Dieses Kapitel stammt aus dem Buch

›Dateiformate – Eine Referenz‹

von Günter Born.

www.borncity.de

ISBN 3-934358-83-7
119,90 DM

Informationen zum Buch
mit Bestellmöglichkeit

www.galileocomputing.de

Galileo Computing

© Copyright 2001 by Galileo Press
Verlag und Autor schließen jede Haftung beim Gebrauch dieser Informationen aus.

26 Interchange File Format (IFF)

Dieser Standard wurde von der Firma Elektronic Arts zum Speichern von Grafiken und Tönen in Dateien eingeführt. Ursprünglich wurden IFF-Dateien auf Amiga-Rechnern benutzt; seitdem jedoch die Programme *DeLuxe Paint* und *DeLuxe Paint II* verfügbar sind, setzt sich das Format auch für PCs durch. IFF-Dateien besitzen in der Regel die Erweiterung .LBM und sind ähnlich wie GIF- oder TIFF-Dateien blockorientiert aufgebaut (Abbildung 26.1).

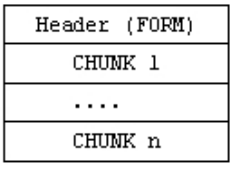

Abbildung 26.1 Aufbau einer IFF-Datei

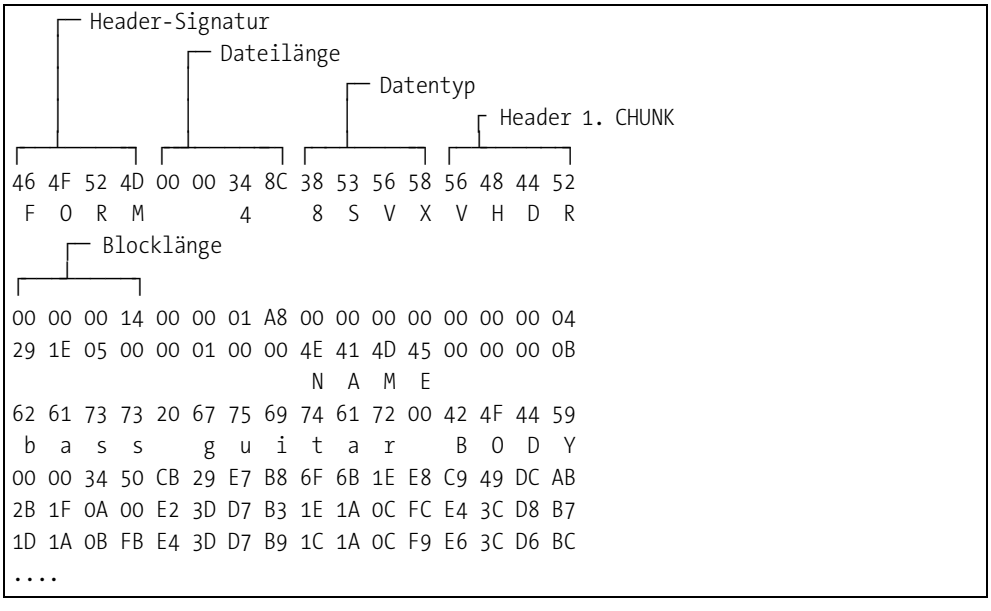

Abbildung 26.2 Auszug aus einer IFF-Datei als Hexdump

Der erste Block der IFF-Datei enthält den Header (FORM), in dem Informationen über die Dateilänge und die Art der gespeicherten Daten abgelegt sind. Daran schließen sich ein oder mehrere Blöcke variabler Länge an, die nachfolgend als *CHUNKs* bezeichnet werden. In diesen CHUNKs befinden sich die Spezifikationen für die Datendekodierung sowie die eigentlichen Daten. Einen Auszug aus einer IFF-Datei sehen Sie in Abbildung 26.2. Die Daten liegen im AMIGA-Format vor, d.h., High- und Low-Byte sind vertauscht. Der genaue Aufbau des Records ist nachfolgend beschrieben.

Der IFF-Header

Am Anfang der Datei steht ein 12 Byte langer Header, der neben der Dateilänge zwei Signaturen enthält. Der Aufbau dieses Headers ist Tabelle 26.1 zu entnehmen:

Offset	Bytes	Bemerkungen
00H	4	Signatur1 im IFF-Header = 'FORM'
04H	4	Dateilänge in Byte (Motorola-Format)
08H	4	Datentyp als ASCII-String (z. B.: ILBM)

Tabelle 26.1 Aufbau des IFF-Headers

Die ersten 4 Bytes des Headers (und damit auch der IFF-Datei) enthalten immer die Signatur 'FORM' als ASCII-String, anhand derer sich gültige IFF-Dateien jederzeit identifizieren lassen. Enthält eine IFF-Datei nur ein Bild, wird die Struktur gemäß Abbildung 26.3 benutzt.

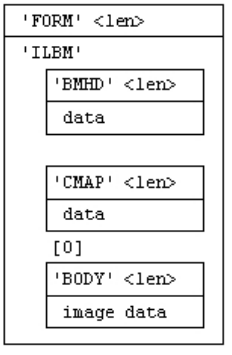

Abbildung 26.3 Struktur einer IFF-FORM-Datei

In einigen Fällen werden mehrere Objekte innerhalb einer IFF-Datei gespeichert (z. B. bei Bildkatalogen). In diesem Fall wird die CAT-Struktur (ConcATenate) aus Abbildung 26.4 verwendet.

Sobald die CAT-Signatur auftritt, enthält die IFF-Datei mehrere Objekte, die in weitere FORM-Strukturen aufgegliedert werden.

