mc
2
Die Mikrocomputer-Zeitschrift
7 DM · 60 öS · 7 sfr. · Februar 1987

Leserbefragung:
Atari 1040 ST zu gewinnen!

Systembus, EGA transparent:

PC-Technik

32 MByte zum Einstecken

1-MBit-EPROMs programmieren

Turbo-Pascal:
DOS-Exec-Funktion
Hercules-Grafik

Im Test:
GfA-Basic für Atari ST
Apple IIGS

Außerdem:
CP/M-68 k,
CP/M 3.0,
mc-68-ECB,
68008-EMUF

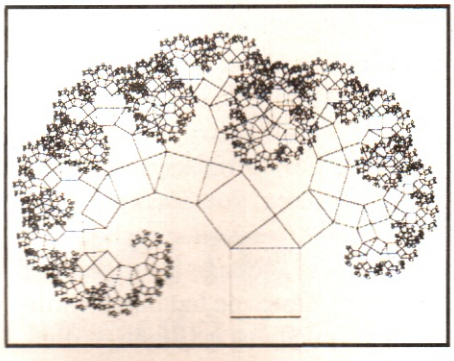
mc-inhalt

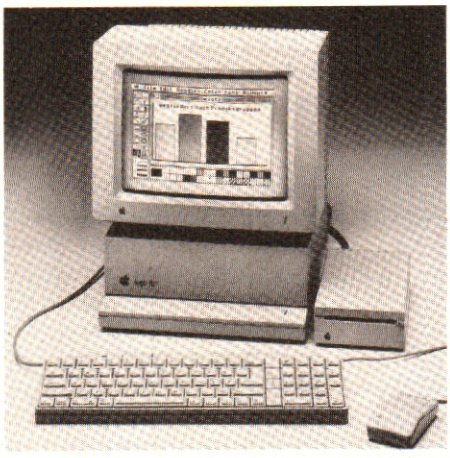
PC-Steckkarten

Ohne eine Erweiterungskarte kommt heute kaum ein PC aus. Neben Tests und Funktionsbeschreibungen von Steckkarten finden Sie einen Beitrag über den PC-Bus, damit selbstentwickelte Schaltungen auch in Ihrem PC funktionieren. Geballte Informationen ab **Seite 30**

PC-Grafik

So schön PC-Grafiken auch sind – ohne detailliertes Hintergrundwissen kann man sie nicht optimal einsetzen. Wir zeigen, wie EGA-Grafiken unverzerrt ausgedruckt und die EGA-Register richtig programmiert werden. Auch für Anwender der Herculeskarte haben wir einen Beitrag. Artikel über PC-Grafik ab **Seite 33**



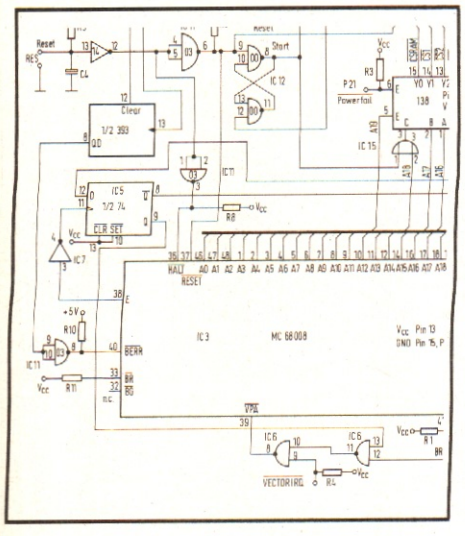


Apple-IIGS

Würde man das Heer der Apple-II-Anwender befragen, welche Fähigkeiten ein neues Modell haben sollte, würde wahrscheinlich ein anderer Computer dabei herauskommen als der jetzt erhältliche Apple-IIGS. Lesen Sie, was Apple's Neuer außer Kompatibilität zum Apple-II zu bieten hat. **Seite 92**

