1 Introduction

Various fonts and colored text content can be saved with this file. In addition, the text width, alignment and separator may be included in the format. For compression, all memory are combined in a bit oriented manner. The text file is created using the program "TextWriter". The file extension is called ".tdct" (Two Dimension Compress Text).

2 Value types

Type	Description	Area
INT8	8Bit with sign	-128 to 127
INT16	16Bit with sign	-32.768 to 32.767
INT32	32Bit with sign	-2.147.483.648 to 2.147.483.647
INT64	64Bit with sign	-9.223.372.036.854.775.808 to 9.223.372.036.854.775.807
BYTE	8Bit unsigned	0 to 255
UINT16	16Bit unsigned	0 to 65.535
UINT32	32Bit unsigned	0 to 4.294.967.295
UINT64	64Bit unsigned	0 to 18.446.744.073.709.551.615
CHAR	8Bit character	0 to 255
WCHAR	16Bit character	0 to 65.535
FLOAT	32Bit floating point	± 1.5e-45 to ± 3.4e38
DOUBLE	64Bit floating point	± 5.0e-324 to ± 1.7e308
MEMORY	Memory in bytes	
...[]	Array	see section 2.1
-> {	Start of the loop	see section 2.2
} <-	End of the loop	see section 2.2
...	Next table	see section 2.3
!	Dependence	see section 2.4

Table 2: Value types

2.1 Array

The set consists of a specific value type. The count of the set is detailed in the information and is usually the previous format value.

Example:

```
A Array INT16[] contains a certain count of INT16 values { INT16, INT16, INT16,
INT16, ... }.

INT16[], BYTE[], UINT32[], WCHAR[], usw.
```

2.2 The loop

In a loop, the format is repeatedly run through. The count of run through is specified in detail in the information and is usually the previous value.

2.3 Next table

The file format is displayed further in the section and the table specified.

2.4 Dependence

The value exists only in the file when a specific condition is met.

3 Description

3.1 File format

Typ	Name	Description	Info
UINT32	IDNumber	The file must have the ID number (0x57544454).	4.1
BYTE	Version	The file version must be 1 for this description.	4.2
BYTE	Alignment	The alignment of the text.	4.3
BYTE	Flags	The bits for the format description.	4.4
INT32!	ThumbnailSize	The size of the thumbnail in bytes.	4.5
MEMORY!	ThumbnailImage	The memory is an image file.	4.6
WCHAR!	Separator	The separator used in the text.	4.7
INT32!	TextWidth	A certain width of the presentation.	4.8
INT32	LetterLength	The number of characters in the text.	4.9
...		3.2 Font format, Table 3.2	

Table 3.1: File format

3.2 Font format

Typ	Name	Description	Info
INT32	FontCount	The number of fonts.	5.1
-> {	Font		
INT32	FontNameLength	The number of characters in the name of the font.	5.2
WCHAR[]	FontName	The name of the font.	5.3
BYTE	FontStyle	The styles of the font.	5.4
FLOAT	FontSize	The height of the font in pixels.	5.5
INT32	LetterCount	The number of characters used for this font.	5.6
UInt16[]	LetterArray	The memory for the characters used.	5.7
INT32	LetterSizeCount	The size of the memory for the character sizes.	5.8
Int16[]	LetterSizeArray	The memory of character sizes.	5.9
} <-	Font		
...		3.3 Color format, Table 3.3	

Table 3.2: Font format

3.3 Color format

Typ	Name	Description	Info
INT32	ColorCount	The number of color.	6.1
-> {	Color		
UINT32	ColorValue	The ARGB colors in the text.	6.2
} <-	Color		
...		3.4 Letter format, Table 3.4	

