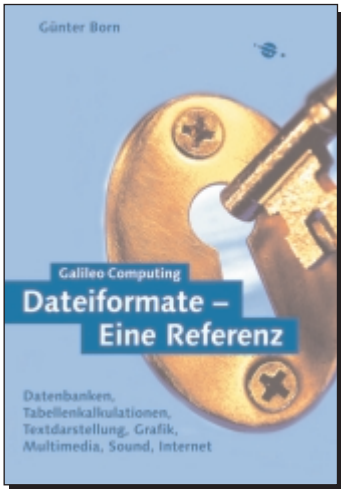
Dieses Kapitel stammt aus dem Buch

›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

16 Das Audio/Video Interleaved-Format (AVI)

Mit Video for Windows führte Microsoft ein neues Format (AVI) zur Darstellung von Bildern und Tönen ein. Dieses Format basiert auf der Microsoft RIFF-Spezifikation.

Die Resource Interchange File Format (RIFF)-Spezifikation

Das Resource Interchange File Format wurde von Microsoft für Windows Multimediaanwendungen definiert. Die RIFF-Dateien bilden dabei einen Container, in dem Formate anderer Spezifikationen abgelegt werden (Tabelle 16.1).

.AVI	Video (Video for Windows)
.WAV	Audio Wave (Windows Wave Files)
.RDI	Bitmap
.RMI	MIDI
.BND	Bündel anderer RIFF-Dateien

Tabelle 16.1 Beispiele für RIFF-Dateien

Eine Anwendung des RIFF-Formats stellt beispielsweise das von Microsoft benutzte WAV-Format dar. Nachfolgend finden Sie jedoch die Beschreibung des AVI-Formats.

Die Struktur der RIFF-Dateien lehnt sich dabei an die IFF-Definition an. RIFF-Dateien werden in einzelne CHUNKs unterteilt, die die Daten enthalten. Ein RIFF-File beginnt mit einem Header, der die Struktur gemäß Tabelle 16.2 aufweist.

Offset	Bytes	Bemerkungen
00H	4	Signatur 'RIFF'
04H	4	Dateigröße in Byte
08H	4	RIFF-Typ

Tabelle 16.2 Struktur eines RIFF-Headers

Die ersten 4 Bytes enthalten die Signatur für die RIFF-Datei. Wurden die Daten im Intel-Format (little endian) gespeichert, findet sich die Signatur 'RIFF'. Dies ist der Standard für RIFF-Dateien. Es wurde aber die Möglichkeit vorgesehen, die Daten im Motorola-Format (big endian) abzulegen. Dann enthalten die ersten 4 Bytes die Signatur 'RIFX'.

Ab Offset 04H schließt sich ein DWORD mit der Längenangabe für die Datei an. Die Längenangabe berücksichtigt nicht die Felder mit der Signatur und der Längenangabe.

Die letzten 4 Bytes nehmen die Signatur für den RIFF-Typ an. Ist dieser String kürzer als 4 Zeichen, sind die Folgebytes mit Leerzeichen (20H) aufzufüllen. Für AVI-Dateien steht hier zum Beispiel die Signatur 'AVI'.

Der Aufbau eines RIFF-CHUNK

Ein RIFF-CHUNK besitzt die in Tabelle 16.3 gezeigte Struktur.

Offset	Bytes	Bemerkungen
00H	4	CHUNK-Signatur
04H	4	CHUNK-Größe in Byte
08H	n	CHUNK-Daten

Tabelle 16.3 Struktur eines RIFF-CHUNK

Die ersten 4 Bytes enthalten die Signatur für den CHUNK-Typ. Ist dieser String kürzer als 4 Zeichen, sind die Folgebytes mit Leerzeichen (20H) aufzufüllen.

Das DWORD ab Offset 04H gibt die Zahl der folgenden Datenbytes an. Bei RIFF-Dateien beginnen CHUNKs immer auf Word-Grenzen. Bei ungerader Länge wird ein Füllbyte angehängt. Die Längenangabe berücksichtigt dieses Füllbyte nicht. Weiterhin werden die ersten 8 Bytes des CHUNK nicht in die Längenangabe einbezogen, d.h., der Wert definiert wirklich die Zahl der folgenden Datenbytes.

Dieser Bereich mit den Datenbytes kann sich in weitere Sub-CHUNKs unterteilen. Diese Sub-CHUNKs enthalten die gleiche Struktur wie ein CHUNK und dienen zur Aufnahme der verschiedenen Daten.

