80-Zeichen-Grafik für den C 128

Auf dieses Grafikpaket für eine 640 x 200 Punkte-Auflösung auf dem 80-Zeichen-Bildschirm haben C 128-Besitzer gewartet! Alle Grafikbefehle des Basic 7.0 stehen damit für den neuen Modus zur Verfügung.

Der neue Basic-Interpreter des Commodore 128 enthält 26 Befehle, mit deren Hilfe man einfach und schnell auch komplizierte Grafik-Programme erstellen kann. Leider unterstützen diese Befehle nur die Programmierung des VIC-Chips mit der schon vom C 64 her bekannten Auflösung. Der zusätzlich eingebaute 80-Zeichen-Chip »VDC« wird nur zum Aufbau eines 80-Zeichen-Text-Bildschirms genutzt. Wer auch diesen Bildschirm für grafische Darstellungen nutzen will, der muß seine eigenen Routinen schreiben. Doch gibt es — wie der vorliegende Artikel zeigt — eine recht einfache Lösung für dieses Problem. Das abgedruckte Basic-Programm (Listing 1) baut ein Maschinenprogramm für den »User-RAM-Bereich« von $1300 bis $1bff auf. Nach dem Initialisieren dieses Maschinenprogramms mit »SYS DEC ("1303")« stehen die Befehle GRAPHIC, BOX, CIRCLE, DRAW und PAINT auch für den VDC-Chip zur Verfügung. Die Befehle LOCATE, SCALE und SCNCLR und die Funktionen RCLR, RDOT und RGT können ohne Einschränkung im 640 x 200-Grafik-Modus benutzt werden. Beide Bildschirme können gleichzeitig im HiRes-Modus arbeiten, beim Aufruf eines Grafikbefehls prüft der Interpreter selbständig, welcher Video-Chip gerade angesprochen ist.

So programmiert man den VDC

Der 8563-Video-Chip ist ein erweiterter »Abkömmling« des viel verwendeten 6845-CTR-Controllers. Er verfügt (im C 128) über einen 16 KByte Bildschirmspeicher und ist nur über zwei Register unter den Adressen 54784 ($d600) und 54785 ($d601) mit den anderen Bausteinen des Computers verbunden. Alle Register und der gesamte Video-RAM-Bereich müssen über diese beiden Adressen angesprochen werden. Das »Einblenden« des Video-RAMs in den Adreßbereich der CPU ist nicht vorgesehen. Will man einen Wert in eines der 36 Register des 8563 übertragen, so muß man zunächst die Nummer des anzusprechenden Registers in die Speicherstelle 54784 und dann den zu übertragenden Wert in die Speicherstelle 54785 schreiben. Genauso geht es beim Lesen eines Registerinhaltes: »POKE 54784, Registernummer: PRINT PEEK (54785):REM Registerinhalt«.

Die HiRes-Grafikregister

Für die Grafikprogrammierung sind von den 36 Registern des VDC vor allem die Register 18, 19, 31, 25 und 26 interessant. Der Inhalt von Register 26 bestimmt (im Grafik-Modus!) Vorder- und Hintergrundfarbe. Über Bit 0 bis 3 kann man 16 verschiedene Hintergrundfarben, über Bit 4 bis 7 16 verschiedene Vordergrundfarben anwählen. Ein Beispiel: die Befehlsfolge »POKE 54784,26: POKE 54785, (3*16+2)« bewirkt, daß im HiRes-Modus eine rote Zeichenfarbe auf weißem Grund erscheint. Im Textmodus wird mit diesem Befehl nur die Hintergrundfarbe verändert. Register 25 ist ein mehrfach belegtes Register. Über die Bits 0 bis 3 kann man den Bildschirm um maximal 16 Pixels horizontal nach links verschieben. Mit Bit 4 kann man zwischen einfacher (0) und doppelter (1) Pixelgröße wählen. Mit den Bits 5 bis 7 kann man zwischen den drei Betriebsarten Text (Bit 6=1), Semigrafik (Bit 5=1) und Grafik (Bit 7=1) wählen. Für uns ist hier nur interessant, daß man durch Beschreiben von Register 25 mit dem Wert 128 den HiRes-Modus wählen und durch Beschreiben des Registers mit dem Wert 64 wieder in den Text-Modus zurückkehren kann. Alle anderen Bits sind standardmäßig mit Null besetzt.

