Der gläserne VC 20

Teil 5

In Folge 4 haben wir die Grundlagen für das Arbeiten mit selbstdefinierten Grafikzeichen besprochen. Diesmal werden wir sehen, welche Fähigkeiten der VIC in bezug auf Grafik sonst noch besitzt.

Beginnen wollen wir heute mit dem Bitmapping, also dem Arbeiten mit hochauflösender Grafik. Hochauflösend deshalb, weil die einzelnen Grafikpunkte sehr klein und das Gesamtbild dadurch sehr fein (eben hochauflösend) ist. Vorher aber noch eine Begriffserklärung: Bitmapping (engl. Map = Landkarte) bedeutet etwa soviel, wie den Bildschirm zu kartografieren, das heißt jede Bildschirmposition ist in hochauflösender Grafik über eine Koordinate erreichbar. Wer das Superexpander-Modul (VC 1211A) besitzt, der kann mittels einfacher Befehle wie PLOT, DRAW, CIRCLE etc. mit der Hires-(high resolution) Grafik arbeiten. Beim »nackten« VC 20 ist dies jedoch nicht so einfach möglich.

Das Bitmapping

Im Gegensatz zu anderen Computern unterstützt der VC 20 diese Art Grafik überhaupt nicht; man muß sich also eine Softwarelösung einfallen lassen.

Die einzige Möglichkeit, einzelne Grafikpunkte (auch Pixels genannt) — aus denen sich ja jedes Zeichen zusammensetzt — anzusprechen, haben wir in der letzten Folge kennengelernt. Ich spreche von der Möglichkeit, sich Zeichen selbst zu definieren.

Wollen wir also Bitmapping betreiben, so bleibt uns nichts anderes übrig, als den gesamten Bildschirm mit verschiedenen Sonderzeichen vollzuschreiben und diese dann umzudefinieren, so daß ein komplettes, neues Bild in Hires-Grafik entsteht.

Doch mit diesem Vorhaben stößt man bereits auf erste Schwierigkeiten, denn der VC 20 kann ja nur 256 verschiedene Zeichen auf einmal auf dem Bildschirm darstellen. Daher müßte man sich mit einer relativ kleinen Fläche für die hochauflösenden Pixels zufrieden geben.

Da die Anzahl der darstellbaren Zeichen (diese setzen sich — um dies noch einmal zu wiederholen — aus 8 x 8 Pixels zusammen) auf 256 beschränkt ist, muß man sich eine andere Lösung einfallen lassen. Diese ist aber — man wird es kaum glauben — bereits in den VIC eingebaut worden. Das sieht praktisch so aus: Man vergrößert die im Bildschirmspeicher abgelegten Zeichen von 8 x 8 auf 16 x 8 Pixels (bei gleicher Auflösung), wodurch sich gleichzeitig die zur Verfügung stehende Zeichenfläche erhöht. Diese Vergrößerung wird über ein bestimmtes Bit im VIC-Kontrollregister #3 eingestellt (vergleiche Folge 4, Tabelle 3). Ist Bit 0 dieser Speicherstelle auf 0, so bleibt alles wie es war, das heißt jedes Zeichen wird innerhalb einer 8 x 8-Matrix dargestellt.

Setzt man dieses Bit nun aber mit »POKE 36867, PEEK (36867) OR 1« auf 1, so sind alle Zeichen plötzlich doppelt so hoch. Ein Charakter wird nämlich innerhalb eines 16 x 8-Gitters abgebildet (Bild 1). Das ist nun alles schön und gut, einen Nachteil hat dieser Betriebszustand aber (wer es selbst ausprobiert hat wird es sicherlich schon bemerkt haben). Denn mit dem normalen Zeichensatz kommt auf dem Bildschirm keine vernünftige Zeichenfolge mehr zustande. So wird beispielsweise aus dem @ (Klammeraffe) das Zeichen

@
A

aus dem A das Zeichen

B
C

und so fort. Drückt man nun eine Taste, so wird nicht der entsprechende Buchstabe abgebildet, sondern irgendwelche anderen Zeichen, die, wie eben beschrieben, übereinander gestapelt sind.

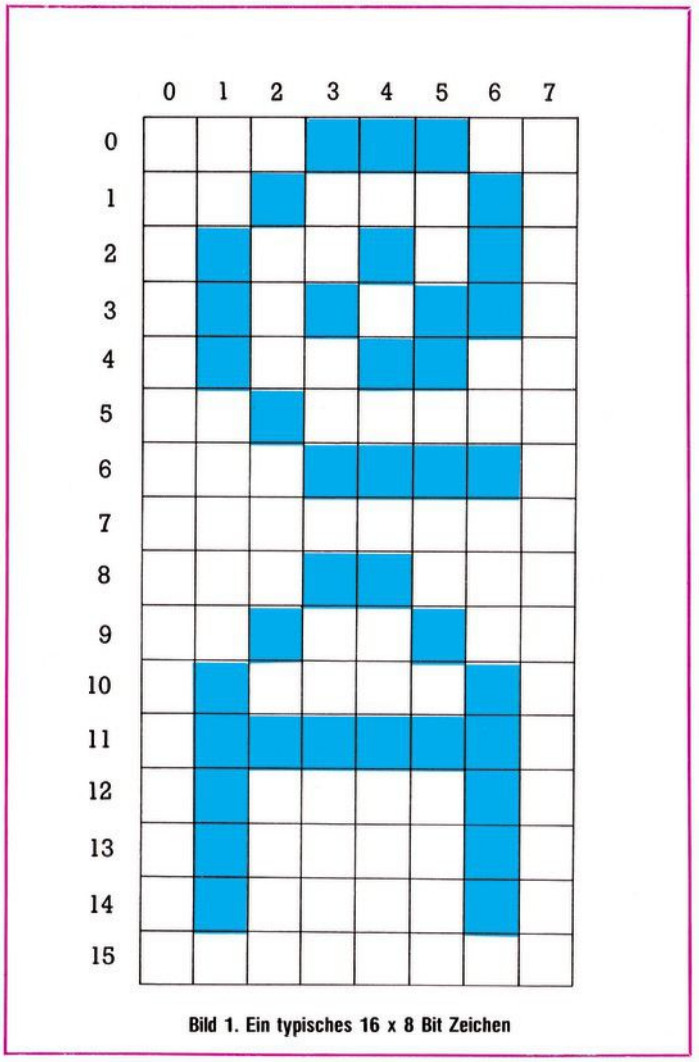

Bild 1. Ein typisches 16 x 8 Bit Zeichen

Doppelt hohe Zeichendarstellung

Die Erklärung dafür ist im Prinzip ganz einfach. Wie wir das letzte Mal gesehen haben, errechnet sich der VIC die relative Adresse eines Charakters im Zeichengenerator (relativ deshalb, weil die Adreßangaben auf eine Anfangsadresse bezogen sind), indem er den Bildschirmcode (auch hier erinnern wir uns daran, daß der Bildschirmcode die Reihenfolge der Zeichen im Charaktergenerator ist) jeweils mit 8 multipliziert.

