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

23 GEM Image File-Format (IMG)

Verschiedene GEM-Programme wie DR PAINT, DR DOODLE oder Publishers Paintbrush benutzen zum Speichern von Grafiken das Format der GEM-IMG-Dateien. Hierbei handelt es sich im wesentlichen um eine pixelorientierte Abspeicherung des Bildschirmfensters. Dieses Fenster wird gemäß Abbildung 23.1 in Zeilen und Spalten mit den Bildpunkten aufgeteilt.

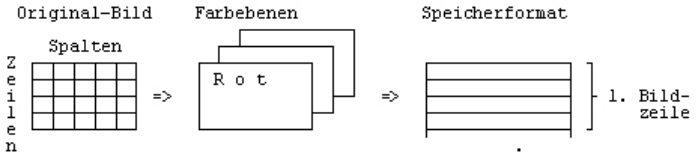

Abbildung 23.1 Umsetzung eines Bildes zum Speichern

Jedes durch eine Zeilen- und Spaltennummer adressierte Element des Originalbildes wird als Pixel (*Picture Element*) bezeichnet. Durch Aneinanderfügen dieser Pixel entsteht das Gesamtbild. Bei der Schwarzweißdarstellung entspricht jedes Pixel einem Bit, das gesetzt oder gelöscht sein kann. In der Regel werden jedoch farbige Bilder bearbeitet und gespeichert. Hier ist jedem Bildelement (Pixel) eine zusätzliche Farbinformation zugeordnet, die nicht mehr mit einem Bit kodierbar ist.

In Abhängigkeit von der Kodierung läßt sich die Farbe eines Bildpunktes in der Regel aus den Grundfarben Rot, Grün, Blau und einer Intensitätsinformation erzeugen. Dies bedeutet, daß das eigentliche Bild in vier Teilbilder mit den jeweiligen Farben zerlegt werden muß. So enthält man zum Beispiel ein Teilbild, in dem alle Pixel des Originalbildes mit einem Rotanteil gesetzt sind. Analoges gilt für das Teilbild mit den grünen Punkten. Durch Mischen der Grundfarben und Verknüpfung mit dem Intensitätsbit lassen sich so im Originalbild 16 Farben darstellen.

Zum Speichern der Bilder lehnt man sich an die Aufteilung in einzelne Zeilen an: Zeile für Zeile des Originalbildes wird gespeichert; die Pixel der einzelnen Farbebenen sind zeilenweise festgehalten. Dies bedeutet, daß der Datenbereich der ersten Zeile zunächst alle Bits der Farbe Rot erfaßt, dann die Zeile mit der Kodierung der grünen Farbe, daraufhin die Zeile mit den blauen Punkten und anschließend die Intensitätsbits. Erst wenn alle diese Informationen abgespeichert sind, kann die Bearbeitung der nächsten Bildzeile beginnen. Dieser Sachverhalt wurde in Abbildung 23.1 schematisch dargestellt.

Bei der Speicherung treten allerdings – wie bei den anderen Formaten (PCX, TIFF etc.) besprochen – einige Schwierigkeiten bezüglich des Speicherbedarfs auf. Ein Grafikbildschirm mit einer Auflösung von 640 x 200 Pixel benötigt bereits 128000 Bit zum Abspeichern eines S/W-Bildes. Hier wird jeder Punkt durch ein gesetztes oder ein gelöschtes Bit

dargestellt. Bei Farben umfaßt ein Bildpunkt mit der beschriebenen Kodierung vier Bit (Rot, Grün, Blau, Intensität). Damit ergibt sich ein vierfacher Speicherbedarf von 512 kBit pro Bild. Zur Reduzierung des Speicherbedarfs gelangen in den IMG-Dateien verschiedene Datenreduktionsverfahren zum Einsatz, die mit dem dazugehörigen Speicherformat nachfolgend beschrieben sind. Abbildung 23.2 zeigt den prinzipiellen Aufbau einer IMG-Datei:

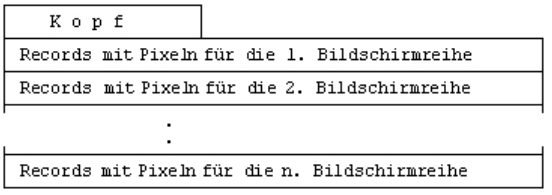

Abbildung 23.2 Prinzipieller Aufbau einer IMG-Datei

Die Datei besteht aus einem Kopfteil mit den Initialisierungsdaten, an den sich die eigentlichen Bildinformationen in der jeweiligen Kodierung anschließen. Diese werden in Abhängigkeit vom Komprimierungsverfahren in Records verschiedener Länge aufgeteilt.