Table 3.3: Color format

3.4 Letter format

Typ	Name	Description	Info
INT32	LetterInfoSize	The size of the information memory.	7.2
BYTE[]	LetterInfoMemory	The information for each character in the text.	7.3
INT32!	LetterFontSize	The size of the memory for the indices of the fonts.	7.4
BYTE[]!	LetterFontMemory	The memory contains the indices of the fonts.	7.5
INT32!	LetterColorSize	The size of the memory for the indices of the colors.	7.6
BYTE[]!	LetterColorMemory	The memory contains the indices of the colors.	7.7
INT32	LetterMaxCount	The maximum size of a character index.	7.8
INT32!	LetterIndexSize	The size of the memory for the indices of the characters.	7.9
BYTE[]!	LetterIndexMemory	The memory contains the indices of the characters.	7.10

Table 3.4: Letter format

4 Information about the file format

4.1 Identification number

The identification number identifies the file format. The number can also be displayed with 4 letters (TDTW: Two Dimension Text Writer).

4.2 Version number

The version number is always 1 for this format description.

4.3 Text alignment

The value tells you how the text is aligned.

Name	Value	Description
Left	0	The text is aligned left justified.
Center	1	The text is centered.
Right	2	The text is aligned right justified.
Justified	3	The text is displayed as a justified sentence.

Table 4.3: Text alignment

4.4 Format bits

The bits determine the further content in the file format. If the (Compress) bit is not set, the format for the uncompressed text must be used. This format is called "TDFormatText".

Name	Value	Description
Separator	0x01	The file format contains a separator (see 4.7).
TextWidth	0x02	The file format contains a specific text width (see 4.8).
Compress	0x04	The file format is compressed (see format: TDCompressText).
Thumbnail	0x08	The file format contains a thumbnail image (see 4.5 and 4.6).

Table 4.4: Format bits

4.5 Thumbnail size

The size is specified in bytes. It can not be negative or 0 (see 4.4 Format bits).

4.6 Thumbnail image

The thumbnail shows a specific text section. The image can be saved in PNG, JPEG, TIFF or BMP formats. By default, the PNG format is used.

4.7 Separator

A separator appears as a hyphen (-) by default. The character is inserted in word syllables to better represent a text visually. If this value is not present, no word breaks are used (see 4.4 Format bits).

4.8 Display width

The value specifies a certain display width. If this value is not specified, the current window width should be used as the text width (see 4.4 Format bits).

4.9 Number of characters in the text

The value indicates the number of characters in the text. This includes the special characters for a new line. The number can not be negative. If the value is 0, the file format is terminated here.

5 Information about the font format

5.1 Number

The number of fonts contained. The value can not be less than 1 and greater than 256. The run of the loop (Font) is determined by this value.

5.2 Name length

The number of characters in the name of the font.

5.3 Name

The name consists of a certain number of characters (letters). The name length (5.2) determines the number of characters. A character is a 16 bit unsigned value. The memory size for the name results from the name length times 2 bytes.

Memory size: `FontNameLength * 2Bytes (16Bit)`

5.4 Font styles

The font styles are given in bits and can therefore be combined.

For example: ***Bold, Italic*** (0x03)

Name	Value	Description
Regular	0x00	The text is displayed without further styles.
Bold	0x01	The characters are drawn in bold.
Italic	0x02	The characters are drawn in italics.
Underline	0x04	The characters are underlined.
Strikout	0x08	The characters are crossed out.

Table 5.4: Font styles

5.5 Font size

The font size is specified in pixels. The value is defined as a floating point number. The decimal places are not used because the font height is specified in pixels.

5.6 Number of characters

The number of characters used with the current font. The value can not be less than 1 or greater than 65536.

5.7 Character memory

The memory contains all the characters for the current font as 16bit values. The special characters "Carriage Return" and "New Line" can also be contained in the memory. The number of characters is given in section 5.6.

Memory size: `LetterCount * 2Bytes (16Bit)`

5.8 The size of the memory for the character sizes

The value indicates the size of the memory in bytes. For each character (see sections 5.6 and 5.7), three 16bit values are used.

Condition: `LetterCount * 6 == LetterSizeCount`

5.9 The memory of character sizes

The size memory contains three 16bit values for each character (see section 4.9). These widths are labeled a, b and c and are indicated in pixels. The values a and c can be negative or 0.