Weiterhin existieren noch IFF-Dateien mit der Signatur 'LIST' im Datei-Header. Diese Signatur ist ein Hinweis, daß die IFF-Datei mehrere Bilder enthält. Die LIST-Struktur kann einen 'PROP'-Block enthalten, mit dem sich Einstellungswerte (z. B. Palette) gemeinsam für mehrere Objekte der Datei verwenden lassen. Die Struktur einer IFF-LIST-Datei ist in Abbildung 26.5 zu sehen.

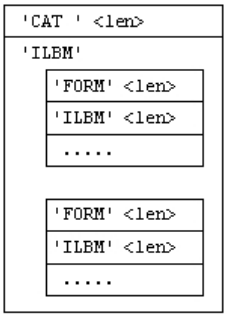

Abbildung 26.4 Struktur einer IFF-CAT-Datei

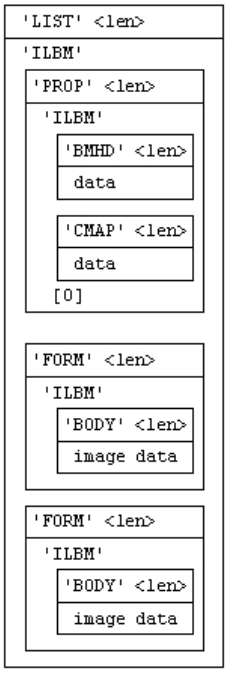

Abbildung 26.5 Struktur einer LIST-IFF-Datei

Die Signaturen 'LIST' und 'CAT' werden selten verwendet. Ein IFF-Reader kann daher diese Strukturen überspringen und die 'FORM'-Strukturen einlesen.

An die Signatur FORM, CAT oder LIST schließt sich ab Offset 04H ein 32-Bit-Wert an, der die Dateilänge in Byte spezifiziert. Die Zählung beginnt ab 1, wobei die Bytes im *Big endian*-Format (Amiga- bzw. Motorola-Konventionen) gespeichert sind. Dieser 32-Bit-Wert ist erforderlich, um das Ende des letzten CHUNKs zu ermitteln. Beachten Sie aber, daß sich in einer Datei mehrere FORMs befinden können, d.h., der Zeiger kann auf eine weitere 'FORM'-Signatur zeigen.

Der IFF-Header wird ab Offset 08H mit einer weiteren 4 Byte langen Signatur abgeschlossen. Dabei handelt es sich wieder um einen ASCII-String, der spezifiziert, um welche Daten (Grafik, Text, Musik) es sich in den nachfolgenden Blocks handelt. Tabelle 26.2 enthält eine Aufstellung gültiger Signaturen. Beachten Sie aber, daß es mittlerweile eine Reihe herstellerspezifischer Erweiterungen (ca. 50) gibt, die aber oft nur unzureichend dokumentiert sind. Nachfolgend werden deshalb nur die in Tabelle 26.2 enthaltenen Varianten behandelt.

Signatur	Datentyp
'WORD'	Textdatei (word data)
'ILBM'	Grafikdatei (interleaved bitmap)
'PBM'	Grafik analog 'ILBM'
'FTXT'	Textdatei (IFF-Format)
'AIFF'	Audio-IFF (Mac Soundmanager)
'SMUS'	Musikdaten (sample music)
'8SVX'	8-Bit-Sample-Daten (8 bit sample voice)

Tabelle 26.2 Signatur des IFF-Headers für verschiedene Datentypen

Mit dieser Information lassen sich in einer IFF-Datei Texte, Grafiken und Töne abspeichern. Findet sich z. B. im Header die Signatur 'WORD', enthält einer der nachfolgenden CHUNKs die Textdaten. Überwiegend wird hier jedoch die Signatur 'ILBM' für Grafiken auftauchen. Dies ist zum Beispiel bei Deluxe Paint der Fall, da dieses Paket nur Grafiken abspeichert. Musikdaten, Sound und Töne werden mit den Signaturen 'SMUS' und '8SVX' versehen.

Normalerweise enthält eine IFF-Datei nur einen FORM-Header mit den zugehörigen CHUNKs. Es ist aber durchaus möglich, Grafik, Text und Sound in einer Datei zu kombinieren, womit die Datei dann mehrere FORMs mit CHUNKs enthält.

IFF-Blockstruktur (CHUNK)

An den Header schließen sich mehrere Blöcke (*CHUNKs*) mit Informationen über Grafikmodus, Farbtabellen und Bilddaten an. Diese CHUNKs besitzen einen schematischen Aufbau gemäß Abbildung 26.6.

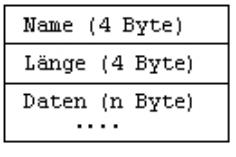

Abbildung 26.6 Aufbau eines IFF-CHUNKs

Der erste Eintrag enthält eine 4 Byte lange Signatur in Form eines ASCII-Strings, der den Typ des jeweiligen CHUNKs (Blocks) angibt. Tabelle 26.3 enthält einige gültige Blocknamen für Grafik und Musikdaten.

Name	Bemerkungen
Grafik (ILBM)	
BMHD	Bitmap-Header
CMAP	Color Map
CRNG	Color Cycle-CHUNK von DPaint
CCRT	Color Cycle-CHUNK von Graphicraft
BODY	Bitmap-Daten der Grafik
Musik (8SVX)	
VHDR	Voice-Daten (Tempo, Oktave etc.)
NAME	Name des Klangs
(c)	Copyright-Vermerk
AUTH	Name des Autors
ANNO	Datum
BODY	Sounddaten
ATAK	Hüllkurveninfo Einschwingdaten
RLSE	Hüllkurveninfo Ausschwingdaten

Tabelle 26.3 IFF-CHUNK-Signaturen

Je nachdem, welches Programm die Datei erzeugt, können weitere Blöcke mit anderen Namen auftreten. Das Konzept erlaubt es, zukünftig neue Blocktypen zu definieren. Ein Block (CHUNK) beginnt immer an einer geraden Adresse innerhalb der Datei. Falls der vorhergehende Block eine ungerade Anzahl von Bytes enthält, wird an diesen Block ein Nullbyte angehängt, das allerdings im Längenfeld des Blocks nicht berücksichtigt wird. Der erste Eintrag des nachfolgenden Blocks enthält dann den 4-Byte-ASCII-String mit der Signatur. Der nächste Eintrag innerhalb des Blocks enthält einen 4-Byte-Zeiger mit der aktuellen Blocklänge in Byte. Daran schließt sich der Datenbereich an, dessen Länge sich aus *Blocklänge – 8* errechnen läßt. Falls ein unbekannter Block auftritt, kann dieser vom Leseprogramm ignoriert werden. *Deluxe Paint* beispielsweise legt weitere Blöcke mit den Bereichen für die Farbverläufe an.