Der IMG-Kopfsatz

Der Kopfsatz besteht in den bisherigen GEM-Versionen (bis 2.x) aus 16 Byte, die jedoch wortweise nach folgender Kodierung interpretiert werden:

Word	Kodierung
1	Dateiversion
2	Kopflänge in Wörtern
3	Bits pro Pixel (Zahl der Farbebenen)
4	Länge eines Musters
5	Pixelbreite (in Mikrometern)
6	Pixelhöhe (in Mikrometern)
7	Länge einer Pixelreihe in Punkten
8	Anzahl der Pixelzeilen (Scanlines)
9	Ventura Erweiterung (Bit Image-Flag)

Tabelle 23.1 Aufbau des Kopfsatzes einer IMG-Datei

Wichtig: Kopfdaten sind entgegen der üblichen Intel-Konvention gespeichert. Auf dem Byte mit der niederwertigen Adresse befindet sich das höherwertige Byte des Speicherwortes. Die Werte *FF 03H* ergeben bei der Abspeicherung gemäß den Intel-Konventionen das Ergebnis *03FF*. Beim IMG-Header ist der Wert als *FF03* zu interpretieren. Dies ist bei allen Zugriffen auf IMG-Dateien zu berücksichtigen.

Der erste Header-Eintrag gibt die GEM-Versionsnummer an, unter der die Bilddatei erstellt wurde. In den Versionen bis 2.x steht hier konstant der Wert 00 01H. Daran schließt sich das Feld mit der Kopflänge an. Auch hier speichert GEM zumindest bis zur Version 2.x konstant den Wert 00 08H ab. Digital Research hält sich hier jedoch die Möglichkeit offen, die Länge des Kopfes zu verändern, weshalb Anwenderprogramme immer die Kopflänge lesen sollten, um auch mit den Datenformaten neuerer GEM-Versionen kompatibel zu bleiben. Das Programm *Ventura Publisher* verwendet zum Beispiel ein extended *IMG-Format*, welches 9 Worte im Header belegt.

Das dritte Feld bestimmt die Anzahl der Bits pro Pixel und ist für die Farbdarstellung relevant. Der Wert 1 markiert eine monochrome Darstellung. Werte zwischen 2 und 16 stehen für Farbbilder.

Diese Information wird beim Speichern und Dekodieren von Bilddaten zur Bestimmung der Anzahl von Farbebenen benötigt. (Das Bild wird zeilenweise in den Pixelreihen der vier Farbebenen abgelegt.) In GEM ist die Reihenfolge der Farbebenen so festgelegt:

```
Rot  Grün  Blau  Intensität
```

In der Speicherdatei stehen daher zunächst alle Bits der Farbe Rot der betreffenden Zeile, gefolgt von den grünen Farbwerten derselben Zeile, der blauen Farbebene und den Intensitätsbits dieser Zeile. Erst danach kann die nächste Bildzeile bearbeitet werden.

Das vierte Word legt die Länge eines Musters fest. Hierbei handelt es sich um die Standardmuster (z. B. Backsteinmuster), die von den Grafikprogrammen zum Füllen von Flächen zur Verfügung gestellt werden. Die Information über die Länge des Musters wird beim *Pattern Run*-Komprimierungsverfahren benötigt und ist in vielen GEM-Versionen zur Zeit konstant auf 2 Byte festgelegt, darf aber zwischen 1 und 8 liegen.