Organisation des Grafik-Bildschirms

Für jedes Pixel (Grafikpunkt) wird ein Bit benötigt. Das macht acht Pixel pro Byte oder 80 Bytes für eine waagrechte Bildzeile bei einer horizontalen Auflösung von 640 Pixel. In vertikaler Richtung erreicht der VDC 200 Zeilen Auflösung, das heißt insgesamt werden 80 x 200 = 16 000 Byte-Video-RAM von einem Bild belegt. Ein Farb-RAM ist nicht vorgesehen. Der VDC kann hochauflösende Grafik nur in einer einzigen Farbe darstellen, ein entscheidender Nachteil des VDC gegenüber dem VIC. Auch in der Adressierung der einzelnen Pixels unterscheiden sich VDC und VIC. Ein Byte beim VDC beschreibt acht nebenanderliegene Pixels, während ein Byte beim VIC acht untereinanderliegende Pixels kontrolliert. Anders ausgedrückt: Das Pixel mit den Koordinaten 7,0 wird von Bit 0 (!) der ersten Bildschirmspeicherstelle repräsentiert, das Pixel 0,7 von der (7 x 80=) 560sten Speicherstelle. Daraus ergibt sich die folgende Formel zur Berechnung der Bildschirmspeicheradresse für ein gegebenes Koordinatenpaar X,Y:

ADR= Yx80 + INT (X/8)

Und zur Berechnung des Bitwertes innerhalb dieser Speicherstelle erhält man (genauso wie beim VIC):

BIT= 2↑ (7-(»LOW-Byte von x«AND7))

Eine Routine »Punkt setzen« könnte man dann formulieren als:

ADR = (»Inhalt von ADR«) OR BIT

Komfortable Zeichenbefehle

Bis hierhin wurde recht ausführlich besprochen, wie man den Grafikschirm einschaltet und Punkte setzt. Nun wird sofort der Wunsch wach, Linien, Kreise oder andere Figuren zu zeichnen. Dazu sind schon recht anspruchsvolle Berechnungen nötig. Doch alle Rechenroutinen sind im Basic-ROM des C 128 enthalten. Allen Routinen ist gemeinsam, daß sie im Unterprogramm »Punkt setzen« münden. Die ROM-Routine dafür kann nur Punkte für den VIC-Bildschirm berechnen. Die oben beschriebene Routine bewirkt das gleiche für den VDC-Chip. Der Gedanke liegt nun nahe, in Routinen wie CIRCLE oder DRAW die Zeiger, die auf das Unterprogramm »Punkt setzen, löschen, testen« zeigen, auf das eigene »Punkt-setzen-Programm« zu »verbiegen«, das den VDC anspricht. Im ROM ist das zwar leider nicht möglich, und die meisten C 128-Benutzer werden ihren neuen 128er nicht gleich mit neuen EPROMs versehen wollen, aber es geht ja auch viel einfacher: Man kopiert einfach die entscheidenden Programmteile ins RAM, paßt die besagten Zeiger an und teilt zuletzt noch dem Interpreter mit, daß er fortan beim Aufruf der Grafik-Befehle die neuen Routinen »anspringen« soll, wenn der 80-Zeichen-Bildschirm eingeschaltet ist. Die Lösung dieser Aufgabe ist das abgedruckte Basic-Programm (Listing 1). Es stellt alle Programmteile zusammen, die benötigt werden, um die Befehle GRAPHIC, BOX, CIRCLE, DRAW und PAINT auch für den VDC wirksam werden zu lassen. Alle Befehle für den 640 x 200-Punkte-Bildschirm sind nur im Programm-Modus ausführbar. Das ist deshalb so eingerichtet, weil der 80-Zeichen-Bildschirm im HiRes-Modus zwangsläufig zerstört wird (siehe oben). Die Befehle könnten im Direkt-Modus also gar nicht vom Bildschirm geholt und interpretiert werden. Man erhielte lediglich »Dreckflecken« auf dem HiRes-Bild. Probieren Sie einfach einmal im Direktmodus die Befehlsfolge

»POKE 54784,25:POKE 54785,128« aus! Man kann sehr viel über die Funktionsweise des VDC im Textmodus lernen. RUN/STOP-RESTORE rückt die Register wieder zurecht.

