NEUES VOM VC

Bild 1. Bildschirmformat 25 x 20

Der Video-Chip des VC 20 kann wesentlich mehr als nur die Bildschirmfarben steuern und Musik machen. So läßt sich zum Beispiel mit wenig Aufwand das Bildschirmformat ändern oder zwischen verschiedenen Bildschirmseiten umschalten. Auch die Erzeugung hochauflösender Grafik ist nicht so schwierig, wie es manchem Anfänger scheinen mag.

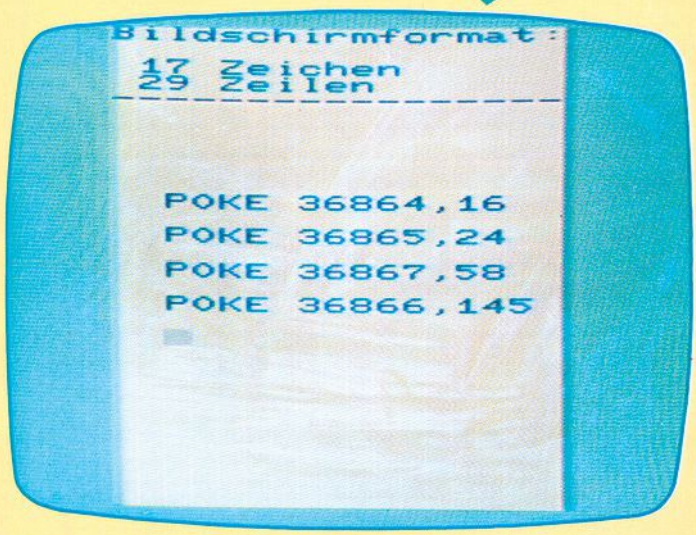

Bild 2. Bildschirmformat 17 x 29

Wie bei seinen »großen Brüdern« CBM 4032/8032 wird die Bildschirmdarstellung des VC 20 mit einem speziellen Videocontroller erzeugt. Das Betriebssystem stellt diesen Baustein nach dem Einschalten auf das bekannte Format von 22 Zeichen bei 23 Zeilen ein. Da diese Festlegung softwaremäßig erfolgt, ist es in Grenzen möglich, das Bildschirmformat nachträglich durch POKE-Befehle zu ändern. Leider werden solche interessanten Möglichkeiten des VC 20 (und C 64) von Commodore nur durch die Befehle PEEK und POKE unterstützt. Deshalb an dieser Stelle zunächst ein Exkurs zu den Bits und Bytes. Ein Byte setzt sich aus acht Bits zusammen, aber das wissen Sie ja sicherlich schon. Um in Basic einzelne Bits anzusprechen, muß eine Bit-Kombination zuerst in den entsprechenden Dezimalwert umgerechnet werden. Dies erreicht man mit der folgenden Beispielrechnung:

Bit: 7 6 5 4 3 2 1 0
Wert: 1 0 0 0 0 0 0 1
=1↑7*1+2↑6*0+2↑5*0+...+2↑0*1
oder: 2↑7+2↑0

Das Ergebnis aus 2↑7+2↑0 lautet 128+1 also 129. Mit den beiden Basic-Befehlen AND und OR können die Bits gezielt verändert werden. Hierzu ein Beispiel: »R=128 : V=1 : PRINT R OR V«. In Bit-Schreibweise sieht die Rechnung wie folgt aus:

R: 1 0 0 0 0 0 0 0 : 128
V: 0 0 0 0 0 0 0 1 : 1
E: 1 0 0 0 0 0 0 1 = 129

Falls ein Bit in R oder in V (oder in beiden) gesetzt ist, wird dieses Bit auch als »1« in das Ergebnis übernommen. Umgekehrt werden jene Bits zu Null, deren entsprechenden Positionen in beiden Variablen zurückgesetzt sind.

Durch den OR-Befehl können gezielt Bits gesetzt werden. Im Bildschirmspeicher bewirkt das gesetzte siebte Bit, daß das Zeichen invertiert dargestellt wird. Schreiben Sie einmal ein Zeichen links oben auf den Bildschirm. Ermitteln Sie nun den »Z=PEEK(7680)« (PEEK(4096) beim erweiterten VC) den entsprechenden Zeichencode. Durch Z=Z OR 128 wird nun das siebente Bit gesetzt. POKE 7680,Z schreibt den Wert zurück und das Zeichen erscheint invertiert.

Das folgende kurze Programm invertiert den gesamten Bildschirm.

```
10 A=7680 : REM ODER 4096!
20 FOR I=0 TO 511 : D=PEEK(I+A)
30 D=D OR 128 : POKE A+I,D :
NEXT
```