Die Größe eines einzelnen Bildpunktes hängt natürlich von den vorherrschenden Geräteeigenschaften ab. Um eine Übertragung und Anpassung bestehender Zeichnungen an andere Geräte zu ermöglichen, werden die Abmessungen eines Bildpunktes im Dateikopf gespeichert. Die Breite eines Punktes in Mikrometern findet sich in Feld 5, während die Pixelhöhe in Feld 6 gespeichert wird. Mit diesen Kenngrößen lassen sich die Bilder weitgehend geräteunabhängig darstellen. Die Einträge müssen jeweils an das Gerät angepaßt werden, das die Datei erzeugt.

Das siebte Word enthält die Zahl der Pixel pro Bildzeile. Diese Information ist wichtig, da die Speicherung ja zeilenweise erfolgt. Das Feld wird deshalb von verschiedenen Komprimierungsverfahren ausgewertet.

Im achten Word steht die Zahl der Pixelzeilen in der IMG-Datei. Dieser Wert definiert damit auch die Bildhöhe.

Das neunte Word ist nur verfügbar, falls die Headerlänge im zweiten Eintrag die Länge mit 9 Word-Einträgen angibt. Dann handelt es sich um das erweiterte IMG-Format des Ventura Publisher, der im neunten Word das *Bit Image*-Flag speichert. Das Flag bestimmt die Interpretation von Bildern mit mehreren Farbebenen (z. B. Farbbilder, Graustufenbilder). Ist Word 9 vorhanden und enthält die Datei mehr als zwei Farbebenen (Word 3 ist größer als 2), enthält das Flag einen gültigen Wert. Der Wert 0 definiert dann, daß es sich um ein Farbgrafikbild handelt. Dabei wird eine feste Palette mit 16 Einträgen gemäß Abbildung 23.3 verwendet. Die Palette-Werte müssen vom lesenden Programm gesetzt werden. Die in Bild 23.3 in Klammern eingefaßten Werte sind als Hexzahlen zu interpretieren und geben den Palette-Wert der Farben *Rot, Grün* und *Blau an.*

```
 0: [3F,3F,3F]  1: [3F,00,00]  2: [00,3F,00]  3: [3F,3F,00]
 4: [00,00,3F]  5: [3F,00,3F]  6: [00,3F,3F]  7: [2B,2B,2B]
 8: [15,15,15]  9: [2B,00,00] 10: [00,2B,00] 11: [2B,2B,00]
12: [00,00,2B] 13: [2B,00,2B] 14: [00,2B,2B] 15: [00,00,00]
```

