Sub Station Alpha v4.00+ Script Format

This document was SSA's format specification originally (can be found at http://www.eswat.demon.co.uk). Updates and differences are marked red.

1. General Information

The information in this document assumes that you are familiar with the terms and concepts used by Sub Station Alpha (SSA). These are documented in SSA's help file, ssa.hlp which is distributed with the program, or can be downloaded separatelyfrom http://www.eswat.demon.co.uk.

1. **The SSA v4.00 script format is different to previous versions of SSA**
   SSA v4.00 will read scripts from older versions, but v4.00 scripts will not load into older versions of SSA correctly.

   Some of the changes in the script format are intended to allow all versions of SSA from v4.00 onwards to read any present or future SSA scripts. In particular, the new "Format" lines allow SSA to read only the information it recognises and discard any new information that is added in future scripts.

2. **Scripts are plain (DOS) text files.**
   This means they can be "manually" editied using any text editor, but caution must be exercised when doing this - SSA assumes that scripts will adhere to the "rules" set out in this document, and any errors may lead to unpredictable results when the script is loaded into SSA.

2. **The script is divided into “ini file” style sections**
   However, SSA scripts are not true Windows .ini files and you cannot do certain things you might expect!

3. **Most lines in each section begin with some sort of code - a "line descriptor"**, to say what information is held in it. The descriptor is terminated by a colon.

3. **The information fields in each line are separated by a commas.**
   This makes it illegal to use commas in character names and style names (SSA prevents you putting commas in these). It also makes it quite easy to load chunks of an SSA script into a spreadsheet as a CSV file, and chop out columns of information you need for another subtitling program.

4. **SSA does not care what order events are entered in.**
   They could be entered in complete reverse order, and SSA would still play everything correctly in the right order ie. you cannot assume that each dialogue line is in chronological order in the script file.

5. **Incorrectly formatted lines are ignored.**
   SSA will discard any lines it doesn't understand, and give a warning after the script has loaded giving the number of lines it discarded.

6. **Lines cannot be split**
   Each entry in a script contains all its information in a single line.

7. **If unknown styles are used in the script, the *Default style will be used.**
   For example, if lines have been pasted in from another script, without the corresponding Style information then when SSA plays the script, the Default style settings will be used.

8. **If a Style specifies a font which is not installed, then Arial will be used instead.**
   This can happen with scripts which you did not create yourself - the original authors may have fonts installed which you don't have.

2. The sections in a Sub Station Alpha script

[Script Info]
This section contains headers and general information about the script.
The line that says “[Script Info]” **must** be the first line in a v4 script.

[v4 Styles]
This section contains all Style definitions required by the script. Each “Style” used by subtitles in the script should be defined here.

ASS uses [v4 Styles+]

[Events]
This section contains all the events for the script - all the subtitles, comments, pictures, sounds, movies and commands. Basically, everything that you see in Sub Station Alpha’s main-screen “grid” is in this section.

[Fonts]
This section contains text-encoded font files, if the user opted to embed non-standard fonts in the script. Only truetype fonts can be embedded in SSA scripts. Each font file is started with a single line in the format:

fontname: <name of file>

The word “**fontname**” must be in lower case (upper case will be interpretted as part of a text-encoded file).

<name of file> is the file name that SSA will use when saving the font file. It is:
the name of the original truetype font,
plus an underscore,
plus an optional “B” if Bold,
plus an optional “I” if Italic,
plus a number specifying the font encoding (character set),
plus “.ttf”
Eg.
fontname: chaucer_B0.ttf
fontname: comic_0.ttf

The fontname line is followed by lines of printable characters, representing the binary values which make up the font file. Each line is 80 characters long, except the last one which may be less.

The conversion from binary to printable characters is a form of Uuencoding, the details of this encoding is contained in "Appendix B", below.