Durch die Umschaltung auf 16-zeilige Zeichen liest der VIC für einen Charakter 16 Zeilen aus dem ROM. Daher werden — da das Zeichengenerator-ROM auf achtzeilige Zeichen ausgelegt ist — eben die Informationen von ursprünglich zwei Zeichen in einem dargestellt. Folglich multipliziert der VIC bei der Adreßermittlung den Bildschirmcode nicht mehr mit 8, sondern mit 16.

Auf diese Weise erklärt sich auch der Zahlensalat in diesem Betriebsmodus. Diese Darstellungsart hat jedoch den erheblichen Vorteil, daß jetzt mehr Zeichen als normalerweise abgebildet werden können. Es sind nämlich bereits in 128 Zeichen die Informationen von ursprünglich 256 Zeichen enthalten. Bei dem 129. Zeichen (RVS ON und @) beginnt daher schon der

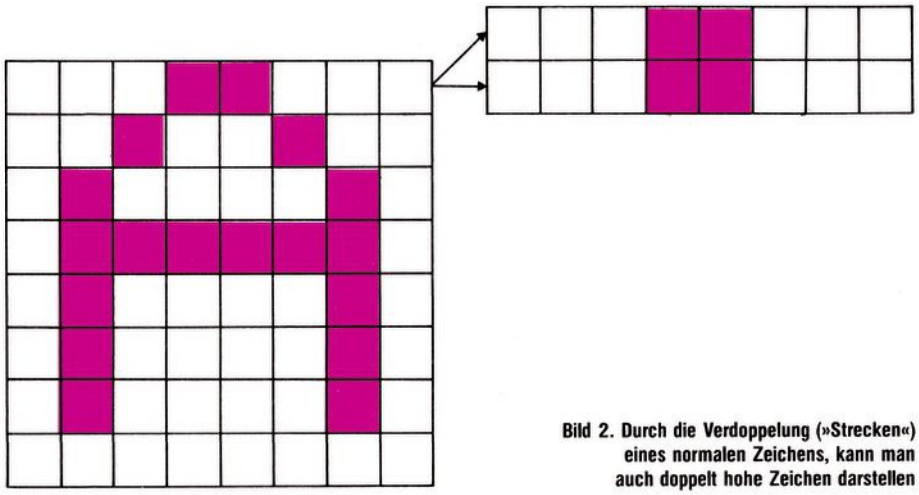

Bild 2. Durch die Verdoppelung (»Strecken«) eines normalen Zeichens, kann man auch doppelt hohe Zeichen darstellen

zweite Zeichensatz des VC. Somit lassen sich diese zwei erstmalig gemeinsam auf dem Bildschirm ausdrucken. Statt bisher 256 »normaler« Zeichen läßt sich nun die gleiche Menge in doppelter Höhe (das entspricht also 512 regulären Zeichen) abbilden. Damit sind wir jetzt in der Lage, den gesamten Bildschirm füllen zu können. Darauf gehen wir aber später noch ausführlicher ein.

Das Naheliegendste wäre es ja nun, den Zeichengenerator ins RAM zu verlegen und dabei die Zeichen so umzubauen, daß man sie wieder richtig lesen kann. Dazu muß beim Kopieren jede Zeile verdoppelt werden; sonst bleibt ja alles beim Alten.

Das Programm in Listing 1 enthält zwei Zähler. Der eine zählt die ROM-, der andere die RAM-Speicherplätze, wobei letzterer genau doppelt so schnell läuft, denn eine ROM-Zeile soll ja zweimal hintereinander ins RAM geschrieben werden (dieses »Strecken« eines Zeichens ist in Bild 2 zu sehen). Da diese Schriftart natürlich besonders auffällig ist, eignet sie sich beispielsweise für Schaufensterwerbung oder ähnliches.

Wir wollen uns aber nun wieder unserem eigentlichen Thema — dem Bitmapping — zuwenden. Wie so oft stellt sich auch hier wieder die Frage nach dem Speicherplatz. Denn egal mit welcher Speicherausbauversion man gerade arbeitet, immer kommt es zu Kollisionen zwischen dem Zeichen- und dem Bildschirmspeicher. Daher müssen wir hier die Anzahl der verfügbaren Zeichen dementsprechend reduzieren. Wenn wir möglichst den ganzen Bildschirm füllen wollen, so muß es immer zu einem Kompromiß zwischen Bildschirmgröße und Speicherplatz kommen — logisch, denn je größer die verfügbare Hires-Fläche sein soll, um so mehr Speicherplatz benötigt man für die Sonderzeichen, über die das Bitmapping abgewickelt wird.

Ich glaube, mit 189 je 16 x 8 Bit-Zeichen einen solchen Kompromiß gefunden zu haben. Anhand von Listing 2, das in mehrere Teile gegliedert ist, möchte ich das Verfahren beim Bitmapping erklären. Programmteil 2 beschreibt den Bildschirm mit den veranschlagten 189 Zeichen. Vorher wird der Rahmen noch entsprechend der etwas »krummen« Zeichenzahl angepaßt. Denn da eben nicht der gesamte Bildschirm genutzt werden kann, wird der leere Rest einfach abgeschnitten, was durch Verkleinerung der Bildschirmfläche geschieht. Wenn man die Zeilen 240 — 260 aus dem Programm herausläßt, sieht man das ganz deutlich. In Teil 3 des Listings wird dann schließlich der gesamte Zeichengenerator von Adresse 5120 bis 8192 gelöscht, damit der Bildschirm auch wirklich restlos leer ist.

Damit sind alle Vorbereitungen getroffen, die wir vor dem eigentlichen Plotten durchführen müssen. Nun geht es darum, die eingegebenen Koordinaten aus den Variablen X und Y so umzuwandeln, daß die entsprechende Zeichenzeile im Charaktergenerator verändert werden kann. Als erstes verschaffen wir uns einen Überblick, wie die Zeichen auf dem Bildschirm angeordnet sind (Bild 3).

Durch die Verkleinerung der Fläche ergeben sich bei 189 verteilten Zeichen 21 Spalten und 9 Zeilen. Eine Spalte ist immer noch 8 Bit breit, denn sie rührt ja von der Zeichenbreite her. Daher ergibt sich durch Multiplikation eine Gesamtbreite von 168 Pixels. Analog verhält es sich mit der Zeilenzahl: 9 Zeilen à 16 Zeichenzeilen ergibt 144 als maximale Y-Koordinate. Übrigens hat das Koordinatensystem seinen Ursprung (X=0/ Y=0) nicht — wie in der Mathematik — links unten, sondern links oben.

Die Koordinaten müssen aus programmtechnischen Gründen in zwei Teile aufgespalten werden; nämlich in den sogenannten Grob- (oder auch Zeichen-) anteil und in den Feinanteil (auch Pixelposition genannt).