Abbildung 23.3 GEM-Standard 16-Farben-Palette

Ein Wert 1 im *Bit Image*-Flag (Word 9) bedeutet, daß ein Graustufenbild in der Datei gespeichert ist. In diesem Fall wird eine feste Palette mit 256 Graustufen verwendet. Die benötigten Werte müssen vom lesenden Programm gemäß Abbildung 23.4 bereitgestellt werden.

```
  0: FF   1: 7F   2: BF   3: 3F   4: DF   5: 5F   6: 9F   7: 1F   8: EF   9: 6F
 10: AF  11: 2F  12: CF  13: 4F  14: 8F  15: 0F  16: F7  17: 77  18: B7  19: 37
 20: D7  21: 57  22: 97  23: 17  24: E7  25: 67  26: A7  27: 27  28: C7  29: 47
 30: 87  31: 07  32: FB  33: 7B  34: BB  35: 3B  36: DB  37: 5B  38: 9B  39: 1B
 40: EB  41: 6B  42: AB  43: 2B  44: CB  45: 4B  46: 8B  47: 0B  48: F3  49: 73
 50: B3  51: 33  52: D3  53: 53  54: 93  55: 13  56: E3  57: 63  58: A3  59: 23
 60: C3  61: 43  62: 83  63: 03  64: FD  65: 7D  66: BD  67: 3D  68: DD  69: 5D
 70: 9D  71: 1D  72: ED  73: 6D  74: AD  75: 2D  76: CD  77: 4D  78: 8D  79: 0D
 80: F5  81: 75  82: B5  83: 35  84: D5  85: 55  86: 95  87: 15  88: E5  89: 65
 90: A5  91: 25  92: C5  93: 45  94: 85  95: 05  96: F9  97: 79  98: B9  99: 39
100: D9 101: 59 102: 99 103: 19 104: E9 105: 69 106: A9 107: 29 108: C9 109: 49
110: 89 111: 09 112: F1 113: 71 114: B1 115: 31 116: D1 117: 51 118: 91 119: 11
120: E1 121: 61 122: A1 123: 21 124: C1 125: 41 126: 81 127: 01 128: FE 129: 7E
130: BE 131: 3E 132: DE 133: 5E 134: 9E 135: 1E 136: EE 137: 6E 138: AE 139: 2E
140: CE 141: 4E 142: 8E 143: 0E 144: F6 145: 76 146: B6 147: 36 148: D6 149: 56
150: 96 151: 16 152: E6 153: 66 154: A6 155: 26 156: C6 157: 46 158: 86 159: 06
```

```
160: FA 161: 7A 162: BA 163: 3A 164: DA 165: 5A 166: 9A 167: 1A 168: EA 169: 6A
170: AA 171: 2A 172: CA 173: 4A 174: 8A 175: 0A 176: F2 177: 72 178: B2 179: 32
180: D2 181: 52 182: 92 183: 12 184: E2 185: 62 186: A2 187: 22 188: C2 189: 42
190: 82 191: 02 192: FC 193: 7C 194: BC 195: 3C 196: DC 197: 5C 198: 9C 199: 1C
200: EC 201: 6C 202: AC 203: 2C 204: CC 205: 4C 206: 8C 207: 0C 208: F4 209: 74
210: B4 211: 34 212: D4 213: 54 214: 94 215: 14 216: E4 217: 64 218: A4 219: 24
220: C4 221: 44 222: 84 223: 04 224: F8 225: 78 226: B8 227: 38 228: D8 229: 58
230: 98 231: 18 232: E8 233: 68 234: A8 235: 28 236: C8 237: 48 238: 88 239: 08
240: F0 241: 70 242: B0 243: 30 244: D0 245: 50 246: 90 247: 10 248: E0 249: 60
250: A0 251: 20 252: C0 253: 40 254: 80 255: 00
```

Abbildung 23.4 GEM-Standard 256-Graustufen-Palette

Der Wert FFH im Index 0 bedeutet, daß alle drei Einträge (Rot, Grün, Blau) den gleichen Wert (FF,FF,FF) verwenden. Die Graustufen-Palette wird niemals innerhalb der IMG-Datei gespeichert.

Speichern von IMG-Daten

Wie erwähnt, unterteilt GEM ein Bild in Zeilen und Spalten, wobei jedes Element dieser Matrix einem Bildpunkt entspricht. Das Speichern der Bilder erfolgt zeilenweise, wobei in der Farbdarstellung jeweils die Daten der verschiedenen Farbebenen der aktuellen Zeile bearbeitet werden. Erst danach wird zur folgenden Zeile übergegangen.

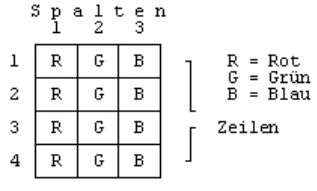

Abbildung 23.5 Muster in drei Spalten aus den Farben Rot, Grün, Blau

Zum Abspeichern des Musters sind *3 Spalten * 4 Zeilen * 4 Farben = 48 Bit* erforderlich. Die Zahl der Farbbits steht im Header einer IMG-Datei. Nun wird das Muster Zeile für Zeile abgetastet und in einer Datei gespeichert (Abbildung 23.5).

Aus diesen Informationen (Abbildung 23.5 und Abbildung 23.6) läßt sich später das Originalbild mit den Farben regenerieren. Die Zahl der Pixel pro Zeile steht im Kopf der IMG-Datei (Word 7). Auch die Zahl der Farbbits pro Pixel findet sich im Header (Word 3). Bei einem Wert von 4 wird jedes Pixel in vier Farbebenen zerlegt und gespeichert.