Der neue GRAPHIC-Befehl

Der GRAPHIC-Befehl wurde um die Funktionen GRAPHIC 6,0 und GRAPHIC 6,1 erweitert. GRAPHIC 6 schaltet den 8563-HiRes-Modus ein. Folgt der »6« eine »1«, so wird der Bildschirm gelöscht, folgt eine »0« so bleibt der Bildschirm, wie er ist. Der Befehl GRAPHIC 6,1 ersetzt auch den in dieser Implementation nicht vorgesehenen Befehl SCNCLR 6.

Die Befehle BOX, CIRCLE, DRAW und PAINT

Diese Befehle funktionieren im 8563-Modus genauso wie es im Bedienungshandbuch für die VIC-Grafik beschrieben ist. Einzige Änderung: Die x-Koordinaten dürfen nun im Bereich von 0 bis 639 liegen. Auch die Implementierung dieser Befehle ist denkbar einfach: Nacheinander werden die Programmteile PAINT ($61a8 bis $62b6), BOX ($62b7 bis $6388), DRAW ($6797 bis $67d6) und CIRCLE ($668e bis $674c) in den RAM-Bereich ab $1672 kopiert (Basic-Programm Zeile 5000 bis 5340). Darunter, in den RAM-Bereich, ab $1952 wird der Programmteil »Strecke zeichnen« aus ROM $9b30 bis $9c18 kopiert. Als nächstes werden die neuen Adressen für Unterprogrammaufrufe eingesetzt (WHILE-DO-Schleife). Es folgt schließlich der neue GRAPHIC-Befehl. Im Basic-Programm ist er in Form von DATA-Zeilen abgelegt. Ebenfalls in Form von DATA-Zeilen sind die schon besprochene Routine »setpoint« (RAM $1400 bis $1671) und die Erweiterung der Interpreterschleife im Basic-Text enthalten.

Die Interpreterschleife

Wie teilen wir dem Interpreter mit, daß er beim Aufruf der Grafikbefehle nun nicht mehr zu den ROM- sondern zu unseren neuen RAM-Routinen springen soll? Beim Starten eines Programms holt sich der Interpreter — genauso wie beim C 64 — die Adresse, die in den Speicherstellen $308 und $309 abgelegt ist. Er arbeitet dann das Programm ab, das bei dieser Adresse beginnt. Setzt man in die Speicherstelle $308/09 nun die Adresse des eigenen Programms ein, dann beginnt der Interpreter nach dem Starten eines Programms mit RUN seine Arbeit bei der neuen Adresse. Eben dieses »Verbiegen« des Interpretervektors bewirkt der Befehl SYS DEC ("1303"). Und noch eine wichtige Kleinigkeit enthält die Initialisierungsroutine: die ins RAM kopierten Programmteile enthalten zahlreiche Unterprogrammsprünge (JSR) in ROM-Routinen. Der Prozessor kann diese Sprünge nur richtig ausführen, wenn ihm der RAM-Speicherbereich, in dem unser Programm liegt, und die angesprochenen ROM-Adressen als ein zusammenhängender 64-KByte-Block erscheinen. Für die Zusammenstellung solcher gemeinsamer Bereiche ist die MMU zuständig (siehe Bedienungshandbuch Anhang B). Die Initialisierungsroutine sorgt dafür, daß die MMU im Bereich von $0 bis $1fff immer die RAMs einschaltet, auf der auch unser Programm liegt. Anders ausgedrückt: Auch wenn man über den BANK-Befehl die ROM-Bank 15 ausgewählt hat, nach dem Durchlaufen unserer Initialisierungsroutine liest die CPU die Adressen zwischen $0 und $1fff immer aus dem RAM-Bereich in Bank 0 aus. Und da taucht nun gleich eine neue Schwierigkeit auf. Wenn die CPU — egal welche Bank ausgewählt wurde — im Bereich von $0 bis $1fff nur Bytes aus Bank 0 erreicht, dann können auch die auf Bank 1 in eben diesem Bereich abgelegten Variablen nicht mehr gelesen werden. Und umgekehrt, beim Anlegen neuer Variablen, würde unser Programm in Bank 0 überschrieben, weil der Interpreter nicht wissen kann, daß er, obwohl er in Bank 1 schreiben will, in Wirklichkeit doch in Bank 0 schreibt. Deshalb muß der Anfang des Variablenspeicherbereichs auf $2000 gesetzt werden. Das bedeutet den Verlust von 7 KByte Variablenspeicherbereich (mit »?FRE(1)« überprüfbar!); aber anders geht es leider nicht, wenn man ständiges, zeitraubendes Umschalten zwischen den Speicherbänken vermeiden will. Doch nun zur eigentlichen Interpreterschleife. Zunächst wird ein Zeichen aus dem Basic-Text geholt und geprüft, ob es um ein Token der in Frage kommenden Grafikbefehle handelt. Ist das nicht der Fall, so fährt das Programm einfach in der alten Interpreterschleife im ROM fort. Ist ein Grafik-Token gefunden, so testet das Programm als nächstes, ob der VIC oder ob der VDC aktiviert ist. Ist der VIC aktiv, so legt sich das Program die dem Token entsprechende ROM-Adresse zurecht (selbstveränderlicher Code hinter dem JSR von »JSR $0000«), arbeitet die normale ROM-Routine ab und kehrt in die Interpreterschleife zurück. Ist der VDC aktiv, so werden die den neuen Befehlen entsprechenden RAM-Adressen geholt.