[Graphics]
This sections contains text-encoded graphic files, if the user opted to embed any pictures they used in the script. The binary picture files are text-encoded, which is inefficient, but ensures that SSA scripts can still be handled by any text editor, if required.

Each graphic file is started with a single line in the format:
filename: <name of file>

The word “**filename**” must be in lower case (upper case will be interpreted as part of a text-encoded file).

<name of file> is the file name that SSA will use when saving the picture file. It will match the filename of a picture used in the script.

SSA saves any files found in the script in a subdirectory off SSA's program directory, "Pictures"
eg. c:\program files\Sub Station Alpha v4.00\Pictures. SSA will attempt to load files using the paths specified in the script, but if they are not found, it will look in the "Pictures" subdirectory for them.

The fontname line is followed by lines of printable characters, fontrepresenting the binary values which make up the picture font file - format is as per embedded font files.

3. The line types in a Sub Station Alpha script

This briefly describes each of the line types that can appear in a Sub Station Alpha Script. Full details of the information held in each line type is in the next chapter.

!: This is a comment used in the script file only. It is not visible when you load the script into SSA.

Title: This is a description of the script

Original Script: The original author(s) of the script

Original Translation: (optional) The original translator of the dialogue

Original Editing: (optional) The original script editor(s), typically whoever took the raw translation and turned it into idiomatic english and reworded for readability.

Original Timing: (optional) Whoever timed the original script

Synch Point: (optional) Description of where in the video the script should begin playback.

Script Updated By: (optional) Names of any other subtitling groups who edited the original script.

Update Details: The details of any updates to the original script made by other subtilting groups.

ScriptType: This is the SSA script format version eg. "V3.00".

Collisions: This determines how subtitles are moved, when preventing onscreen collisions

PlayResY: This is the height of the screen used by the authors when playing the script.

PlayResX: This is the width of the screen used by the authors when playing the script.

PlayDepth: This is the colour depth used by the authors when playing the script.

Timer: This is the Timer Speed for the script, as a percentage.
eg. "100.0000" is exactly 100%.

The timer speed is a time multiplier applied to SSA's clock to provide a ramp time.

Style: This is a Style definition, used to format text displayed by the script.

Dialogue: This is a Dialogue event, ie. Some text to display.

Comment: This is a "comment" event.
This contains the same information as a Dialogue, Picture, Sound, Movie, or Command event, but it is ignored during script playback.

Picture: This is a "picture" event, which means SSA will display the specified .bmp, .jpg, .gif, .ico or .wmf graphic.

Sound: This is a "sound" event, which means SSA will play the specified .wav file.

Movie: This is a "movie" event, which means SSA will play the specified .avi file.

Command: This is a "command" event, which means SSA will execute the specified program as a background task.

4. Header lines, [Script Info] section

; Semicolon. Any text can follow the semicolon
This is a comment used in the script file only. It is not visible when you load the script into SSA. The semicolon **must** be the first character in the line. This replaces the **!:** line type used in previous script versions.

Title: This is a description of the script. If the original author(s) did not provide this information then <untitled> is automatically substituted.

Original Script: The original author(s) of the script. If the original author(s) did not provide this information then <unknown> is automatically substituted.

Original Translation: (optional) The original translator of the dialogue. This entry does not appear if no information was entered by the author.

Original Editing: (optional) The original script editor(s), typically whoever took the raw translation and turned it into idiomatic english and reworded for readability. This entry does not appear if no information was entered by the author.

Original Timing: (optional) Whoever timed the original script. This entry does not appear if no information was entered by the author.

Synch Point: (optional) Description of where in the video the script should begin playback. This entry does not appear if no information was entered by the author.

Script Updated By: (optional) Names of any other subtitling groups who edited the original script. This entry does not appear if subsequent editors did not enter the information.

Update Details: The details of any updates to the original script - made by other subtitling groups. This entry does not appear if subsequent editors did not enter any information.