Pixel 1. Zeile (Rot)	1	0	0
Pixel 1. Zeile (Grün)	0	1	0
Pixel 1. Zeile (Blau)	0	0	1
Pixel 1. Zeile (Int.)	0	0	0
		...	
Pixel 4. Zeile (Rot)	1	0	0
Pixel 4. Zeile (Grün)	0	1	0
Pixel 4. Zeile (Blau)	0	0	1
Pixel 4. Zeile (Int.)	0	0	0

Abbildung 23.6 Speicherschema des unkodierten Musters

Bildkomprimierung bei IMG-Dateien

Bereits bei einer Auflösung von 640 x 300 Bildpunkten ist der Speicherbedarf eines Bildes recht hoch und nimmt bei steigender Auflösung und besserer Farbdarstellung noch drastisch zu. Aus diesem Grund benutzt GEM verschiedene Komprimierungsverfahren zur Datenreduzierung:

- *Vertical Replication Count*
- *Solid Run*
- *Bit String*
- *Pattern Run*

Diese Verfahren werden in Abhängigkeit von den Bildelementen gewählt.

Nach dem Header beginnt der Bereich mit den Bilddaten, wobei diese zeilenweise unter Berücksichtigung der Farbebenen gespeichert sind. Die Pixel werden in Records gemäß folgendem Format abgelegt:

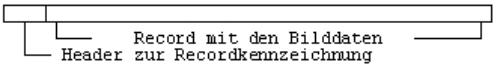

Abbildung 23.7 Datenspeicherung in Records

Der Aufbau der Records variiert dabei in Abhängigkeit vom Kodierungsverfahren.

Pixelkodierung

Naheliegend ist es, die einzelnen Pixel in ihre Farbebenen aufzuteilen und bitweise abzuspeichern. Bei vier Farbbits und 640 Pixel pro Zeile ergibt sich dann pro Zeile folgender Speicherbedarf:

```
640 Pixel x 4 Farbebenen / 8 Bit = 320 Byte
```

Hier versucht GEM, durch Anwendung geeigneter Kodierungen Speicherplatz zu sparen.

Das Solid Run-Format

Der einfachste Fall liegt vor, wenn mehrere Pixel einer Zeile identisch sind. Bei der Zerlegung der Pixel in die einzelnen Bits der Farbebenen sind häufig mehrere aufeinanderfolgende Bits mit dem gleichen Wert (0 oder 1) belegt. Zur Darstellung des Abschnitts genügt also die Angabe des Wertes und die Zahl der betroffenen Bits. Reserviert man nun ein Byte als Wiederholungszähler und benutzt Bit 7 als Markierung für *Solid Run*, sind maximal 127 Bit pro Datenbyte abbildbar, was bereits eine gute Komprimierung bedeutet. Die Spezifikation geht aber noch einen Schritt weiter: Die Bits mit gleichem Inhalt werden in Gruppen zu je 8 Bit (1 Byte) zusammengefaßt. Im *Solid Run*-Format werden dann zusammenhängende Bytes mit gleichen Werten gemäß Abbildung 23.8 gespeichert:

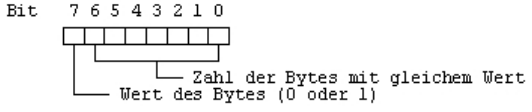

Abbildung 23.8 Das Format der Solid Run-Records

Als Basis für den Code dient immer ein Byte, dessen oberstes Bit den Wert der komprimierten Pixel (1 oder 0) eines Bytes anzeigt. Die restlichen Bits 0 bis 6 dienen als Zähler für die Menge der komprimierten Bytes. Liegt nun zum Beispiel eine Reihe von 320 Bit mit dem Wert 1 vor, sind dies genau 320/8 = 40 Byte. Analoges gilt für Bitreihen, die den Wert 0 enthalten. Damit sind folgende Codes möglich:

```
Bits = 0:01H (1 Byte bzw. 8 Bit)
...
7FH (127 Byte bzw. 1016 Bit)
Bits = 1:81H (1 Byte bzw. 8 Bit)
FFH (127 Byte bzw. 1016 Bit)
```