Der aufmerksame Leser wird sich wohl schon gefragt haben, warum die Grafik-Routinen gerade in den Bereich ab $1300 »gequetscht« wurden. Ein Grund dafür wurde gerade schon genannt: Die Routinen enthalten sehr viele Sprünge in ROM-Unterprogramme, die es erforderlich machen, daß die gesamten 48 KByte ROM-I/O ab $4000 eingeblendet sind. Will man umständliche und vor allem zeitraubende Bank-Umschaltungen (JSRFAR im Kernal $ff6e) vermeiden, so muß man das Programm in den RAM-Bereich unterhalb von $4000 legen. Es soll jedoch möglich sein, auf beiden Grafik-Bildschirmen gleichzeitig zu arbeiten. Dann verbietet es sich auch noch, den Bereich von $1c00 bis $4000 zu benutzen, da dort der Farb- und der Bildschirmspeicher für die VIC-HiRes-Grafik angeordnet sind. Bleibt der einzige Bereich der freie RAM-Bereich von $1300 bis $1bff.

Das »Kochrezept«

Wer bis hier aufmerksam gelesen hat, wird gleichsam als Belohnung eine Menge interessanter Details zur Programmierung seines C 128 erfahren haben. Aber auch ungeduldige Leser sollten spätestens hier einhalten, es folgt die Bedienungsanleitung für das Programm »Graphik-80«:

1) Tippen Sie das Basic-Programm »Graphik 80« (Listing 1) sorgfältig ein! Für diese Arbeit sollten Sie sich Zeit nehmen, damit Ihnen das »DATA-Grab« am Ende des Programms nicht zur Falle wird.

2) Speichern Sie das Basic-Programm auf Diskette. Wenn Sie später Änderungen am Grafik-Paket vornehmen wollen, brauchen Sie es wieder.

3) Starten Sie das Programm mit »RUN«! Das Diskettenlaufwerk (1541 oder 1570/1571) muß beim Start des Programms eingeschaltet und mit einer Diskette versehen sein.

4) Wenn keine Fehlermeldungen aufgetreten sind, und Ihnen auch keine Fehler beim Abschreiben der DATA-Zeilen unterlaufen sind, dann befindet sich jetzt das fertige »Graphik-80«-Paket unter dem Namen »graphik-80.m« auf Diskette und fertig initialisiert im Speicher. Mit Hilfe des kleinen Testprogramms (Listing 2), das noch mit abgedruckt ist, können Sie leicht überprüfen, ob alles richtig gelaufen ist. Von jetzt ab brauchen Sie nur noch als erste (!) Programmzeile »bload"graphik-80.m":sys dec("1303")« einzugeben, und das Grafik-Paket steht zur Verfügung.