Die AVI-Struktur

AVI-Dateien werden von Video for Windows verwendet, um Audio- und Videodaten zu speichern. Die AVI-Datei beginnt mit einem RIFF-Header ('RIFF' xx xx xx xx 'AVI'), dem mehrere CHUNKs folgen. Diese CHUNKs unterteilen sich in zwei LIST-CHUNKs, die die Informationen über die Datenkodierung (1. LIST CHUNK, 'hdrl') und die eigentlichen Daten (2. LIST CHUNK, 'movi') enthalten (Abbildung 16.1).

```
RIFF ('AVI '
 LIST ('hdrl'
       ....
     )
 LIST ('movi'
       ...
     )
 ['idx1' <optional AVI index>]
  ....
   )
```

Abbildung 16.1 Struktur eines AVI-Files

Auf die beiden LIST-CHUNKs folgen weitere Sub-CHUNKs mit den eigentlichen Daten (Abbildung 16.2).

```
RIFF ('AVI '
 LIST ('hdrl'
       'avih' <main AVI header>
    LIST ('strl' <stream line>
         'strh' <stream header>
         'strf' <stream format>
         'strd' <stream data>
          ....
        )
      )
 LIST ('movi'
       subchunk oder
    LIST ('rec' <data>
         subchunk 1
          ....
         subchunk n
        )
      )
 ['idx1' <optional AVI index>]
  ....
   )
```

Abbildung 16.2 Sub-CHUNK-Struktur in AVI-Dateien

Ein AVI-Reader muß deshalb nach der ersten LIST-Signatur die Sub-CHUNKs mit der Definition des Datenformats lesen. Sobald ein weiterer LIST-CHUNK auftritt, ist dessen Typ zu prüfen. Es kann sich um eine weitere Liste im Header oder um den Videodaten-Stream handeln. Der Beginn der Videodaten wird mit der Signatur 'movi' markiert. Dann folgen die Sub-CHUNKs mit den eigentlichen Videodaten. Tritt eine 'idx1'-Signatur in einem CHUNK auf, ist der Videodatenbereich beendet, und es folgen optionale AVI-Indexdefinitionen. Die Struktur der einzelnen CHUNKs wird nachfolgend vorgestellt.

Der AVI-Header-CHUNK (hdrl)

Der erste LIST-CHUNK enthält den Haupt-AVI-Header-CHUNK (main AVI header Sub CHUNK). Der CHUNK wird mit dem CHUNK-Typ 'hdrl' markiert.

Offset	Bytes	Bemerkungen
00H	4	CHUNK Signatur ('LIST')
04H	4	Länge Sub-CHUNK
08H	4	CHUNK-Typ ('hdrl')
0CH	n	Sub-CHUNKs

Tabelle 16.4 Struktur eines hdrl-CHUNK

Die Informationen in den Sub-CHUNKs des Headers definieren das Format des gesamten AVI-CHUNKs.

Der avih-Sub-CHUNK

Dieser Sub-CHUNK enthält die globalen Daten für den AVI-CHUNK. Der Datenbereich des avih-CHUNK besitzt folgende Struktur.

Offset	Bytes	Bemerkungen
00H	4	Signatur 'avih'
04H	4	CHUNK-Länge (38H)
08H	4	Time Delay zwischen Frames
0CH	4	AVI-Datenrate
10H	4	Reserviert
14H	4	Flags
18H	4	Zahl der Frames
1CH	4	Initial-Frames
20H	4	Zahl der Data-Streams
24H	4	Größe Playback Buffer
28H	4	Video Frame Width (Pixel)
2CH	4	Video Frame Height (Pixel)
30H	4	Unit Time Scale
34H	4	Playback-Datenrate
38H	4	Startzeit
3CH	4	Größe AVI Daten-CHUNK

Tabelle 16.5 Struktur eines avih-Sub-CHUNK

Das erste Feld enthält die Signatur ('avih'). Daran schließt sich ab Offset 04H der Datenbereich an. Dieser Datenbereich des Sub-CHUNK wird in 4-Byte-Felder (DWORD) aufgeteilt. Das erste Feld gibt die Verzögerungszeit in Millisekunden (Microseconds) zwischen zwei Einzelbildern (Frames) an. Das nächste Feld enthält die maximale Datenrate in Byte pro Sekunde. Dieser Wert gibt an, wie viele Bytes pro Sekunde das System aus der

AVI-Datei lesen muß, um die Bilder in der korrekten Geschwindigkeit auszugeben. Erreicht das System diese Datenrate nicht (abhängig von der Bildgröße), wird die Sequenz verzögert ausgegeben.