VIDEO CHIP

20

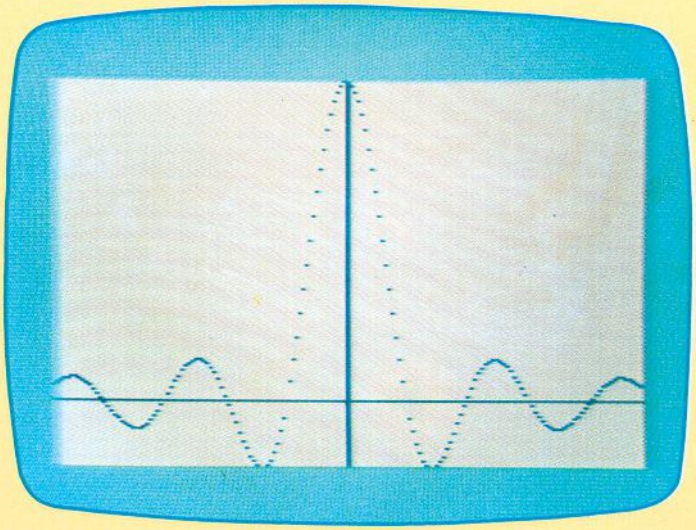

Bild 4. Ein weiteres Beispiel für die Anwendung der neuen Grafikbefehle

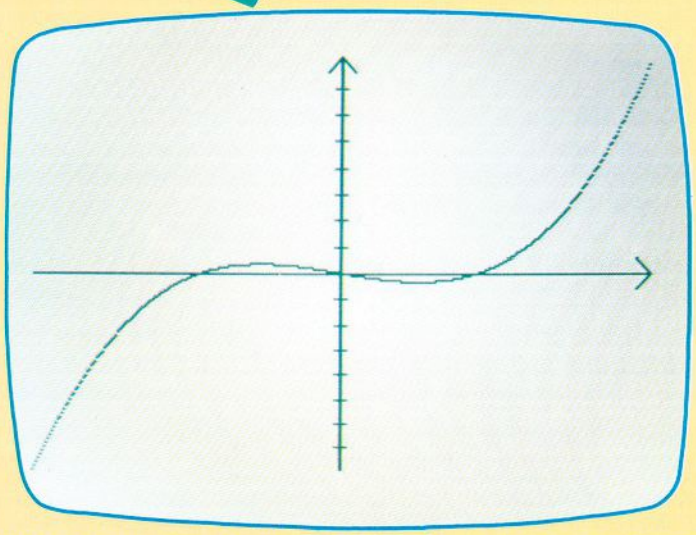

Bild 3. Das Plotten von Funktionen ist kein Problem mit der Grafikerweiterung

Falls Sie die Zeile 30 durch »D = D AND 127:...« ersetzen, werden die zuvor invertierten Zeichen wieder »normal« dargestellt. Durch die BIT-Schreibweise wird dies anschaulich:

```
R : 1 0 0 0 0 0 0 0     1 : 129)
V : 0 1 1 1 1 1 1 1     1 : 127)
E : 0 0 0 0 0 0 0 0     1 =   1
```

Alle BIT-Positionen von R werden in das Ergebnis übernommen, sobald die entsprechenden Positionen des Vergleichswertes auf »1« gesetzt sind. Durch D AND 127 wird somit das siebte Bit in D gelöscht.

Die Tabelle 1 zeigt die für unsere Zwecke wichtigsten Register des Video-Chips sowie deren Funktion.

Die folgende Tabelle 2 verdeutlicht, wie die Register verändert werden können. Mit dem PEEK-Befehl wird der Inhalt der Speicheradresse abgefragt. Der AND-Befehl löscht nun einen Teilbereich und mit OR X wird zuletzt ein neuer Wert angeknüpft.

Außerdem gibt die Tabelle 2 die Originalwerte für die Variable X an, sowie — in Klammern — den zulässigen Zahlenbereich.

Die Adressen 36864/36865 zentrieren die Bilddarstellung auf dem Fernsehgerät oder Monitor. Mit dem Basicprogramm »BILD ZENTRIEREN« (Listing 1) können Sie dies testen. Der Bildausschnitt wird mit den Cursor-Tasten verschoben. Falls Sie die RETURN-Taste drücken, zeigt das Programm die notwendigen POKE-Befehle für die letzte Einstellung und setzt den Bildschirm zuletzt wieder in den Ausgangszustand zurück.

Das Programm »BILDSCHIRMFORMAT« (Listing 2) verändert die Adressen 36866/36867. Mit den Cursor-Tasten wird die Anzahl der Zeilen beziehungsweise die Anzahl der Zeichen pro Zeile verändert. Auch hier können Sie durch Drücken der RETURN-Taste die erforderlichen POKE-Befehle für die letzte Einstellung ermitteln.

Die Bilder 1 und 2 zeigen zwei geänderte Bildformate mit den dazugehörigen POKE-Befehlen.

Zwischen verschiedenen Bildschirmbereichen umschalten

Das Register 36869 legt die Adressen für den Zeichensatz und den Bildschirmspeicher fest. Die Tabelle 2 zeigt vier POKE-Befehle mit den entsprechenden Bildschirm-Startadressen. Falls Sie eine Änderung durchführen, müssen Sie dem Computer die neue Startadresse in dem Zeiger 648 mitteilen. Außerdem müssen noch 25 weitere Zeiger (MSB der Zeilenanfänge) geändert werden. Durch SYS 58775 führt der Computer die notwendigen Berechnungen aus. Die POKE-Befehle in die Adressen 52/53 begrenzen den oberen Speicherbereich und schützen so das Bild-RAM vor dem Überschreiben durch Basic-Variablen.

Die mit einem Doppelkreuz »#« gekennzeichneten Adressen gelten, falls die Bildadresse zusätzlich um 512 Byte nach unten verschoben wurde. Diese »Feinverschiebung«

NEUES VOM VC 20 VIDEO CHIP

erfolgt durch das siebte Bit der Adresse 36866. Die Tabelle 2 zeigt die beiden POKE-Anweisungen, mit denen diese Umschaltung — beispielsweise zwischen zwei Text- oder Grafikseiten — durchgeführt wird.

