CLEAR WINDOWS - (CLW)

Name: **ClearWindows** - Clears Text from currently active window(s)

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)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
1	**0**	**0**	**0**	**1**	**0**	**0**	**0**	command
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	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 currently active window(s)

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)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
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 (w_6), 5 (w_5), and 2 (w_2). 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** - Displays currently active window(s)

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)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
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 (w_7), 4 (w_4), 2 (w_2), and 1 (w_1). 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** - Hides currently active window(s)

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)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
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 (w_6), 5 (w_5), 4 (w_4), and 1 (w_1). 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 currently active window(s)

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)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
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 (w_7), 1 (w_1), and 0 (w_0). 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)**

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0	
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 styles to a dynamic preset style number.

Command Type: Pen

Format: **SetPenAttributes** (*pen size, font, offset, italics, underline, edge type)*

Parameters: *pen size* (s) defines which of three pen sizes is to be used for text written to the current window. Note that the pen size displayed on the screen can be overridden by the user. [SMALL, STANDARD, LARGE] == [0, 1, 2].

font style (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).

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

text tag (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).

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

offset (o) specifies sub-scripting and super-scripting attributes for text written to the current window. [SUBSCRIPT, NORMAL, SUPERSCRIPT] == [0, 1, 2].

italics (i) specifies if text written to the current window is italicized. [YES, NO] == [1, 0].

underline (u) specifies if text written to the current window is underlined. [YES, NO] == [1, 0].

edge type (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. 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. The next group of text written to the current window will start at the specified row and column. 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	
1	0	0	0	1	1	1	1	command

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, SDTVand 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 (ANSI/EIA-608) 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-A, 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 Bpsper 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 (™
- 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 ($^1/_8$), G2 char code 0x76	G0 percent sign (%), char code 0x25
three-eighths ($^3/_8$), G2 char code 0x77	G0 percent sign (%), char code 0x25
five-eighths ($^5/_8$), G2 char code 0x78	G0 percent sign (%), char code 0x25
seven-eighths ($^7/_8$), 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 minimum screen coordinate recommendations 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
4:3	75v x 160h	15v x 32h	4
16:9	75v x 210h	15v x 42h	4
other	75v x (5 x H)	15v x H*	4

Table 18 Minimum Screen Coordinates

*H = 32 x (the width of the screen in relation to a 4:3 display). For example, the 16:9 format is 4/3 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 maximum 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

For fixed-spaced (i.e., non-proportionally-spaced) fonts, all decoders should implement "left", "right", and "center" caption-text justification, with "full" justification implementation being optional. "Full" justification is to be converted to "left" justification when "full" justification is not implemented

For proportionally-spaced fonts, all decoders should implement left, right and center justification. For center and right justification, decoders need not display any portion of a received row of text until the entire row is received (i.e., until a CR or any non-text related command is received). This easement allows the decoder to calculate the "width" of a text row before displaying it, and thus, giving the decoder all of the parameters necessary to calculate a reasonable offset of the row within a window to effect center or right justification.

Receipt of a character for a displayed row which already contains text and which has already been "justified" 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

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	*NTSC Style PopUp Captions*
2	LEFT	LEFT -TO- RIGHT	BOTTOM -TO- TOP	NO	SNAP	n/a	n/a	n/a	TRANS- PARENT	NONE	n/a	*PopUp Captions w/o Black Background*
3	CNTR	LEFT -TO- RIGHT	BOTTOM -TO- TOP	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style Centered PopUp Captions*
4	LEFT	LEFT -TO- RIGHT	BOTTOM- TO- TOP	YES	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style RollUp Captions*
5	LEFT	LEFT -TO- RIGHT	BOTTOM- TO- TOP	YES	SNAP	n/a	n/a	n/a	TRANS- PARENT	NONE	n/a	*RollUp Captions w/o Black Background*
6	CNTR	LEFT -TO- RIGHT	BOTTOM- TO- TOP	YES	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*NTSC Style Centered RollUp Captions*
7	LEFT	TOP -TO- BOTTOM	RIGHT -TO- LEFT	NO	SNAP	n/a	n/a	(0,0,0) Black	SOLID	NONE	n/a	*Ticker Tape*

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	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*Default NTSC Style**
2	STNDR	1	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Mono w/ Serif*
3	STNDR	2	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Prop w/ Serif*
4	STNDR	3	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Mono w/o Serif*
5	STNDR	4	NORMAL	NO	NO	NONE	(2,2,2) White	SOLID	(0,0,0) Black	SOLID	n/a	*NTSC Style* Prop w/o Serif*
6	STNDR	3	NORMAL	NO	NO	UNIFRM	(2,2,2) White	SOLID	n/a	TRANS-PARENT	(0,0,0) Black	*Mono w/o Serif, Bordered Text, No BG*
7	STNDR	4	NORMAL	NO	NO	UNIFRM	(2,2,2) White	SOLID	n/a	TRANS-PARENT	(0,0,0) Black	*Prop. w/o Serif, Bordered Text, No BG*

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 flexibilty 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 readablility 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 Encoder Manufacturer Recommendations

This section discusses issues and recommendations relevant to DTVCC encoder manufacturers.[1]

[1] EIA-708-A users should note that, at the time of publication, standards develoment efforts are underway in other forums to address in-plant distribution of video, audio, PSIP and closed captioning data. Alternatives to the examples included in section 10, including those with a separate port for PSIPdata (as well as other alternatives), are under consideration in standards forums as are individual manufacturer implementations.