Script Type: This is the SSA script format version eg. "V4.00". It is used by SSA to give a warning if you are using a version of SSA older than the version that created the script.

ASS version is “V4.00+”.

Collisions: This determines how subtitles are moved, when automatically preventing onscreen collisions.

If the entry says "**Normal**" then SSA will attempt to position subtitles in the position specified by the "margins". However, subtitles can be shifted vertically to prevent onscreen collisions. With "normal" collision prevention, the subtitles will "stack up" one above the other - but they will always be positioned as close the vertical (bottom) margin as possible - filling in "gaps" in other subtitles if one large enough is available.

If the entry says "**Reverse**" then subtitles will be shifted upwards to make room for subsequent overlapping subtitles. This means the subtitles can nearly always be read top-down - but it also means that the first subtitle can appear half way up the screen before the subsequent overlapping subtitles appear. It can use a lot of screen area.

PlayResY: This is the height of the screen used by the script's author(s) when playing the script. SSA v4 will automatically select the nearest enabled setting, if you are using Directdraw playback.

PlayResX: This is the width of the screen used by the script's author(s) when playing the script. SSA will automatically select the nearest enabled, setting if you are using Directdraw playback.

PlayDepth: This is the colour depth used by the script's author(s) when playing the script. SSA will automatically select the nearest enabled setting if you are using Directdraw playback.

Timer: This is the Timer Speed for the script, as a percentage.

eg. "100.0000" is exactly 100%. It has four digits following the decimal point.

The timer speed is a time multiplier applied to SSA's clock to stretch or compress the duration of a script. A speed greater than 100% will reduce the overall duration, and means that subtitles will progressively appear sooner and sooner. A speed less than 100% will increase the overall duration of the script means subtitles will progressively appear later and later (like a positive ramp time).

The stretching or compressing only occurs during script playback - this value does not change the actual timings for each event listed in the script.

Check the SSA user guide if you want to know why "Timer Speed" is more powerful than "Ramp Time", even though they both achieve the same result.

WrapStyle: **Defines the default wrapping style.**
0: smart wrapping, lines are evenly broken
1: end-of-line word wrapping, only \N breaks
2: no word wrapping, \n \N both breaks
3: same as 0, but lower line gets wider.

5. Style Lines, [v4+ Styles] section

Styles define the appearance and position of subtitles. All styles used by the script are are defined by a Style line in the script.

Any of the the settings in the Style, (except shadow/outline type and depth) can overridden by control codes in the subtitle text.

The fields which appear in each Style definition line are named in a special line with the line type “Format:”. The Format line must appear before any Styles - because it defines how SSA will interpret the Style definition lines. The field names listed in the format line must be correctly spelled! The fields are as follows:

Name, Fontname, Fontsize, PrimaryColour, SecondaryColour, TertiaryColour, BackColour, Bold, Italic, Underline, StrikeOut, ScaleX, ScaleY, Spacing, Angle, BorderStyle, Outline, Shadow, Alignment, MarginL, MarginR, MarginV, AlphaLevel, Encoding

The format line allows new fields to be added to the script format in future, and yet allow old versions of the software to read the fields it recognises - even if the field order is changed.

Field 1: **Name**. The name of the Style. Case sensitive. Cannot include commas.

Field 2: **Fontname**. The fontname as used by Windows. Case-sensitive.

Field 3: **Fontsize**.

Field 4: **PrimaryColour.** A long integer BGR (blue-green-red) value. ie. the byte order in the hexadecimal equivelent of this number is BBGGRR

This is the colour that a subtitle will normally appear in.

Field 5: **SecondaryColour.** A long integer BGR (blue-green-red) value. ie. the byte order in the hexadecimal equivelent of this number is BBGGRR

This colour may be used instead of the Primary colour when a subtitle is automatically shifted to prevent an onscreen collsion, to distinguish the different subtitles.