Das Programm »BILD-RAM« (Listing 3) verdeutlicht dies. Die Programmzeile 11 schützt den oberen Speicherbereich. Mit GOSUB 80 wird die erste Seite »vorgeblendet«. CHR$(147) löscht die Seite. Zuletzt wird der Text »ERSTE SEITE« gedruckt. Die Zeilen 40 bis 42 wiederholen den Vorgang für die zweite Seite. Die Programmzeilen 50 bis 61 schalten nun — mit einer Verzögerung — abwechselnd die beiden Testseiten ein. So schnell und flimmerfrei kann dies kein anderes Commodore-System.

Falls Sie nach dem Umschalten mit dem Register 36866 die Zeile »POKE 648,...:SYS 58775« weglassen, wird sich die Bildanzeige zwar ändern; PRINT-Befehle gelangen jedoch weiterhin in die alte — nun unsichtbare — Seite.

Tabelle 2. So können die Register aus Tabelle 1 verändert werden

D=36864 / X: (0-127) / Orig. 12	:	Horizontal-Position
POKE D, (PEEK(D) AND 128) OR X	:	verschieben.
D=36865 / X: (0-255) / Orig. 38:		Vertikal-Position
POKE D,X	:	verschieben.
D=36866 / X: (0-27) / Orig. 150	:	Anzahl Zeichen
POKE D,(PEEK(D) AND 128) OR X	:	pro Zeile.
POKE D,PEEK(D) OR 128 : oben	:	Adresse des Bildschirm-RAM
POKE D,PEEK(D) AND 127 # unten	:	um 512 Byte verschieben.
D=36867 / X: (0-64) / Orig. 174		
POKE D,(PEEK(D) AND 129) OR (2*X):		Anzahl der Zeilen ändern.
POKE D,PEEK(D) OR 1 (ein)	:	Einfache oder doppelte
POKE D,PEEK(D) AND 254 (aus)	:	Matrix einstellen.
D=36869 (# = > untere 512 Byte		
		Bildschirm-Adresse:
POKE D,240:POKE 52/56/648,30 (#28)	:	7680 (#7168)
POKE D,224:POKE 52/56/648,26 (#24)	:	6656 (#6144)
POKE D,208:POKE 52/56/648,22 (#20)	:	5632 (#5120)
POKE D,192:POKE 52(56/648,18 (#16)	:	4608 (#4096)
	:	
		Zeichensatz-Adresse: (von-bis)
POKE D,(PEEK(D) AND 240) OR 15	:	7168-8191 (1)
POKE D,(PEEK(D) AND 240) OR 14	:	6144-8181
POKE D,(PEEK(D) AND 240) OR 13	:	5120-7168
POKE D,(PEEK(D) AND 240) OR 12	:	4096-6144
POKE D,(PEEK(D) AND 240) OR 11	:	3072-5120 (2)
POKE D,(PEEK(D) AND 240) OR 10	:	2048-4096 (2)
	:	
	:	(1) invertierte Zeichen vom ROM
	:	(2) mit 3 KByte-Erweiterung
Beispiel: POKE D,PEEK(D), 208+14	= >	Bildschirm-Adresse: 5632-6143
	= >	Zeichens. -Adresse: 6144-8191

Eigene Zeichen definieren

Aus der Tabelle 1 kann man ablesen, daß nur die oberen vier Bits der Adresse 36869 das Bild-RAM verschieben. Die restlichen Bits bestimmen die Lage des Zeichensatzes im Speicher. Durch die in der Tabelle 2 angegebenen OR-Verknüpfungen kann der Zeichensatz ins RAM gelegt werden. Nur »OR 15« stellt eine Ausnahme dar, da hierdurch die unteren 128 Zeichen aus dem RAM, die oberen — invertierten — Zeichen weiterhin aus dem ROM geholt werden. Diese Zeichen erscheinen dann aber »nichtinvertiert«.

In dem Programm »PROG. ZEICHENSATZ« (Listing 4) werden die notwendigen Änderungen in der Programmzeile 30 vorgenommen. Nach »POKE 36869,240« lautet die Startadresse des Bildschirm-RAMs 7680. Der Wert »+15« verschiebt den Zeichensatz zur Adresse 7168.

Im einzelnen schreibt das Programm den gesamten Zeichensatz des VC 20 auf den Bildschirm. Dabei werden die Zeichen durch »POKE 38400+I,0« in der Farbe Schwarz gedruckt. Die Zeile 30 verändert dann die Zeiger, so daß die ersten 128 Zeichen den zufälligen Inhalt des Zeichensatz-RAMs wiedergeben. Der Zeichensatzspeicher wird dann in den folgenden Zeilen gesetzt und anschließend gelöscht. Zuletzt werden noch die Originalzeichen aus dem ROM in das RAM »gePOKEt«. Die Zeile 55 schaltet den programmierten Zeichensatz wieder ab.

Falls Sie mit dem Zeichensatz experimentieren wollen, brauchen Sie nur die letzte Zeile wegzulassen. Alle »ungeshifteten« Zeichen können dann verändert werden. Zuerst müssen Sie jedoch ermitteln, wohin Sie »POKEn« müssen. Schreiben Sie hierzu das Zeichen, das Sie ändern wollen, links oben auf den Bildschirm. Mit »PRINT PEEK(7680)« kann dann der dazugehörige Code ermittelt werden. Der Wert muß nun mit acht multipliziert werden, da zu jedem Zeichen acht Byte gehören. Zuletzt müssen Sie noch die Startadresse des Zeichensatzes (hier 7168) hinzuaddieren. Für den Buchstaben »A« lautet die richtige Adresse 7174. POKEn Sie einmal den Wert 129 an diese Adresse. In Bit-Form sieht das wie folgt aus: 129 gibt 10000001.

