1 Allgemeiner Aufbau

1.1 File

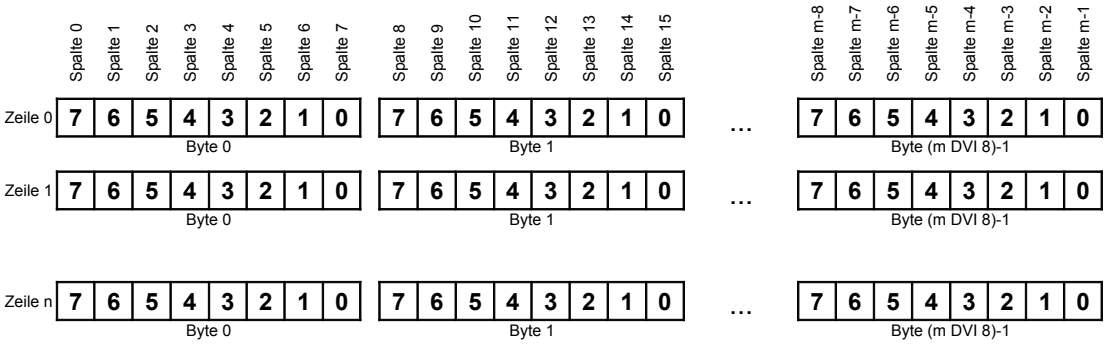
Das Fontfile beginnt mit einem Identifikationsstring gefolgt mit dem jeweiligen Fontparametern. Fontparameter und Identifikationsstring belegen die ersten 32 Byte der File. Es folgen eine variable Anzahl von Bytes, die die Zeichen Bitmap und sonstige Informationen zu den Zeichen enthalten.

```
Byte        Inhalt

00h..1Fh    Fileheader mit Identifikationsstring und Fontparameter
20..        Zeichencodes
```

1.2 Zeichen

Ein Zeichen besteht aus dem eigentlichem Bitmap und zeichenbezogenen Parametern. Das Bitmap ist für alle Typen der Fontfiles identisch. Dabei wird ein gesetztes Bit durch eine 1 repräsentiert. Jede Zeile des Zeichenbitmaps besteht aus eine durch 8 teilbaren Anzahl von spalten, wobei nicht benutze Spalten mit 0 beschrieben werden.



Die Anzahl der von einen Zeichenbitmap benötigten Byte errechnet sich nach folgenden Gleichung:

$$Byte_{Bitmap} = Höhe_{Zeichensatz} * (Breite_{Zeichen\ maximal} + 7)\ DIV\ 8$$

Die Anzahl pro Zeichen gespeicherten Bytes errechnet sich wie folgt:

$$Byte_{Zeichen} = Byte_{Bitmap} + Verwaltungsdaten_{Zeichen}$$

1.3 Kerninginformatinen

Zur Realisierung von Unterschneidungen können Kerning Informationen gespeichert werden. Dazu werden pro Zeichen Bereiche definiert, in die andere Zeichen hineinragen dürfen. Die Daten werden in einen 8 Byte Großen Struktur gespeichert.

```
struct kerning(
      struct NWKerning( byte x; byte y );
      struct NOKerning( byte x; byte y );
      struct SWKerning( byte x; byte y );
      struct SOKerning( byte x; byte y );
)
```

2 Fileformate

2.3 Type 1

2.3.1 Header

Byte	Inhalt
00h	Länge des folgenden Identifikationsstrings (27d = 1Bh)
01h..1Bh	Identifikationsstring „SS-FONTFILE TYPE 001“
1Ch..1Dh	Zeichenbreite (intel)
1Eh..1Fh	Zeichenhöhe (intel)

2.3.2 Zeichencodes

Die Zeichencodes werden ab 20h gespeichert und enthalten für jedes Zeichen nur die Bitmapdaten.