Und nun viel Freude beim »Malen«! Vielleicht vermissen Sie den Befehl CHAR im Grafik-Paket. Er wird von den Verfassern dieses Artikels gerade zusammengebaut und soll in einer späteren Ausgabe der 64'er erscheinen. Die dem C 128 mitgelieferte CHAR-Routine ist wirklich etwas zu simpel. Es können nur 40 oder 80 Zeichen in 25 Zeilen dargestellt werden, genauso wie auf dem Textbildschirm. Die in Vorbereitung befindliche Routine kann hingegen Buchstaben auf jedem beliebigen Pixel beginnen lassen. Indizes wie H_2O und Exponenten wie x^2 sind dann kein Problem mehr. Auch senkrecht schreiben, etwa zum Beschriften von Koordinatenachsen, wird dann möglich sein. Und schließlich wird noch ein Hardcopy-Programm für Epson- und ähnliche Drucker geliefert, damit Sie Ihre Bilder auch in Händen halten können.

(Thomas Rumbach/Dieter Winkler/ev)

```
1 rem "         GRAPHIK-80
2 :
3 rem "erstellt Graphik-Paket für den
4 rem " 80-Zeichen-HIRES-Bildschirm
5 rem "         des  C-128
6 :
7 rem "        Version 1.00
8 :
9 rem "        1985  by  KRW
10 :
11 :
1000 fast
1020 restore
1040 bank 15
1060 scnclr 5: print chr$(17) chr$(17) t
ab(20)"Erstellen eines Graphik-Paketes"
1080 print chr$(17) tab(17)"für den 8563
-HIRES-Graphik-Bildschirm"
1100 print chr$(17) chr$(17) chr$(17) "
 Bearbeitet:"
1120 print: print
1140 :
2000 rem "Erweiterung der Interpretersch
leife laden
2020 :
2040 print tab(25) "Neue Interpreterschl
eife
2060 a= dec("1300"): e= dec("1398")
2080 gosub 8000
2100 :
3000 rem "Neue Routine für 'Punkt setzen
, löschen und testen' laden
3020 :
3040 print tab(25) "Punkt setzen, lösche
n, testen
3060 a= dec("1400"): e= dec("1671")
3080 gosub 8000
3100 :
4000 rem "Erweiterung der GRAPHIC-Routin
e laden
4020 :
4040 print tab(25) "Erweiterung der GRAP
HIK-Routine
4060 a= dec("1a3e"): e= dec("1ab6")
4080 gosub 8000
4100 :
5000 rem "ROM-Routinen in RAM-Bereich ko
pieren
5020 :
5040 print tab(25) "ROM -> RAM - Kopie
5060 a= dec("61a8"): e= dec("6388"): rem
 "PAINT und BOX
5080 g= dec("1672")
5100 gosub 8500
5120 :
5140 a= dec("6797"): e= dec("67d6"): rem
 "DRAW
5160 if g<>dec("1853") then print "Fehle
r !": stop
5180 gosub 8500
5200 :
5220 a= dec("668e"): e= dec("674c"): rem
 "CIRCLE
5240 if g<>dec("1893") then print "Fehle
r !": stop
5260 gosub 8500
5280 :
5300 a= dec("9b30"): e= dec("9c18"): rem
 "Strecke zeichnen
5320 if g<>dec("1952") then print "Fehle
r !": stop
5340 gosub 8500
5360 :
5380 read ad$
5400 do while ad$<>"ende"
5420 read mn$, al$, ah$
5440 ad= dec(ad$)
5460 poke ad, dec(mn$): poke ad+1, dec(a
l$): poke ad+2, dec(ah$)
5480 read ad$
5500 loop
5520 :
5540 print tab(25) "Fertiges Maschinenpr
ogramm abspeichern
5560 if ds >20 then print ds$: stop
5580 bsave "graphik-80.m",d0,u8,on b0,p(
dec("1300")) to p(dec("1ab7"))
5600 if ds >20 then print ds$: stop
5620 sys dec("1303")
5640 print chr$(17) chr$(17) "     Graph
ik-Paket fertig installiert, gespeichert
 und initialisiert."
5660 end
6000 end
7960 rem "Kopierroutine 1
7980 :
8000 for i=a to e
8020 read d
8040 poke i,d
8060 next i
8080 return
8100 :
8460 rem "Kopierroutine 2
8480 :
8500 for i=a to e
8520 poke g, peek(i)
8540 g= g+1
8560 next i
8580 return
8600 :
9000 :
10000 rem "datas für neue Interpretersch
leife
10010 :
10020 data  76,45,19,76,6,19,120,169,45,
141,8,3,169,19,141,9,3,88,169,0,141,0
10030 :
10040 data 255,169,6,141,6,213,169,32,13
3,48,133,50,133,52,169,0,133,47,133
10050 :
10060 data  49,133,51,96,32,128,3,201,22
2,144,31,201,233,176,27,170,36,215,16
10070 :
10080 data  52,189,123,18,141,78,19,189,
135,18,141,79,19,138,32,128,3,32,0,0
10090 :
10100 data  76,246,74,32,134,3,76,243,74
,62,114,215,129,147,141,43,86,88,226
10110 :
10120 data 121,96,26,22,103,23,24,101,10
0,24,105,105,106,105,189,162,18,141
10130 :
10140 data  78,19,189,174,18,141,79,19,7
6,73,19,90,168,215,183,142,141,43,151
10150 :
10160 data  85,226,121,96,107,97,103,98,
```