Genauso wird anschließend die erste Zeile in dem Buchstaben »A« aussehen; rechts und links außen je ein Pünktchen. Verändert Sie auch einmal die nächsten Adressen oder

Tabelle 1. Einige wenig bekannte, aber recht nützliche Register des VIC-Chips.

Adresse / BIT:	7	6	5	4	3	2	1	0	
36864 =	—	HZ	HZ	HZ	HZ	HZ	HZ	HZ	: Horizontal-Zentrierung
36865 =	VZ	VZ	VZ	VZ	VZ	VZ	VZ	VZ	: Vertikal-Zentrierung
36866 =	BA	ZE	ZE	ZE	ZE	ZE	ZE	ZE	: Bildschirm-Adresse : ZEichen pro Zeile
36867 =	—	ZL	ZL	ZL	ZL	ZL	ZL	MA	: ZeiLen-Anzahl : Matrix: (8*8 / 8*16)
36869 =	BA	BA	BA	BA	ZA	ZA	ZA	ZA	: Bildschirm-Adresse : Zeichensatz-Adresse

versuchen Sie den Umlaut »Ä« zu gestalten.

Nach der Theorie nun ein praktisches Grafikbeispiel. Das abschreckend lange Basic-Ladeprogramm (Listing 5) erweitert den Basic-Befehlssatz des VC 20 (ohne Speichererweiterung) um sieben weitere Anweisungen (COPY, GNEW, GON, GOFF, PLOT, TURN und CLEAR). Beim Eintippen sollten Sie jedoch die Zeilen 10 bis 32 durch das Programm »Prüfsumme« (Listing 6) ersetzen. Nach RUN können Sie mit Hilfe der Tabelle 3 überprüfen, ob Sie die Daten fehlerfrei eingegeben haben. Die Prüfsummen gelten jeweils für eine DATA-Zeile.

Nachdem Sie das Ladeprogramm (Zeile 10 bis 32) hinzugenommen haben, muß das Programm unbedingt abgespeichert werden. Nach RUN sucht das Programm die Zeile 32 (also nicht weglassen) und überträgt die Daten ab dieser Adresse in den Speicher. In den Programmzeilen 22 bis 24 wird das Maschinenprogramm zuletzt in den oberen RAM-Bereich übertragen und durch die folgenden POKE-Befehle geschützt. Der NEW-Befehl löscht zuletzt das — ohnehin zerstörte — Programm.

Mit SYS 6273 können die Befehle jetzt eingeschaltet werden. Bei 2173 BYTES FREE nähert man sich natürlich gefährlich dem »Sinclair-Syndrom«.

Nach dem Einschalten fällt zuerst ein neuer Cursor auf. In der invertierten Blinkphase des Cursors wird ein »graues« Viereck (COMMODORE-SHIFT »+«) gedruckt. Dies ist notwendig, da der Zeichensatz keine invertierten Zeichen mehr umfaßt und deshalb der Cursor nicht mehr blinken würde. Durch das Grafikprogramm werden alle 64 umgeshifteten Zeichen programmierbar. Der Zeichensatzspeicher beginnt — wie in dem kleinen Textprogramm »PROG. ZEICHENSATZ« (Listing 4) — an der Adresse 7168. Mit dem Befehl »COPY« wird der Originalzeichensatz aus dem ROM in diesen Bereich kopiert.

Der Befehl »GON« schaltet nun den Grafik-Modus ein. Da zuvor der Originalzeichensatz ins RAM kopiert wurde, werden sich die ersten 64 Zeichen nicht verändern. Falls invertierte Zeichen auf dem Bildschirm stehen, werden diese jetzt nichtinvertiert wiedergegeben. Nur bei den geshifteten Zeichen (außer den invertierten) gibt es einen Totalausfall, aber bei 3,5 KByte Speicherraum muß man schon gewisse Mängel in Kauf nehmen. Mit »GOFF« wird der Zeichensatz wieder ins ROM verlegt, so daß alle Zeichen sichtbar werden.

Die drei Befehle PLOT/CLEAR/TURN setzen, löschen oder invertieren einen Bildpunkt. Natürlich muß hinter den Befehlen eine X- und Y-Koordinate angegeben werden. Beispielsweise »PLOT 176,1«; allerdings führt diese Eingabe zu der Fehlermeldung X-ERROR (IN...), da der zulässige Wertebereich für die X-Koordinate 0 bis 175 (22 Zeichen * 8 — 1), und für die Y-Koordinate 0 bis 183 (23 Zeichen * 8 — 1) beträgt.

Der Wertebereich für die Koordinatenangabe wird durch den letzten Befehl »GNEW« ermittelt, da das Bildschirmformat — wie zuvor beschrieben — per Software geändert werden kann. Bei einer Einstellung mit beispielsweise 25 Zeichen pro Zeile kann der Wert für X zwischen 0 und 199 variieren.

Außerdem löscht der GNEW-Befehl die 64 programmierbaren Zeichen sowie alle nichtinvertierten Zeichen auf dem Bildschirm. Da sich eine Grafik jedoch aus genau solchen Zeichen zusammensetzt, wird — falls sich eine Zeichnung auf dem Bildschirm befindet — diese gelöscht. Invertierte Zeichen werden hingeben nicht verändert, so daß Texte oder Beschriftungen erhalten bleiben.