Die Koordinatenumrechnung

Der Grobanteil ist nötig, damit zunächst einmal die Anfangsadresse eines Zeichens im Zeichengenerator festgestellt werden kann. Mit Hilfe des Feinanteils adressiert man dann die benötigte Zeichenzeile und in dieser dann die Pixelposition (aber dazu später mehr).

Anhand eines konkreten Beispiels wollen wir den zur Berechnung nötigen Algorithmus entwickeln: Der Punkt mit den Koordinaten X= 43 und Y= 106 soll auf dem Bildschirm gesetzt werden.

Nun wird als erstes festgestellt, in welchem Zeichen eine Änderung vorgenommen werden muß. Zu diesem Zweck wird die Koordinate in die besagten Grobanteile aufgespalten, was durch einfache Division geschieht. Die Spaltenkoordinate wird durch 8 (Zeichenbreite), der Zeilenanteil wird durch 16 (Zeilenhöhe) dividiert:

X: 43 ÷ 8 = 5 Rest 3
Y: 106 ÷ 16 = 6 Rest 10

Das Ergebnis ist jeweils der Grobanteil, der Rest ist dann automatisch die Pixelposition (= Feinanteil). Der erste Teil dieser Rechnung wird in Listing 2 in den Zeilen 640 und 650 durchgeführt.

Danach wird die relative Position eines Zeichens im Charaktergenerator ermittelt. Die dafür nötigen »Formeln« haben wir ja bereits das letzte Mal besprochen:

Position = ZY x Zeichen pro Zeile + ZX
= 6 x 21 + 5 = 131.

Das betreffende Zeichen hat also den Bildschirmcode 131. Da ein Charakter (nach der Umschaltung auf eine 16 x 8 Matrix) den Platzbedarf von 16 Byte hat, kann man auch ganz leicht die Anfangsadresse der ersten Zeichenzeile errechnen:

ADRESSE = Position x Platzbedarf + Basisadresse = 131 x 16 + 5120 = 7216

Dann ermitteln wir als nächstes über den Pixelanteil der Y-Koordinate die benötigte Zeichenzeile. In unserem Beispiel muß zur Zeichenadresse der Wert von PY — also 10 — dazuaddiert werden. Damit haben wir die endgültige Adresse der an-

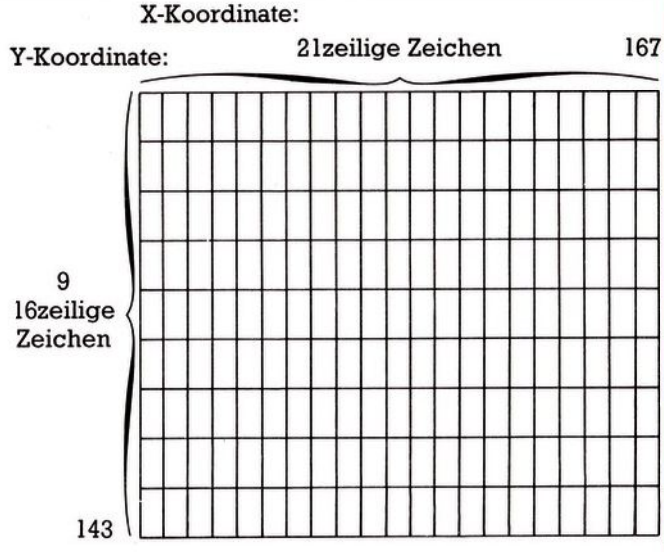

Bild 3. Durch das Beschreiben des Bildschirms mit 189 aufeinanderfolgenden Zeichen kann das Bitmapping realisiert werden

```
100 REM ********************************  <009>
110 REM ***                            ***  <249>
120 REM ***  DOPPELT HOHE ZEICHEN-     ***  <114>
130 REM ***  DARSTELLUNG.              ***  <128>
140 REM ***                            ***  <023>
150 REM ***   A C H T U N G  ! !       ***  <109>
160 REM ***  BEI 8 KBYTE ERWEITER.     ***  <071>
170 REM ***    VOR  DEM LADEN          ***  <102>
180 REM ***   >> POKE  44,32 <<        ***  <091>
190 REM ***     EINGEBEN !!!!          ***  <010>
200 REM ********************************  <109>
210 POKE 55,0:POKE 56,24:CLR             <211>
220 REM *** ZEILE 210 ENTFAELLT BEI      <100>
230 REM *** DER 8 KBYTE ERWEITERUNG.     <004>
240 AW=6144:EW=7147:G=14                 <252>
250 REM *** BEI 8 KBYTE ERWEITERUNG      <223>
260 REM *** ZEILE 230 FOLGENDERMASSEN    <125>
270 REM *** AENDERN :                    <083>
280 REM *** AW=5120:EW=8192:G=13         <204>
290 Z1=32768                             <106>
300 FOR Z2=AW TO EW STEP 2               <159>
310 WE=PEEK(Z1)                          <035>
320 POKE Z2,WE:POKE Z2+1,WE              <044>
330 Z1=Z1+1:NEXT                         <170>
340 POKE 36869,PEEK(36869)OR G           <066>
350 POKE 36865,21                        <145>
360 POKE 36867,33                        <160>
```

Listing 1. Doppelt hohe Zeichendarstellung

gewählten Zeichenzeile berechnet. Dies klingt alles viel komplizierter, als es in Wirklichkeit ist, denn alle drei Schritte können zu einem (Zeile 690) zusammengefaßt werden.

Damit sind wir schon fast am Ziel (das kann man hier sogar wörtlich nehmen) angelangt. Als letztes muß das durch die Pixel-X-Koordinate vorgegebene Bit in der Zeichenzeile gesetzt werden. Dies wird durch eine ODER-Verknüpfung des Wertes mit der Zeichenzeile erreicht.

Vorher ist aber noch eine letzte Hürde zu überwinden, die Koordinate ist nämlich nicht Byte-identisch. Was bedeutet das? Nun, unser Bildschirm entspricht beim Bitmapping ja auch einem Koordinatensystem. Denn über die horizontale (X-) und die vertikale (Y-) Koordinate läßt sich jeder beliebige Pixelpunkt durch ein Zahlenpaar (eben durch den X- und Y-Wert) eindeutig adressieren. Dabei hat der Punkt links oben die Koordinaten (0/0), der rechts unten die Koordinaten (167/143).

Für die X-Achse bedeutet dies, daß der Wert nach rechts ansteigt. Für die Fein-X-Koordinate gilt natürlich das Gleiche; sie kann von links nach rechts folgende Positionen annehmen: 0,1,2,3,4,5,6,7.

