```
26a9 : ff ff ff ff 6b ff 00 ff  5e
26b1 : d5 00 00 ff 00 ff ff ff  86
26b9 : 00 00 00 ba ff ff ff ff  11
26c1 : 00 00 00 45 c1 00 e5 00  1e
26c9 : 00 00 ff c0 00 ff ff 00  e1
26d1 : 00 00 00 00 00 ff 00 00  d1
26d9 : 00 00 df 00 00 ff ff 00  d1
26e1 : 00 00 00 00 00 ff 00 00  e1
26e9 : 00 00 f5 ff 00 ff ff 00  67
26f1 : ff ff ff ff ff ff ff ff  f0
26f9 : ff ff ff ff ff ff ff ff  f8
2701 : ff ff ff ff ff ff ff ff  00
2709 : ef ff 00 ff ff ff ff fb  f0
2711 : ff ff ff ff ff ff ff ff  10
2719 : ff ff ff ff ff ff ff ff  18
2721 : ff ff ff ff ff ff ff ff  20
2729 : ff ff ff ff 6b ff 00 ff  de
2731 : d5 00 00 ff 00 ff ff ff  06
2739 : 00 00 00 ba ff ff ff ff  91
2741 : 00 00 00 45 c1 00 e5 00  9e
2749 : 00 00 ff c0 00 ff ff 00  61
2751 : 00 00 00 00 00 ff 00 00  51
2759 : 00 00 df 00 00 ff ff 00  51
2761 : 00 00 00 00 00 ff 00 00  61
2769 : 00 00 f5 ff 00 ff ff 04  ef
2771 : ff ff ff ff ff ff ff ff  70
2779 : ff ff ff ff ff ff ff ff  78
2781 : ff ff ff ff ff ff ff ff  80
2789 : ef ff 00 ff ff ff ff fb  70
2791 : ff ff ff ff ff ff ff ff  90
2799 : ff ff ff ff ff ff ff ff  98
27a1 : ff ff ff ff ff ff ff ff  a0
27a9 : ff ff ff ff 6b ff 00 ff  5e
27b1 : d5 00 00 ef 00 ff ff ff  84
27b9 : 00 00 00 ba ff ff ff ff  11
27c1 : 00 00 00 45 c1 00 e5 00  1e
27c9 : 00 00 ff c0 00 ff ff 00  e1
27d1 : 00 00 00 00 00 ff 00 00  d1
27d9 : 00 00 df 00 00 ff ff 00  d1
27e1 : 00 00 00 00 00 ff 00 00  e1
27e9 : 00 00 f5 ff 00 ff ff 00  67
27f1 : ff ff ff ff ff ff ff ff  f0
27f9 : ff ff ff ff ff ff ff ff  f8
2801 : ff ff ff ff ff ff ff ff  00
2809 : ef ff 00 ff ff ff ff fb  f0
2811 : ff ff ff ff ff ff ff ff  10
2819 : ff ff ff ff ff ff ff ff  18
2821 : ff ff ff ff ff ff ff ff  20
2829 : ff ff ff ff 6b ff 00 ff  de
2831 : d5 00 00 ff 00 ff ff ff  06
2839 : 00 00 00 ba ff ff ff ff  91
2841 : 00 00 00 45 c1 00 e5 00  9e
2849 : 00 00 ff c0 00 ff ff 00  61
2851 : 00 00 00 00 00 ff 00 00  51
2859 : 00 00 df 00 00 ff ff 00  51
2861 : 00 00 00 00 00 ff 00 00  61
2869 : 00 00 f5 ff 00 ff ff ff  e7
```

Listing 1. »Ascompiler 64« (Schluß)

```
10 REM  HIRES-PLOT-DEMO
20 REM FUER ASCOMPILER 64
30 REM
60000 POKE 53265,59
60010 POKE 53272,24
60011 POKE 20000,128
60012 POKE 20001,64
60013 POKE 20002,32
60014 POKE 20003,16
60015 POKE 20004,8
60016 POKE 20005,4
60017 POKE 20006,2
60018 POKE 20007,1
60020 LET I=1024
60030 POKE I,1
60040 LET I=I+1
60050 IF I<2023 THEN 60030
60060 LET I=8192
60070 POKEI,0
60075 LET I=I+1
60080 IF I<16384 THEN 60070
60090 GOTO 63000
61000 LET W=Y/8
61010 LET V=X/8
61020 LET H=W*40
61030 LET H=H+V
61040 LET H=H*8
61050 LET H=H+8192
61060 LET G=W*8
61070 LET G=Y-G
61080 LET P=H+G
61100 IF P>16384 THEN 63900
61110 LET B=X/8
61120 LET B=B*8
61130 LET B=X-B
61140 LET Q=20000+B
61150 LET Q=PEEK(Q)
61155 LET L=PEEK(P)
61156 LET L=LORQ
61160 POKE P,L
61170 RETURN
63000 LET X=0
63005 LET Y=X*X
63010 LET Y=Y+X
63020 LET Y=Y/200
63030 GOSUB 61000
63040 LET X=X+1
63050 IF X<400 THEN 63005
63900 LET Q=PEEK(203)
63910 IF Q=64 THEN 63900
63940 POKE 53265,27
63950 POKE 53272,23
63960 PRINT"J";
63970 END

READY.
```