Die Farbbereiche im Palettenmenü sind einstellbar; die Information wird für die Ausgabe der Bilddaten jedoch nicht benötigt und kann demnach überlesen werden. Nachfolgend wird die Struktur der einzelnen CHUNKs beschrieben.

Die CHUNKs des ILBM-FORM

Zum Speichern von Grafikdaten wird ein ILBM-FORM mit verschiedenen CHUNKs benutzt. Der Aufbau der gebräuchlichsten CHUNKs wird nachfolgend beschrieben.

Bitmap-Header-CHUNK (BMHD)

Dieser CHUNK enthält Informationen über Grafikmodus, Bildgröße etc. und muß vor allen anderen CHUNKs stehen. Tabelle 26.4 gibt den Aufbau dieses Blocks wieder.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'BMHD' als ASCII-String
04H	4	Blocklänge in Byte (00 00 00 14)
08H	2	Bildbreite in Pixel
0AH	2	Bildhöhe in Pixel
0CH	2	X-Koordinate obere linke Bildecke (bei Deluxe Paint immer 0)
0EH	2	Y-Koordinate obere linke Bildecke (bei Deluxe Paint immer 0)
10H	1	Zahl der Farbebenen (Bitplanes) (bei EGA immer 4)
11H	1	reserviert (Maskierung)
12H	1	Bildkodierung 0 = ungepackte Speicherung 1 = gepackte Speicherung
13H	3	reserviert
16H	1	Breite eines Pixels (x-aspect ratio)
17H	1	Höhe eines Pixels (y-aspect ratio)
18H	2	maximale Bildbreite (X-Max)
1AH	2	maximale Bildhöhe (Y-Max)

Tabelle 26.4 Aufbau des Bitmap-Header-CHUNKs

Der CHUNK beginnt mit dem ASCII-String 'BMHD' als Signatur. Daran schließt sich ein 4-Byte-Wert mit der Blocklänge in Byte an; dieser Eintrag sollte den Wert 14H besitzen. Wichtig dabei ist, daß die Zahl im Motorola-Format gespeichert wird, damit der Wert 00000014H dann im Format 00 00 00 14 in der Datei abgelegt wird. Bei Intel CPUs werden die Daten in der Schreibweise 14 00 00 00 erwartet, d.h., es muß eine Umsetzung erfolgen (siehe TIFF-Format).

Die nächsten beiden 2-Byte-Vektoren beschreiben die aktuellen Bildabmessungen in der Maßeinheit Pixel. Die Koordinaten der linken oberen Bildecke werden ab Offset 0CH als 2-Byte-Werte (X, Y) geführt. Die maximal möglichen Abmessungen finden sich ab Offset

18H. Das Programm *DeLuxe Paint* trägt hier immer den Wert (0,0) ein. Beim Speichern von Bildausschnitten lassen sich dort unter Umständen andere Koordinaten ablegen.

IFF-Dateien mit ILBM-CHUNKs dienen hauptsächlich zum Übertragen farbiger Bilder. Ab Offset 10H steht in einem Byte die Zahl der Farbebenen (*Bitplanes*), bei Schwarzweißdarstellungen wird hier der Wert 1 eingetragen. Standardmäßig sind für EGA-Karten 16 unterschiedliche Farben vorgesehen, so daß der Wert auf 4 Bitplanes gesetzt wird: Der Wert FFH entspricht so 256 Farbebenen. Diese Informationen sind für die Dekodierung der Color Map wichtig.

Ab Offset 12H findet sich ein Byteflag mit der Spezifikation der Bilddatenkomprimierung. Der Wert 0 zeigt an, daß sie unkomprimiert im BODY-Block abgelegt sind. Da sich bei 640 x 350 Bildpunkten und 4 Bit pro Farbe jedoch 112000 Byte pro Bild ergeben, existiert hier ein einfaches Komprimierungsverfahren zum Speichern der Daten. Werte größer 0 signalisieren, daß die Daten im BODY-CHUNK komprimiert abgelegt sind. Zur Zeit sind jedoch nur die Codes 0 (*unverschlüsselt*) und 1 (*verschlüsselt*) definiert. Der Verschlüsselungsalgorithmus ist im Abschnitt über den BODY-Block beschrieben. Die Bytes ab Offset 16H geben die Größe eines Originalpunktes im Bild an. Dies ist für die Umrechnung in neue Bildabmessungen wichtig.