In den Zeilen 10 bis 20 wird der Bildschirm gelöscht und der invertierte Text »TEXT-PROGRAMME« geschrieben. Der Befehl »GNEW« löscht den Grafikspeicher und den Bildschirm, wobei der invertierte Text unverändert bleibt. GON schaltet dann die Grafik ein und in der Schleife wird eine Linie von links unten nach rechts oben (in der augenblicklichen Schriftfarbe) gezeichnet. Sobald Sie eine Taste drücken, wird die Grafik wieder abgeschaltet und Sie sehen eine schräge Linie aus Fragezeichen.

Das Maschinenprogramm schreibt ein programmierbares Zeichen auf den Bildschirm und ändert den Grafikinhalt dieses Zeichens. Anschließend wird geprüft, ob schon ein Zeichen mit genau dieser Grafik umdefiniert wurde. Ist dies der Fall, so wird das gefundene Zeichen auf den Bildschirm gebracht und das alte Zeichen ist wieder frei. Wegen dieser Optimierungs-Routine »verbraucht« das Beispielprogramm nur ein Zeichen aus dem schmalen Vorrat von 64 programmierbaren Zeichen.

Da diese Optimierung über einen Vektor verläuft, kann sie durch »POKE 1,PEEK(1)—1« abgeschaltet werden. Lassen Sie das Testprogramm erneut laufen und sehen Sie sich das neue Ergebnis an. Mit »POKE 1,PEEK(1)+1« optimiert das Programm wieder, wird aber auch etwas langsamer.

Durch »PEEK(0)« können Sie außerdem abfragen, wie viele Zeichen noch frei sind. Hierdurch kann die Fehlermeldung »CHARACTER-ERROR (IN...)« verhindert werden. Falls PEEK(0) den Wert Null ergibt, sollten keine Grafikbefehle mehr folgen.

Ändern Sie nun die Zeile 60 des Testprogramms in »60 TURN I,I« und erweitern Sie folgende Zeile mit »GOTO 50«.

Tabelle 3. Prüfsummen zum Grafikprogramm

Zeile 50-54	:	2390	2436	2823	2640	3449
Zeile 55-59	:	3487	3010	3321	2882	2608
Zeile 60-64	:	2930	2741	2458	3089	2285
Zeile 65-69	:	2480	2156	2935	3048	2241
Zeile 70-74	:	2495	2564	3256	2723	2393
Zeile 75-79	:	1864	2230	2187	2148	3300
Zeile 80-84	:	3135	2640	3023	2734	1918
Zeile 85-88	:	2258	1431	1564	449	—

Doch nun ein Beispiel:

```
10 PRINT CHR$(147);CHR$(18);
20 PRINT »TEST-PROGRAMM«
30 GNEW
40 GON
50 FOR I=0 TO 111
60 : PLOT I,I
70 NEXT
80 POKE 198,0:WAIT 198,1
90 GOFF
```

Das Programm wird jetzt eine Linie ziehen, anschließend löschen und so fort, da die Bildpunkte jetzt fortlaufend invertiert werden. Falls Sie in der Zeile 40 den Befehle »GON« durch »GOFF« ersetzen, können Sie anschaulich verfolgen, wie das Programm — und speziell die Optimierung — arbeitet.

Eine Besonderheit müssen Sie jedoch unbedingt beachten. Die neuen Basicbefehle können nicht direkt hinter einer THEN-Anweisung stehen. Am einfachsten ist es, die beiden Befehle dann durch einen Doppelpunkt (...THEN : COPY) zu trennen.

Listing 1. »Bild zentrieren«

```
 10 REM ******* BILD ZENTRIEREN ********
 15 :
 20 H=12        : V=38
 25 POKE 198,0 : WAIT 198,1 : GET A$
 30 IF A$=CHR$( 13)              THEN 60
 35 IF A$=CHR$( 17) AND V<255 THEN V=V+1
 40 IF A$=CHR$(145) AND V>0   THEN V=V-1
 45 IF A$=CHR$( 29) AND H<23  THEN H=H+1
 50 IF A$=CHR$(157) AND H>1   THEN H=H-1
 55 POKE 36864,H :POKE 36865,V:GOTO 25
 60 POKE 36864,12:POKE 36865,38
 65 PRINT "POKE 36864 ,";H
 70 PRINT "POKE 36865 ,";V
READY.
```

Zuletzt eine Grafik-Anwendung. Die Bilder 3 und 4 zeigen, welche Ergebnisse mit der Grafik-Erweiterung möglich sind. Das Basicprogramm »FUNKTIONS-PLOT« (Listing 7) vereinfacht das programmierte Plotten von Funktionen doch erheblich.

Nachdem Sie das Programm eingegeben und gestartet haben, werden Sie nach dem Wertebereich gefragt. Anschließend wird die Bildumrahmung gezeichnet. Die Größe sowie die Position des Diagramms auf dem Bildschirm ist durch die Variablen Unten, Oben, Rechts und Links (U,O,R,L) in der Zeile 110 festgelegt. Die Zeilen 170 bis 230 berechnen nun das Minimum sowie das Maximum der Funktion. Außerdem werden die Werte in das Feld Y(A) übertragen. Die Variablen MI und MA werden in Zeile 250 auf vier Nachkommastellen begrenzt. Diese Werte sowie die Funktion selbst werden nun (invertiert) im unteren Bildteil gedruckt. Nachdem die Funktion grafisch dargestellt wurde, läuft das Programm in Zeile 390 in einer Warteschleife.