Listing 1. »Graphik-80« (Basic-Lader). Bitte geben Sie dieses Programm im 40-Zeichen/DIN-Modus (128er-Modus) ein.

```
102,101,100,103,105,105,106,105,85
10170 :
12000 rem "datas für Routine Punkt setze
n, löschen, testen
12010 :
12020 data  76,48,22,76,9,20,76,93,22,17
3,0,255,72,169,0,141,0,255,169,7,141
12030 :
12040 data  6,213,104,141,0,255,96,173,0
,255,133,158,169,0,141,0,255,133,253
12050 :
12060 data 133,254,173,52,17,208,90,169,
199,205,51,17,144,83,169,127,237,49
12070 :
12080 data 17,169,2,237,50,17,144,71,172
,51,17,185,150,20,133,253,185,99,21
12090 :
12100 data 133,254,174,50,17,173,49,17,4
1,248,74,74,74,24,125,147,20,133,252
12110 :
12120 data 24,165,253,101,252,133,253,16
5,254,105,0,133,254,162,18,32,204,205
12130 :
12140 data 162,19,165,253,32,204,205,32,
216,205,133,252,173,49,17,41,7,170,189
12150 :
12160 data 139,20,24,96,56,96,128,64,32,
16,8,4,2,1,0,32,64,0,80,160,240,64,144
12170 :
12180 data 224,48,128,208,32,112,192,16,
96,176,0,80,160,240,64,144,224,48,128
12190 :
12200 data 208,32,112,192,16,96,176,0,80
,160,240,64,144,224,48,128,208,32,112
12210 :
12220 data 192,16,96,176,0,80,160,240,64
,144,224,48,128,208,32,112,192,16,96
12230 :
12240 data 176,0,80,160,240,64,144,224,4
8,128,208,32,112,192,16,96,176,0,80
12250 :
12260 data 160,240,64,144,224,48,128,208
,32,112,192,16,96,176,0,80,160,240,64
12270 :
12280 data 144,224,48,128,208,32,112,192
,16,96,176,0,80,160,240,64,144,224,48
12290 :
12300 data 128,208,32,112,192,16,96,176,
0,80,160,240,64,144,224,48,128,208,32
12310 :
12320 data 112,192,16,96,176,0,80,160,24
0,64,144,224,48,128,208,32,112,192,16
12330 :
12340 data 96,176,0,80,160,240,64,144,22
4,48,128,208,32,112,192,16,96,176,0
12350 :
12360 data 80,160,240,64,144,224,48,128,
208,32,112,192,16,96,176,0,80,160,240
12370 :
12380 data 64,144,224,48,128,208,32,112,
192,0,0,0,0,1,1,1,2,2,2,3,3,3,4,4,4
12390 :
12400 data 5,5,5,5,6,6,6,7,7,7,8,8,8,9,9
,9,10,10,10,10,11,11,11,12,12,12,13
12410 :