EMUF-08

Mit dem EMUF-08 stellen wir einen leistungsfähigen Einplatinencomputer vor, der besonders Anwender eines 68000er-Computers interessieren wird. Mit einem Arbeitsspeicher von 1 MByte beherrscht der EMUF-08 auch die komplizierteste Aufgabe schnell und zuverlässig. **Seite 101**



wird 4SEITEN gar nicht mehr durchlaufen. Einige Programme setzen sich auch an den Anfang des INT 9, lesen selbst den Tastaturkanal und durchlaufen danach ganz normal den INT 9, dem dann natürlich eine Tastaturinformation fehlt. Nun wird vielleicht auch klarer, warum die Anbieter residenter Programme empfehlen, ihr Programm immer als letztes zu laden, was bereits bei zwei Programmen sehr schwierig wird. Das zuletzt geladene Programm trifft die Interrupts immer so an, wie sie von ihm gesetzt wurden, und wird deshalb höchstwahrscheinlich laufen.
Allerdings traten bisher keine Konflikte mit anderen residenten Programmen auf, auch dann nicht, wenn 4SEITEN als erstes Programm geladen wurde. 4SEITEN ist nicht gegen mehrfaches Laden geschützt. Mehrere residente Versionen in Reihe behindern sich nicht.
Ein residentes Programm vor mehrfachem Laden zu schützen, oder mit einem zweites Laden die alten Interrupts wiederherstellen zu lassen, ist keinesfalls einfach, wenn nicht gar unmöglich, denn später geladene Programme könnten die Interrupts geändert haben.
Nun noch einige Bemerkungen zu den Erscheinungsbildern, die 4SEITEN auf dem Bildschirm hervorrufen kann.
Überhaupt keine Probleme entstehen mit Programmen, die sich nach der aktiven Seite richten, wie beispielsweise das ganze MS-DOS. Am ärgerlichsten sind Programme, die den Bildschirmmodus ändern, da das BIOS dann immer alle 16K Bildschirmspeicher normiert. Damit sind natürlich alle Seiten gelöscht. Beim Umschalten in den Grafikmodus muß diese Normierung sein, weil ja die ganzen 16K gebraucht werden. Aber wenn Programme erst den Modus erfragen um ihn dann genauso wieder zu setzen, so ist das zwar eine einfache, aber keine weitsichtige Methode, den Bildschirm zu löschen.

Direktes Schreiben in den Bildschirmspeicher

Ein weiteres Problem entsteht, wenn Programme direkt in den Bildschirmspeicher auf Adresse B8000H schreiben. Dieses Vorgehen wird gemeinhin als schlechter Programmierstil bezeichnet. Die Ausgaben solcher Programme erscheinen natürlich nur auf Seite 1. Wenn dann zusätzlich auch noch Ausgaben an den Cursor gehen, der immer eine Variable der Seite ist, kann es passieren, daß ein Teil der Ausgaben auf Seite 1 und der Rest auf der aktiven Seite steht. Mit 4SEITEN besteht also auch die Möglichkeit der Qualitätsprüfung, denn gute Programme sollten die aktuelle Seite erfragen, und Ihre Ausgaben dorthin schicken, anstatt fest auf eine nicht mehr änderbare Seite zu schreiben.
Direktes Schreiben in den Bildschirmspeicher erzeugt eine sehr schnelle Ausgabe . Wenn ein Programm diese Anforderung erfüllen muß, empfiehlt es sich, je nach aktiver Seite auf die Adresse B8000H, B9000H, BA000H oder BB000H zu schreiben; das sind die Anfangsadressen der 4 Bildschirmseiten. Diese Vorgehensweise macht ein Programm flexibler und erfordert lediglich 5 Bytes mehr Code. Anstatt beispielsweise das Register BX direkt mit dem Segment B800H zu laden, müßten dafür vier neue Befehle eingegeben werden (*Tabelle*).
Mit etwas Geduld und Glück lassen sich manchmal auf diese Weise auch fertige Programme abändern, die danach auf jeder Seite lauffähig sind.

