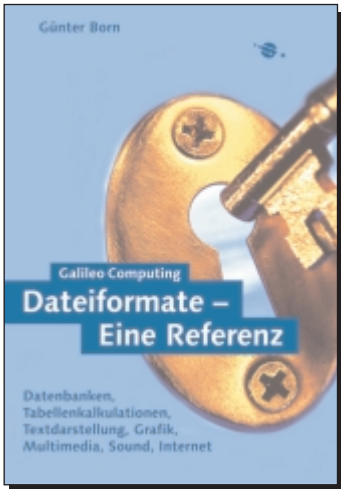
Dieses Kapitel stammt aus dem Buch

›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

34 Das PCPAINT/Pictor-Format (PIC)

Das Programm PCPAINT wurde 1984 für Mouse Systems entwickelt. PCPAINT existiert in verschiedenen Versionen, wobei ab der Version 2.0 das Format stabil wurde. Die folgende Beschreibung beschränkt sich deshalb auf das Format von PCPAINT 2.0 und Pictor 3.0/3.1. Die Bilddateien bestehen aus einem Header, gefolgt von einem Bilddatenbereich für Monochrom- oder Farbbilder. Die Daten werden innerhalb der PIC-Datei immer im Intel-Format abgelegt. Der Bildaufbau erfolgt von unten nach oben.

Der PCPAINT/Pictor-Header

Der PIC-Header besitzt eine variable Länge und enthält die Felder aus Tabelle 34.1.

Offset	Bytes	Bemerkungen
00H	2	Signatur 1234H
02H	2	X Bildbreite in Pixel
04H	2	Y Bildhöhe in Pixel
06H	2	X-Kordinate unten links
08H	2	Y-Koordinate unten links
0AH	1	Image-Flag Bit 0-3: Bit per Pixel und Plane Bit 4-7: Bitplanes
0BH	1	Flag (FFH = folgende Einträge gültig)
0CH	1	Video Mode (0 Standard) 0: 40 Zeichen Text 1: 80 Zeichen Text 2: Text Monochrom 3: 43 Zeilen Text A: 320 x 200 x 4 (CGA) B: 320 x 200 x 16 Tandy 1000 etc. C: 640 x 200 x 2 (CGA) D: 640 x 200 x 16 (EGA) E: 640 x 350 x 2 (EGA) F: 640 x 350 x 4 (EGA) G: 640 x 350 x 16 (EGA) H: 720 x 348 x 2 (Hercules) I: 320 x 200 x 16 (Plantronics) J: 320 x 200 x 16 (EGA) K: 640 x 400 x 2 (AT&T, Toshiba) L: 320 x 200 x 256 (VGA) M: 640 x 480 x 16 (EGA+, VGA) N: 720 x 384 x 16 (Herc. Incolor) O: 640 x 480 x 2 (VGA)

Offset	Bytes	Bemerkungen
0DH	2	Extra Info-Descriptor 0: nichts 1: Palette (1 Byte), Border (1 Byte) [CGA] 2: PCjr oder nicht ECD 16 Color-Register (0-15), je 1 Byte 3: EGA mit EDC 16 Color-Register (0-63), je 1 Byte 4: VGA 256 Color Info
0FH	2	Size Extra Info-Feld
11H	n	Extra Info-Block (optional)
..H	2	Nummer gepackte Blocks

Tabelle 34.1 Struktur des PCPAINT/Pictor PIC-Headers

Ab Offset 0 steht die Signatur 1234H für eine PIC-Datei. Die folgenden zwei Worte definieren die Bildbreite und Bildhöhe in Pixel. Ab Offset 06H findet sich der Nullpunkt des Bildes in Bildschirmkoordinaten. Dieser Wert wird standardmäßig auf 0,0 gesetzt. Das Bild ist von unten nach oben aufzubauen.

