc_info_start bit in the CDP header section.

svc_info_change – This bit shall be set to '1' during the packet which begins a complete set of service information to indicate that the service information in the following set of information has changed from the previously delivered set of information. Otherwise, this bit shall be set to '0'. This bit is duplicated in the CDP header section. The value of this bit shall not be different from the value of the svc_info_change bit in the CDP header section.

svc_info_complete – This bit shall be set to '1' to indicate that the current packet concludes a full set of service information. Otherwise this bit shall be set to '0'. This bit is duplicated in the CDP header section. The value of this bit shall not be different from the value of the svc_info_complete bit in the CDP header section.

NOTE--If a single packet contains a complete set of service information, then both the svc_info_start and svc_info_end bits would be set to '1'.

svc_count – This 4-bit field shall be set to a value equal to the number of services which have service information included in this service information section.

caption_service_number – This 5-bit field carries the caption service number for the service described by the following 6 service data bytes. This field shall have a value of 0x00 when the service data applies to the line 21 (EIA-608-A) service, and shall have a value (between 0x01 – 0x10 inclusive) that matches the caption_service_number contained within svc_data_byte_4 when the service data applies to one of the DTVC services. This field shall not have a value between 0x11 and 0x1f (inclusive).

svc_data_byte_n – These 6 bytes shall carry the caption service data for one service, encoded as described by the caption service descriptor loop in ATSC A/65, section 6.7.3, table 6.17.

11.2.5.1 Service information signalling

The `svc_info` start, change, and end bits allow the receiver of a CDP stream to build up a complete set of cc service information from multiple packets, and to follow changes in the captioning services which are intentionally introduced by the source of the CDP stream. These changes may include beginning and ending of one or more captioning services, or an alteration in a particular service.

When a service is terminated, the `svc_info_change` bit shall be set to '1' during the first packet containing the next full set of service information. The following full sets of service information shall not contain any service information for the terminated service. The caption service number of the terminated service shall not appear in any of the `svc_info` sections.

When a service is changed, the `svc_info_change` bit shall be set to '1' during the first packet containing the next full set of service information. The following full sets of service information shall contain service information representing the new service information for the changed service.

If a CDP stream is switched, there should be a discontinuity in the sequence counters. If the switch occurs between packets, the discontinuity will occur between the previous footer value and the following header value. If the switch occurs during a packet, the balance of the new packet may not be received correctly and there will not be a correct sequence number or checksum in the footer. If a receiver detects that a CDP stream switch has occurred, the receiver should assume all service information has changed and take the next full set of service information as current.

NOTE--In this case, the `svc_info_change` bit will not signal a change of service information. Uncontrolled switches of CDP streams may cause momentary glitches in the display of captions on receivers.

11.2.6 cdp_footer

The CDP footer shall be present in all CDP packets. The CDP footer syntax shall be as defined in Table 29. This section contains a section id byte, a sequence counter value, and a checksum. The sequence counter provides good detection ability as to whether a switch occurred during the transmission of the current CDP packet. The check sum provides a simple means of detecting most transmission errors. The total length of this section is 4 bytes.

Syntax	Bits	Format	Comment
<code>cdp_footer() {</code>			
<code>cdp_footer_id</code>	8	0x74	Indicates CDP footer section
<code>cdp_ftr_sequence_cntr</code>	16	uimsbf	
<code>packet_checksum</code>	8	uimsbf	
<code>}</code>			

Table 29 CDP Footer Syntax

cdp_ftr_sequence_cntr – This 16-bit unsigned integer shall be set to the same value as the `cdp_hdr_sequence_cntr`. Receivers may use the values of `cdp_hdr_sequence_cntr` and `cdp_ftr_sequence_cntr` to detect that the entire packet has been received.

packet_checksum – This 8-bit field shall contain the 8-bit value necessary to make the arithmetic sum of the entire packet (first byte of `cdp_identifier` to `packet_checksum`, inclusive) modulo 256 equal zero.

11.2.7 future_section()

It is possible to define new sections to be included in the CDP. Any newly defined sections shall follow the syntax defined in this clause and in Table 30. All equipment that can receive the CDP shall be capable of ignoring these new sections. The length value is provided so that decoders will know how many bytes of data to skip.