Field 6: **OutlineColor** (~~**TertiaryColour**~~). A long integer BGR (blue-green-red) value. ie. the byte order in the hexadecimal equivelent of this number is BBGGRR

This colour may be used instead of the Primary or Secondary colour when a subtitle is automatically shifted to prevent an onscreen collsion, to distinguish the different subtitles.

Field 7: **BackColour**. This is the colour of the subtitle outline or shadow, if these are used. A long integer BGR (blue-green-red) value. ie. the byte order in the hexadecimal equivelent of this number is BBGGRR.

Field 4-7: The color format contains the alpha channel, too. (AABBGGRR)

Field 8: **Bold**. This defines whether text is bold (true) or not (false). -1 is True, 0 is False. This is independant of the Italic attribute - you can have have text which is both bold and italic.

Field 9: **Italic**. This defines whether text is italic (true) or not (false). -1 is True, 0 is False. This is independant of the bold attribute - you can have have text which is both bold and italic.

Field 9.1: Underline. [-1 or 0]

Field 9.2: Strikeout. [-1 or 0]

Field 9.3: ScaleX. Modifies the width of the font. [percent]

Field 9.4: ScaleY. Modifies the height of the font. [percent]

Field 9.5: Spacing. Extra space between characters. [pixels]

Field 9.6: Angle. The origin of the rotation is defined by the alignment. Can be a floating point number. [degrees]

Field 10: **BorderStyle**. 1=Outline + drop shadow, 3=Opaque box

Field 11: **Outline.** If BorderStyle is 1, then this specifies the width of the outline around the text, in pixels. Values may be 0, 1, 2, 3 or 4.

Field 12: **Shadow.** If BorderStyle is 1, then this specifies the depth of the drop shadow behind the text, in pixels. Values may be 0, 1, 2, 3 or 4. Drop shadow is always used in addition to an outline - SSA will force an outline of 1 pixel if no outline width is given.

Field 13: **Alignment**. This sets how text is "justified" within the Left/Right onscreen margins, and also the vertical placing. Values may be 1=Left, 2=Centered, 3=Right. Add 4 to the value for a "Toptitle". Add 8 to the value for a "Midtitle".
eg. 5 = left-justified toptitle

Field 13: Alignment, but after the layout of the numpad (1-3 sub, 4-6 mid, 7-9 top).

Field 14: **MarginL**. This defines the Left Margin in pixels. It is the distance from the left-hand edge of the screen.The three onscreen margins (MarginL, MarginR, MarginV) define areas in which the subtitle text will be displayed.

Field 15: **MarginR**. This defines the Right Margin in pixels. It is the distance from the **right-hand** edge of the screen. The three onscreen margins (MarginL, MarginR, MarginV) define areas in which the subtitle text will be displayed.

Field 16: **MarginV**. This defines the vertical Left Margin in pixels.
For a **subtitle**, it is the distance from the **bottom** of the screen.
For a **toptitle**, it is the distance from the **top** of the screen.
For a **midtitle**, the value is ignored - the text will be vertically centred

Field 17: **AlphaLevel.** This defines the transparency of the text. SSA does not use this yet.

Field 17: Not present in ASS.

Field 18: **Encoding.** This specifies the font character set or encoding and on multi-lingual Windows installations it provides access to characters used in multiple than one languages. It is usually 0 (zero) for English (Western, ANSI) Windows.

When the file is Unicode, this field is useful during file format conversions.

5. Dialogue event lines, [Events] section

These contain the subtitle text, their timings, and how it should be displayed.
The fields which appear in each Dialogue line are defined by a **Format:** line, which must appear before any events in the section. The format line specifies how SSA will interpret all following Event lines. The field names must be spelled correctly, and are as follows:

Marked, Start, End, Style, Name, MarginL, MarginR, MarginV, Effect, Text

The last field will always be the Text field, so that it can contain commas. The format line allows new fields to be added to the script format in future, and yet allow old versions of the software to read the fields it recognises - even if the field order is changed.