Das Byte ab Offset 0AH wird in zwei Nibbles aufgeteilt. Die unteren 4 Bits definieren die Zahl der Bits pro Pixel und Bildebene. Die oberen 4 Bit definieren die Zahl der Bildebenen. Für CGA mit 4 Farben enthält das Byte den Wert 02H, eine EGA-Darstellung mit 16 Farben wird mit 31H (3 Bildebenen, 1 Bit pro Pixel und Plane) kodiert. Der Plantronics-16-Farbmodus wird mit dem Eintrag 12H markiert.

Das Byte ab Offset 0BH definiert, ob die Datei Daten über die Palette und den Videomodus enthält. Dies ist ab Version 2.0 möglich. Ist der Wert ab Offset 0BH auf FFH gesetzt, sind die folgenden Felder gültig. Enthält Offset 0BH andere Werte, folgen die Bilddaten ab Offset 0CH.

Ist Offset 0BH = FFH, speichert das PIC-Format den Bildschirmmodus ab Offset 0CH. Das Wort ab Offset 0DH definiert, ob im Header ein Datenbereich mit Extra-Infos auftritt. In diesem Block lassen sich Palettendaten etc. ablegen. Ist das Word auf 00H gesetzt, fehlt der *Extra Info*-Block. Die Länge des *Extra Info*-Blocks (in Byte) ist im Wort ab Offset 0FH abgelegt. Steht hier der Wert 0, fehlt der Block.

Der optionale *Extra Info*-Block beginnt ab Offset 17H und enthält die Palettendaten. Deren Struktur ist in Tabelle 34.1 grob definiert. Zum Beispiel werden bei einer CGA-Karte zwei Byte (Palette, Border) benötigt. Beim Modus 3 liegen dagegen 64 Byte mit den Registerwerten vor.

An den *Extra Info*-Block schließt sich dann noch ein Wort mit der Zahl der folgenden gepackten Bilddatenblocks an. Ist hier der Wert 0 eingetragen, enthält die Datei ungepackte Daten.

Der PIC-Datenbereich

Auf den Header folgt der PIC-Datenbereich mit gepackten oder ungepackten Daten. In PCPAINT 2.0 werden die Bitplanes zusammengepackt, so daß sich die Daten sehr einfach auf dem PCjr anzeigen lassen (z. B. 4 Bit pro Pixel, 1 Bit pro Plane). Ab Pictor 3.0 werden die Bitplanes vor dem Speichern separiert und ebenenweise abgelegt. Eine PIC-Datei kann Bilddaten mit 1, 4 und 8 Bit pro Pixel enthalten.

Monochrombilder (1 Bit pro Pixel)

Bei 1 Bit pro Pixel werden die Daten in einer Ebene gespeichert. Jedes Byte definiert 8 Bildpunkte.

Farbbilder (4 Bit pro Pixel)

Hier werden die Bilddaten in 4 Ebenen aufgeteilt. Jede Ebene enthält dann ein Bit pro Pixel, d. h., ein gelesenes Bilddatenbyte beschreibt 8 Bildpunkte einer Farbebene. Für EGA- und VGA-Karten muß dann die Bildinformation aus den vier Farbebenen (Rot, Grün, Blau, Intensität) zusammengesetzt werden.

Farbbilder (8 Bit pro Pixel)

Hier werden die Bilddaten entweder zu 8 Bit pro Pixel in einer Ebene oder mit 1 Bit pro Pixel in 8 Ebenen abgelegt. Je nach Kodierung muß dann die Farbinformation für einen Bildpunkt zusammengesetzt werden.

Die Bilddatenblocks (PIC-File)

Die Bilddaten werden in einzelnen Blöcken abgelegt. Dabei können die Farbinformationen in einzelnen Ebenen oder zusammengefaßt abgespeichert werden. Die Art der Kodierung der Ebenen findet sich im Header ab Offset 0AH. Hier ist sowohl die Zahl der Bits pro Pixel und Ebene als auch die Zahl der Ebenen vermerkt.