Tabelle: Direktes Schreiben auf den Bildschirm

```
mov ah, 0fh ;Bildschirmseite erfragen
int 10h     ;wird in BH zurückgegeben
add bh,b8h  ;Offset High-Byte Bildspeicher addieren
xor bl,bl   ;Low-Byte = 00H; neues Bildschirmsegment in BX
```

Spruch des Monats

„Es gibt einen weiteren Typ von Automaten: Solche, die nicht einfache Aktionen, sondern durchdachte Handlungen eines Menschen imitieren und den Menschen sogar ersetzen können."

Leonardo Torres Y Quevedo, Computerpionier, 1914.

Startadresse von Apple-Binärdateien

Bei DOS 3.3 kann man Binärdateien mit oder ohne Angabe der Anfangsadresse laden oder starten. Während an der Adresse $AA60 die Länge der Datei steht, fehlt die Möglichkeit, die Startadresse festzustellen, die in den ersten zwei Byte der Datei notiert ist.
Um die ORG-Adresse eines abgespeicherten Programms feststellen zu können, genügt ein einfacher Patch im DOS 3.3:

```
bisher:
A371: 20 7A A4          JSR $A47A

geändert:
A371: 20 __ __          JSR PATCH

neu:
__ __: 20 7A A4 PATCH   JSR $A47A
       8D F9 B7         STA $B7F9
       8C FA B7         STY $B7FA
       60               RTS
```

Man kann diesen 10 Byte langen Patch im Bereich $BA69...$BA95 bzw. $BCDF...$BCFF unterbringen, wenn darin noch genügend Platz vorhanden ist (DOS prüfen). Für „PATCH" ist die richtige Adresse einzusetzen. Mit INIT kann diese Änderung dauerhaft auf eine neue Diskette übertragen werden.
In $B7F9...$B7FA findet man nach dem Laden bzw. Starten einer Binärdatei deren ORG-Adresse, in $AA60...$AA61 die Länge, in $AA72...$AA73 die Ladeadresse.
In Applesoft kann man die Parameter so abfragen:

```
ORGADR =
PEEK(47097)+256*PEEK(47098)
LAENGE =
PEEK(43616)+256*PEEK(43617)
LODADR =
PEEK(43634)+256*PEEK(43635)
```

Wolfdieterich Müller

Frank Wolter

Sprites für den Apple-II

Obwohl der Apple von Haus aus über sogenannte „Shapes" verfügt, soll hier eine schnellere und speicherfreundlichere Methode zur Darstellung von Grafiksymbolen vorgestellt werden.

Jeder, der schon einmal versucht hat, Shapes zu entwerfen, wird wohl bestätigen können, daß dieses Vorhaben ohne ein Programm, das die Codierung der Shapes vornimmt, so gut wie unmöglich ist oder zumindest einen ganzen Batzen Zeit in Anspruch nimmt. Zudem sind solche Shapes in Programmen, in denen es auf schnelle Grafik ankommt, oft nicht einzusetzen, da die Ausführung der Vektorkommandos auch für einen Mikroprozessor eine zeitraubende Angelegenheit ist. Hinzu kommt noch, daß die Vektortabellen der Shapes sich im Speicher schnell auf ungeahnte Größen aufblähen, wenn einige größere Objekte definiert wurden.
Eine elegantere Möglichkeit bieten Sprites, auch Blockshapes genannt. Sie bestehen im Gegensatz zu den Shapes nicht aus Kommandocodes, sondern direkt aus dem Bitmuster der Grafik, so wie es später auf dem Bildschirm ausgegeben wird. Daß es schneller geht, eine bestehende Grafik von einem Bereich des RAMs in einen anderen zu kopieren, als Vektorkommandos zu interpretieren und auszuführen, wird jedem einleuchten.