Type	Name	Description
INT16	a	The overhang at the beginning of the character.
INT16	b	The width of the displayed character.
INT16	c	The overhang at the end of the character.

Table 5.9: Size memory

Example:

Figure 5.9: Letter widths

The letter "b" starts with a distance (a) to the previous character "A". The width (b) of the letter passes over the following character "c". At the end of the letter "b" the negative distance (c) for the overhang is deducted. The character "c" is displayed at the current position.

6 Information about the color format

6.1 Number

The number of colors in the text. The value can not be less than 1 or greater than 256. The run of the loop (Color) is determined by this value.

6.2 Color

The color (ARGB) is given as a 32bit value. The color component "Alpha" is not used.

7 Information about the character format

7.1 Decompression of a bit oriented memory

A memory can contain only bytes as the smallest unit. One byte (8 bits) can represent a number from 0 to 255. The required values can also be smaller than the maximum number. If, for example, only values from 0 to 7 are used, the memory is divided into 3bit blocks. The program 7.1 converts a bit oriented memory into a byte oriented memory.

Bits	Values	Number	Description
0	0	1	The memory is not used.
1	0..1	2	The memory contains at least 2 different values.
2	0..3	4	The memory contains at least 3 to 4 different values.
3	0..7	8	The memory contains at least 5 to 8 different values.
4	0..15	16	The memory contains at least 9 to 16 different values.
5	0..31	32	The memory contains at least 17 to 32 different values.
6	0..63	64	The memory contains at least 33 to 64 different values.
7	0..127	128	The memory contains at least 65 to 128 different values.
8	0..255	256	The memory does not have to be decompressed.

Table 7.1: Bit oriented memory

```
Int32 LetterLength; //See 4.9
Int32 ValueMax;     //Number

Byte[] ValueArray;  //Input memory
Byte[] ResultArray = new Byte[LetterLength]; //Output memory

Int32 BitCountMax = (Int32) Math.Ceiling(Math.Log(ValueMax, 2)); //Bits
Int32 BitCount = 0;
Int32 Index = 0;

for(Int32 i = 0; i < LetterLength; i++) {
  Byte Value = ValueArray[Index];

  Value <<= BitCount;
  BitCount += BitCountMax;

  if(BitCount >= 8) {
    if(BitCount == 8) {
      BitCount = 0;

      Value >>= 8 - BitCountMax;

      Index++;
    } else {
      BitCount -= 8;

      Byte ValueNext = ValueArray[++Index];
      ValueNext >>= 8 - BitCount;

      Value >>= 8 - BitCountMax;
      Value |= ValueNext;
    }
  } else {
    Value >>= 8 - BitCountMax;
  }

  ResultArray [i] = Value;
}
```

Program 7.1: Decompression

7.2 Size of the memory for the character information

The value indicates the size of the information memory in bytes. The value can not be smaller than 1 or greater than the number of characters (see section 4.9) in the text.

7.3 Information memory

The information memory contains a 3bit value for each character (see section 4.9). The additional information can only have the maximum value 0x07 (3 bits) if the value "Control" and the bit "NewLine" are contained. For conversion to byte memory see program 7.1.

```
Int32 ValueMax = 8; //Number
Byte[] ValueArray = LetterInfoMemory; //Input memory
```

The memory contains additional information about the character used. A word that can be split contains the value 0x01 (Seperator) at the separation point. In addition, a separator is displayed after the letter (see section 4.7). For a space, the value 0x02 (Space) is specified. A line break consists of two special characters. At the end the character "Carriage Return" with the value 0x0D is inserted. The character at the beginning of the line has the value 0x0A and is called "New Line". Both special characters have the information (Control) with the value 0x03. The beginning, the new line is additionally marked with the bit 0x04 (NewLine), the result is the information with the value 0x07. If a specific text width is used (see section 4.4), the bit (NewLine) may also have been set at the first letter of a word.