12420 data 13,13,14,14,14,15,15,15,15,16
,16,16,17,17,17,18,18,18,19,19,19,20
12430 :
12440 data 20,20,20,21,21,21,22,22,22,23
,23,23,24,24,24,25,25,25,25,26,26,26
12450 :
12460 data 27,27,27,28,28,28,29,29,29,30
,30,30,30,31,31,31,32,32,32,33,33,33
12470 :
12480 data 34,34,34,35,35,35,35,36,36,36
,37,37,37,38,38,38,39,39,39,40,40,40
12490 :
12500 data40,41,41,41,42,42,42,43,43,43,
44,44,44,45,45,45,45,46,46,46,47,47
12510 :
12520 data47,48,48,48,49,49,49,50,50,50,
50,51,51,51,52,52,52,53,53,53,54,54
12530 :
12540 data54,55,55,55,55,56,56,56,57,57,
57,58,58,58,59,59,59,60,60,60,60,61
12550 :
12560 data61,61,62,62,62,63,63,63,32,28,
20,176,34,166,131,208,5,73,255,37,252
12570 :
12580 data  44,5,252,133,252,162,18,165,
254,32,204,205,162,19,165,253,32,204
12590 :
12600 data 205,162,31,165,252,32,204,205
,165,158,141,0,255,96,32,28,20,176,245
12610 :
12620 data 37,252,240,6,32,87,22,162,0,9
6,32,87,22,162,255,96
12630 :
14000 rem "datas für Erweiterung des GRA
PHIC-Befehls
14010 :
14020 data 201,158,208,11,32,34,160,32,1
28,3,169,0,133,216,96,32,244,135,224
14030 :
14040 data 6,240,41,176,36,138,72,169,0,
141,0,255,162,25,169,64,32,204,205,32
14050 :
14060 data 12,206,165,215,72,169,128,133
,215,32,66,193,104,133,215,104,72,170
14070 :
14080 data  76,110,107,76,40,125,169,0,1
41,0,255,162,25,169,128,32,204,205,32
14090 :
14100 data 28,158,169,0,141,0,255,224,2,
176,229,224,0,240,29,160,64,132,8,160
14110 :
14120 data 0,152,162,18,32,204,205,162,1
9,32,204,205,162,31,32,204,205,136,208
14130 :
14140 data 250,198,8,208,246,96
19900 :
20000 data 1a3b, 4c, 00, 14
20010 data 16d5, 20, 00, 14
20020 data 1a2d, 20, 00, 14
20030 data 188d, 20, 1d, 1a
20040 data 19c7, 20, 1d, 1a
20050 data 19eb, 20, 0c, 1a
20060 data 1a07, 20, 0c, 1a
20070 data 1887, 4c, 6c, 18
20080 data 1890, 4c, 6c, 18
20090 data 17b1, 20, 52, 19
20100 data 17f1, 20, 52, 19
20110 data 1884, 20, 52, 19
20120 data 1948, 20, 52, 19
20130 data 1692, 20, 06, 14
20140 data 16c6, 20, 06, 14
```