Drücken Sie jetzt eine Taste, so gelangen Sie in ein Mini-Menü. Sie können nun eine neue Funktion eingeben oder die alte mit neuen Werten untersuchen.

Eine neue Funktion wird in der Zeile 450 mit INPUT A$ übergeben. Salopp gesagt, programmiert sich der Copmputer in den folgenden Zeilen selbst. Wie das?

Zuerst wird der Bildschirm gelöscht und in der obersten Zeile die Zeilennummer 190 und der String A$, der die neue Funktion enthält, gedruckt. Die Zeile 460 druckt darunter die Nummer 190 und einen PRINT-Befehl. Anschließend folgt ein Anführungszeichen, dann die Funktion und schließlich wieder ein Anführungszeichen. Zuletzt wird in der dritten Bildschirmzeile der Befehl »RUN« gedruckt. Jetzt könnte auf dem Bildschirm der folgende Text stehen:

```
190 Y=COS(X)
290 ?"Y=COS(X)"
RUN
```

Wäre das Programm hier zu Ende, so müßten Sie nur die HOME-Taste drücken und der Cursor würde auf der Zeile 190 stehen. Drückt man jetzt RETURN, so wird diese Zeile einprogrammiert. Ein erneuertes RETURN übernimmt die Zeile 290 in das Programm und beim dritten RETURN wird der Befehl RUN ausgeführt. Da sich der Computer bis zu neun Tastatureingaben »merken« kann, brauchen Sie diesen Ablauf nicht »von Hand« einzugeben. Der erste POKE-Befehl in der Zeile 470 sagt dem Computer, daß — angeblich — sechs Tasten gedrückt worden sind. Zuerst »CURSOR HOME« (POKE 631,19), und dann fünfmal »RETURN« (Zeile 480). Nach dem END-Befehl führt er diese — untergeschobenen — Eingaben aus, programmiert die beiden Programmzeilen ein und startet sich zuletzt wieder mit dem RUN-Befehl.

Dieses Verfahren ist übrigens sehr praktisch, falls man ein Maschinenprogramm in DATA-Zeilen umwandeln muß. Vorausgesetzt das notwendige Programm ist fehlerfrei, kann man eigentlich sicher sein, daß es der Basic-Lader anschließend auch ist.

Wie Sie an dem Funktions-Plotter sehen können, ermöglicht die Grafikhilfe die Programmierung auch komplexerer Grafiken in einfacher und überschaubarer Form. Da die Erstellung der Grafik recht schnell erfolgt, ist eine große Vielfalt von Anwendungen denkbar.

(Heino Verder/ev)

Listing 2. »Bildschirmformat«

```
 10 REM ****** BILDSCHIRM-FORMAT *******
 15 :
 20 V=46        : H=22+128
 25 POKE 198,0 : WAIT 198,1 : GET A$
 30 IF A$=CHR$( 13)              THEN 60
 35 IF A$=CHR$( 17) AND V<100 THEN V=V+2
 40 IF A$=CHR$(145) AND V>2   THEN V=V-2
 45 IF A$=CHR$( 29) AND H<155 THEN H=H+1
 50 IF A$=CHR$(157) AND H>128 THEN H=H-1
 55 POKE 36866,H :POKE 36867,V  :GOTO 25
 60 POKE 36867,46:POKE 36866,150
 65 PRINT "POKE 36866 ,";H
 70 PRINT "POKE 36867 ,";V
READY.
```

Listing 3. »Bild-RAM« — Umschalten zwischen zwei Bildschirmseiten

```
  1 REM ***** BILD-RAM *****
  2 :
 10 D=36866:P1=30:P2=28
 11 POKE 52,P2:POKE 56,P2
 15 :
 30 GOSUB 80
 31 PRINT CHR$(147):PRINT
 32 PRINT" ERSTE SEITE"
 35 :
 40 GOSUB 90
 41 PRINT CHR$(147):PRINT
 42 PRINT" ZWEITE SEITE"
 45 :
 50 FOR Z=0 TO 300:NEXT
 51 GOSUB 80
 60 FOR Z=0 TO 300:NEXT
 61 GOSUB 90       :GOTO 50
 70 :
 80 POKE D,PEEK(D) OR 128
 81 POKE 648,P1:SYS 58775
 82 RETURN
 85 :
 90 POKE D,PEEK(D) AND 127
 91 POKE 648,P2:SYS 58775
 93 RETURN
READY.
```

Listing 4. »Programmierbarer Zeichensatz«

```
 10 REM ******* PROG. ZEICHENSATZ ******
 11 :
 12 POKE 52,26          :POKE 56,26
 15 :
 20 FOR I=0 TO 255      :POKE 7680+I,I
 25 POKE 38400+I,0      :NEXT
 30 POKE 36869,240+15
 35 FOR I=7168 TO 7679 :POKE I,255:NEXT
 40 FOR I=7168 TO 7679 :POKE I,0   :NEXT
 45 FOR I=0    TO  511 :C=PEEK(32768+I)
 50 POKE 7168+I,C       :NEXT
 55 POKE 36869,240+0
READY.
```