Name	Value	Description
Letter	0x00	A character without additional information.
Seperator	0x01	The word can be divided with a separator.
Space	0x02	The character is a space.
Control	0x03	The special character is specified for a line break.
NewLine	0x04	The bit is set when a new line starts.

Table 7.3: Character information

7.4 Size of the memory for the indices of the fonts

The value specifies the size of the memory for the indexes of the fonts in bytes. The value can not be smaller than 1 or greater than the number of characters (see section 4.9) in the text. If only one font is used (see section 5.1), the value is not included in the character format.

7.5 Memory for the indices of the fonts

The memory contains a zero based index for a font for each character (see section 4.9). If only one font is used, the memory is not included in the character format. The number of fonts (see section 5.1) determines the bit orientation of the memory using table 7.1. For conversion to byte memory see program 7.1.

```
Int32 ValueMax = FontCount; //See 5.1
Byte[] ValueArray = LetterFontMemory; //Input memory
```

7.6 Size of the memory for the indices of the colors

The value specifies the size of the memory for the indices of the colors in bytes. The value can not be smaller than 1 or greater than the number of characters (see section 4.9) in the text. If only one color is used (see section 6.1), the value is not included in the character format.

7.7 Memory for the indices of the colors

The memory contains a zero based index for each character (see Section 4.9) for a color. If only one color is used (see section 6.1), the memory is not included in character format. For conversion to byte memory see program 7.1.

```
Int32 ValueMax = ColorCount; //See 6.1
Byte[] ValueArray = LetterColorMemory; //Input memory
```

7.8 Maximum index

The value specifies the maximum index of all characters within all fonts. The value can not be less than 1 or greater than 65536. The maximum index can also be determined in the font format (see section 3.2).

7.9 Size of the memory for the indices of the characters

The value indicates the amount of memory for the indices of the characters in bytes. The value can not be smaller than 1 or greater than the number of characters (see section 4.9) in the text. If only one character is used in all fonts (see section 7.8), the value is not included in the character format.

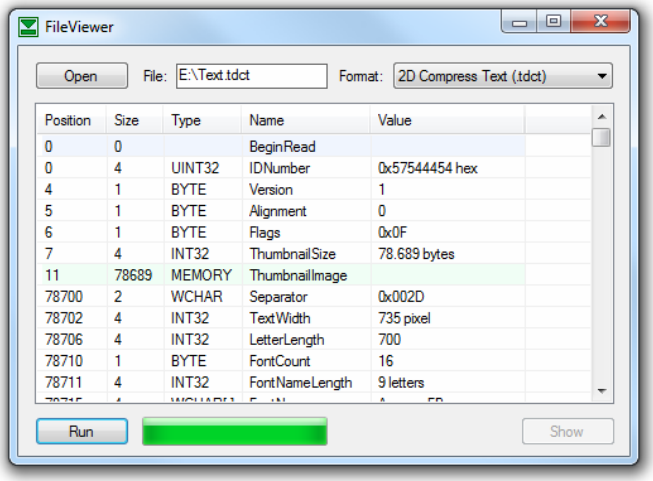
7.10 Memory for the indices of the characters

The memory contains a zero based index for each character (see section 4.9). Based on the font index (see section 7.5), the character can now be determined. If only one character per font is used (see section 7.8), the memory is not in character format. For conversion to byte memory see program 7.1. If more than 256 characters have been used for a font, the 7.1 program can not be used. In this case, a new program code with a 16bit result memory must be created.

```
Int32 ValueMax = LetterMaxCount; //See 7.8
Byte[] ValueArray = LetterIndexMemory; //Input memory
```

8 Program for reading the file format

On the PanotiSoft website, there is a test program under technical documents, with which the file format can be read out in a structured manner. In addition, the program code can also be downloaded. The program was written under Visual Studio 2008 with the programming language C#.



Program: FileViewerX64.zip oder FileViewerX32.zip

Project file: FileViewerCode.zip

Description: FileViewer.pdf

FileViewerCode:

Format file: FileViewerFormat.cs
Format class: FileViewerTDCompressText