Hier liegt nun der springende Punkt. Position 0 entspricht nämlich Bit 7 in der Zeichenzeile, Position 1 Bit 6, die Position 2 Bit 5 und so fort. Diese Bits laufen also genau in entgegengesetzter Richtung. Folglich muß die Pixel-X-Koordinate dementsprechend »umgepolt« werden. Dies wird ganz einfach dadurch erreicht, indem man diesen Wert von der 7 subtrahiert:

Bitformat = 7 — (Pixel-X-Koordinate)

In unserem Beispiel ergibt sich:

Bitposition = 7 — 3 = 4

Bit 4 soll nun in der Zeichenzeile gesetzt werden. Dies erreichen wir — wie bereits erwähnt — durch die ODER-Verknüpfung des Wertes mit dem Zeichenbyte. Wer den letzten Teil aufmerksam verfolgt hat, dem wird dies nicht schwerfallen. Da Bit 4 die Wertigkeit 16 (= 2^4) hat, wird die Zeichenzeile eben mit 16 ODER-verknüpft.

Wenn man alle drei Schritte zusammenfaßt, ergibt sich folgende Zeile:

POKE AD, PEEK (AD) OR 2 ↑ (7-PX).

Damit ist der entsprechende Pixelpunkt gesetzt. Mit diesem letzten Teilstück haben wir nun endlich den kompletten Routinenteil beieinander, um ihn als Unterprogramm in Teil 4 von Listing 2 zu verwenden. Natürlich können mit Hilfe dieser Methode auch Punkte gelöscht oder abgefragt werden. Das Löschen wird mit Hilfe der AND-Operation bewerkstelligt. Das zu löschende Bit muß im Operanden auf Null, alle anderen, die unberührt bleiben sollen, auf eins gesetzt werden. Folgende Zeile löscht den adressierten Pixelpunkt:

POKE AD, PEEK (AD) AND 255—2↑ (7-PX).

Die Abfrage von Punkten wird ebenfalls über die AND-Operation abgewickelt. Das gewünschte Bit wird im Operanden gesetzt und danach mit der Speicherstelle UND-verknüpft. Ist das adressierte Pixel gesetzt, so ist die IF-THEN-Bedingung erfüllt, ansonsten nicht:

IF 2↑ (7-PX) = (PEEK(AD) AND 2↑ (7-PX)) THEN...

Natürlich sollte man anstelle des Ausdrucks 2↑ (7-PX) eine Variable definieren, damit das Programm kürzer und schneller wird.

Zeichnen auf dem Bildschirm — der Joypainter

Soweit also die Erklärung des Bitmapping beim VC 20. Um einmal zu zeigen, was man mit diesen Erkenntnissen anfangen kann, habe ich ein Joypainter-Programm in Maschinensprache entwickelt. Der Basic-Lader (Listing 3 und 4) transferiert das Programm aus den DATA-Zeilen automatisch in die Speicherbereiche ab $2000 und nur dort ist es lauffähig (Der Speicher muß also um mindestens 8 KByte erweitert sein). Beide Listings müssen nacheinander geladen und gestartet werden. Zunächst zur Bedienung der Routine, die mit »SYS 9682« gestartet wird.

Der Joystickpainter arbeitet — wie der Name bereits sagt — mit dem Joystick. Der kleine »Zeichencursor« kann mit dem Steuerknüppel in alle vier Himmelsrichtungen und in alle Diagonalen bewegt werden.

Es ist aber auch möglich, den Cursor über die Tasten E, S, D und X zu bewegen (ganz nach Belieben). Auch die Cursortasten können für die Steuerung herangezogen werden, über die Tastatur hat man allerdings nur vier Bewegungsrichtungen für den Zeichencursor zur Verfügung. Die Tasten verwendet man sinnvollerweise dann, wenn es darum geht, besonders exakt zu zeichnen. Aus diesem Grund kann man auch die Bewegungsgeschwindigkeit über die Funktionstasten F5 und F7 auf schnell beziehungsweise langsam stellen.

Gezeichnet wird mit den Funktionstasten F1 und F3. Die obere setzt einen Punkt (und rückt den Cursor um eins nach rechts), die andere löscht einen Punkt (diese Tasten haben eine Wiederholungsfunktion). Auch der Feuerknopf kann zum Zeichnen verwendet werden. Ein kurzer Druck bewirkt das Setzen, ein langer das Löschen eines Pixels. Über die CTRL-Taste können darüber hinaus noch andere Funktionen wie Dauerzeichnen oder Dauerlöschen angewählt werden. Tabelle 1 zeigt den kompletten Befehlsvorrat.

Natürlich fehlen diesem Programm — da es nicht allzu lang ist — einige Funktionen, die es noch komfortabler machen würden, wie beispielsweise die Verschiebung eines Zeichenblocks auf dem Bildschirm. Solch ein Programm würde aber samt Erklärung den Rahmen dieses Kurses sprengen.

Diejenigen, die einige Routinen wie das Bitmapping-Unterprogramm oder ähnliches in ihren eigenen Programmen verwenden möchten, finden in Tabelle 2 eine Auflistung der wichtigsten Programmteile.

Das Unterprogramm zur Joystickabfrage wird übrigens in der nächsten Folge ausführlicher besprochen. Einen bedeutenden Nachteil hat die Hires-Grafik aber, sie ist nämlich ziemlich farblos. Warum ist dies so?

Da sich diese Grafikart aus den 8 x 8- (oder 16 x 8-) Basiszei-

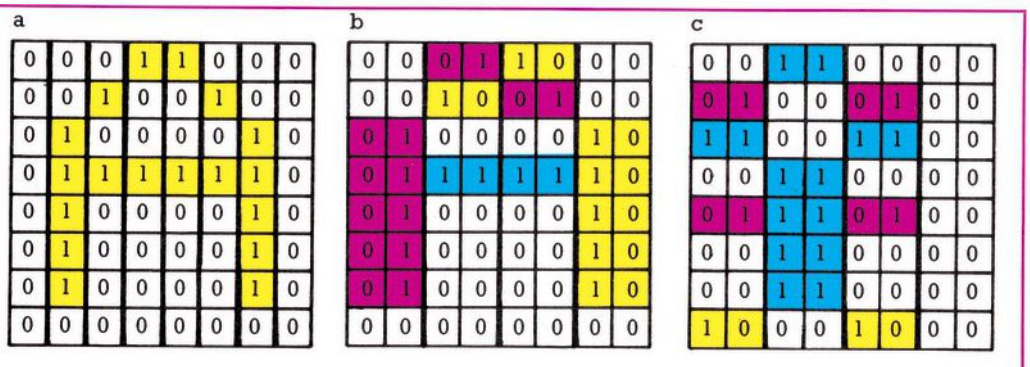

Bild 4. Das Hires-Zeichen »A« (4a) wird im Multicolormodus (4b) zu einem mehrfarbigen Gebilde. Der Vielfarbmodus eignet sich aber besonders für Spielfiguren (4c) oder ähnliches.

chen zusammensetzt, kann man — wie es auch bei »normalen« Buchstaben und Grafikzeichen üblich ist — jeder Videospeicherposition über das Farb-RAM jeweils nur eine Farbe (von 0 bis 7) zuweisen.