Listing 6. Prüfsummenprogramm zum Basic-Lader aus Listing 5. Das Zeichen » â « in Zeile 28 ist der Hochpfeil (»↑«)

```
 10 REM ********* PRUEFSUMME **********
 12 ZD=50:A=1:FOR I=0 TO 896:READ A$
 14 ZL=PEEK(60)+PEEK(61)*256:D=0
 16 IF ZL=ZD THEN 20
 18 PRINT "ZEILE";ZD;C : ZD=ZL : C=0
 20 FOR L=1 TO 2:IF LEN(A$)<2THEN A$="0"+A$
 22 W$=MID$(A$,L,1):W=0:IF W$=""THEN 24
 24 W=ASC(W$)-48    :IF W>9 THEN W=W-7
 26 :
 28 D=D+W*16^(2-L) :NEXT : C=C+D : S=S+1
 30 NEXT : PRINT "ZEILE";ZD,C    : END
 32 :
 50 DATA EA,EA,EA,78,A9,BC,8D,14,3,A9...
READY.
```

NEUES VOM VC 20 VIDEO CHIP

Listing 5. Basic-Ladeprogramm zur Befehlserweiterung

```
 10 FOR I=4096 TO 9999:IF PEEK(I)<>42THEN NEXT
 12 A=I:FOR I=0 TO 896:READ A$:D=0:PRINT"";I
 14 FOR L=1 TO 2:IF LEN(A$)<2THEN A$="0"+A$
 16 W$=MID$(A$,L,1):W=0:IF W$=""THEN 18
 18 W=ASC(W$)-48   :IF W>9 THEN W=W-7
 20 :
 22 D=D+W*16^(2-L) :NEXT:POKE I+A,D:NEXT
 24 FOR I=0 TO 896:D=PEEK(A+I):POKE 6272+I,D:NEXT
 26 POKE 51,128 : POKE 52,24
 28 POKE 55,128 : POKE 56,24  :NEW
 30 :
 32 REM **********
 50 DATA EA,EA,EA,78,A9,BC,8D,14,3,A9,18,8D,15,3,A9,1B,8D,8,3,A9,19,8D,9
 51 DATA 3,A9,11,85,1,A9,1B,85,2,8D,F2,1B,58,60,78,A9,BF,8D,14,3,A9,EA,8D
 52 DATA 15,3,A9,E4,8D,8,3,A9,C7,8D,9,3,58,60,AD,F2,1B,F0,D,CE,F2,1B,A4,D3
 53 DATA B1,D1,8D,F1,1B,8D,F0,1B,A9,18,48,A9,DB,48,8,8,8,8,4C,BF,EA,AE,5
 54 DATA 90,E8,D0,0,A4,D3,A6,CC,D0,2E,AD,F1,1B,F0,8,8E,F1,1B,B1,D1,8D,F0
 55 DATA 1B,A5,C6,F0,E,A5,CB,C9,1B,F0,5,EE,F2,1B,D0,9,EE,F1,1B,A9,E6,A6,CF
 56 DATA D0,3,AD,F0,1B,D1,D1,F0,2,91,D1,4C,18,EB,A9,7,85,58,A9,0,48,AA,CA
 57 DATA A0,0,C8,E8,BD,AB,1B,F0,F,D1,7A,F0,F5,18,68,69,8,C6,58,D0,E8,4C,E4
 58 DATA C7,68,88,98,18,65,7A,85,7A,90,2,E6,7B,BD,AC,1B,85,F7,BD,AD,1B,85
 59 DATA F8,20,55,19,4C,18,19,6C,F7,0,A2,FF,D0,2,A2,F0,8E,5,90,60,A0,0,A2
 60 DATA 2,84,62,84,64,A9,80,85,63,A9,1C,85,65,B1,62,91,64,C8,D0,F9,E6,63
 61 DATA E6,65,CA,D0,F2,60,AD,2,90,29,7F,8D,F5,1B,A,A,A,8D,F4,1B,AD,3,90
 62 DATA 29,7E,A,A,8D,F3,1B,CE,F3,1B,A9,1C,85,63,A9,1E,85,65,A0,0,84,62,84
 63 DATA 64,A2,1,A9,0,91,62,B1,64,30,4,A9,A0,91,64,C8,D0,F1,E6,63,E6,65,CA
 64 DATA 10,EA,A2,3F,86,0,A2,7F,A9,0,9D,50,3,CA,10,FA,60,A2,0,2C,A2,2,2C
 65 DATA A2,4,BD,E0,1B,8D,40,3,BD,E1,1B,8D,41,3,20,73,0,20,8A,CD,20,F7,D7
 66 DATA A0,58,A6,15,D0,29,8E,43,3,A5,14,8D,42,3,CD,F4,1B,B0,1C,20,79,0,20
 67 DATA FD,CE,20,8A,CD,20,F7,D7,A5,15,D0,8,AD,F3,1B,38,E5,14,B0,17,A9,59
 68 DATA D0,2,A9,58,20,D2,FF,A9,2D,20,D2,FF,20,7A,C6,C8,A9,6B,4C,67,C4,AA
 69 DATA 20,98,1B,8E,45,3,A9,1E,85,5B,A9,0,C0,0,F0,B,18,6D,F5,1B,90,2,E6
 70 DATA 5B,88,D0,F5,85,5A,AE,42,3,20,98,1B,AD,43,3,F0,5,98,18,69,20,A8,A9
 71 DATA 0,E8,38,6A,CA,D0,FC,86,FD,8D,44,3,B1,5A,8D,47,3,30,15,C5,0,90,11
 72 DATA AA,BD,50,3,D0,6,B1,5A,C5,0,B0,3A,DE,50,3,E6,FD,A5,FB,F0,9,C6,FB
 73 DATA A6,FB,BD,E8,3,D0,11,A5,0,D0,B,58,A9,E6,A0,1B,20,1E,CB,4C,22,1A,C6
 74 DATA 0,91,5A,AA,A5,5A,18,69,0,85,58,A5,5B,69,78,85,59,AD,86,2,91,58,8A
 75 DATA 78,85,FC,8C,46,3,A2,0,86,59,A,26,59,A,26,59,A,26,59,18,69,0,85,58
 76 DATA A5,59,69,1C,85,59,A5,FD,F0,24,A9,0,85,5E,AD,47,3,A,26,5E,A,26,5E
 77 DATA A,26,5E,18,69,0,85,5D,A5,5E,69,1C,85,5E,A0,7,B1,5D,91,58,88,10,F9
 78 DATA AC,45,3,AD,44,3,20,8A,1B,91,58,58,6C,1,0,60,78,A0,7,84,FD,B1,58
 79 DATA 99,E0,3,D0,2,C6,FD,88,10,F4,E6,FD,D0,4,A9,A0,D0,24,A9,1E,85,5E,A9
 80 DATA 0,85,5D,A2,40,D0,42,A0,7,E4,FC,F0,3C,E4,0,F0,4B,B1,5D,D9,E0,3,D0
 81 DATA 31,88,10,F6,FE,50,3,8A,AC,46,3,91,5A,A6,FC,BD,50,3,F0,3,DE,50,3
 82 DATA 8A,E6,0,C5,0,F0,9,C6,0,A6,FB,9D,E8,3,E6,FB,A0,7,A9,0,91,58,88,10
 83 DATA FB,30,10,A5,5D,38,E9,8,B0,2,C6,5E,85,5D,CA,E4,0,D0,AE,58,60,6C,40
 84 DATA 3,11,58,60,51,58,60,49,FF,31,58,60,A0,0,8A,10,4,A0,10,29,7F,AA,38
 85 DATA E9,8,90,3,C8,D0,F7,60,50,4C,4F,54,0,D1,19,0,54,55,52,4E,0,D4,19
 86 DATA 0,43,4C,45,41,52,0,D7,19,43,4F,50,59,0,62,19,0,CB,4E,0,58,19,0,0
 87 DATA 0,CB,46,46,0,5C,19,0,0,47,A2,0,81,19,8D,1B,90,1B,93,1B,43,48,41
 88 DATA 52,41,43,54,45,52,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
READY.
```

