1 Einleitung

Mit dieser Datei können verschiedene Schriften und farbige Textinhalte gespeichert werden. Zusätzlich können die Textbreite, Ausrichtung und Trennzeichen im Format enthalten sein. Für die Komprimierung werden alle Speicher bitorientiert zusammengefasst. Die Textdatei wird mit dem Programm "TextWriter" erstellt. Die Dateierweiterung wird ".tdct" (Two Dimension Compress Text) genannt.

2 Wertetypen

Typ	Beschreibung	Bereich
INT8	8Bit mit Vorzeichen	-128 bis 127
INT16	16Bit mit Vorzeichen	-32.768 bis 32.767
INT32	32Bit mit Vorzeichen	-2.147.483.648 bis 2.147.483.647
INT64	64Bit mit Vorzeichen	-9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807
BYTE	8Bit ohne Vorzeichen	0 bis 255
UINT16	16Bit ohne Vorzeichen	0 bis 65.535
UINT32	32Bit ohne Vorzeichen	0 bis 4.294.967.295
UINT64	64Bit ohne Vorzeichen	0 bis 18.446.744.073.709.551.615
CHAR	8Bit Zeichen	0 bis 255
WCHAR	16Bit Zeichen	0 bis 65.535
FLOAT	32Bit Gleitkommazahl	± 1.5e-45 zu ± 3.4e38
DOUBLE	64Bit Gleitkommazahl	± 5.0e-324 zu ± 1.7e308
MEMORY	Speicher in Bytes	
...[]	Wertemenge	siehe Abschnitt 2.1
-> {	Beginn der Schleife	siehe Abschnitt 2.2
} <-	Ende der Schleife	siehe Abschnitt 2.2
...	Nächste Tabelle	siehe Abschnitt 2.3
!	Abhängigkeit	Siehe Abschnitt 2.4

Tabelle 2: Wertetypen

2.1 Wertemenge

Die Menge besteht aus einem bestimmten Wertetyp. Die Anzahl der Menge wird in den Informationen detailliert angegeben und ist meistens der vorherige Formatwert.

Beispiel:

```
Eine Menge INT16[] enthält eine bestimmte Anzahl von INT16 Werten { INT16, INT16,
INT16, INT16, ... }.

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

2.2 Die Schleife

Bei einer Schleife wird das Format wiederholt durchlaufen. Die Anzahl der Durchläufe wird in den Informationen detailliert angegeben und ist meistens der vorherige Wert.

2.3 Nächste Tabelle

In dem angegebenen Abschnitt und der Tabelle wird das Dateiformat weiter fortgesetzt.

2.4 Abhängigkeit

Der Wert ist nur in der Datei vorhanden, wenn eine bestimmte Bedingung erfüllt ist.

3 Beschreibungen

3.1 Dateiformat

Typ	Name	Beschreibung	Info
UINT32	IDNumber	Die Datei muss die ID Nummer (0x57544454) haben.	4.1
BYTE	Version	Die Dateiversion muss für diese Beschreibung 1 sein.	4.2
BYTE	Alignment	Die Ausrichtung des Textes.	4.3
BYTE	Flags	Die Bits zur Formatbeschreibung.	4.4
INT32!	ThumbnailSize	Die Größe des Miniaturbildes in Bytes.	4.5
MEMORY!	ThumbnailImage	Der Speicher ist eine Bilddatei.	4.6
WCHAR!	Separator	Das verwendete Trennzeichen im Text.	4.7
INT32!	TextWidth	Eine bestimmte Breite der Darstellung.	4.8
INT32	LetterLength	Die Anzahl der Zeichen im Text.	4.9
...		3.2 Schriftformat, Tabelle 3.2	

Tabelle 3.1: Dateiformat

3.2 Schriftformat

Typ	Name	Beschreibung	Info
INT32	FontCount	Die Anzahl der Schriften.	5.1
-> {	Font		
INT32	FontNameLength	Die Anzahl der Zeichen im Namen des Schrifttyps.	5.2
WCHAR[]	FontName	Der Name der Schrift.	5.3
BYTE	FontStyle	Die Stile der Schrift.	5.4
FLOAT	FontSize	Die Höhe der Schrift in Bildpunkte.	5.5
INT32	LetterCount	Die Anzahl der verwendeten Zeichen für diese Schrift.	5.6
UInt16[]	LetterArray	Der Speicher für die verwendeten Zeichen.	5.7
INT32	LetterSizeCount	Die Größe des Speichers für die Zeichengrößen.	5.8