Der Color-Map-CHUNK (CMAP)

Bei Standardanwendungen werden Bilder aus 16 Farben zusammengesetzt. Diese Farben lassen sich bei EGA/VGA-Karten jedoch aus 16 Millionen Farbkombinationen zusammenstellen. Die Informationen über die jeweils aktuelle Farbpalette sind deshalb in einer eigenen Tabelle im CMAP-CHUNK-Block gespeichert, der sich an den BMHD-CHUNK anschließt und folgenden Aufbau besitzt:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'CMAP' als ASCII-String
04H	4	Blocklänge in Byte (3 x n)
08H	3 x n	Farbtabelle mit *n* Einträgen á 3 Byte für die Farben Rot, Grün und Blau

Tabelle 26.5 Aufbau des Color Map-CHUNKs

Die ersten 4 Bytes enthalten wieder den Namen CMAP zur Markierung des CHUNK-Typs, darauf folgt die Blocklänge in Byte. Ab Offset 08H schließt sich eine Tabelle mit den Farbdefinitionen an: Jeder Farbe sind drei Bytes für den Rot-, Grün- und Blauanteil zwischen 0 und 255 zugeordnet. Mit 24 Bit lassen sich demnach 16 Millionen Farbstufen darstellen, bei 16 selektierten Farben besitzt die Tabelle *16 x 3 = 48 Byte*. Ein Bildpunkt läßt sich in diesem Fall durch 4 Bit abbilden. Bei zukünftigen Versionen der IFF-Dateien kann die Tabelle auch 256 Einträge zu je 3 Byte aufweisen, die Blocklänge läßt die Berechnung der Zahl der Einträge jederzeit zu. Bei der Auswertung der Farben müssen eventuell in Abhängigkeit vom Bildschirmmodus die Farben angepaßt bzw. umgesetzt werden.

Der CRNG-CHUNK (DeLuxe Paint)

Der CHUNK wird von Electronic Arts für das Programm DeLuxe Paint benutzt, um aufeinanderfolgende Farbregister für Schattierung und Farbwechsel anzugeben. Der CHUNK ist optional und muß vor einem BODY-CHUNK gesetzt werden. Ist das unterste Bit im Flag (Offset 0CH) auf 1 gesetzt, wird ein Farbwechsel aktiviert. Die Farbe wechselt dann zum nächsten (Register-)Wert. Ist das zweite Bit im Flag auf 1 gesetzt, erfolgt der Farbwechsel in entgegengesetzter Richtung. Die Werte ab Offset 14 (0EH) und 15 (0FH) geben dabei den Bereich der Farbregister an. Die Geschwindigkeit der Farbwechsel findet sich im Wert *Color cycle rate* (Offset 0AH). Bei 60 Farbwechseln pro Sekunde wird der Wert 16384 hinterlegt. Dieser CHUNK besitzt jedoch nur auf dem Amiga-Rechner eine Bedeutung.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'CRNG' (Color Cycle)
04H	4	CHUNK-Länge in Byte
08H	2	reserviert (00 00)
0AH	2	Color Cyle Rate
0CH	2	Flag: aktiviert/nicht aktiviert
0EH	1	unterer Farbwert (Farbregister)
0FH	1	oberer Farbwert (Farbregister)

Tabelle 26.6 Aufbau des CRNG-CHUNKs

Der CCRT-CHUNK (Graphicraft)

Ähnlich wie bei DeLuxe Paint legen die Produkte der Firma Graphicraft einen *Color Cycle*-CHUNK an. Dieser erhält jedoch die Signatur CCRT und ist wie in Tabelle 26.7 beschrieben aufgebaut:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'CCRT' (Color Cycle)
04H	4	CHUNK-Länge in Byte
08H	2	Richtung 0 = kein Cycling 1 = vorwärts -1 = rückwärts
0AH	1	Startfarbe (Farbregister low)
0BH	1	Endfarbe (Farbregister high)
0CH	4	Sekunden Wartezeit für Vertauschungen
10H	4	Mikrosekunden Wartezeit
14H	2	unbenutzt (00H 00H)

Tabelle 26.7 Aufbau des CCRT-CHUNKS

Die genaue Bedeutung der Felder konnte nicht in Erfahrung gebracht werden.

Der BODY-CHUNK mit den Daten

Die eigentlichen Bilddaten stehen im BODY-CHUNK. An diese Signatur schließt sich gemäß Tabelle 26.8 der 4-Byte-Zeiger mit der Blocklänge an:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'BODY' als ASCII-String
04H	4	Blocklänge in Byte
08H	n	Bilddaten

Tabelle 26.8 Aufbau des BODY-CHUNK (Datenblock)

Die eigentlichen Bilddaten folgen ab Offset 08H. Ein Eintrag 0 im BMHD ab Offset 12H signalisiert, daß die Daten unkodiert vorliegen, während Werte größer 0 (ab Offset 12H) kodierte Daten anzeigen. Enthält das Flag im BMHD-Block den Wert 1, liegen die Daten in kodierter FORM vor. Die Bilddaten einer Bitplane werden dann zeilenweise nach folgender Recordstruktur gespeichert:

```
1. Byte = Steuerbyte
2. Byte = Datenbytes
...
n. Byte =    "
```

Das erste Byte wird jeweils als Steuerbyte interpretiert und bestimmt die Auswertung:

- Enthält es Werte kleiner 128 (80H), folgen anschließend *n* unterschiedliche Datenbytes. Der Wert *n* entspricht dabei dem Wert des Steuerbytes + 1.
- Bei Werten größer 128 (80H) enthält das Folgebyte die zu entschlüsselnde komprimierte Bildfolge. Dieses Byte ist dann n-fach zu duplizieren. Die Zahl *n* errechnet sich aus dem Wert des Steuerbytes gemäß folgender Anweisung:

  ```
  n = 100H – Wert des Steuerbytes + 1
  ```

- Ein Eintrag FDH im Steuerbyte ergibt damit einen Wiederholungsfaktor von *100H – FDH + 1 = 4*. So lassen sich einheitliche Bildbereiche (z. B. gefüllte Flächen) recht effizient speichern.
- Ein Wert von 80H im Steuerbyte ist zu überlesen und bewirkt, daß in diesem Fall das Folgebyte als Steuerbyte interpretiert wird.

Die Bildpunkte im Datenbereich werden zeilenweise abgespeichert.