Bei gepackter PIC-Datei (Offset 17H > 0) werden die Bilddaten in Blöcken zusammengefaßt. Beim Wechsel einer Bildebene muß auch ein neuer Block angelegt werden. Die Zahl der gepackten Blöcke folgt im Header hinter dem Extra-Info-Block.

Struktur eines Datenblocks

Die gepackten Daten werden in einzelnen Paketen (Datenblocks) abgelegt. Ein solches Paket besteht aus einem Header, gefolgt von den gepackten und ungepackten Bilddaten. Der Block-Header besitzt den in Tabelle 34.2 gezeigten Aufbau.

Bytes	Feld
2	Blockgröße (in Byte)
2	Größe Datenbereich ungepackt (in Byte)
1	Marker für gepackte Subrecords

Tabelle 34.2 Struktur eines Block-Headers

Die Blocklänge und die Länge der unkomprimierten Daten werden als Worte kodiert. Die Blocklänge schließt dabei diese beiden Längenbytes mit ein.

Ab Offset 05H wird ein Code als Markerbyte vereinbart. Der Wert dient innerhalb des Blockes zur Markierung gepackter Subrecords. In diesem Byte sollte deshalb ein Wert eingetragen werden, der innerhalb des entpackten Datenbereiches nicht oder nur selten vorkommt. Andernfalls vergrößert sich die PIC-Datei (siehe unten).

An diesen Header schließen sich dann die ungepackten Daten und/oder die Subrecords mit den komprimierten Daten an. Das PIC-Format nutzt dabei ein sehr einfaches Komprimierungsverfahren:

- Tritt ein Datenbyte mehrfach hintereinander auf, wird es in einem Subrecord als *Run Length* gespeichert.
- Ungleiche Datenbytes werden unkodiert in einem Block abgelegt.

Die Subrecords mit den gepackten Daten besitzen den in Tabelle 34.3 gezeigten Aufbau.

Bytes	Feld
1	Marker (markiert einen Run)
1	Run Count; IF Run Count = 0, 16-Bit-Run Count
1	Datenbyte Run

Tabelle 34.3 Struktur eines komprimierten Subrecords

Das Markerbyte dient zur Anzeige des gepackten Subrecords. Der Wert wurde im Header des Blocks ab Offset 05H vereinbart. Das folgende Byte enthält den Wiederholungszähler für das nächste Datenbyte. Ist der Wert größer 0, muß das Datenbyte n mal (max. 255) kopiert werden. Um größere Wiederholungsfaktoren abzubilden, wurde eine erweiterte Struktur definiert. Ist der Wiederholungszähler = 0, folgt ein 16-Bit-Wiederholungszähler. Dann ist das letzte Byte im Subblock n mal zu wiederholen.

Ungepackte Daten werden dagegen als einfache Bytefolge im Block abgelegt. Tritt ein Code mit dem Wert des Markerbytes auf, muß dieser Code in einem komprimierten Subrecord mit dem Wiederholungszähler 1 gespeichert werden. Deshalb ist es wichtig, daß der Markercode nicht oder nur selten in den Bilddaten auftritt. Andernfalls wird ein Bilddatenbyte mit drei Byte gespeichert, was die Dateilänge erhöht.

Das Lesen eines Datenblockes läßt sich mit folgendem Algorithmus durchführen:

```
Get block size
Get orig size
Get marker byte
Repeat (for all bytes in block)
 Get next byte
 IF byte = marker code THEN
/* komprimierter Subrecord
  Get run count byte
  IF run count = 0 THEN
   Get 16-Bit run count
  ENDIF
  Get data byte
  Repeat run count
   write data byte
  END Repeat
 ELSE
/* unkomprimiertes Datenbyte
  write data byte
 ENDIF
END Repeat
```

Abbildung 34.1 Pseudocode PIC-Reader

Diese Speicherstruktur ist sehr ökonomisch, falls ein Bild nur wenige Informationen enthält. Dann liegen große Bereiche mit gleichen Bilddaten vor, die sich gut packen lassen. Bei Bildern mit Mustern werden die Bilddaten unkomprimiert abgespeichert.