Int16[]	LetterSizeArray	Der Speicher der Zeichengrößen.	5.9
} <-	Font		
...		3.3 Farbformat, Tabelle 3.3	

Tabelle 3.2: Schriftformat

3.3 Farbformat

Typ	Name	Beschreibung	Info
INT32	ColorCount	Die Anzahl der Farben.	6.1
-> {	Color		
UINT32	ColorValue	Die ARGB Farben im Text.	6.2
} <-	Color		
...		3.4 Zeichenformat, Tabelle 3.4	

Tabelle 3.3: Farbformat

3.4 Zeichenformat

Typ	Name	Beschreibung	Info
INT32	LetterInfoSize	Die Größe des Informationsspeichers.	7.2
BYTE[]	LetterInfoMemory	Die Informationen für jedes Zeichen im Text.	7.3
INT32!	LetterFontSize	Die Größe des Speichers für die Indizes der Schriften.	7.4
BYTE[]!	LetterFontMemory	Der Speicher enthält die Indizes der Schriften.	7.5
INT32!	LetterColorSize	Die Größe des Speichers für die Indizes der Farben.	7.6
BYTE[]!	LetterColorMemory	Der Speicher enthält die Indizes der Farben.	7.7
INT32	LetterMaxCount	Die maximale Größe eines Zeichenindex.	7.8
INT32!	LetterIndexSize	Die Größe des Speichers für die Indizes der Zeichen.	7.9
BYTE[]!	LetterIndexMemory	Der Speicher enthält die Indizes der Zeichen.	7.10

Tabelle 3.4: Zeichenformat

4 Informationen zum Dateiformat

4.1 Identifikationsnummer

Die Identifikationsnummer kennzeichnet das Dateiformat. Die Nummer kann auch mit 4 Buchstaben (TDTW: Two Dimension Text Writer) dargestellt werden.

4.2 Versionsnummer

Die Versionsnummer ist für diese Formatbeschreibung immer 1.

4.3 Textausrichtung

Der Wert gibt Auskunft darüber, wie der Text ausgerichtet ist.

Name	Werte	Beschreibung
Left	0	Der Text wird linksbündig ausgerichtet.
Center	1	Der Text wird mittig ausgerichtet.
Right	2	Der Text wird rechtsbündig ausgerichtet.
Justified	3	Der Text wird als Blocksatz dargestellt.

Tabelle 4.3: Textausrichtung

4.4 Formatbits

Die Bits bestimmen den weiteren Inhalt im Dateiformat. Ist das Bit (Compress) nicht gesetzt, muss das Format für den unkomprimierten Text verwendet werden. Dieses Format heißt "TDFormatText".

Name	Werte	Beschreibung
Separator	0x01	Das Dateiformat enthält ein Trennzeichen (siehe 4.7).
TextWidth	0x02	Das Dateiformat enthält eine bestimmte Textbreite (siehe 4.8).
Compress	0x04	Das Dateiformat ist komprimiert.
Thumbnail	0x08	Das Dateiformat enthält ein Miniaturbild (siehe 4.5 und 4.6).

Tabelle 4.4: Formatbits

4.5 Miniaturbildgröße

Die Größe wird in Bytes angegeben. Sie kann nicht negativ oder 0 sein (siehe 4.4 Formatbits).

4.6 Miniaturbild

Das Miniaturbild zeigt einen bestimmten Textausschnitt. Das Bild kann in den Formaten PNG, JPEG, TIFF oder BMP gespeichert sein. Als Standard wird das PNG Format verwendet.

4.7 Trennzeichen

Ein Trennzeichen wird standardmäßig als Bindestrich (-) angezeigt. Das Zeichen wird bei Wortsilben eingefügt, um einen Text optischer besser darzustellen. Ist dieser Wert nicht vorhanden, werden keine Wortumbrüche verwendet (siehe 4.4 Formatbits).

4.8 Darstellungsbreite

Der Wert gibt eine bestimmte Darstellungsbreite vor. Wird dieser Wert nicht angegeben, soll die aktuellen Fensterbreite als Textbreite verwendet werden (siehe 4.4 Formatbits).

4.9 Anzahl der Zeichen im Text

Der Wert gibt die Anzahl der Zeichen im Text an. Dazu gehören auch die Sonderzeichen für eine neue Zeile. Die Zahl kann nicht negativ sein. Ist der Wert 0 ist das Dateiformat hier beendet.