Das Feld ab Offset 10H ist reserviert und definiert die Option *Padding Granularity*. Der Wert wird meist auf 2048 gesetzt.

Die Flags ab Offset 14H enthalten Informationen über die nachfolgenden Daten. Zur Zeit sind folgende Flags definiert:

10H	AVIF_HASINDEX: AVI-File hat idx1-CHUNK.
20H	AVIF_MUSTUSEINDEX: Index CHUNK zur Bestimmung der Bildreihenfolge verwenden
100H	AVIF_ISINTERLEAVED: Interleaved AVI-File (wird bei CD-ROMs verwendet)
10000H	AVIF_WASCAPTUREFILE: AVI-File zur Speicherung von Real-Time-Videos verwenden
20000H	AVIF_COPYRIGHTED: AVI-File mit Copyright-Daten

Die Definition dieser Flag-Konstanten erfolgt in den Header-Dateien der AVI-Bibliothek.

Der Wert ab Offset 18H definiert die Zahl der Einzelbilder (Frames) in der AVI-Datei. Das Feld mit der Zahl der *Initial Frames* wird nur bei Interleave-Bildern verwendet. Der Wert gibt die Zahl der Frames an, die vor dem *Initial Frame* in der AVI-Datei stehen.

Das Feld ab Offset 20H definiert die Zahl der Streams in der AVI-Datei. Ein AVI-File mit Audio- und Videodaten besitzt zwei Streams.

Das Feld ab Offset 24H gibt die minimale Größe des Puffers für die Wiedergabe (Playback Buffer) an.

Ab Offset 28H finden sich zwei Felder, die die Abmessungen des Bildausschnitts für die Videosequenz angeben.

Ab Offset 30H steht die Einheit für die Zeitskala (Time Scale Unit). Das folgende Feld gibt die Datenrate (Frames) bei der Ausgabe an. Die Rate in *Samples per Second* läßt sich zu:

```
Datenrate / Time Scale
```

berechnen. Das Feld *Start* definiert die Startzeit des Videos und sollte auf 0 gesetzt werden. Das letzte Feld gibt die Länge des AVI-Files an. Die Einheiten werden durch die vorhergehenden Felder (Time Unit und Datenrate) definiert.

Der AVI-Stream-Line-Header-CHUNK (strl)

Auf den AVI-Header-CHUNK (hdrl) folgen ein oder mehrere Stream-Line-Header-CHUNKs (strl). Für jeden Stream existiert ein strl-CHUNK. Diese CHUNKs enthalten Informationen zu den einzelnen Streams der AVI-Datei. Die Informationen in diesen Streams beziehen sich auf den jeweiligen Daten-Stream im movi-CHUNK mit der gleichen Nummer.

Der CHUNK besitzt dabei den in Tabelle 16.5 gezeigten Aufbau. Als CHUNK-Typ wird allerdings die Signatur 'strl' eingetragen.

Jeder dieser strl-CHUNKs muß einen *stream header* (strh) und einen *stream format* (strf) Sub-CHUNK enthalten. Weiterhin können noch Stream-Data Sub-CHUNKs folgen.

Der strh-Sub-CHUNK

Dieser Header enthält Informationen zum zugehörigen Stream und besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Signatur ('strh')
04H	4	Länge Sub-CHUNK (38H)
08H	4	Typ (4 Zeichen)
0CH	4	Handler (4 Zeichen)
10H	4	Flags
14H	4	reserviert
18H	4	Initial Frame
1CH	4	Scale
20H	4	Rate
24H	4	Start
28H	4	Length
2CH	4	Buffer Size
30H	4	Quality
34H	4	Sample Size

Tabelle 16.6 strh-Sub-CHUNK in einer AVI-Datei

Die ersten 4 Bytes enthalten die Signatur 'strh', gefolgt von der Längenangabe (in Byte) für den Datenbereich. Die 4 Bytes ab Offset 08H definieren den Typ des Streams. Hier werden die Strings 'vids' (video stream) oder 'auds' (audio stream) eingetragen.