Listing 7. Basicprogramm »Funktions-Plot« zur Demonstration der neuen Grafikbefehle

```
CURSOR-CODE:  PRINT"C"  =>  CLEAR HOME
              "H"  =>  CURS. HOME
              "D"  =>  CURS. DOWN
              "R"  =>  REVERSE


 100 CLR:INPUT"CR WERTEBEREICH";X1,X2
 110 U=80:O=183:L=0:R=175:D=R-L:K=10000
 120 GNEW:GON:DIM Y(D):IF X1=X2 THEN 100
 130 FOR A=L TO R:PLOT A,U:PLOT A,O:NEXT
 140 FOR A=U TO O:PLOT L,A:PLOT R,A:NEXT
 150 IF X1>X2 THEN Z=X1:X1=X2:X2=Z
 160 :
 170 SX=(X2-X1)/D:X=X1:MI=1E38:P=MI:MA=-MI
 180 FOR A=0 TO D
 190 Y=SIN(X)+SIN(2*X)
 200 IF Y>MA THEN MA=Y
 210 IF Y<MI THEN MI=Y
 220 IF ABS(Y)<P THEN P=ABS(Y):N=A
 230 Y(A)=Y:X=X+SX:NEXT:SY=(MA-MI)/(O-U-5)
 240 :
 250 MI=INT(MI*K)/K:MA=INT(MA*K)/K:K=0
 260 PRINT "SDDDDDDDDDDDDDDDDDR  ";T1=K
 270 T2=LEN(STR$(X1))-LEN(STR$(MI))
 280 IF T2<K THEN T1=-T2:T2=K
 290 PRINT "Y=SIN(X)+SIN(2*X)"
 300 PRINT TAB(T1);"RD";X1;"<X<";X2
 310 PRINT TAB(T2);"R" ;MI;"<Y<";MA
 320 IF MI=>K AND MA=>K OR MI<=K AND MA<=K THEN 340
 330 P=U+3+INT((Y(N)-MI)/SY):FOR A=L TO R:PLOT A,P:NEXT
 340 IF X1=>K AND X2=>K OR X1<=K AND X2 <=K THEN 360
 350 P=INT(ABS(X1)/SX):FOR A=U TO O:PLOT P,A:NEXT
 360 :
 370 FOR A=L TO R:P=U+3+INT((Y(A-L)-MI)/SY)
 380 PLOT A,P:IF PEEK(K) THEN NEXT
 390 POKE 198,0:WAIT 198,1:GET A$
 400 PRINT "CR F => FUNKTION"
 410 PRINT "R W => WERTEBEREICH"
 420 WAIT 198,1:GET A$
 430 IF A$= "W" THEN 100
 440 IF A$<>"F" THEN 420
 450 COPY:INPUT "C FUNKTION";A$:PRINT "C 190";A$
 460 PRINT"290 ?";CHR$(34);A$;CHR$(34)
 470 PRINT "RUN ":POKE 198,6:POKE 631,19
 480 FOR I=632 TO 636:POKE I,13:NEXT:END
READY.
```