Die Darstellung von verschiedenen Farben innerhalb eines Zeichens ist also nicht möglich. Für manche Zwecke benötigt man aber nun gerade mehrere Farben innerhalb einer Zeichenmatrix. In diesem Fall wird dann ganz einfach auf den Multicolormodus — den wir im folgenden genauer besprechen werden — umgeschaltet.

Buntes Allerlei: Multicolor

Ist innerhalb eines 8 x 8 Zeichengitters irgendwo ein Bit mit dem Wert 1, so wird die entsprechende Bildschirmzelle mit dem Farbwert, der in der zugehörigen Farbspeicherstelle steht, ausgefüllt. Befindet sich in dieser Pixelzelle der Binärwert 0, so wird diese in der Hintergrundfarbe auf dem Bildschirm abgebildet (sie ist also nicht zu sehen).

In einem normalen hochauflösenden Zeichen können also zwei verschiedene Farben dargestellt werden:
Bit auf 0: Pixeldarstellung in Hintergrundfarbe
Bit auf 1: Pixeldarstellung in Vordergrundfarbe (Farb-RAM)

Wie Sie sehen, kann man die zwei Zustände auch als Farbcodes interpretieren. Ebenso verhält es sich im Multicolormodus: Hier werden nun zwei Bit in einer Zeichenzeile zusammengefaßt, die ebenfalls einen solchen Farbcode (allerdings für vier Farben) bilden. Ein Pixel ist in diesem Fall dann aber auch doppelt so breit wie normal, wodurch sich auch die Auflösung um die Hälfte reduziert. Bild 4 verdeutlicht den Unterschied zwischen diesen beiden Grafikarten. Diese vier möglichen Farbcodes sind auch hierbei wiederum nur die Zeiger auf Speicherstellen, in denen die eigentlichen Farbinformationen enthalten sind. Hier die möglichen Kombinationen:

00 : Hintergrundfarbe
01 : Rahmenfarbe
10 : Zeichenfarbe
11 : Hilfsfarbe

Die Werte 00 und 01, also Hintergrund- und Rahmenfarbe zeigen auf das VIC-Register # 15 (36879), in dem ja die Informationen über Rahmen- und Hintergrundfarbe abgelegt sind. Diese Farben sind also für alle Multicolorzeichen einheitlich vorgewählt. Auch der Code 11 — die Hilfsfarbe — wird einheitlich vorgewählt. Dieser Farbwert wird allerdings in einem Register gespeichert, das eigentlich mit Grafik überhaupt nichts zu tun hat, nämlich dem VIC-Register # 14 (36878), mit dem ja auch die Lautstärke festgelegt wird. Die Bits 0 bis 3 enthalten also den Lautstärkewert, die oberen vier Bit die Hilfsfarbe. Diese muß demnach speziell für Multicolor eingestellt werden, denn normalerweise ist diese Speicherstelle auf Null gesetzt (damit die Lautstärke ebenfalls null ist).

Schließlich ist noch Farbcode 10 zu erwähnen. Dieser wird für jedes Zeichen einzeln in der entsprechenden Farbspeicherstelle — wie wir es ja von den Hires-Zeichen her kennen — bestimmt. Die Bits 0 bis 2 speichern die gewünschte Farbe, Bit 3 hat eine besondere Funktion: Über dieses Bit wird — und das ist neu — der Grafikmodus ausgewählt. Ist es auf 0 (was ja der Normalfall ist), so stellt der Computer den in der dazugehörigen Videospeicherposition abgelegten Charakter im Zweifarbmodus (also in Hires-Grafik) dar. Setzt man das Bit 3 nun aber auf 1, so wird der Vierfarbmodus für dieses Zeichen eingeschaltet.

Folglich kann also die Darstellungsart für jedes Zeichen im Farb-RAM selektiert werden. Daher kann also sowohl der eine, als auch der andere Grafiktyp gleichzeitig auf dem Bildschirm abgebildet werden.

Probieren Sie es gleich einmal aus: die 1, also das Zeichen A, wird in die Bildschirmspeicherstelle 7680 (4096) gePOKEt. Danach ist in der entsprechenden Farbspeicherstelle der Colorwert mit POKE 38400,6 (beziehungsweise 37888,6) zu vermerken. In der linken oberen Ecke steht also nun ein blaues »A«. Als nächstes wird Bit 3 im Farb-RAM für diese Zeichen gesetzt:
POKE 38400, PEEK (38400) OR 8.

Nun verschwimmen die Konturen des Charakters und ein farbiges Gebilde erscheint, das nur noch entfernt an den Buchstaben A erinnert. Nun, wie Sie schon bei den doppelt hohen Zeichen gesehen haben, ist das Zeichengenerator-ROM für eine bestimmte Betriebsart konzipiert. Natürlich werden die Zeichen deshalb in hochauflösender Grafik dargestellt, damit sie besser unterscheidbar sind. Da der Zeichengenerator wiederum nur für Hires-Zeichen konzipiert wurde, kann man in Multicolor nicht viel mit ihnen anfangen.

Wegen ihrer geringen Auflösung eignet sich diese Betriebsart sowieso nur für Spielfiguren oder ähnliche Zeichen. Dazu muß der Zeichengenerator wieder ins RAM verlegt werden, was ja durch eine Änderung im Register #5 geschieht. Dann können die Zeichen wie in Bild 4 programmiert werden. In Listing 5 ist eine solche Routine zu finden. Sie erzeugt ein kleines Multicolor-Männchen (der Klammeraffe @ wird entsprechend umdefiniert).

Soweit unser etwas ausgedehnter Exkurs in die Welt der VC 20-Grafik, mit dem ich diese Folge beschließen möchte. In der nächsten Folge unseres Kurses möchte ich noch einmal auf das Betriebssystem und dessen Routinen eingehen, denn auch hier tut sich dem Programmierer ein weites Betätigungsfeld auf.