An diesen Bereich schließen sich die Felder des Stream-Headers an. Die 4 Bytes ab Offset 0CH definieren den Typ des Handlers für die Kompression/Dekompression der Daten. Der Typ wird dabei mit 4 Zeichen (z. B. 'msvc') definiert. Das DWORD ab Offset 10H enthält Flags für den Datenstrom:

AVISF_DISABLED (Wert 01H)	Data Stream nur anzeigen, wenn er explizit freigegeben wird.
AVISF_VIDEO_PALCHANGES (Wert 10000H)	AVI-Datei enthält Informationen zum Palette-Wechsel.

Die Bits der einzelnen Flags sind in den Headerdateien der AVI-Bibliothek definiert.

Das DWORD ab Offset 18H definiert die Zahl der Frames, die vor dem *Initial Frame* auftreten. Dieses Feld wird nur bei Interleaved AVI-Files benutzt. Die restlichen Felder beschreiben die Abspielcharakteristik (playback characteristics) der AVI-Datei. Das Feld *Scale* definiert die Zeiteinheit für die Wiedergabe. Die Felder *Scale*, *Rate*, *Start*, *Length* und *Buffer Size* besitzen die gleiche Bedeutung wie die Felder im hdrl-CHUNK. Im Feld *Quality* wird ein Integerwert zwischen 0 und 10000 abgelegt. Dieser gibt die Qualität an, die beim Aufbereiten (encoding) der Daten benutzt werden soll. Das Feld *Sample Size* gibt die Größe eines einzelnen Bildes (sample) an. Ist der Wert 0, liegen die Einzelbilder in ungleichmäßiger Größe vor und werden in Sub-CHUNKs gespeichert. Wurde ein Wert eingetragen, besitzen alle Einträge (samples) die gleiche Größe.

Einige der Felder treten auch im Stream-Line-Header auf. In diesem Fall gelten die Daten im Stream-Header für die gesamte Datei, während sich die Daten in der strl-Struktur nur auf den folgenden Stream beziehen.

Der Stream-Format-CHUNK (strf)

Dieser Sub-CHUNK muß in jedem Stream-Header CHUNK hinter dem Stream Header (strh)-CHUNK auftreten. Dieser CHUNK enthält die Formatbeschreibung der Daten im zugehörigen Stream. Enthält der Stream Videodaten, wird der Stream-Format-CHUNK als BITMAPINFO-Header-Struktur definiert.

Offset	Bytes	Bemerkungen
00H	4	Header Size
04H	4	Bildbreite (in Pixel)
08H	4	Bildhöhe (in Pixel)
0CH	2	Zahl der Planes
0EH	2	Bits per Pixel
10H	4	Kompressionstyp
14H	4	Bildgröße in Byte
18H	4	X Pels per Meter
1CH	4	Y Pels per Meter
20H	4	Colors Used
24H	4	Colors Important
28H	n*4	RGB-Quad-Struktur 1 Byte blau 1 Byte grün 1 Byte rot 1 Byte reserviert

Tabelle 16.7 BITMAPINFO-Header-Struktur

Die Struktur wird auch in den Windows-BMP-Dateien verwendet. Das erste DWORD gibt die Länge des Headers (einschließlich des RGB-Quad-Records) an.

Die folgenden zwei Felder definieren die Bildabmessungen. Ab Offset 0CH findet sich die Zahl der Farbebenen, in denen die Daten gespeichert sind (sollte auf 1 gesetzt werden). Das folgende Word definiert die Zahl der Bits per Pixel. Ab Offset 10H folgt ein Flag (DWORD), welches die Komprimierung der Bilddaten angibt.

0: RGB, unkomprimierte Daten als Bitmap

1: RLE8, 8 Bit werden nach dem RLE-Verfahren komprimiert

2: RLE4, 4 Bit werden nach dem RLE-Verfahren komprimiert

Eine Beschreibung der Kompressionsmethoden finden Sie beim Windows-BMP-Format. Das DWORD ab Offset 14H gibt die Bildgröße in Byte an. Danach folgen zwei 4-Byte-Felder mit der horizontalen und vertikalen Auflösung. Diese wird in Picture Elements (Pels) per Meter angegeben.

Die letzten beiden Felder dienen zur Verwaltung der Farbinformationen. Es wird die Zahl der benutzten Farben und die Zahl der wichtigen Farben definiert.