Der Aufbau der Sprites

Das Programm orientiert sich anhand eines Zeigers, wo es die Sprites im Speicher finden kann. Dieser Zeiger kann vom Benutzer bequem per Basic-Befehl definiert werden. So besteht die Möglichkeit, die Tabelle an einer beliebigen Stelle im Speicher des Rechners zu plazieren. Natürlich ist hierbei darauf zu achten, daß die Daten nicht mit reservierten Speicherbereichen (Basic-Bereich, Sprite-Routinen, DOS usw.) in Konflikt kommen.
Wie der Zeiger beeinflußt werden kann, wird später gezeigt. An der Stelle, auf die der Zeiger deutet, befinden sich wiederum zwei Bytes, die auf das nächste Sprite hinweisen. Dies soll ein schnelles Auffinden ähnlich wie bei den Basic-Zeilen ermöglichen. Das folgende Byte gibt die vom Benutzer festgelegte Nummer des Sprites an. Sie darf im Bereich von 1 bis 255 liegen und kann vom Programmierer frei gewählt werden.
Als nächstes folgen einige Informationen, auf die der Benutzer allerdings nur indirekt Einfluß hat. Es handelt sich um Angaben zur Darstellung der Sprites (*Bild 1*) die intern von den Sprite-Routinen verwaltet werden. Sie beinhalten: Die Breite in Bytes, die erforderlich ist, um alle Punkte des Objektes in der Breite zu erfassen. In der Sprite-Tabelle werden jeweils in einem Byte acht Punkte der Grafik abgelegt. Hat eine Grafik in der Breite z. B. elf Punkte, so sind zwei Bytes zur Darstellung der Punkte erforderlich; acht Punkte in dem einen Byte und drei Bits des zweiten Bytes werden benutzt, die anderen sind auf Null gesetzt.
Gefolgt wird dieses Byte von der Gesamtlänge in Bytes, die zur Darstellung des Musters erforderlich ist. Hier wird eine Einschränkung für den Benutzer gemacht. Es dürfen maximal 255 Bytes für ein Sprite verwendet werden. Diese Länge wird vom Programm aus kontrolliert, und gegebenenfalls wird der Programmierer bei Mißachtung dieser Regel gewarnt.
Abschließend folgen dann endlich die Bytes, aus denen das Sprite besteht, wobei das Objekt zeilenweise im Speicher abgelegt ist.
Mit dem Befehl „&Clear (Ausdruck)" können alle vorhandenen Sprite-Definitionen gelöscht werden, gleichzeitig wird der Zeiger auf die Sprites neu festgelegt.

```
Sprite # 1
Zeiger auf Sprites --->     1.  Zeiger auf nächstes Sprite -+
                            2.  High-Byte vom Zeiger       -+
                                                            I
                            3.  Kennummer des Sprites       I
                                                            I
                            4.  Breite in Bytes             I
                            5.  Anzahl der folgenden Bytes  I
                                                            I
                            6.  Sprite als Bitmuster        I
                                      ''                    I
                                      ''                    I
                      max. 255.  Ende des Sprites           I
                                                            I
Sprite # 2                  1.  $00                 <------+
                            2.  $00
                            3.  $00
```

Bild 1. Aufbau eines Sprite im Speicher: Drei Null-Bytes kennzeichnen das Ende der Tabelle

Die Eingabe von Sprites

Eine Definition (*Bild 2*) beginnt mit dem Kommando „&Def (Ausdruck)", wobei der Ausdruck die Nummer des Sprites angibt (1...255). Jetzt folgt das eigentliche Sprite auf eine übersichtliche und einfache Weise:

Beispiel:

```
&Def ..X..
&Def XXXXX
&Def ..X..
```

Dabei erscheint später auf dem Bildschirm für jedes X ein heller Punkt. Abschließend wird das Ende der Definition mit „&Def End" gekennzeichnet. So, und nun endlich kann das Sprite in der gewohnten Weise, wie ein Shape auf den Bildschirm gezaubert werden. Hierzu dienen die Befehle
&DRAW (Nummer) AT (X-Position), (Y-Position)
&XDRAW (Nummer) AT (X-Position), (Y-Position)
&DRAW zeichnet das Sprite über schon bestehende Grafiken, wohingegen &XDRAW das Sprite invertierend mit dem Hintergrund ausgibt, was dazu benutzt werden kann, um ein Sprite wieder vom Bildschirm zu löschen. Jeder Ausdruck, der hier in Klammern geschrieben wurde, kann eine Zahl, eine Variable oder auch ein Ausdruck sein.