5 Informationen zum Schriftformat

5.1 Anzahl

Die Anzahl der enthaltenen Schriften. Der Wert kann nicht kleiner als 1 und größer als 256 sein. Der Durchlauf der Schleife (Font) wird mit diesem Wert bestimmt.

5.2 Namenlänge

Die Anzahl von Zeichen im Namen des Schrifttyps. Der Wert kann nicht kleiner als 1 sein.

5.3 Name

Der Name besteht aus einer bestimmten Anzahl von Zeichen (Buchstaben). Die Namenlänge (5.2) bestimmt die Anzahl der Zeichen. Ein Zeichen ist ein 16Bit Wert ohne Vorzeichen. Die Speichergröße für den Namen ergibt sich aus der Namenlänge mal 2Bytes.

Speichergröße: `FontNameLength * 2Bytes (16Bit)`

5.4 Schriftstile

Die Schriftstile sind in Bits angegeben und können somit auch kombinierte werden.

Zum Beispiel: ***Bold, Italic*** (0x03)

Name	Werte	Beschreibung
Regular	0x00	Der Text wird ohne weitere Stile dargestellt.
Bold	0x01	Die Zeichen werden Fett gezeichnet.
Italic	0x02	Die Zeichen werden kursive gezeichnet.
Underline	0x04	Die Zeichen werden unterstrichen dargestellt.
Strikout	0x08	Die Zeichen werden durchgestrichen dargestellt.

Tabelle 5.4: Schriftstile

5.5 Schriftgröße

Die Schriftgröße wird in Bildpunkte angegeben. Der Wert ist als Gleitkommazahl definiert. Die Nachkommastellen werden nicht verwendet, weil die Schrifthöhe in Bildpunkte angegeben ist.

5.6 Anzahl der Zeichen

Die Anzahl der Zeichen, die mit der aktuellen Schrift verwendet werden. Der Wert kann nicht kleiner als 1 oder größer als 65536 sein.

5.7 Zeichenspeicher

Der Speicher enthält alle Zeichen für die aktuelle Schrift als 16Bit Werte. Auch die Sonderzeichen "Carriage Return" und "New Line" können im Speicher enthalten sein. Die Anzahl der Zeichen ist im Abschnitt 5.6 angegeben.

Speichergröße: `LetterCount * 2Bytes (16Bit)`

5.8 Die Größe des Speichers für die Zeichengrößen

Der Wert gibt die Größe des Speichers in Bytes an. Für jedes Zeichen (siehe Abschnitt 5.6 und 5.7) werden drei 16Bit Werte verwendet.

Bedingung: `LetterCount * 6 == LetterSizeCount`

5.9 Speicher der Zeichengrößen

Der Größenspeicher enthält für jedes Zeichen (siehe Abschnitt 5.6) drei 16Bit Werte. Diese Breiten werden mit a, b und c gekennzeichnet und sind in Bildpunkte angegeben. Die Werte a und c können negativ oder auch 0 sein.

Typ	Name	Beschreibung
INT16	a	Der Überhang am Anfang des Zeichen.
INT16	b	Die Breite des dargestellten Zeichens.
INT16	c	Der Überhang am Ende des Zeichen.

Tabelle 5.9: Größenspeicher

Beispiel:

Bild 5.9: Buchstabenbreiten

Der Buchstabe "b" beginnt mit einem Abstand (a) zum vorherigen Zeichen "A". Die Breite (b) des Buchstaben geht über das folgenden Zeichen "c" hinweg. Am Ende des Buchstaben "b" wird noch der negativen Abstand (c) für den Überhang abgezogen. An der aktuellen Position wird dann das Zeichen "c" dargestellt.

6 Informationen zum Farbformat

6.1 Anzahl

Die Anzahl der enthaltenen Farben im Text. Der Wert kann nicht kleiner als 1 oder größer als 256 sein. Der Durchlauf der Schleife (Color) wird mit diesem Wert bestimmt.

6.2 Farbe

Die Farbe (ARGB) wird als 32Bit Wert angegeben. Die Farbkomponente "Alpha" wird nicht verwendet.