(Christoph Sauer/ev)

Taste	Funktion
F1	Punkt setzen
F3	Punkt löschen
F5	Cursor auf schnelle Bewegung umschalten
F7	Cursor auf langsame Bewegung umschalten
E S D X	Steuertasten für den Zeichencursor
CTRL C	Bildschirm löschen
CTRL F	Farbe des Bildschirmes mit den Tasten + und — ändern. Mit RETURN beendet man die Farbeinstellung
CTRL L	LOAD von Band oder Floppy
CTRL S	SAVE von Band oder Floppy
CTRL F1	Zeichenmodus: Der Joystick (oder die Tasten) werden zu einem Zeichenstift, das heißt, bei jeder Bewegung wird ein Punkt gesetzt.
CTRL F3	Radiermodus: Joypainter wird auf Dauerlöschen umgeschaltet.
SHIFT	»hebt« den Zeichenstift, beziehungsweise den Radiergummi. Solange man diese Taste drückt, kann man den Cursor bewegen, ohne daß das Bild verändert wird.
RUN/STOP	Hebt den Zeichen- beziehungsweise Radiermodus ganz auf

Tabelle 1. Die Steuerfunktionen beim Joypainter-Programm

Anfangsadresse	Funktion
$2000	Cursortasten abfragen
$2047	Joystickabfrage
$2084	Startbild (64'er....)
$21B2	Grafik (Bitmapping) einschalten
$21DC	Grafik löschen
$21F5	Koordinaten-Umrechnung
$225F	Punkt setzen
$226B	Punkt löschen
$2279	Punkt abfragen
$22D9	Feuerknopf Verarbeitung (kurzer Druck, langer Druck)
$233D	Joypaintroutine
$2410	Kontrolltastenabfrage und -verarbeitung
$25D2	Startadresse

Tabelle 2. Die wichtigsten Unterroutinen im Joypainter-Programm

```
100 REM ******************************    <009>
110 REM *** VC 20 BITMAPPINGROUTINE ***    <005>
120 REM *** FUER SPEICHER  >8 KBYTE ***    <125>
130 REM ***                         ***    <013>
140 REM ***     A C H T U N G !     ***    <066>
150 REM ***     VOR  DEM  LADEN     ***    <082>
160 REM ***    >> POKE  44,32 <<    ***    <071>
170 REM ***      EINGEBEN !!!!      ***    <246>
180 REM ******************************    <089>
190 REM                                   <077>
200 REM                                   <087>
210 REM ---------- TEIL 1 ----------       <203>
220 REM                                   <107>
230 REM *** BILDSCHIRM EINRICHTEN         <189>
240 POKE 36864,13  : REM LINKER RAND      <213>
250 POKE 36865,44  : REM OBERER RAND      <222>
260 POKE 36866,21  : REM 21 SPALTEN       <123>
270 POKE 36867,19  : REM 18 ZEILEN (16*8) <093>
280 POKE 36869,205 : REM ZEICHEN AB 5120  <157>
290 REM                                   <178>
300 REM ---------- TEIL 2 ----------       <039>
310 REM                                   <198>
320 REM *** 189 ZEICHEN IN DEN BILD-      <172>
330 REM *** SCHIRMSPEICHER POKEN.         <028>
340 FOR T=0 TO 188:POKE 4096+T,T          <115>
350 POKE 37888+T,6:NEXT                   <036>
360 REM                                   <248>
370 REM ---------- TEIL 3 ----------       <110>
380 REM                                   <012>
390 REM *** HIRESSCREEN LOESCHEN          <032>
400 FOR T=0 TO 3024:POKE 5120+T,0:NEXT    <100>
410 REM                                   <042>
420 REM ---------- TEIL 4 ----------       <161>
430 REM                                   <062>
440 REM                                   <072>
450 REM *** HIER  WURDE  ALS  DEMON-      <255>
460 REM *** STRATIONSBEISPIEL   EINE      <015>
470 REM *** SIN-FUNKTION EINGESETZT.      <153>
480 FOR X=0 TO 167                        <222>
490 Y=(SIN(X*2*π/167)+1)*71.7             <215>
500 GOSUB 640                             <028>
510 NEXT                                  <129>
520 GOTO 520                              <042>
530 REM                                   <163>
540 REM ---------- TEIL 5 ----------       <027>
550 REM                                   <183>
560 REM ***     BITMAPPINGROUTINE         <080>
570 REM ***     ================          <086>
580 REM *** DIE HOCHAUFLOESENDEN PIXELS   <149>
590 REM *** KOENNEN  UEBER  KOORDINATEN   <028>
600 REM *** ANGESPROCHEN WERDEN.          <215>
610 REM *** DIE X-KOORDINATE MUSS <168,   <007>
620 REM *** Y MUSS  <144 SEIN.            <078>
630 REM                                   <007>
640 ZX=INT(X/8) : PX=X-ZX*8               <086>
650 ZY=INT(Y/16): PY=Y-ZY*16              <195>
660 REM *** DIE KOORDINATEN  WERDEN IN    <015>
670 REM *** EINEN ZEICHEN-  UND  EINEN    <165>
680 REM *** PIXELANTEIL AUFGESPALTEN.     <163>
690 AD=5120 +ZX*16 +ZY*336 +PY            <026>
700 REM *** AD  IST  DIE ZEILENADRESSE    <224>
710 REM *** EINES  CHARAKTERS  IM ZEI-    <092>
720 REM *** CHENGENERARAOR RAM.           <176>
730 POKE AD,PEEK (AD) OR (2↑ (7-PX))      <120>
740 REM *** IN DIESER  ZEILE WIRD  DAS    <205>
750 REM *** GEWUENSCHTE PIXEL GESETZT.    <021>
760 RETURN                                <136>
```