Der Eintrag 83H bedeutet zum Beispiel, daß die folgenden drei Bytes mit dem Wert 1 darzustellen sind. Mit dieser Methode lassen sich maximal 127 gleiche Bytes (entspricht 7FH oder FFH) zusammenfassen. Der Record ist lediglich ein Byte lang und besitzt Werte zwischen 01H und FFH. Der Wert 00H als Zähler ist nicht zulässig, da er einerseits sinnlos ist und andererseits zu Konflikten mit den Headerbytes der Formate *Bit String* (80H) und *Pattern Run* (00H) führt. Ein Wert von 80H ist in diesem Format ebenfalls nicht zulässig, da er 0 Wiederholungen eines Bytes mit dem Wert FFH definiert.

Das Bit String-Format

Dieses Format dient der Darstellung unkomprimierbarer Bildabschnitte. Es erlaubt das Speichern nicht wiederholbarer Muster als *Bitreihe*. Bit String-Records werden gemäß Abbildung 23.9 aufgebaut.

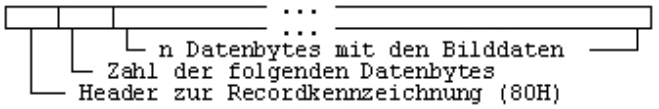

Abbildung 23.9 Format der Bit String-Records

Der Satz beginnt mit der Kennung 80H im ersten Byte. Das zweite Byte enthält einen Zähler, der die Anzahl der Folgebytes mit den Daten des Records enthält. In diesen Folgebytes sind die Pixel als Bitstrings abgespeichert, womit immer nur ein ganzzahliges Vielfaches von 8 als Pixelzahl speicherbar ist. Im *Bit String*-Format lassen sich maximal 255 Byte (2040 Bit) pro Satz darstellen, bei längeren Mustern muß ein zweiter Satz angefügt werden. GEM benutzt dieses Format nur, falls kein anderes Kodierverfahren paßt.

Das Pattern Run-Format

Eine weitere Kodierung wird für die Musterdarstellung verwendet. Solche Muster, die man zum Füllen von Flächen und Bildhintergründen benötigt, werden in der Regel von den Toolboxen der Grafikprogramme geliefert. Sie haben meist geometrische Strukturen und lassen sich mit einem Format gemäß Abbildung 23.10 darstellen.

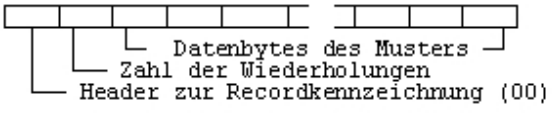

Abbildung 23.10 Format des Pattern Run-Records

Die Sätze beginnen mit dem Wert 00H als Recordkennung. Daran schließt sich ein Zählerbyte an, das die Wiederholungsrate des Musters angibt. Der Wert 5 bedeutet, daß das Muster in der betreffenden Zeile fünfmal in aufeinanderfolgenden Bits zu erzeugen ist. Der Rest des Satzes wird dann durch die Datenbytes des eigentlichen Musters belegt, wobei jedes Pixel des Musters durch ein Datenbit repräsentiert wird. Auch hier ist zu beachten, daß ein Muster aus einer durch 8 teilbaren Anzahl von Pixel bestehen muß. Nun fehlt aber noch die Information über die Länge des Musters. Wie viele Datenbytes sind durch dieses Muster belegt? GEM definiert in IMG-Dateien eine feste Datenlänge (im IMG-Header in Word 4 abgelegt) für alle angebotenen Muster, das in der Regel auf 2 Byte beschränkt ist. Ein solches Muster muß dann durch *2 * 8 = 16 Bit* dargestellt werden.

Ein aus n Byte bestehendes Muster läßt sich im *Pattern Run*-Format bis zu 255mal wiederholen, bevor ein neuer Record angelegt werden muß.