Der GRAB-CHUNK

Dieser CHUNK ist optional, wird äußerst selten verwendet und beschreibt einen »hot spot« (z. B. Mauszeiger). Der CHUNK besitzt dabei folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'GRAB' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	X-Koordinate (relativ)
0AH	2	Y-Koordinate (relativ)

Tabelle 26.9 Aufbau des GRAB-CHUNK

Der DEST-CHUNK

Dieser CHUNK ist ebenfalls optional und beschreibt, wie ein Grafikausschnitt in eine Ziel-Bitmap-Datei eingefügt werden kann. Der CHUNK besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'DEST' als ASCII-String
04H	4	Blocklänge in Byte
08H	1	Anzahl der Bitplanes der Quelle
09H	1	reserviert
0AH	2	Bits für Bitplane des Ziels
0CH	2	Bitplane-Flag 2
0EH	2	Maske Bitplane (Ziel)

Tabelle 26.10 Aufbau des DEST-CHUNK

Ab Offset 0AH befindet sich eine Maske, deren einzelne Bits signalisieren, ob die einzelnen Bitplanes des Zielbildes bearbeitet werden müssen. Ist das betreffende Bit = 1 gesetzt, darf in der Bitplane des Zielbildes nichts verändert werden. Ist das Bit = 0 gesetzt, muß das Bit im Folgewort ausgewertet werden. Falls dort das Bit = 1 gesetzt ist, ist die komplette Bitplane im Zielbild mit 1-Bits zu füllen. Das letzte Wort enthält eine Maske, wobei ein Bit = 0 bedeutet, daß die Bits der betreffenden Bitplanes nicht verändert werden sollen.

Der SPRT-CHUNK

Der SPRT-CHUNK definiert (optional) die Priorität eines Sprite. Der CHUNK besitzt dabei folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'SPRT' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	Priorität des Sprite (0..7)

Tabelle 26.11 Aufbau des SPRT-CHUNK

Der Wert 0 signalisiert die höchste Priorität, d.h., das Sprite überlagert alle anderen Sprites.

Der CAMG-CHUNK

Der CAMG-CHUNK ist spezifisch für den Amiga und gibt den Grafikmodus an:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'CAMG' als ASCII-String
04H	4	Blocklänge in Byte
08H	4	Daten des Viewmode-Registers

Tabelle 26.12 Aufbau des CAMG-CHUNK

Der CLUT-CHUNK

Der CLUT-CHUNK definiert eine Farbtabelle (Color Lock up Table):

Offset	Bytes	Bemerkungen
00H	4	Blockname 'CLUT' als ASCII-String
04H	4	Blocklänge in Byte
08H	4	Typ der Tabelle
0CH	4	reserviert
10H	256	Farbtabelle (8 Bit)

Tabelle 26.13 Der CLUT-CHUNK

Ab Offset 08H steht der Typ der Tabelle (Mono, RGB etc.). Die folgenden Bytes sind reserviert. Ab Offset 10H folgt eine Tabelle mit 256 Einträgen für die 8-Bit-Farbdefinitionen.

Für das ILBM-FORM sind weitere CHUNK-Namen reserviert, die aber teilweise sehr spezifisch belegt und nur unzureichend dokumentiert sind. So benutzt DPaint II den DPPV-CHUNK, um Informationen über die Perspektive der Sicht für einzelne rotierende Objekte abzulegen. Weiterhin existiert in der PC-Version von Deluxe Paint II Enhanced der TINY-CHUNK, in dem Teilbilder abgelegt werden. (Nach verschiedenen Informationen soll die Längenangabe in diesem CHUNK in einigen Fällen um 1 zu niedrig liegen).

Das Programm Deluxe Paint II erwartet die CHUNKs in der festen Reihenfolge: BMHD (Offset 0CH), CMAP (Offset 28H), BODY. Falls diese Reihenfolge nicht stimmt, kann das Bild nicht gelesen werden.

Als dritte Hürde wurde für die Speicherung von VGA-Bildern mit 256 Farben ein neues FORM ('PBM') definiert. Die CHUNKS entsprechen dem Aufbau der BMHD-CHUNKs. Allerdings werden die gespeicherten Bits anders interpretiert. Beim ILBM-FORM werden die einzelnen Bits eines Punktes in Farbebenen aufgeteilt. Bei 256 Farben ergibt dies 8 (Bit) Ebenen. Dann wird das Bild Bit für Bit gespeichert. Im ersten Durchlauf wird Bit 1 jedes Bytes der ersten Zeile zusammengefaßt und gespeichert. Im folgenden Durchlauf wird dann Bit 2 eines jeden Bytes gelesen und zeilenweise gespeichert. Sobald alle 8 Bits einer Zeile so gespeichert wurden, beginnt die Bearbeitung der zweiten Bildzeile. Dies ist ein recht umständliches Verfahren. Das PBM-FORM signalisiert nun, daß die Daten (8 Bit) eines Bildpunktes als Byte direkt gespeichert werden. Damit läßt sich jeder Punkt direkt aus dem Grafikspeicher bearbeiten, was einfacher und schneller ist. Die Daten befinden sich im BODY-Block und werden nach dem oben beschriebenen Verfahren komprimiert.

Die CHUNKs des 8SVX-FORM

Ähnlich wie bei Grafiken besteht auch die Möglichkeit, in IFF-Dateien digitalisierte Töne (*sampled data*) im 8-Bit-Format zu speichern. Für diesen Zweck ist der 8SVX-CHUNK vorgesehen. Der Aufbau des Headers (8SVX-FORM) ist mit dem ILBM-FORM identisch. Lediglich ab Offset 08H findet sich der 4-Byte-String 8SVX als Signatur. Die zugehörigen CHUNKs werden nachfolgend beschrieben.