Listing 2. Die Bitmapping-Routine

```
100 REM *** JOYPAINT TEIL 1               <062>
110 REM >> VOR DEM LADEN <<               <034>
120 REM POKE 44,39:POKE9984,0             <217>
130 REM >> EINGEBEN <<                    <066>
140 FOR T=8192 TO 9015:READ D:S=S+D:POKE T,D
    :NEXT                                 <048>
150 IF S<>92869 THEN PRINT"{CLR,3DOWN}FEHLER
    IN DATAS !!":END                      <234>
160 PRINT"BITTE TEIL 2 LADEN"             <185>
```

```
170 DATA 095,234,152,072,165,197,162,003  <050>
180 DATA 221,059,032,240,005,202,016,248  <039>
190 DATA 208,005,189,063,032,208,027,162  <064>
200 DATA 001,221,067,032,240,005,202,016  <045>
210 DATA 248,208,005,189,069,032,208,004  <090>
220 DATA 169,000,240,006,172,141,002,208  <075>
230 DATA 001,010,133,184,104,168,234,234  <089>
240 DATA 165,184,096,041,018,026,049,004  <118>
250 DATA 008,002,001,023,031,004,001,120  <077>
260 DATA 152,072,169,000,141,019,145,169  <135>
270 DATA 127,141,034,145,173,032,145,041  <134>
280 DATA 128,073,128,074,074,074,074,133  <163>
290 DATA 184,169,255,141,034,145,173,017  <173>
300 DATA 145,041,060,073,060,074,074,133  <166>
310 DATA 185,041,008,074,074,074,133,250  <184>
320 DATA 165,185,041,007,005,184,170,104  <189>
330 DATA 168,138,088,096,169,059,141,015  <230>
340 DATA 144,162,000,189,160,032,032,210  <196>
350 DATA 255,232,208,247,189,160,033,240  <229>
360 DATA 006,032,210,255,232,208,245,096  <229>
370 DATA 147,031,206,163,163,205,032,032  <231>
380 DATA 032,206,163,165,032,207,165,013  <245>
390 DATA 165,207,208,186,032,032,206,206  <006>
400 DATA 165,165,032,204,165,013,165,165  <019>
410 DATA 032,032,032,206,206,032,165,165  <014>
420 DATA 032,032,032,206,163,163,205,167  <028>
430 DATA 208,206,208,013,165,163,163,205  <045>
440 DATA 032,165,204,164,165,204,032,032  <046>
450 DATA 032,165,207,208,167,167,032,164  <074>
460 DATA 186,013,165,207,208,167,032,165  <085>
470 DATA 032,032,032,167,032,032,032,165  <070>
480 DATA 163,163,167,167,032,165,013,165  <105>
490 DATA 204,186,167,032,163,163,163,165  <116>
500 DATA 207,032,032,032,165,207,163,163  <109>
510 DATA 167,032,165,013,205,164,164,206  <126>
520 DATA 032,032,032,032,204,165,032,032  <113>
530 DATA 032,205,163,163,208,167,032,165  <148>
540 DATA 013,032,032,032,032,032,032,032  <124>
550 DATA 032,032,032,032,032,032,032,163  <140>
560 DATA 163,163,032,163,013,017,017,018  <169>
570 DATA 032,032,032,032,032,032,032,032  <155>
580 DATA 032,032,032,032,032,032,032,032  <165>
590 DATA 032,032,032,032,032,032,032,032  <175>
600 DATA 032,074,079,089,083,084,073,067  <240>
610 DATA 075,032,080,065,073,078,084,069  <243>
620 DATA 082,032,032,032,032,032,032,032  <210>
630 DATA 032,032,032,032,032,032,032,032  <215>
640 DATA 032,032,032,032,032,032,032,032  <225>
650 DATA 032,032,013,017,017,144,032,040  <243>
660 DATA 067,041,032,049,057,056,052,032  <020>
670 DATA 032,066,089,032,067,046,083,065  <043>
680 DATA 085,069,082,013,017,017,029,029  <049>
690 DATA 029,029,029,018,156,032,032,032  <048>
700 DATA 084,065,083,084,069,032,032,032  <065>
710 DATA 013,000,169,013,141,000,144,169  <054>
720 DATA 044,141,001,144,169,021,141,002  <060>
730 DATA 144,169,019,141,003,144,169,205  <095>
740 DATA 141,005,144,162,189,138,157,000  <101>
750 DATA 016,169,006,157,000,148,202,224  <106>
760 DATA 255,208,242,096,169,020,133,177  <134>
770 DATA 169,000,133,176,160,000,145,176  <128>
780 DATA 230,176,208,002,230,177,166,177  <149>
790 DATA 224,032,208,242,096,134,176,132  <154>
800 DATA 177,138,041,248,133,178,165,176  <153>
810 DATA 056,229,178,133,180,165,177,041  <188>
820 DATA 240,133,179,165,177,056,229,179  <208>
830 DATA 133,181,169,000,133,176,133,182  <192>
840 DATA 169,020,133,177,162,021,165,179  <211>
850 DATA 032,083,034,202,208,248,165,178  <219>
860 DATA 024,010,168,169,000,101,182,133  <208>
870 DATA 182,152,032,083,034,169,000,133  <225>
880 DATA 182,165,181,032,083,034,056,169  <253>
890 DATA 007,229,180,170,240,009,169,001  <249>
900 DATA 010,202,208,252,133,182,096,169  <007>
910 DATA 001,208,249,024,101,176,133,176  <013>
920 DATA 165,182,101,177,133,177,096,032  <036>
930 DATA 245,033,160,000,177,176,005,182  <033>
940 DATA 145,176,096,032,245,033,160,000  <042>
950 DATA 165,182,073,255,049,176,145,176  <080>
960 DATA 096,032,245,033,160,000,177,176  <067>
970 DATA 037,182,197,182,208,002,160,001  <073>
980 DATA 096,160,048,032,071,032,165,250  <085>
990 DATA 208,008,136,208,246,160,000,132  <087>
```

Listing 3. Basic-Lader »Joypaint« (Teil 1)

```
1000 DATA 069,096,169,240,141,012,144,160  <111>
1010 DATA 037,162,255,234,234,234,234,234  <120>
1020 DATA 202,208,248,136,208,243,160,255  <130>
1030 DATA 032,071,032,165,250,208,013,136  <126>
1040 DATA 208,246,160,001,132,069,169,000  <142>
1050 DATA 141,012,144,096,169,144,141,012  <152>
1060 DATA 144,160,032,162,255,234,234,202  <159>
1070 DATA 208,251,136,208,246,160,002,208  <174>
1080 DATA 227,169,015,141,014,144,032,137  <184>
1090 DATA 034,192,000,240,006,165,069,201  <186>
1100 DATA 002,240,001,096,169,000,141,014  <186>
1110 DATA 144,032,137,034,192,002,240,249  <212>
1120 DATA 160,002,132,069,208,237,165,197  <236>
1130 DATA 201,039,240,009,201,047,240,014  <220>
1140 DATA 160,000,132,069,096,169,240,032  <245>
1150 DATA 032,035,160,001,208,007,169,144  <248>
1160 DATA 032,032,035,160,002,132,069,096  <002>
1170 DATA 141,012,144,169,015,141,014,144  <009>
1180 DATA 160,037,162,255,234,234,202,208  <029>
1190 DATA 251,136,208,246,169,000,141,012  <034>
```