Bei Audiodaten enthält der Sub-CHUNK eine PCMWAVEFORMAT- oder eine WAVEFORMATEX-Struktur. Die WAVEFORMATEX-Struktur stellt dabei eine Erweiterung der PCMWAVEFORMAT-Struktur dar.

Offset	Bytes	Bemerkungen
00H	2	Formattyp
02H	2	Kanalzahl
04H	4	Abtastrate (Sample Rate)
08H	4	Bytes pro Sekunde
0CH	2	Blockgröße Daten
10H	2	Bits per Sample
12H	2	Byte Counter Extend-Daten

Tabelle 16.8 WAVEFORMATEX-Struktur

Das erste Feld gibt den Formattyp (WAVE_FORMAT_PCM) oder (WAVE_FORMAT_EX) an. Ab Offset 02H steht die Zahl der Audiokanäle (1 = mono, 2 = stereo). Das DWORD ab Offset 04H gibt die Datenrate (sample rate) des Audiokanals in Samples per Sekunde an. Das folgende Feld enthält die mittlere zu übertragende Datenrate (in Byte/Sekunde) und dient zur Bestimmung der Puffergröße. Ab Offset 0CH findet sich die Blockgröße, in der die Daten zu lesen sind. Dies ist bei Interleaved-Daten (CD-ROM) erforderlich, da hier 2-kByte-Blöcke gebildet werden. Das folgende Bit per Sample schließt sich an die WAVE_FORMAT_PCM-Struktur an und definiert die Bits per Sample. Das letzte Feld tritt nur in der WAVE_FORMAT_EX-Struktur auf und gibt die Länge der folgenden Zusatzinformationen (extra informations) an.

Der Stream-Data-CHUNK (strd)

An den Stream-Format-CHUNK kann sich ein Stream-Data-CHUNK anschließen. Das Format dieses CHUNK wird durch den installierbaren Kompressions- oder Dekompressionstreiber festgelegt. Der Datenblock enthält in der Regel Informationen zur Konfigurierung des Treibers.

Der movi-CHUNK

An die CHUNKs der ersten LIST-Struktur schließt sich eine movi-CHUNK-Struktur mit den eigentlichen Daten an. Diese Daten können in einem Block oder in verschiedenen Sub-CHUNKs (rec-CHUNKs) vorliegen. Die rec-CHUNKs finden bei CD-ROMs Verwendung, um interleaved Daten zu speichern. Die Daten werden dabei in 2-kByte-Blöcken abgelegt. Der Treiber sollte in diesem Fall die komplette Datenstruktur lesen. Der movi-CHUNK besitzt folgende Struktur:

Offset	Bytes	Bemerkungen
00H	4	Signatur 'LIST'
04H	4	CHUNK-Länge in Byte
08H	4	CHUNK-Typ 'movi'
0CH	n	Datenbereich oder Sub-CHUNKs

Tabelle 16.9 Struktur eines movi-CHUNK

Der CHUNK beginnt mit einer 4-Byte-LIST-Signatur. Daran schließt sich Angabe der Länge des folgenden Datenbereiches an. Die letzten 4 Bytes enthalten den String 'movi' zur Markierung des CHUNK-Typs.

Der rec-CHUNK

Wurden die Daten in rec-Sub-CHUNKs aufgeteilt, schließt sich an den Header folgende Struktur an:

Offset	Bytes	Bemerkungen
00H	4	Signatur 'LIST'
04H	4	Länge Datenbereich
08H	4	Sub-CHUNK Typ 'rec '
0CH	n	Datenbereich

Tabelle 16.10 Struktur eines rec-CHUNK

Der Sub-CHUNK wird wieder mit der Signatur LIST eingeleitet. Die folgende Längenangabe bezieht sich auf den Datenbereich des Sub-CHUNKs. Das DWORD ab Offset 08H definiert den CHUNK-Typ (rec). Ab Offset 0CH folgen dann die Daten. Umfaßt der Datenbereich nicht ein Vielfaches von 2 kByte, kann sich anschließend ein JUNK-CHUNK zum Füllen anschließen (siehe unten).