Listing 1. »Graphik-80« (Fortsetzung)

```
20150 data 1712, 20, 06, 14
20160 data 1747, 20, 06, 14
20170 data 16e5, 20, 46, 17
20180 data 16fa, 20, 46, 17
20190 data 1740, 4c, b5, 16
20200 data 1869, 4c, 1d, 1a
20210 data 1672, 20, 32, 9e
20220 data 1781, 20, 32, 9e
20230 data 1893, 20, 32, 9e
20240 data 1853, ea, ea, ea
20250 data ende
50000 :
60000 zz$="graphik-80": un=8
60010 open 15,un,15,"s0:"+zz$
60020 gosub 60100
60030 save zz$,un
60040 gosub 60100
60050 verify zz$,un
60060 close 15
60070 end
60100 input#15, s1, s$, s2, s3
60110 if s1=1 then print s2; s$
60120 if s1<20 then return
60130 print s1 ", " s$ "," s2; ","
60140 close 15

ready.
```

Listing 1. »Graphik-80« (Schluß)

```
10 fast
20 bank15
30 bload"graphik-80.m"
40 :
50 sys dec("1303")
60 :
90 graphic 6,1
100 circle,320,100,250,99,70,260
110 box, 145,100, 115,110, 160, 1
112 box, 145,100, 115,110,  70, 1
120 circle ,110,45,40,15,1
130 circle,500,40,37,17,,,,45
140 circle,350,100, 160,60, 145,220
150 draw 1, 300,50 to 300,110 to 400,120
 to 300,50
160 circle0,320,100,250,99,70,260
170 circle,320,100,250,99,70,260
182 paint ,320,90
184 circle, 130,85, 200,80, 330,10
186 circle, 509,85, 200,80, 330,10
188 circle, 110,0, 100,55, 158,202
189 circle, 490,0,  97,50, 155,190
190 get q$: if q$<>"" then 200
192 circle1, 340,196, 160,63, 325,40
193 for i=1 to 500: next
194 circle0, 340,196, 160,63, 325,40
195 for i=1 to 500: next
196 goto190
200 graphic 5
220 end

ready.
```

Listing 2. Eine kleine Demonstration der Fähigkeiten von »Graphik-80«

Seekrieg per Telefon

»Seeschlacht«, ein Spiel für zwei Spieler mit zwei Akustikkopplern und zwei C 64. Gespielt wird diese Version des bekannten »Schiffe versenken« über Telefon oder ein »Nullmodem«.

Mit dem Eintritt der Akustikkoppler in die erschwinglichen Preisregionen wurde eine neue Art von Spielen möglich gemacht: Die »Play by Mail«-Spiele. Unser Listing »Seeschlacht« ist ein Vertreter dieser Gruppe. Zwei C 64 sind über Akustikkoppler und Telefon miteinander verbunden. Das Programm steuert selbständig alle Vorgänge, die für ein solches Unternehmen notwendig sind, zum Beispiel das Timing. Für diejenigen, die noch nie etwas von dem fast schon legendären Spiel »Schiffe versenken« gehört haben, hier noch einmal kurz die Regeln:

Jeder der beiden Spieler hat ein Schlachtfeld von neun mal neun Feldern Größe. Auf dieses Feld verteilt er nun waagerecht und senkrecht seine Flotte. (Der Gegner darf die eigene Aufstellung natürlich nicht sehen !). Diese besteht aus einem Flugzeugträger von vier Kästchen Länge, einem Zerstörer von drei Kästchen Länge, drei Schnellbooten von zwei Kästchen Länge und zwei U-Booten von einem Kästchen Länge. Die Schiffe dürfen sich nicht gegenseitig berühren, müssen also mindestens ein Kästchen voneinander entfernt sein. Haben beide Gegner Ihre Aufstellung gemacht, so beginnt die Schlacht: Beide Spieler schießen nun abwechselnd in die Schlachtfelder des Gegners durch die Angabe der Koordinaten des Schusses. Wer getroffen hat, darf noch einmal. Jeder Spieler hat neben seinem eigenen Schlachtfeld noch ein Feld, in das er seine abgegebenen Schüsse einträgt, um die Übersicht zu behalten, und um nicht in ein Kästchen zweimal zu schießen. In der vorliegenden Computerversion übernimmt der C 64 selbständig das Eintragen der Schüsse.

Doch nun zum eigentlichen Programm: Das Eintippen des Listings mit Hilfe des Checksummers dürfte wohl keine Probleme bereiten. Wir empfehlen allen Lesern wärmstens, das Programm mit einem leistungsfähigen Compiler zu compilieren. Ansonsten kann die Seeschlacht schnell zur Familienschlacht beim Eintreffen der Telefonrechnung werden, da die Basic-Version an manchen Stellen recht langsam ist. Die Leserservice-Diskette enthält neben der Basic- auch die schnelle compilierte Version des Spiels.

Beim Testen des Spiels in der 64'er-Redaktion hat sich folgende Vorgehensweise bewährt:

1. Ein spielwütiger C 64-Fan ruft seinen ebenso spielwütigen Freund an, und beide vereinbaren, ihre Aggressionen in einer Partie »Seeschlacht« abzukühlen.
2. Beide haben Ihren Computer neben dem Telefon stehen (beziehungsweise umgekehrt), den Akustikkoppler an den User-Port des C 64 angeschlossen und das Super-Listing aus der 64'er geladen und gestartet.