Syntax	Bits	Format	Comment
future_section() {			
future_section_id	8	uimbsf	Value in range 0x75-0xEF
Length	8	uimbsf	Number of bytes of data
for (i = 0 ; i < length ; i++)			
{			
new_data_byte(i)	8		New data content
}			
}			

Table 30 future_section syntax

future_section_id -- Sections defined in the future shall be specified to have a section id value in the range 0x75 to 0xEF, inclusive. Decoders shall be designed to skip over sections that have id values that are not understood.

11.3 Serial Interface

This section defines one specific serial interface which may convey a CDP stream.

11.3.1 Physical Interface

The physical interface for a CDP stream shall be an TIA/EIA-574 interface. The source of the CDP stream shall be DTE with a 9-pin "D" male connector, and the receiver of the CDP stream shall be DCE with a 9-pin "D" female connector.

Table 31 and Table 32 detail the pin connections for the CDP serial interface. The terminology under the "Signal Name" column reflects that used in TIA/EIA-574. The source of the CDP stream shall follow the settings shown in Table 11-7. The receiver of the CDP stream shall follow the settings shown in Table 11-8. The minimum implementation of this serial interface only requires two wires: serial data and ground. In the case where the other 7 lines are connected to drivers, the state of those lines is specified.

Pin	In or Out	Signal Name	Setting	Comments
1	Input	Receive Line Signal Detect	NA	NC or ignored
2	Input	Receive Data	NA	NC or ignored
3	Output	Transmit Data	Active Data	CDP data stream
4	Output	DTE Ready	NC or "ON"	If connected, set "ON"
5	---	Common	GND	Signal common
6	Input	DCE Ready	NA	NC or ignored
7	Output	RTS	NC or "ON"	If connected, set "ON"
8	Input	CTS	NA	NC or ignored
9	Input	Ring Indicator	NA	NC or ignored

Table 31 CDP Source Connector Pinout (DTE Male)

Pin	In or Out	Signal Name	Setting	Comments
1	Output	Receive Line Signal Detect	NC or "ON"	If connected, set "ON"
2	Output	Receive Data	NC or "ONE"	If connected, set to "ONE" (Mark)
3	Input	Transmit Data	Active Data	DCCCP data stream
4	Input	DTE Ready	NA	NC or ignored
5	---	Common	GND	Signal common
6	Output	DCE Ready	NC or "ON"	If connected, set to "ON"
7	Input	RTS	NA	NC or ignored
8	Output	CTS	NC or "ON"	If connected, set to "ON"
9	Output	Ring Indicator	NC or "OFF"	If connected, set to "OFF"

Table 32 CDP Receiver Connector Pinout (DCE Female)

NOTE--NC means not connected; NA means not applicable.

The serial interface shall support operation with the parameters indicated in Table 33. The recommended baud rate is 38,400, but, if necessary, the interface may also be operated at 57,600 or 115,200 baud.

Parameter	Setting
Baud Rate	38,400 (default); 57,600 (optional); 115,200 (optional)
Data Bits	8
Parity	None
Stop Bits	1
Start Bits	1

Table 33 TIA/EIA-574 Interface Parameters for CDP Stream

11.3.2 Operation of the CDP Serial Interface

This section describes the typical application of the CDP serial interface. Other applications are not precluded.

In a typical application, captioning intentions are captured in a high level representation and then rendered into EIA-708 captioning packets. SMPTE time code may be employed to provide means for synchronizing the captioning intentions to the picture. With knowledge of the picture frame rate and time code, the EIA-708 captioning packets may be formed into CDP packets, where one CDP packet corresponds to each picture frame. During real-time streaming of pictures over a video interface and the corresponding CDP packets over the CDP serial interface, the CDP packet for picture frame *n* should be presented to the serial interface during the time window between the beginning of picture frame *n* and the beginning of picture frame *n*+1. In an ideal application, both the CDP packets and the individual pictures will have timecodes, and the MPEG-2 picture encoder will rely on these time codes to establish synchronization between the MPEG-2 encoded pictures and the captioning data included in the user data space of those coded pictures.

When the CDP is conveyed by this serial interface, the CDP shall be preceded by four null bytes (0x00). These null bytes, plus the `cdp_identifier` (0x9669), form a unique 48-bit sync code that allows the serial receiver to synchronize to the CDP stream.