Kollisionen feststellen

Manchmal ist es sinnvoll, feststellen zu können, ob das gerade gezeichnete Sprite mit einem Objekt auf dem Bildschirm (gesetzte weiße Punkte) kollidiert ist. Dies ist nach dem Draw-Kommando

durch einen einfachen Test der Speicherstelle 255, z. B. mit Koll=Peek (255) zu ermitteln. Ergibt das den Wert Null, so wurde das Sprite auf schwarzem Grund ausgegeben, ansonsten wurde ein weißer Punkt überschrieben.

Fehlermeldungen

Sprite too long wird ausgegeben, wenn der Gesamtumfang des Sprites 255 Bytes überschreitet. Hier hilft nur, das Sprite in zwei kleinere zu unterteilen, und diese dann nacheinander auf dem Bildschirm auszugeben.

Duplicate Spritenumber: Sie haben versucht, einem Sprite eine Nummer zuzuweisen, die im Speicher schon existiert. Benutzen Sie eine andere Nummer, oder führen sie ein &Clear aus, um alte Definitionen aus dem Speicher zu löschen.

Undefined Sprite erscheint, falls Sie ein Draw- bzw. XDraw-Kommando ausführen und dabei ein Sprite aufrufen, das noch nicht definiert wurde.

End without Def erhalten Sie beim Aufruf des &Def-End-Befehls, bevor eine Definition durch &Def (Nummer) eingeleitet wurde. Entfernen Sie den überflüssigen Befehl.

Missing End: Sie haben eine Definition begonnen, und ohne diese mit dem End-Befehl abzuschließen, wurde versucht, ein Sprite mittels Draw-Kommandos zu zeichnen oder ein weiteres Sprite zu definieren. Die Sprites werden nur zur Benutzung freigegeben, wenn alle Definitionen vorschriftsmäßig abgeschlossen sind. Fügen Sie den fehlenden End-Befehl ein.