Field 1: **Marked**
Marked=0 means the line is not shown as "marked" in SSA.
Marked=1 means the line is shown as "marked" in SSA.

Field 1: Layer (any integer)

Subtitles having different layer number will be ignored during the collusion detection.

Higher numbered layers will be drawn over the lower numbered.

Field 2: **Start**
Start Time of the Event, in 0:00:00:00 format ie. Hrs:Mins:Secs:hundredths. This is the time elapsed during script playback at which the text will appear onscreen. Note that there is a **single** digit for the hours!

Field 3: **End**
End Time of the Event, in 0:00:00:00 format ie. Hrs:Mins:Secs:hundredths. This is the time elapsed during script playback at which the text will disappear offscreen. Note that there is a **single** digit for the hours!

Field 4: **Style**
Style name. If it is "Default", then your **own** *Default style will be subtituted.

However, the Default style used by the script author IS stored in the script even though SSA ignores it - so if you want to use it, the information is there - you could even change the Name in the Style definition line, so that it will appear in the list of "script" styles.

Field 5: **Name**
Character name. This is the name of the character who speaks the dialogue. It is for information only, to make the script is easier to follow when editing/timing.

Field 6: **MarginL**
4-figure Left Margin override. The values are in pixels. All zeroes means the default margins defined by the style are used.

Field 7: **MarginR**
4-figure Right Margin override. The values are in pixels. All zeroes means the default margins defined by the style are used.

Field 8: **MarginV**
4-figure Bottom Margin override. The values are in pixels. All zeroes means the default margins defined by the style are used.

Field 9: **Effect**
Transition Effect. This is either empty, or contains information for one of the three transition effects implemented in SSA v4.x

The effect names are case sensitive and must appear exactly as shown. The effect names do **not** have quote marks around them.

"Karaoke" means that the text will be successively highlighted one word at a time.

Karaoke as an effect type is obsolete.

"Scroll up;y1;y2;delay[;fadeawayheight]"means that the text/picture will scroll up the screen. The parameters after the words "Scroll up" are separated by semicolons.

The y1 and y2 values define a vertical region on the screen in which the text will scroll. The values are in pixels, and it doesn't matter which value (top or bottom) comes first. If the values are zeroes then the text will scroll up the full height of the screen.

The delay value can be a number from 1 to 100, and it slows down the speed of the scrolling - zero means no delay and the scrolling will be as fast as possible.

“Banner;delay” means that text will be forced into a single line, regardless of length, and scrolled from right to left accross the screen.

The delay value can be a number from 1 to 100, and it slows down the speed of the scrolling - zero means no delay and the scrolling will be as fast as possible.

"Scroll down;y1;y2;delay[;fadeawayheight]"

“Banner;delay[;lefttoright;fadeawaywidth]”

lefttoright 0 or 1. This field is optional. Default value is 0 to make it backwards compatible.

When delay is greater than 0, moving one pixel will take (1000/delay) second.

(WARNING: Avery Lee’s “subtitler” plugin reads the “Scroll up” effect parameters as delay;y1;y2)

fadeawayheight and fadeawaywidth parameters can be used to make the scrolling text at the sides transparent.

Field 10: **Text**

Subtitle Text. This is the actual text which will be displayed as a subtitle onscreen. Everything after the 9th comma is treated as the subtitle text, so it can include commas.

The text can include \n codes which is a line break, and can include Style Override control codes, which appear between braces { }.

6. Comment event lines, [Events] section

These can contain the same information as any of the other event line types, but they will be ignored when the script is played.

7. Picture event lines, [Events] section

These contain the same information as Dialogue events, but Field 10 contains the full path and filename of the picture to display, instead of subtitle text.

The Style specified is ignored. The "scroll up" transition effect can be used for picture events.

The Left and Vertical Margin Overrides specify the bottom-left corner position of the picture. A left margin of all zeroes means that the picture will be horizontally centered. A vertical margin of all zeroes means that the picture will be vertically centered.