NOTE—These null bytes are not considered part of the CDP, and are not required when the CDP is carried by other interfaces.

BIBLIOGRAPHY

ATSC A/54, Guide to the Use of the ATSC Digital Television Standard (1995)

- Advanced Television Systems Committee (ATSC), 1750 K Street N.W., Suite 1200, Washington, DC 20006; Phone 202-828-3130; Fax 202-828-3131; Internet <http://www.atsc.org/standards.html>

Annex A (Informative)

Figure 22 illustrates one example of a decoder implementation for processing the PMT, EIT and User Data in the DTVCC Transport Stream. It shows the separate paths of the Service Descriptors in the PMT and EIT and the service data in the Picture User Data bits.

NOTE--Different block diagrams may exist for other implementations.

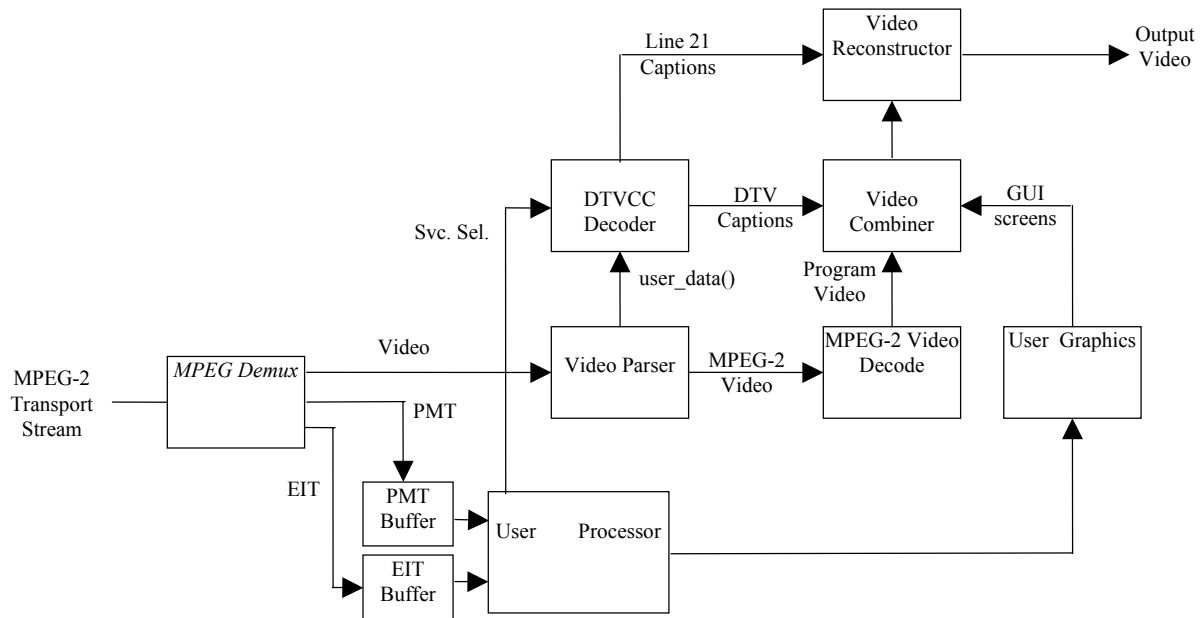


Figure 22 DTVCC Transport Stream Decoder

CEA Document Improvement Proposal

If in the review or use of this document, a potential change is made evident for safety, health or technical reasons, please fill in the appropriate information below and email, mail or fax to:

Consumer Electronics Association
Technology & Standards Department
2500 Wilson Blvd.
Arlington, VA 22201
FAX: 703 907-7693
standards@ce.org

Document No.	Document Title:
Submitter's Name: Submitter's Company:	Telephone No.: FAX No.: e-mail:
Address:	
Urgency of Change: Immediate: <input type="checkbox"/> At next revision: <input type="checkbox"/>	
Problem Area: a. Clause Number and/or Drawing: b. Recommended Changes: c. Reason/Rationale for Recommendation:	
Additional Remarks:	
Signature:	Date:
FOR CEA USE ONLY Responsible Committee: Chairman: Date comments forwarded to Committee Chairman:	