Missing Spritenumber wird gemeldet, falls sofort mit der Definition eines Sprites (&Def ...X..X..X.) begonnen wurde, ohne die Startzeile, in der dem Sprite eine Nummer zugewiesen wird, ausgeführt zu haben. Weisen Sie dem Sprite zu Anfang eine Nummer zu.

```
9200- A9 4C 8D F5 03 A9 AB 8D +=045B
9208- F6 03 A9 93 8D F7 03 A0 +=045C
9210- 00 84 1E A9 60 85 1F 98 +=02E7
9218- 91 1E C8 91 1E C8 91 1E +=039D
9220- A9 29 85 73 A9 92 85 73 +=03FD
9228- 60 00 00 00 00 00 00 00 +=0060
9230- 00 00 00 A5 1E 85 CE A5 +=02BB
9238- 1F 85 CF A0 00 60 A0 00 +=0313
9240- B1 CE AA C8 B1 CE 85 CF +=05C4
9248- 86 CE D0 10 8A D0 0D 68 +=0403
9250- 68 AD 32 92 F0 03 4C 6F +=0387
9258- 95 4C 8A 95 60 20 B1 00 +=0331
9260- 20 67 DD 20 52 E7 A9 00 +=0366
9268- 8D 30 92 8D 31 92 A8 91 +=03D8
9270- 50 C8 91 50 A5 50 85 1E +=0391
9278- A5 51 85 1F 60 A2 00 8E +=032A
9280- 2D 92 8E 2E 92 86 FF 86 +=0418
9288- FA A5 30 29 7F C9 01 F0 +=0431
9290- 0C 4A 38 66 FA EE 2D 92 +=039B
9298- 6A 90 F7 46 FA 60 AA 25 +=0460
92A0- FA 8D 2F 92 8A 18 AE 2D +=03C5
92A8- 92 F0 04 0A CA D0 FC 0D +=0433
92B0- 2E 92 F0 12 AA 31 26 29 +=02EC
92B8- 7F F0 04 A9 01 85 FF 8A +=042B
92C0- 11 26 29 7F 91 26 AD 2F +=0272
92C8- 92 AE 2D 92 F0 07 E8 E8 +=04C6
92D0- 18 2A CA D0 FC 8D 2E 92 +=0425
92D8- 60 A9 51 8D C0 92 4C ED +=0472
92E0- 92 20 B1 00 20 B9 F6 60 +=0392
92E8- A9 11 8D C0 92 AD 31 92 +=0409
92F0- F0 03 4C 8F 95 20 B1 00 +=0334
92F8- 20 67 DD 20 52 E7 A5 51 +=03B3
9300- D0 0D A5 50 F0 09 8D 29 +=0381
9308- 92 20 33 92 4C 15 93 4C +=02B7
9310- 99 E1 20 3E 92 A0 02 B1 +=03BD
9318- CE CD 29 92 D0 F4 C8 B1 +=0593
9320- CE 85 FD F0 BC C8 B1 CE +=0643
9328- 85 FC A9 05 85 FB 20 B1 +=0480
9330- 00 20 B9 F6 8E 2A 92 8C +=03A5
9338- 2B 92 8D 2C 92 20 11 F4 +=032D
9340- 20 7D 92 4C 59 93 AE 2A +=033F
9348- 92 AC 2B 92 EE 2C 92 AD +=0454
9350- 2C 92 C9 C0 F0 54 20 11 +=03BC
9358- F4 A5 FD 85 FE 8C 29 92 +=0560
9360- A4 FB B1 CE AC 29 92 20 +=04A5
9368- 9E 92 E6 FB C6 FC F0 30 +=05F3
9370- C6 FE F0 17 C8 C0 28 90 +=050B
9378- E4 E6 FB C6 FC F0 2B C6 +=0668
9380- FE D0 F1 A9 00 8D 2E 92 +=04B5
9388- 4C 46 93 C8 C0 28 B0 08 +=038D
9390- A9 00 20 9E 92 4C 46 93 +=031E
9398- A9 00 8D 2E 92 4C 46 93 +=031B
93A0- A9 00 C8 C0 28 B0 03 20 +=032C
93A8- 9E 92 60 A0 00 8C 32 92 +=0380
93B0- B1 B8 C9 BD D0 03 4C 5D +=046B
93B8- 92 C9 B8 D0 03 4C D1 93 +=0496
93C0- C9 94 D0 03 4C E8 92 C9 +=04BF
93C8- 95 D0 03 4C D9 92 4C C9 +=0434
93D0- DE 20 B1 00 EE 32 92 A0 +=0401
93D8- 00 B1 B8 C9 B0 F0 22 C9 +=048D
93E0- 2E D0 03 4C 84 94 C9 58 +=0386
93E8- D0 03 4C 84 94 20 67 DD +=039B
93F0- 20 52 E7 A5 51 D0 07 A5 +=03CB
93F8- 50 F0 03 4C 32 94 4C 99 +=033A
9400- E1 A4 FB AD 31 92 D0 03 +=04C3
9408- 4C 65 95 CE 31 92 A9 00 +=0380
9410- 91 CE C8 91 CE C8 91 CE +=05AD
9418- 88 88 18 98 65 CE 48 A5 +=03E0
9420- CF 69 00 A0 01 91 CE 88 +=03C0
9428- 68 91 CE EE 30 92 20 B1 +=0448
9430- 00 60 AD 31 92 F0 03 4C +=030F
9438- 8F 95 20 33 92 EE 31 92 +=03BA
9440- AC 30 92 8C 29 92 F0 1B +=03C0
9448- A0 02 B1 CE C5 50 F0 0B +=0431
9450- 20 3E 92 A0 02 B1 CE C5 +=03D6
9458- 50 D0 03 4C 6F 95 CE 29 +=036A
9460- 92 D0 ED A0 02 A5 50 91 +=0477
9468- CE A9 00 85 66 85 EB 85 +=0457
9470- EC C8 91 CE C8 91 CE A0 +=05DA
9478- 05 84 FB 60 68 68 4C 6A +=036A
9480- 95 20 B1 00 AD 31 92 C9 +=039F
9488- 01 D0 68 A0 00 B1 B8 C9 +=040B
9490- 58 F0 05 C9 2E D0 43 18 +=036F
9498- 66 66 E6 EB A5 EB C9 07 +=04FD
94A0- D0 DF A4 FB A5 66 4A 91 +=0534
94A8- CE E6 FB F0 CF A9 00 85 +=059C
94B0- 66 85 EB A0 04 B1 CE AA +=04A3
94B8- E8 8A 91 CE 88 E6 EC A5 +=05D0
94C0- EC 91 CE 20 B1 00 A0 00 +=03BC
94C8- B1 B8 C9 2E F0 04 C9 58 +=0475
94D0- D0 03 4C 84 94 A9 00 85 +=0365
94D8- EC 60 A4 EB C0 07 F0 06 +=0498
94E0- 46 66 C8 4C DC 94 C6 B8 +=04AE
94E8- A5 B8 C9 FF D0 02 C6 B9 +=0576
94F0- 4C A2 94 A0 56 4C 76 95 +=03CF
94F8- BF C5 CE C4 A0 D7 C9 D4 +=062A
9500- C8 CF D5 D4 A0 C4 C5 C6 +=062F
9508- 00 BF D3 D0 D2 C9 D4 C5 +=0596
9510- A0 D4 CF CF A0 CC CF CE +=061B
9518- C7 00 BF C4 D5 D0 CC C9 +=0584
9520- C3 C1 D4 C5 A0 D3 D0 D2 +=0632
9528- C9 D4 C5 CE D5 CD C2 C5 +=0659
9530- D2 00 BF D5 CE C4 C5 C6 +=0583
9538- A7 C4 A0 D3 D0 D2 C9 D4 +=061D
9540- C5 00 BF CD C9 D3 D3 C9 +=0589
9548- CE C7 A0 C5 CE C4 00 BF +=054B
9550- CD C9 D3 D3 C9 CE C7 A0 +=063A
9558- D3 D0 D2 C9 D4 C5 CE D5 +=067A
9560- CD C2 C5 D2 00 A0 FF 4C +=0511
9568- 76 95 A0 10 4C 76 95 A0 +=03B2
9570- 21 A9 00 8D 31 92 20 8E +=02C8
9578- FD C8 B9 F8 94 F0 06 20 +=0520
9580- ED FD 4C 79 95 68 68 4C +=0460
9588- 2D D4 A0 39 4C 76 95 A0 +=03D1
9590- 49 4C 76 95 +=01A0
```