8. Sound event lines, [Events] section

These contain the same information as Dialogue events, but Field 10 contains the full path and filename of the wav file to play, instead of subtitle text.

The Style and margins are ignored. The End time is also ignored - the wav will play until it finishes, or until another wav file is played.

If an avi movie is played at the same time as a wav is already playing, then any sound in the avi will not be heard. Similarly, if a wav starts playing when an avi movie with sound is already playing then the wav will not be heard.

9. Movie event lines, [Events] section

These contain the same information as Dialogue events, but Field 10 contains the full path and filename of the avi file to play, instead of subtitle text.

The Style is ignored. Transition effects are ignored.

The End time specifies when the movie picture will disappear - but if th eavi file lasts longer, then the sound will continue to be heard.

The Left and vertical Margin Overrides specify the TOP-LEFT corner position of the picture (unlike picture events). A left margin of all zeroes means that the picture will be horizontally centered. a vertical margin of all zeroes means that the picture will be verticall centered.

If an avi movie is played at the same time as a wav is already playing, then any sound in the avi will not be heard. Similarly, if a wav starts playing when an avi movie with sound is already playing then the wav will not be heard.

10. Command event lines, [Events] section

These contain the same information as Dialogue events, but Field 10 contains the full path and filename of the program to execute, instead of subtitle text.

The Style is ignored. The margins are ignored. Transition effects are ignored. The End time is also ignored - the program will execute until it ends, or is "manually" closed.

There are also internal SSA commands which can appear in SSA scripts - the "SSA:Pause", “SSA:Wait for trigger” command events, and genlock control commands. These all begin with "SSA:"

The SSA:Pause command has the same effect as pressing "P" during script playback. It is useful as a second "synch point" to resume subtitling after switching sides of a laserdisk.

The “SSA:Wait for audio trigger” command has the same effect as pressing "P" during script playback, but pausing is automatically cancelled if the audio input to the computer exceeds a specified “trigger” level. It is useful as a second "synch point" to resume subtitling after switching sides of a laserdisk. The audio triggering can be overridden to resume playback - by pressing "P".

Audio triggering "times out" after 10 minutes - If no audio peak of sufficient magnitude is received, and "P" is not pressed within 10 minutes - then playback will resume anyway.

Appendix A: Style override codes

This is a reference which may be useful for those of you who wish to learn the style override codes, so you can type them in manually without using the "override style" dialogue box.

All Override codes appear within braces { } except the newline \n and \N codes.
All override codes are always preceded by a backslash \
Several overrides can be used within one set of braces.

Each override affects all text following the override. To apply an override only to selected text, you need a second "cancelling" override after the selected text, to "undo" the effect of the first override.

Some overrides automatically apply to ALL the text - currently this is just alignment overrides, but more may be added later (eg. Shadow/outline depth overrides).

\n — New line (carriage return)
\n is ignored by SSA if “smart-wrapping” is enabled
eg. This is the first line\nand this is the second

\N — New line (carriage return). This is used by SSA instead of \n if “smart-wrapping” is enabled.

\b<0 or 1> — \b1 makes the text bold. \b0 forces non-bold text.
eg. There is a {\b1}bold {\b0}word here

When this parameter is greater than 1, it will be used as the weight of the font. (400 = Normal, 700 = Bold, note: most fonts will quantize to 2 or 3 levels of thickness)

\i<0 or 1> — \i1 makes the text italic. \i0 forces non-italic text.
eg. There is an {\i1}italicised {\i0}word here

\u<0 or 1> — **underline**

\s<0 or 1> — **strikeout**

\bord<width> — **border**

\shad<depth> — **shadow**

\be<0 or 1> — **blur edges**

\fn<font name> — <font name> specifies a font which you have installed in Windows. This is case sensitive.
eg. Here is some {\fnCourier New}fixed space text

If you use a font name that doesn't exist, then Arial will be used instead.