7 Informationen zum Zeichenformat

7.1 Dekomprimierung eines bitorientierten Speichers

Ein Speicher kann als kleinste Einheit nur Bytes enthalten. Ein Byte (8Bits) kann eine Zahl von 0 bis 255 darstellen. Die benötigten Werte können aber auch kleiner als die maximale Zahl sein. Werden zum Beispiel nur Werte von 0 bis 7 gebraucht, wird der Speicher in 3Bit Blöcke aufgeteilt. Das Programm 7.1 wandelt einen bitorientierten Speicher in einen byteorientierten Speicher um.

Bits	Werte	Anzahl	Beschreibung
0	0	1	Der Speicher wird nicht verwendet.
1	0..1	2	Der Speicher enthält mindesten 2 unterschiedliche Werte.
2	0..3	4	Der Speicher enthält mindesten 3 bis 4 unterschiedliche Werte.
3	0..7	8	Der Speicher enthält mindesten 5 bis 8 unterschiedliche Werte.
4	0..15	16	Der Speicher enthält mindesten 9 bis 16 unterschiedliche Werte.
5	0..31	32	Der Speicher enthält mindesten 17 bis 32 unterschiedliche Werte.
6	0..63	64	Der Speicher enthält mindesten 33 bis 64 unterschiedliche Werte.
7	0..127	128	Der Speicher enthält mindesten 65 bis 128 unterschiedliche Werte.
8	0..255	256	Der Speicher muss nicht dekomprimiert werden.

Tabelle 7.1: Bitorientierter Speicher

```
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;
}
```

Programm 7.1: Dekomprimierung

7.2 Größe des Speichers für die Zeicheninformationen

Der Wert gibt die Größe des Informationsspeichers in Bytes an. Der Wert kann nicht kleiner als 1 oder größer als die Anzahl der Zeichen (siehe Abschnitt 4.9) im Text sein.

7.3 Informationsspeicher

Der Informationsspeicher enthält für jedes Zeichen (siehe Abschnitt 4.9) einen 3Bit Wert. Die Zusatzinformation kann nur den maximal Wert 0x07 (3 Bits) haben, wenn der Wert "Control" und das Bit "NewLine" enthalten sind. Für die Umwandlung in einen Bytespeicher siehe Programm 7.1.

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

Der Speicher enthält Zusatzinformationen zum verwendeten Zeichen. Ein Wort, dass geteilt werden kann, enthält an der Trennstelle den Wert 0x01 (Seperator). Zusätzlich wird nach dem Buchstaben ein Trennzeichen angezeigt (siehe Abschnitt 4.7). Bei einem Leerzeichen wird der Wert 0x02 (Space) angegeben. Ein Zeilenumbruch besteht aus zwei Sonderzeichen. Am Ende wird das Zeichen "Carriage Return" mit dem Wert 0x0D eingefügt. Das Zeichen am Anfang der Zeile hat den Wert 0x0A und wird mit "New Line" bezeichnet. Beide Sonderzeichen haben die Information (Control) mit dem Wert 0x03. Der Anfang, der neuen Zeile wird zusätzlich mit dem Bit 0x04 (NewLine) gekennzeichnet, daraus ergibt sich die Information mit dem Wert 0x07. Wird eine bestimmte Textbreite verwendet (siehe Abschnitt 4.4), kann das Bit (NewLine) auch beim ersten Buchstaben eines Wortes gesetzt worden sein.

Name	Wert	Beschreibung
Letter	0x00	Ein Zeichen ohne zusätzlichen Informationen.
Seperator	0x01	Das Wort kann mit einem Trennzeichen geteilt werden.
Space	0x02	Das Zeichen ist ein Leerzeichen.
Control	0x03	Das Sonderzeichen wird bei einem Zeilenumbruch angegeben.
NewLine	0x04	Das Bit wird gesetzt, wenn eine neue Zeile beginnt.

Tabelle 7.3: Zeicheninformation

7.4 Größe des Speichers für die Indizes der Schriften

Der Wert gibt die Größe des Speichers für die Indizes der Schriften in Bytes an. Der Wert kann nicht kleiner als 1 oder größer als die Anzahl der Zeichen (siehe Abschnitt 4.9) im Text sein. Wird nur eine Schriftart (siehe Abschnitt 5.1) verwendet, ist der Wert nicht im Zeichenformat enthalten.