Listing 3. Basic-Lader »Joypaint« (Teil 1, Schluß)

```
100 REM *** JOYPAINT TEIL 2                <063>
110 REM >> VOR DEM LADEN <<                <034>
120 REM >> POKE 44,39 EINGEBEN <<          <103>
130 FOR T=9016 TO 9825:READ D:S=S+D:POKE T,D
   :NEXT                                   <043>
140 IF S<>93663 THEN PRINT"{CLR,3DOWN}FEHLER
    IN DATAS !!":END                       <217>
150 SYS 9682                               <013>
160 DATA 144,141,014,144,096,032,178,033   <030>
170 DATA 169,000,170,149,064,232,224,006   <038>
180 DATA 208,249,170,168,032,095,034,032   <058>
190 DATA 217,034,208,003,032,254,034,208   <052>
200 DATA 124,032,016,036,032,071,032,208   <055>
210 DATA 009,032,002,032,208,004,169,000   <060>
220 DATA 240,229,133,183,165,068,240,007   <096>
230 DATA 032,098,034,169,000,240,003,032   <088>
240 DATA 110,034,166,183,189,249,035,168   <129>
250 DATA 189,239,035,024,101,064,201,255   <122>
260 DATA 240,006,201,166,176,006,208,006   <124>
270 DATA 169,000,240,002,169,166,133,064   <140>
280 DATA 133,065,170,024,152,101,066,201   <137>
290 DATA 255,240,006,201,143,176,006,208   <155>
300 DATA 006,169,000,240,002,169,143,133   <161>
310 DATA 066,133,067,168,032,121,034,133   <180>
320 DATA 068,168,240,007,032,110,034,169   <192>
330 DATA 000,240,003,032,098,034,164,050   <184>
340 DATA 032,042,035,165,070,240,128,169   <211>
350 DATA 000,133,070,240,132,165,069,201   <206>
360 DATA 002,240,014,169,008,162,001,134   <215>
370 DATA 068,133,183,169,001,133,070,208   <244>
380 DATA 131,162,000,169,004,208,240,234   <239>
390 DATA 000,000,000,255,255,255,000,001   <227>
400 DATA 001,001,255,001,000,000,255,001   <228>
410 DATA 000,000,255,001,000,000,000,000   <223>
420 DATA 000,000,000,000,000,000,000,000   <220>
430 DATA 173,141,002,201,001,240,006,165   <021>
440 DATA 053,048,002,133,068,173,141,002   <046>
450 DATA 201,004,240,018,165,197,201,024   <054>
460 DATA 208,003,076,211,036,201,055,240   <062>
470 DATA 067,201,063,240,068,096,173,015   <091>
480 DATA 144,133,055,169,008,141,015,144   <095>
490 DATA 165,197,162,021,221,079,038,240   <113>
500 DATA 016,202,208,248,173,141,002,201   <101>
510 DATA 004,240,237,165,055,141,015,144   <119>
520 DATA 096,138,010,170,234,189,084,038   <151>
530 DATA 133,048,189,085,038,133,049,165   <168>
540 DATA 055,141,015,144,160,144,032,042   <143>
550 DATA 035,108,048,000,169,003,133,050   <155>
560 DATA 096,169,043,208,249,169,004,141   <194>
570 DATA 014,144,169,160,141,011,144,173   <180>
580 DATA 015,144,133,051,141,015,144,165   <186>
590 DATA 197,201,064,240,250,201,005,240   <192>
600 DATA 010,201,061,240,010,201,015,240   <178>
610 DATA 032,208,236,169,001,208,002,169   <226>
620 DATA 255,133,052,165,051,024,101,052   <224>
```

Listing 4. Basic-Lader »Joypaint« (Teil 2)

```
630 DATA 133,051,160,048,162,255,234,234   <246>
640 DATA 234,202,208,250,136,208,245,240   <252>
650 DATA 203,169,000,141,011,144,141,014   <244>
660 DATA 144,096,169,001,133,053,096,169   <036>
670 DATA 000,240,249,169,255,208,245,165   <041>
680 DATA 068,240,007,032,098,034,169,000   <041>
690 DATA 240,003,032,110,034,032,024,229   <026>
700 DATA 096,032,178,033,076,195,035,169   <083>
710 DATA 000,133,198,032,215,036,162,000   <055>
720 DATA 189,255,037,032,210,255,232,224   <084>
730 DATA 017,208,245,162,000,032,015,225   <077>
740 DATA 201,013,240,008,157,000,002,232   <072>
750 DATA 224,016,144,241,134,250,162,000   <094>
760 DATA 189,016,038,032,210,255,232,224   <120>
770 DATA 049,208,245,032,228,255,240,251   <137>
780 DATA 201,049,240,006,201,050,240,006   <118>
790 DATA 208,241,162,001,208,002,162,008   <136>
800 DATA 134,251,169,000,133,193,133,172   <127>
810 DATA 133,174,169,020,133,194,133,173   <176>
820 DATA 169,032,133,175,165,250,133,183   <189>
830 DATA 169,000,133,187,169,002,133,188   <201>
840 DATA 165,251,133,186,032,249,253,169   <218>
850 DATA 060,133,178,169,003,133,179,096   <227>
860 DATA 032,239,036,032,130,246,032,134   <212>
870 DATA 037,076,233,036,032,239,036,166   <241>
880 DATA 193,164,194,169,000,032,066,245   <252>
890 DATA 032,134,037,076,233,036,169,013   <253>
900 DATA 032,210,255,165,251,201,001,240   <240>
910 DATA 032,169,008,032,180,255,169,111   <018>
920 DATA 032,150,255,032,165,255,201,013   <013>
930 DATA 240,006,032,066,231,184,080,243   <029>
940 DATA 032,066,231,032,171,255,024,144   <038>
950 DATA 012,032,183,255,240,007,169,105   <051>
960 DATA 160,195,032,030,203,169,008,133   <060>
970 DATA 255,162,255,160,255,234,234,136   <084>
980 DATA 208,251,202,208,246,198,255,208   <098>
990 DATA 240,096,032,132,032,032,228,255   <090>
1000 DATA 201,000,240,249,169,027,141,015  <095>
1010 DATA 144,169,037,133,050,169,255,133  <126>
1020 DATA 053,165,254,201,210,240,003,032  <105>
1030 DATA 220,033,169,210,133,254,076,061  <132>
1040 DATA 035,234,234,234,234,234,234,234  <146>
1050 DATA 147,017,018,070,073,076,069,013  <163>
1060 DATA 018,078,065,077,069,058,146,032  <188>
1070 DATA 013,017,017,018,068,069,086,073  <187>
1080 DATA 067,069,058,146,032,049,046,032  <200>
1090 DATA 084,065,080,069,032,032,032,040  <190>
1100 DATA 049,041,013,032,032,032,032,032  <182>
1110 DATA 032,032,032,050,046,032,070,076  <200>
1120 DATA 079,080,080,089,032,040,056,041  <229>
1130 DATA 013,017,017,070,079,080,080,089  <241>
1140 DATA 032,069,082,082,079,082,058,032  <003>
1150 DATA 034,042,041,021,039,047,220,033  <239>
1160 DATA 125,036,104,037,116,037,202,036  <002>
1170 DATA 207,036                          <119>
```

Listing 4. Basic-Lader »Joypaint« (Teil 2, Schluß)

```
100 REM ********************************  <051>
110 REM ***                            ***  <249>
120 REM ***  MULTICOLOR-FIGUR  FUER    ***  <233>
130 REM ***  ALLE   AUSBAUVERSIONEN    ***  <165>
140 REM ***                            ***  <023>
150 REM ********************************  <101>
160 PRINT"{CLR}"                           <016>
170 FOR T=0 TO 7:READ D:POKE 7168+T,D:NEXT <018>
180 POKE PEEK(648)*256+69,0                <190>
190 REM *** @ IN BILDSCHIRMPOSITION        <248>
200 POKE 38469,10                          <250>
210 REM *** BEI 8 KBYTE ERWEITERUNG        <183>
220 REM *** DEN POKE DURCH 37957,10        <001>
230 POKE 36869,PEEK(36869)OR 15            <242>
240 REM *** ZEICHENGENERATOR INS RAM       <116>
250 GOTO 250                               <026>
260 DATA 48,68,204,48,100,48,48,136        <155>
```

Listing 5. Ein Multicolor-Demo