Struktur des Datenrecords

Werden die Daten in einem Block im movi-CHUNK gespeichert, besitzt dieser Bereich folgende Struktur:

Offset	Bytes	Bemerkungen
00H	4	Signatur
04H	n	Datenbereich

Tabelle 16.11 Format eines Datenrecords

Da die Formatinformationen im Header-Stream stehen, enthält der Datenbereich nur eine 4-Byte-Signatur zur Identifikation der Daten. Hierbei gilt folgender Code:

- xxwb: Wave-Daten folgen
- xxdb: DIB-Bitmap-Daten (unkomprimiert) folgen
- xxdc: DIB-Bitmap-Daten (komprimiert) folgen

Die Zeichen xx dienen dabei zur Identifizierung des Streams. Ein Bitmap-Bild wird dabei als unkomprimierte oder komprimierte DIB-Struktur abgelegt, d.h., die Daten eines Pixels stehen zum Beispiel in einem Byte (bei 8 Bit per Pixel).

Der AVI_PALCHANGE-CHUNK

Innerhalb einer AVI-Datei kann die Palette zwischen einzelnen Bildern geändert werden. Dieser CHUNK besitzt folgende Struktur:

Offset	Bytes	Bemerkungen
00H	2	Signatur 'xxpc'
02H	1	1. Palette zu ändern
03H	1	Zahl der Einträge
04H	2	Flags
06H	n*4	Palette-Einträge

Tabelle 16.12 Format eines AVI-PALCHANGE-CHUNK

Der CHUNK wird durch einen 2-Byte-String eingeleitet. Die Signatur pc steht dabei für *palette change*. Das nächste Byte definiert den Index zum ersten zu ändernden Palette-Eintrag (0 bis 255). Daran schließt sich ein Byte mit der Zahl der Palette-Einträge des CHUNK an. Die Belegung des Flags ist nicht bekannt. Ab Offset 06H folgen n Einträge mit den Palette-Daten (grün, blau, rot, reserviert).

Anmerkung: Sofern AVI_PALCHANGE-CHUNKs im Datenstrom auftreten, muß im Stream-Header das VIDEO_PALCHANGE-Flag gesetzt sein.

Der idx1-CHUNK

Eine AVI-Datei kann optional mehrere idx-CHUNKs im Anschluß an einen movi-CHUNK enthalten. Hierbei handelt es sich um eine Liste mit Zeigern auf die einzelnen Daten-CHUNKs, mit der die Reihenfolge definiert wird. Damit ist auch ein Direktzugriff auf die Daten möglich, ohne die gesamte AVI-Datei analysieren zu müssen. Der CHUNK besitzt folgenden Aufbau:

Offset	Bytes	Bemerkungen
00H	4	Signatur ('idx1')
04H	4	Flags
08H	4	CHUNK-Offset
0CH	4	CHUNK-Länge

Tabelle 16.13 Struktur eines idx1-CHUNK

Die ersten 4 Bytes enthalten die Signatur (idx1). Daran schließt sich ein DWORD mit Flags an.

AVIIF_KEYFRAME	Das Flag zeigt an, dass Key Frames in der Videosequenz auftreten.
AVIIF_NOTIME	Der CHUNK beeinflußt nicht das Video-Timing (z. B. ein Palette-Change-CHUNK).
AVIIF_LIST	Markiert einen LIST-CHUNK.

Ab Offset 08H steht der Offset (relativ zum movi-CHUNK) auf den betreffenden CHUNK, gefolgt von der Längenangabe (ohne den 8-Byte-Header) für den CHUNK.

Der obige idx-CHUNK kann sich mehrfach wiederholen. Enthält der movi-CHUNK die Daten in rec-Sub-CHUNKs, wird für jeden Record je ein idx-CHUNK definiert.

Andere Daten-CHUNKs

Die RIFF-Definition erlaubt die Definition weiterer CHUNKs. Diese werden von Microsoft registriert und periodisch veröffentlicht. Der folgende CHUNK tritt zum Füllen von Datenbereichen auf:

Der JUNK-CHUNK

Dieser CHUNK besitzt lediglich die Aufgabe, eine Datenstruktur bis zur Blockgrenze (z. B. bei CD-ROM-Daten) aufzufüllen. Es gilt folgende Struktur:

Offset	Bytes	Bemerkungen
00H	4	Signatur 'JUNK'
04H	n	Daten

Tabelle 16.14 Struktur eines JUNK-CHUNK

Alternativ kann auch die Signatur 'PAD' auftreten, die ebenfalls zum Füllen von Datenbereichen benutzt wird. Der JUNK-CHUNK wurde von IBM registriert, während der PAD-CHUNK von Microsoft vorgeschlagen wurde.