Der Voice-Header-CHUNK (VHDR)

Dieser Block enthält die Informationen über die Speicherung der digitalisierten Töne im BODY-CHUNK. Der VHDR-CHUNK hat einen Aufbau gemäß Tabelle 26.14:

Offset	Bytes	Bemerkungen
00H	4	Signatur 'VHDR' (Voice Header)
04H	4	CHUNK-Länge in Byte
08H	4	OneShotHiSamples (Anzahl der digitalisierten Bytes für einen Anschlag)
0CH	4	RepeatHiSamples (Anzahl der Bytes für einen Klang)
10H	4	Samples per HiCycle (gerade Anzahl Bytes für eine Schwingung)
14H	2	Samples per Second (Abtastrate)

Offset	Bytes	Bemerkungen
16H	1	Anzahl Oktaven im BODY-CHUNK
17H	1	Kodierung 0 = ungepackt 1 = gepackt
18H	4	Volume (Lautstärke)

Tabelle 26.14 Der Aufbau des Voice-Header-CHUNKs

Die Werte müssen zur Interpretation des BODY-CHUNKs ausgewertet werden. Ab Offset 17H findet sich das Flag, das die Datenspeicherung im BODY-CHUNK spezifiziert. Falls hier der Wert 1 steht, liegen die Daten gepackt (Fibonacci-Delta) vor. Für die Einstellung der Lautstärke ab Offset 18H gilt als Standard der Wert 0001 0000H.

Der NAME-CHUNK (Name)

In diesem CHUNK läßt sich ein Text ablegen, der Hinweise auf das klangerzeugende Instrument gibt. Es gilt die Recordstruktur gemäß Tabelle 26.15.

Offset	Bytes	Bemerkungen
00H	4	Signatur 'NAME' (Name)
04H	4	CHUNK-Länge in Byte
08H	4	String mit dem Namen

Tabelle 26.15 Der Aufbau des NAME-CHUNKs

Im Textfeld dürfen beliebige Strings abgelegt werden, da diese bei der Auswertung als Kommentare gelten. Bei einer Stringlänge mit ungerader Bytezahl muß an den CHUNK ein Nullbyte (00) angehängt werden, das im Längenfeld des CHUNK jedoch nicht mitgezählt wird. Durch das Nullbyte beginnt ein eventuell folgender CHUNK auf einer geraden Offset-Adresse.

Der BODY-CHUNK (Daten)

In diesem CHUNK finden sich die digitalisierten Daten. Es gilt dabei der gleiche Aufbau wie beim ILBM-BODY-CHUNK (Tabelle 26.8). Für die gespeicherten Voice-Daten lassen sich zwei Verwendungsarten unterscheiden:

One Shot Sound

Hier wurde ein einmaliger Klang (*Effekt*) digitalisiert und abgespeichert. Im Voice-Header-CHUNK finden sich die Informationen über die Wandlungsrate *Samples per Second*, *OneShotHiSamples* und *RepeatHiSamples* für die Anzahl der Daten. Ist der Wert *SamplesperHiCycle* unbekannt, wird eine Null eingesetzt. Die Anzahl der Oktaven wird immer auf 1 gesetzt.

Musical Instrument

Die zweite Verwendungsart dient zum Abtasten und Abspeichern des Klangs eines Instrumentes. Hier wird folgende Strategie zur Digitalisierung verfolgt: Zuerst zeichnet man den Klang des Instruments mit einem *OneShotHiSample* auf. Im zweiten Teil wird die Klangdauer in *RepeatHiSamples* gespeichert. Um den Klang naturgetreu wiederzugeben, sind die Klänge über mehrere Oktaven gespeichert. Die Angaben im Voice Header beziehen sich immer auf die höchste gespeicherte Oktave. Ist in diesem Header die Zahl der Oktaven mit einem Wert größer 1 angegeben, liegen die Sounddaten von Oktave zu Oktave im BODY-CHUNK hintereinander. Es ist zu beachten, daß für jede Oktave *n* Byte für *OneShotSample* und *n* Byte für *RepeatHiSample* gespeichert werden, wodurch sich der Speicherbedarf von Oktave zu Oktave verdoppelt. Im Feld *Samples per HiCycle* im Voice Header wird die Anzahl der Bytes pro Schwingung für einen Klang der höchsten Oktave angegeben.

In der Regel sind die Daten unkomprimiert abgelegt, um Verzerrungen zu vermeiden. Beim Wert 1 im Voice Header ab Offset 17H sind die Daten nach dem Fibonacci-Delta-Algorithmus gepackt. Wie bereits erwähnt, liegen die Daten nach Oktaven geordnet hintereinander im Datenblock. Bei jeder Abtastung wird die Amplitude vorzeichenbehaftet mit 8 Bit verschlüsselt gespeichert. Dabei wird bei negativen Werten nur das oberste Bit invertiert. Die kleinste Amplitude läßt sich durch 1000 0000B darstellen, während die größte Amplitude den Wert 0111 1111B besitzt.

Der ATAK-CHUNK

Dieser CHUNK ist optional und beschreibt eine ansteigende Hüllkurve (Atak). Der CHUNK besitzt dabei 6 Datenbytes mit folgender Struktur:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'ATAK' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	Zeit bis Lautstärke erreicht
0AH	4	Lautstärkenfaktor

Tabelle 26.16 Der Aufbau des ATAK-CHUNKs

Die genaue Bedeutung dieser Daten ist aber nicht bekannt.

Der RLSE-CHUNK

Dieser CHUNK definiert eine absteigende Hüllkurve (Release), wobei der Datenbereich der Belegung des ATAK-CHUNKs entspricht. Auch dieser CHUNK ist optional.

Die CHUNKs des AIFF-FORM

Dies ist ein herstellerspezifisches FORM für den Sound-Manager des Apple Macintosh. Er dient zur Aufzeichnung von Musik und Sprache und bietet mehr Möglichkeiten als das 8SVX-Form. Da die Datenstruktur dokumentiert ist, möchte ich dieses Format hier kurz vorstellen.