\fs<font size> — <font size> is a number specifying a font point size.
eg. {\fs16}This is small text. {\fs28}This is large text

\fsc<x or y><percent>
<x or y> x scales horizontally, y scales vertically
<percent>

\fsp<pixels > — **<pixels> changes the distance between letters. (default: 0)**

\fr[<x/y/z>]<degrees>
<degrees> sets the rotation angle around the x/y/z axis.

\fr defaults to \frz.

\fe<charset> — <charset> is a number specifying the character set (font encoding)

\c&H<bbggrr>**&** <bbggrr> is a hexadecimal RGB value, but in reverse order. Leading zeroes are not required.

eg. {\c&HFF&}This is pure, full intensity red
{\c&HFF00&}This is pure, full intensity Green
{\c&HFF0000&}This is pure, full intensity Blue
{\c&HFFFFFF&}This is White
{\c&HA0A0A&}This is dark grey

\1c&Hbbggrr&, \2c&Hbbggrr&, \3c&Hbbggrr&, \4c&Hbbggrr& to set specific colors.

\1a&Haa&, \2a&Haa&, \3a&Haa&, \4a&Haa& to set specific alpha channels.

\alpha defaults to \1a

\a<alignment> <alignment> is a number specifying the onscreen alignment/positioning of a subtitle.

A value of 1 specifies a left-justified subtitle
A value of 2 specifies a centered subtitle
A value of 3 specifies a right-justified subtitle

Adding 4 to the value specifies a "Toptitle"
Adding 8 to the value specifies a "Midtitle"

0 or nothing resets to the style default (which is usually 2)

eg. {\a1}This is a left-justified subtitle
{\a2}This is a centered subtitle
{\a3}This is a right-justified subtitle

{\a5}This is a left-justified toptitle
{\a11}This is a right-justified midtitle

Only the first appearance counts.

\an<alignment> **numpad layout**

Only the first appearance counts.

\k<duration> <duration> is the amount of time that each section of text is highlighted for in a dialogue event with the Karaoke effect. The durations are in hundredths of seconds.

eg. {\k94}This {\k48}is {\k24}a {\k150}karaoke {\k94}line

\k<duration> highlight by words
\kf or \K<duration> fill up from left to right
\ko<duration> outline highlighting from left to right

\q<num> **<num> wrapping style**

\r[<style>] This cancels all previous style overrides in a line

<style> Restores to <style> instead of the dialogue line default.

Any style modifier followed by no recognizable parameter resets to the default.

Functions:

\t([<t1>, <t2>,] [<accel>,] <style modifiers>)

<t1>, <t2> Animation beginning, ending time offset [ms] (optional)

<accel> Modifies the linearity of the transformation (optional)

The following calculation is performed to get the coefficient needed to interpolate between the given style modifiers: pow((t-t1)/(t2-t1), accel), where t is the time offset for the subtitle.

The meaning of <accel>:
1: the transformation is linear
between 0 and 1: will start fast and slow down
greater than 1: will start slow and get faster

As an example, using 2 will make growing the letters (by {\fscx200\fscy200}) look linear rather than slowering.

<style modifiers>Any style modifier which can be animated:
\c,\1-4c,\alpha,\1-4a,\fs,\fr,\fscx,\fscy,\fsp,\bord,\shad,\clip (only the rectangular \clip)

\move(<x1>, <y1>, <x2>, <y2>[, <t1>, <t2>])

<x1>, <y1> The coordinate to start at.
<x2>, <y2> The coordinate to end at.
<t1>, <t2> Animation beginning, ending time offset [ms] (optional)

The origin of the movement is defined by the alignment type.

\pos(<x>, <y>) Defaults to \move(<x>, <y>, <x>, <y>, 0, 0)

\org(<x>, <y>) Moves the default origin at (x,y). This is useful when moving subtitles in the direction of rotation.