7.5 Speicher für die Indizes der Schriften

Der Speicher enthält für jedes Zeichen (siehe Abschnitt 4.9) einen null basierten Index für eine Schrift. Wird nur eine Schriftart verwendet, ist der Speicher nicht im Zeichenformat enthalten. Die Anzahl der Schriften (siehe Abschnitt 5.1) bestimmt anhand der Tabelle 7.1 die Bitorientierung des Speichers. Für die Umwandlung in einen Bytespeicher siehe Programm 7.1.

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

7.6 Größe des Speichers für die Indizes der Farben

Der Wert gibt die Größe des Speichers für die Indizes der Farben in Bytes an. Der Wert kann nicht kleiner als 1 oder größer als die Anzahl der Zeichen (siehe Abschnitt 4.9) im Text sein. Wird nur eine Farbe (siehe Abschnitt 6.1) verwendet, ist der Wert nicht im Zeichenformat enthalten.

7.7 Speicher für die Indizes der Farben

Der Speicher enthält für jedes Zeichen (siehe Abschnitt 4.9) einen null basierten Index für eine Farbe. Wird nur eine Farbe (siehe Abschnitt 6.1) verwendet, ist der Speicher nicht im Zeichenformat enthalten. Für die Umwandlung in einen Bytespeicher siehe Programm 7.1.

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

7.8 Maximaler Index

Der Wert gibt den maximalen Index aller Zeichen innerhalb aller Schriftarten an. Der Wert kann nicht kleiner als 1 oder größer als 65536 sein. Der maximale Index kann auch im Schriftformat (siehe Abschnitt 3.2) bestimmt werden.

7.9 Größe des Speichers für die Indizes der Zeichen

Der Wert gibt die Größe des Speichers für die Indizes der Zeichen in Bytes an. Der Wert kann nicht kleiner als 1 oder größer als die Anzahl der Zeichen (siehe Abschnitt 4.9) im Text sein. Wird nur ein Zeichen in allen Schriften verwendet (siehe Abschnitt 7.8), ist der Wert nicht im Zeichenformat enthalten.

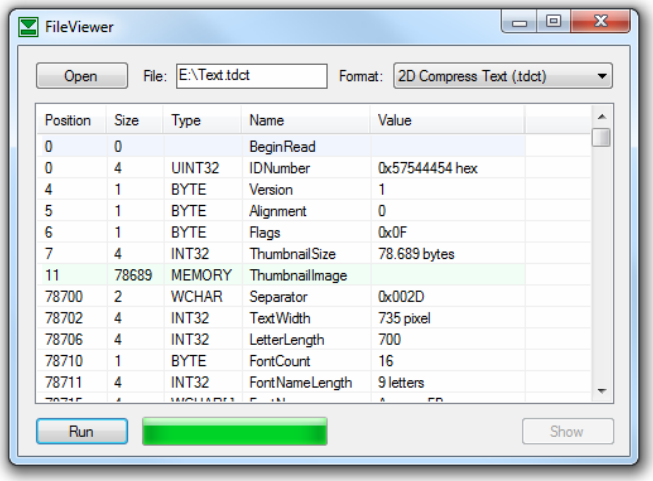
7.10 Speicher für die Indizes der Zeichen

Der Speicher enthält für jedes Zeichen (siehe Abschnitt 4.9) einen null basierten Index. Anhand des Schriftindex (siehe Abschnitt 7.5) kann nun das Zeichen bestimmt werden. Wird nur ein Zeichen pro Schrift verwendet (siehe Abschnitt 7.8), ist der Speicher nicht im Zeichenformat enthalten. Für die Umwandlung in einen Bytespeicher siehe Programm 7.1. Sind für eine Schriftart mehr als 256 Zeichen verwendet worden, kann das Programm 7.1 nicht verwendet werden. Für diesen Fall muss ein neuer Programmkode mit einem 16Bit Ergebnisspeicher erstellt werden.

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

8 Programm zum Auslesen des Dateiformats

Auf der Internetseite von PanotiSoft ist unter technische Dokumente ein Testprogramm vorhanden, mit dem das Dateiformat strukturiert ausgelesen werden kann. Zusätzlich kann auch der Programmkode herunter geladen werden. Das Programm wurde unter Visual Studio 2008 in der Programmiersprache C# geschrieben.



Programm: FileViewerX64.zip oder FileViewerX32.zip

Projektdatei: FileViewerCode.zip

Beschreibung: FileViewer.pdf

FileViewerCode:

Formatdatei: FileViewerFormat.cs
Formatklasse: FileViewerTDCompressText