Der COMM-CHUNK

Der COMM-CHUNK beschreibt die Aufzeichnung (Samples) und besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'COMM' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	Anzahl der Kanäle
0AH	4	Anzahl FRAMES (Abschnitte in SNDD)
0EH	2	Bitanzahl der Samples
10H	10	Sample Rate (Frames/Sek.)

Tabelle 26.17 Der COMM-CHUNK

Der Wert für Sample Rate wird als 10-Byte-Fließkommazahl im Macintosh Pascal-Format eingetragen.

Der SNDD-CHUNK

Der SNDD-CHUNK beschreibt die Sounddaten der einzelnen Abschnitte:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'SNDD' als ASCII-String
04H	4	Blocklänge in Byte
08H	4	Offset Beginn erstes FRAME
0CH	4	Blockgröße nn erstes FRAME
12H	x	Daten

Tabelle 26.18 Der SNDD-CHUNK

Die Kodierung der Daten ist mir aber nicht bekannt.

Der INST-CHUNK

Der INST-CHUNK spezifiziert für die MIDI-Schnittstelle das abzuspielende Instrument:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'INST' als ASCII-String
04H	4	Blocklänge in Byte
08H	1	Notenbasis
09H	1	Veränderung des Tons (Pitch)
0AH	1	niedrigste Note Notenbereich
0BH	1	höchste Note Notenbereich
0CH	1	kleinster Velocity-Wert (Anschlag)
0DH	1	größter Velocity-Wert
0EH	2	Ton Variation (in db)
10H	6	Sustain Loop
16H	6	Release Loop

Tabelle 26.19 Der INST-CHUNK

Das FORM besitzt noch weitere CHUNKs (MIDI, AESD, APPL, COMT), die hier aber nicht weiter besprochen werden, da die genaue Belegung teilweise unbekannt ist.

Die CHUNKs des SMUS-FORM

Dieses FORM (SMUS = Simple Musical Score) speichert ebenfalls Daten für Musik und Ton, die mit MIDI-Werkzeugen erstellt wurden. Das FORM besitzt dabei folgende CHUNKs:

Der SHDR-CHUNK

Der CHUNK beschreibt den SCORE-Header und besitzt folgendes Format:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'SHDR' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	Tempo (1/4 Noten pro 128 Minuten)
0AH	1	Lautstärke (0..127)
0BH	1	Anzahl Stimmen des folgenden Tracks

Tabelle 26.20 Der SHDR-CHUNK

Die genaue Kodierung der Daten ist mir aber nicht bekannt.

Der INS1-CHUNK

Der CHUNK enthält die Informationen zum verwendeten Instrument für die folgenden Tracks und besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'INS1' als ASCII-String
04H	4	Blocklänge in Byte
08H	1	Registernummer Instrument (0-255)
09H	1	Flag-Auswahlinstrument: 0 = Auswahl über Name 1 = Auswahl über 2 Folgebytes
0AH	1	MIDI-Kanal (falls Flag = 0)
0BH	1	MIDI-Preset (falls Flag = 0)
0CH	x	Name des Instruments

Tabelle 26.21 Der INS1-CHUNK

Die genaue Kodierung der Daten ist mir aber nicht bekannt.

Der TRAK-CHUNK

Der TRAK-CHUNK enthält die Musikdaten einer MIDI-Spur und besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'TRAK' als ASCII-String
04H	4	Blocklänge in Byte
08H	n	Interpretation Folgebyte (Byte) und Datenbyte (Note)

Tabelle 26.22 Der TRAK-CHUNK

Pro Note werden 2 Bytes abgespeichert: Das erste Byte definiert, wie die im folgenden Byte gespeicherte Note zu interpretieren ist. Die genaue Kodierung der Daten ist mir aber nicht bekannt.

Die CHUNKs des FTXT-FORM

Für formatierte Texte wurde das FORM FTXT definiert. Dieses FORM wird aber bisher nur durch das Programm TextCraft genutzt, das nicht allzu verbreitet ist. Es existieren zwei CHUNKs für dieses FORM:

Der FONS-CHUNK

Der FONS-CHUNK beschreibt den im Text verwendeten Font:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'FONS' als ASCII-String
04H	4	Blocklänge in Byte
08H	1	Fontnummer
09H	1	reserviert
0AH	1	Flag Proportional Font
0BH	1	Flag Serifen 0 = unbekannt 1 = Serifenschrift 2 = serifenlose Schrift
0CH	x	Fontname

Tabelle 26.23 Der FONS-CHUNK

Die genaue Kodierung der Daten ist mir aber nicht bekannt.

Der CHRS-CHUNK

Der CHUNK enthält den Text und besitzt folgende Struktur:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'CHRS' als ASCII-String
04H	4	Blocklänge in Byte
08H	x	Text

Tabelle 26.24 Der CHRS-CHUNK

Die genaue Kodierung der Textformatierung ist mir aber nicht bekannt.

Die CHUNKs des WORD-FORM

Für formatierte Texte wurde das FORM WORD definiert. Dieses FORM wird aber bisher nur durch das Programm ProWrite von Horizon Software genutzt. Im Gegensatz zum FTXT-Format lassen sich hier auch Formatierungsanweisungen speichern. Es existieren zwei CHUNKs für dieses FORM:

Der FONT-CHUNK

Der FONT-CHUNK ist optional und beschreibt den im Text verwendeten Font:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'FONT' als ASCII-String
04H	4	Blocklänge in Byte
08H	1	Fontnummer (0-255)

Offset	Bytes	Bemerkungen
09H	2	Fontgröße
0BH	x	Fontname

Tabelle 26.25 Der FONT-CHUNK

Für jeden verwendeten Font sollte ein solcher CHUNK vor den Daten stehen. Die genaue Kodierung der Daten ist mir aber nicht bekannt.