Das Vertical Replication Count-Format

Die bisher beschriebenen Kodierungsverfahren beziehen sich auf die Darstellungen einzelner Pixel in einer Reihe. Nun kommt es in Bildern aber immer wieder vor, daß komplette Zeilen identisch sind. Man denke nur an den oberen und unteren Bildrand, der häufig mit einer Farbe ausgefüllt ist: Hier reicht es, eine Zeile zu beschreiben und diese Zeile mehrfach zu kopieren. Zur Unterstützung dieser Technik wurde das *Vertical Replication Count*-Format eingeführt. Es kommt nur zur Anwendung, wenn sich mehrere aufeinanderfolgende Zeilen in der gleichen Kodierung darstellen lassen. Die Bilddaten werden nach wie vor mit einem der oben beschriebenen Formate (Pattern Run, Bit String, Solid Run) kodiert. Bei Anwendung des *Vertical Replication Count* wird vor dem Header des Datensatzes ein zweiter Header mit folgendem Aufbau kopiert:

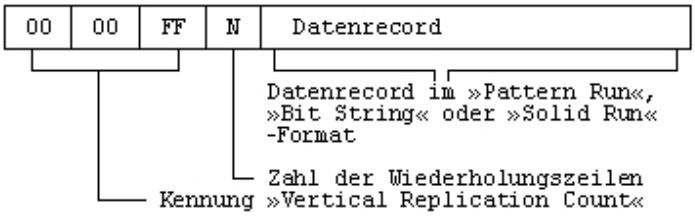

Abbildung 23.11 Vertical Replication Count-Format

Der Record beginnt immer mit einem vier Byte langen Header. Die ersten drei Bytes enthalten zur Formatmarkierung diese Signatur:

```
00 00 FF
```

Diese Kennung wird von keinem anderen Kodierverfahren benutzt und nur dann eingesetzt, wenn sich mehrere Zeilen mit *Vertical Replication Count* darstellen lassen. Das vierte Byte enthält einen Zähler mit der Anzahl der auszugebenden Zeilen. Ein Eintrag FEH veranlaßt, daß 254mal die gleiche Zeile ausgegeben wird. Es sind maximal 255 Zeilen in der gleichen Darstellung möglich. Die Kodierung der einzelnen Pixel erfolgt gemäß den oben beschriebenen Verfahren.

Bei der Erstellung eines Bildes im IMG-Format werden alle grafischen Objekte (Rechtecke, Kreise, Texte) im Pixelformat abgebildet und in den beschriebenen Formaten gespeichert. Abbildung 23.13 zeigt ein mit DR PAINT erstelltes Bild. Es enthält mehrere Objekte, die in einer Datei als Pixelgrafiken abgespeichert wurden. Auch Texte bildet GEM in einzelnen Punkten ab. Sobald ein Text gesichert wurde, lassen sich zum Beispiel die Darstellungsattribute (Größe, Stil etc.) nicht mehr ändern. Die Textausgabe G E M ergibt in der S/W-Darstellung folgendes Pixelmuster:

```
00111100  11111110  11000110 ┐
01100110  01100010  11101110 │
11000000  01100100  11111110 ┘
11001110  01111100  11111110 ┐ Pixelreihen
01100110  01100100  11010110 │
00111110  01100010  11000110 │
00000000  00000000  00000000 ┘
   G         E         M
```

Abbildung 23.12 Bitmuster des Textes GEM

Farbige Buchstaben lassen sich durch Mischen der Grundfarben (Rot, Grün, Blau und Intensität) erzeugen. In diesen Fällen teilen sich die Bits auf die vier Farbebenen auf. Durch Überlagerung der Bits ergibt sich dann eine der 16 möglichen Farben für den betreffenden Buchstaben. Bei der Bearbeitung einer IMG-Datei lassen sich die verschiedenen Recordformate leicht erkennen. Abbildung 23.13 zeigt eine solche Datei:

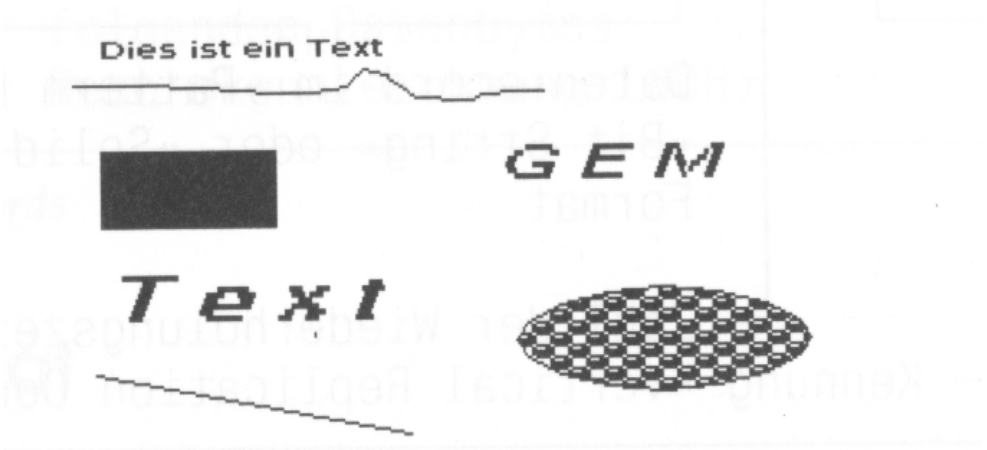

Abbildung 23.13 Darstellung des Originalbildes im PAINT-Format

Die IMG-Datei läßt sich anschließend als Hexdump ausgeben. Ein Ausschnitt aus dieser Anzeige wird in Abbildung 23.14 wiedergegeben. Bei der Konstruktion eines Bildes aus der IMG-Datei ist jeweils eine Zeile der entsprechenden Farbebene zu ermitteln; danach folgen die Daten der nächsten Farbebene. Sobald eine Bildreihe vorliegt, kann entschieden werden, ob eine Wiederholung mit *Vertical Replication Count* möglich ist.

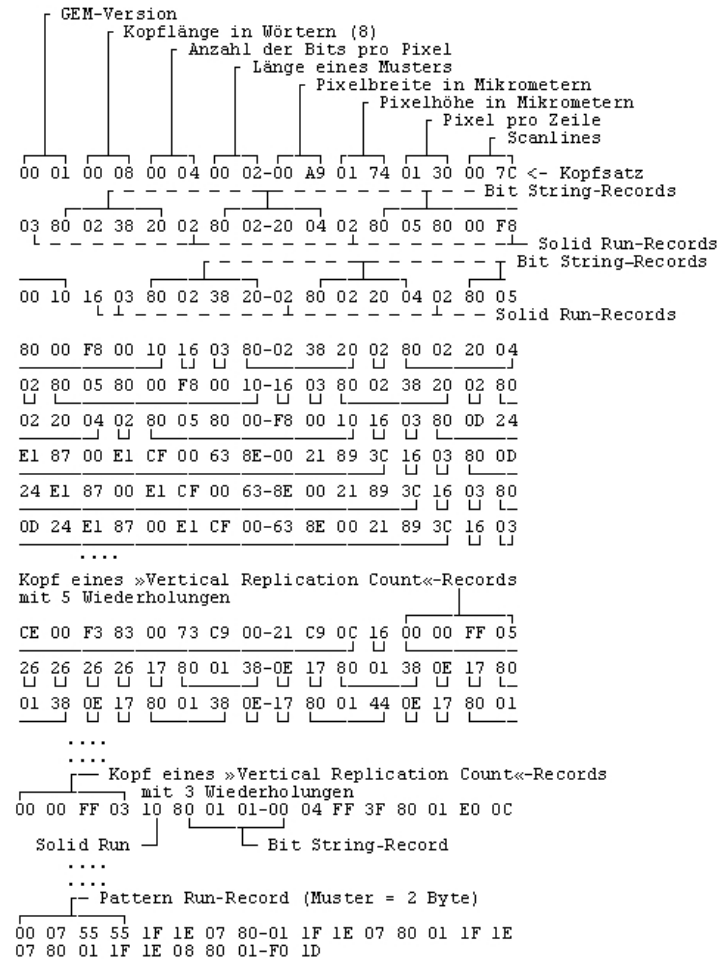

Abbildung 23.14 Hexdump einer IMG-Datei