Offset des 1.Byte	letztes Byte		des Zeichens
0	$Byte_{Bitmap}-1$		00h
$Byte_{Bitmap}-1$	$2*Byte_{Bitmap}-1$		01h
$n*Byte_{Bitmap}-1$	$(n+1)*Byte_{Bitmap}-1$		n
$255*Byte_{Bitmap}-1$	$256*Byte_{Bitmap}-1$		FFh

Der vom einem Zeichen belegte Speicher entspricht $Byte_{Bitmap}$.

2.4 Type 2

2.4.1 Header

Byte	Inhalt
00	Länge des folgenden Identifikationsstrings (25d = 19h)
01h..19h	Identifikationsstring „<scharsoft>-FONT 002“+20h+20h+20h+20h+00h
1Ah..1Bh	BaseLine des Zeichensatzes (intel)
1Ch..1Dh	Zeichenbreite (intel)
1Eh..1Fh	Zeichenhöhe (intel)

2.4.2 Zeichencodes

Die Zeichencodes werden ab 20h gespeichert und enthalten für jedes Zeichen die Bitmapdaten gefolgt von der realen Zeichenbreite.

Offset des 1.Byte	letztes Byte		des Zeichen
0	$Byte_{Bitmap}-1$	Bitmap	00h
$Byte_{Bitmap}$	$Byte_{Bitmap}$	Breite	00h
$Byte_{Bitmap}+1$	$2*(Byte_{Bitmap}+1)-1$	Bitmap	01h
$2*(Byte_{Bitmap}+1)$	$2*(Byte_{Bitmap}+1)$	Breite	01h
$n*(Byte_{Bitmap}+1)+1$	$(n+1)*(Byte_{Bitmap}+1)-1$	Bitmap	n
$(n+1)*(Byte_{Bitmap}+1)$	$(n+1)*(Byte_{Bitmap}+1)$	Breite	n
$255*(Byte_{Bitmap}+1)+1$	$256*(Byte_{Bitmap}+1)-1$	Bitmap	FFh
$256*(Byte_{Bitmap}+1)$	$256*(Byte_{Bitmap}+1)$	Breite	FFh

Der vom einem Zeichen belegte Speicher entspricht Byte $_{Bitmap}+1$.

2.5 Type 3

2.5.1 Header

Byte	Inhalt
00	Länge des folgenden Identifikationsstrings (25d = 19h)
01h..19h	Identifikationsstring „<scharsoft>-FONT 003“+20h+20h+20h+20h+00h
1Ah..1Bh	BaseLine des Zeichensatzes (intel)
1Ch..1Dh	Zeichenbreite (intel)
1Eh..1Fh	Zeichenhöhe (intel)

2.5.2 Zeichencodes

Die Zeichencodes werden ab 20h gespeichert und enthalten für jedes Zeichen die Zeichenbreite, die Kerningbereiche und im Anschluss die Bitmapdaten.

Offset des 1.Byte	letztes Byte		des Zeichen
0	0	Breite	00h
1	8	Kerning	00h
9	$(Byte_{Bitmap}+8)$	Bitmap	00h
$Byte_{Bitmap}+9$	$Byte_{Bitmap}+9$	Breite	01h
$Byte_{Bitmap}+10$	$Byte_{Bitmap}+17$	Kerning	01h
$Byte_{Bitmap}+18$	$2*(Byte_{Bitmap}+9)-1$	Bitmap	01h
$n*(Byte_{Bitmap}+9)$	$n*(Byte_{Bitmap}+9)$	Breite	n
$n*(Byte_{Bitmap}+9)+1$	$n*(Byte_{Bitmap}+9)+8$	Kerning	n
$n*(Byte_{Bitmap}+9)+9$	$(n+1)*(Byte_{Bitmap}+9)-1$	Bitmap	n
$255*(Byte_{Bitmap}+9)$	$255*(Byte_{Bitmap}+9)$	Breite	FFh
$255*(Byte_{Bitmap}+9)+1$	$255*(Byte_{Bitmap}+9)+8$	Kerning	FFh
$255*(Byte_{Bitmap}+9)+9$	$256*(Byte_{Bitmap}+9)-1$	Bitmap	FFh

Der vom einem Zeichen belegte Speicher entspricht Byte $_{Bitmap}$+9 (Zeichenbreite + Kerning).