Bild 2. So definiert man ein Sprite. Die zugehörige Tabelle, die im Speicher angelegt wird, ist im Bild unten zu sehen

Hinweise

Wird nach dem Starten des Maschinenprogramms gleich mit der Definition von Sprites begonnen, so steht der Zeiger auf die Sprites bei $6000, also hinter der zweiten HGR-Seite.
Beachten Sie, daß folgendes möglich ist:

```
&Def 1 : &DefXXXXX
For I = 1 to 10
&Def X...X
Next
&Def XXXXX
&Def End
```

Dieser kleine Trick kann einem manchmal viel Arbeit ersparen.
Achten Sie darauf, daß in allen Definitionszeilen des Sprites die Anzahl der Punkte („.“ oder „X“) gleich ist und keine Schwankungen in der Breite des Sprites vorliegen. Ansonsten könnte es passieren, daß die Sprites nicht richtig in der Tabelle abgelegt werden.

```
Die Definition des Sprites:
100 &CLEAR 6*256*16          150 &DEF XXXXX
110 &DEF 1                   160 &DEF X...X
120 &DEF ..X..               170 &DEF END
130 &DEF .X.X.               200 HGR 2
140 &DEF X...X               210 &DRAW 1 AT 10,10

Die Tabelle im Speicher:
Adresse  Pointer  Nummer Breite Bytes Bitmuster

6000:    0A 60    01     01     05    04 0A 11 1F 11
600A:    00 00 00 (=Ende)
```

Bild 3. Das Programm ist mit BSAVE SPRITES, A$9200, L$394 abzuspeichern und vor seiner Verwendung mit BRUN SPRITES zu laden [1]

Literatur

[1] *Feichtinger, Herwig:* Prüfsummenprogramm für den Apple-II. mc 1984, Heft 6, Seite 64...65.