Der COLR-CHUNK

Der CHUNK beschreibt die Übersetzungstabelle für ISO-Farben. Es sind insgesamt 8 Einträge in der Tabelle vorhanden; deren Kodierung ist mir aber nicht bekannt.

Der DOC-CHUNK

Der CHUNK leitet den Anfang eines DOCUMENT-Abschnitts ein. Ab diesem CHUNK folgen die Anweisungen für die Formatierung des Textes.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'DOC ' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	erste Seitennummer
0AH	1	Typ der Seitennumerierung
0BH	5	reserviert

Tabelle 26.26 Der DOC-CHUNK

Die Seitennumerierung läßt sich als Zahl (1, 2 etc.) oder über Buchstaben (i, I, A, a) durchführen und wird im Feld *Typ der Seitennumerierung* angegeben.

Der FOOT/HEAD-CHUNK

Diese beiden CHUNKs besitzen einen identischen Aufbau und beschreiben Kopf- und Fußzeile eines Dokuments.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'FOOT' oder 'HEAD'
04H	4	Blocklänge in Byte
08H	1	Position Kopf- oder Fußzeile
09H	1	erste Seite (0 = nicht Seite 1)
0AH	4	reserviert

Tabelle 26.27 Der FOOT/HEAD-CHUNK

Ist der Wert ab Offset 09H mit 0 belegt, wird die Ausgabe des Kopf- und Fußtextes auf der ersten Seite des Dokuments unterdrückt. Der Text für Kopf- und Fußzeile gilt bis zum nächsten CHUNK vom Typ DOC, HEAD, FOOT oder PCTS.

Der PARA-CHUNK

Dieser CHUNK beschreibt die Formatierung des folgenden Absatzes.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'PARA' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	einrücken (in 1/10 Punkt)
0AH	2	linker Rand
0CH	2	rechter Rand
0EH	1	Spacing (Abstand zwischen Absätzen)
0FH	1	Zeilenausrichtung (justify)
10H	1	Fontnummer
11H	1	Flag Fontstyle
12H	1	normal, sub- oder superscript
13H	1	interne Farbnummer
14H	4	reserviert

Tabelle 26.28 Der PARA-CHUNK

Die genaue Kodierung der einzelnen Einträge ist mir aber nicht bekannt.

Der TABS-CHUNK

Dieser CHUNK beschreibt die Position der Tabulatoren.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'TABS' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	TAB-Position in 1/10 Punkt
0AH	1	Ausrichtung (links, Mitte, ...)
0BH	1	reserviert

Tabelle 26.29 Der TABS-CHUNK

Die genaue Kodierung der einzelnen Einträge ist mir aber nicht bekannt.

Der PAGE-CHUNK

Dieser CHUNK beschreibt den Seitenumbruch im Text.

Offset	Bytes	Bemerkungen
00H	4	Blockname 'PAGE' als ASCII-String
04H	4	Blocklänge in Byte

Tabelle 26.30 Der PAGE-CHUNK

Der CHUNK besitzt keinen Datenbereich, sondern definiert nur eine Marke.

Der TEXT-CHUNK

Dieser CHUNK enthält den Text eines Absatzes und besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'TEXT' als ASCII-String
04H	4	Blocklänge in Byte
08H	x	Text

Tabelle 26.31 Der TEXT-CHUNK

Der Text sollte keine RETURN- oder LF-Zeichen enthalten.

Der FSCC-CHUNK

Dieser CHUNK definiert die Parameter FONT, STYLE und COLOR für den vorhergehenden Text:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'FSCC' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	Zeichenposition im Absatz
0AH	1	Fontnummer
0BH	1	Flags für Fontstyle
0CH	1	normal, subscript, superscript
0DH	1	interne Farbnummer
0EH	1	reserviert

Tabelle 26.32 Der FSCC-CHUNK

Genauere Informationen sind mir allerdings nicht bekannt.

Der PCTS-CHUNK

Dieser CHUNK definiert, daß ein Bild in den Text eingebunden werden soll:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'PCTS' als ASCII-String
04H	4	Blocklänge in Byte
08H	1	Anzahl Farbebenen (Bitplanes)
09H	1	reserviert

Tabelle 26.33 Der PCTS-CHUNK

An diesen CHUNK schließt sich dann die Definition der Bilddaten (siehe unten) an.

Der PINF-CHUNK

Dieser CHUNK definiert weitere Infomationen zum Bild, das in den Text eingebunden werden soll:

Offset	Bytes	Bemerkungen
00H	4	Blockname 'PINF' als ASCII-String
04H	4	Blocklänge in Byte
08H	2	Breite in Pixel
0AH	2	Höhe in Pixel
0CH	2	Seitennummer für Bild
0EH	2	X-Position des Bildes
10H	2	Y-Position des Bildes
12H	1	Maske (siehe ILBM-BMHD)
13H	1	Kompression
14H	1	transparent
15H	1	reserviert

Tabelle 26.34 Der PINF-CHUNK

An diesen CHUNK schließt sich dann die Definition der Bilddaten an. Diese Bilddaten werden in einem BODY-CHUNK gespeichert, welcher der Kodierung des ILBM-FORMS entspricht.

Allgemeine CHUNKs

Zusätzlich wurden einige CHUNKs definiert, die in allen FORMS auftreten können. Sie definieren Kommentare und zusätzliche Informationen zu den Daten. Nachfolgende Tabelle zeigt die Namen und die Struktur dieser CHUNKs.

Offset	Bytes	Bemerkungen
00H	4	Blockname-CHUNK als ASCII-String '(c) ' Copyright-Vermerk 'ANNO' Anmerkung (Annotations) 'AUTH' Name des Autors 'CHRS' Text 'NAME' Name einer Grafik, Musik 'TEXT' unformatierter Text
08H	x	n Byte mit dem betreffenden Text

Tabelle 26.35 Allgemeine CHUNKs