WARNING: \t, \move and \pos will ignore collusion detection.

\fade(<a1>, <a2>, <a3>, <t1>, <t2>, <t3>, <t4>)

<a1> Alpha value before <t1>
<a2> Alpha value between <t2> and <t3>
<a3> Alpha value after <t4>
<t1>, <t4> Animation beginning, ending time offset [ms]
<t1> - <t2> Alpha value will be interpolated between <a1> and <a2>
<t2> - <t3> Alpha value will be set to <a2>
<t3> - <t4> Alpha value will be interpolated between <a2> and <a3>

\fad(<t1>, <t2>) <t1> the time length of fading in
<t2> the time length of fading out

\clip(<x1>, <y1>, <x2>, <y2>)

Clips any drawing outside the rectangle defined by the parameters.

\clip([<scale>,] <drawing commands>)

Clipping against drawn shapes.

<scale> has the same meaning as in the case of \p<scale>

Drawings:

\p<scale> <scale>

Turns on drawing mode and sets the magnification level of the coordinates at the same time. Scale is interpreted as two to the power of (<scale> minus one). For

example {\p4} and the coordinate (8,16) will mean the same as {\p1} and (1,2). This feature can be useful for sub-pixel accuracy.

If 0, drawing mode is turned off and the text is interpreted as usual.

\pbo<y> <y> baseline offset. By default any drawings are positioned on the current baseline. With this value it is possible to move them up or down by <y> pixels. (up: y<0, down: y>0)

Drawing commands:

m <x> <y> Moves the cursor to <x>, <y>

n <x> <y> Moves the cursor to <x>, <y> (unclosed shapes will be left open)

l <x> <y> Draws a line to <x>, <y>

b <x1> <y1> <x2> <y2> <x3> <y3>

3rd degree bezier curve to point 3 using point 1 and 2 as the control points

s <x1> <y1> <x2> <y2> <x3> <y3> .. <xN> <yN>

3rd degree uniform b-spline to point N, must contain at least 3 coordinates

p <x> <y> extend b-spline to <x>, <y>

c close b-spline

Things you should know:

Commands must appear after {\p1+} and before {\p0}. (except for \clip(..))

Drawings must always start with a move to command.

Drawings must form a closed shape.

All unclosed shape will be closed with a straight line automatically.

Overlapping shapes in the Dialogue line will be XOR-ed with each-other.

If the same command follows another, it isn't needed to write its identifier letter again, only the coordinates.

The coordinates are relative to the current cursor position (baseline) and the alignment mode.

Commands p and c should only follow other b-spline commands.

Examples:

Square: m 0 0 l 100 0 100 100 0 100

Rounded square: m 0 0 s 100 0 100 100 0 100 c (c equals to "p 0 0 100 0 100 100" in this case)

Circle (almost): m 50 0 b 100 0 100 100 50 100 b 0 100 0 0 50 0 (note that the 2nd 'b' is optional here)

Appendix B: embedded font/picture encoding

SSA's font and picture file embeddeding is a form of UUEncoding.

It takes a binary file, three bytes at a time, and converts the 24bits of those bytes into four 6-bit numbers. 33 is added to each of these four numbers, and the corresponding ascii character for each number is written into the script file.

The offset of 33 means that lower-case characters cannot appear in the encoded output, and this is why the "filename" lines are always lower case.

Each line of an encoded file is 80 characters long, except the last one, which may be shorter.

If the length of the file being encoded is not an exact multiple of 3, then for odd-number filelengths, the last byte is multiplied by hexadecimal 100, and the most significant 12 bits are converted to two characters as above. For even-number filelengths, the last two bytes are multiplied by hexadecimal 10000, and the most significant 18 bits are converted to three characters as above.

There is no terminating code for the embedded files. If a new [section] starts in the script, or if another filename line is found, or the end of the script file is reached then the file is considered complete.