Listing 2. Ein Mini-Plot-Programm für den Ascompiler 64

```
1 REM  SPLITSCREEN-DEMO
2 REM FUER ASCOMPILER 64
3 REM
10 LET A=PEEK(53266)
20 IF A>0 THEN 10
30 POKE 53281,2
40 LET A=PEEK(53266)
50 IF A<128 THEN 40
60 POKE 53281,3
70 LET A=PEEK(197)
80 IF A=64 THEN 10

READY.
```

Listing 3. Dieses »Screen-Split«-Demo zeigt die mit dem Ascompiler erreichbaren Geschwindigkeiten

```
1 REM DREI NUETZLICHE ROUTINEN
2 REM FUER DEN ASCOMPILER 64
3 REM
7 REM
8 REM GET Z
9 REM
10 LET Z=PEEK(198)
11 IF Z=0 THEN 14
12 LET Z=PEEK(631)
13 POKE 198,0
14 RETURN
17 REM
18 REM PRINT CHR$(Z)
19 REM
20 POKE 2,Z
21 SYS 65490
22 RETURN
27 REM
28 REM INPUT(Z)
29 REM
30 SYS 42336
31 LET Z=512
32 RETURN

READY.
```

Listing 4. Drei nützliche Unterroutinen

Hardcopy leicht gemacht

Jetzt bekommt Ihr Drucker was zu tun! Wir zeigen Ihnen, wie Sie die Basic-Hardcopy-Routine aus der Ausgabe 9/85 in Maschinensprache umsetzen können. Ein gedrucktes »Foto« des Textbildschirms auf Tastendruck.

Eine Hardcopy vom Textbildschirm des C 64 wird häufig gebraucht. Die Dokumentation eines selbstgeschriebenen Programms oder ein Beispielausdruck für eine Programmanleitung sind nur zwei Beispiele.

In der letzten Folge war die Hardcopy-Routine auf Commodore-Drucker zugeschnitten. Diesmal sind alle 8-Nadel-Drukker an der Reihe. Das Programm ist auf die Epson-Drucker FX und RX zugeschnitten. Ein verwendetes Interface muß sich durch die Sekundäradresse 1 im OPEN-Befehl in den Linearmodus schalten lassen. Lassen Sie sich davon aber nicht abschrecken, falls Ihr Interface oder Drucker auf andere Weise angesteuert wird. Wenn Sie Grundkenntnisse in der Assembler-Programmierung haben, sind Sie in der Lage, das Programm auch an Ihren Drucker anzupassen. Voraussetzung für das Gelingen der Anpassung ist nur, daß Ihr Drucker über einen Bitmustermodus mit acht ansteuerbaren Nadeln verfügt.

Wenn Sie keine Änderungen am Programm vornehmen möchten, sollten Sie den MSE-Lader (Listing 1) abtippen. Wollen Sie aber Änderungen und Verbesserungen daran vornehmen, verwenden Sie besser das Quellisting (Listing 2) und einen Assembler. Aber das werden Sie dann sowieso machen. Da in jeder Zeile nur ein einziger Befehl vorkommt, sollte das Quellisting ohne weiteres auf jeden Assembler, beispielsweise dem Hypra-Ass, übertragen werden können. Es müssen ja prinzipiell nur die Assembler-Anweisungen in den Zeilen 1000, 1010, 1100, 1110, 4070 und 4080 geändert werden. Die Bedeutung dieser Zeilen:

1000 Programmdatei für Maschinen-Code öffnen
1010 Starten des Assemblers
1100 Maschinen-Code in offene Datei schreiben
1110 Startadresse des Programms

4070 Ende des Quell-Codes
4080 Initialisierung der Hardcopy-Routine (nur sinnvoll, wenn der Maschinen-Code in den Speicher geschrieben wird und nicht auf Diskette gespeichert wird).

Hardcopy des Text-Bildschirms

Aber nun Schluß mit den allgemeinen Hinweisen. Im folgenden wird das Source-Listing anhand der Zeilennummern erklärt. Ab Zeile 1150 im Sourcelisting (Quellisting), werden den benötigten Betriebssystem-Routinen und Speicherzellen Labels (Namen) zugeordnet. Was die einzelnen Routinen leisten, entnehmen Sie bitte den Kommentaren im Quellisting.

Eine Hardcopy-Funktion für Textbildschirme wird erst dann so richtig interessant, wenn sie durch einfachen Tastendruck gestartet werden kann. Unabhängig davon, was der Computer gerade macht. Denn nur so kann man sich Beispiele von Bildschirmmasken etc. drucken lassen. Aus diesem Grund wurde dieser Hardcopy-Routine eine Tastenabfrage vorgeschaltet. So wird erreicht, daß durch kurzes Drücken der F1-Taste das Hardcopy-Programm startet. Wie man eine solche Tastenabfrage realisieren kann, sehen Sie ab Zeile 1460 im Quellisting. Dort wird die Abfrage-Routine in den Systeminterrupt des C 64 eingebunden. Danach wird die Taste pro Sekunde etwa 60mal abgefragt. Ist sie nicht gedrückt, wird der normale Systeminterrupt fortgesetzt. Zum Einbinden einer Routine in den Systeminterrupt (IRQ) muß der Interruptvektor in den Speicherzellen $314 und $315 auf die Routine »umgeleitet« werden. Im Normalfall zeigt der IRQ-Vektor auf den Beginn der Systeminterrupt-Routine bei $EA31. In diesem Fall wird er so verändert, daß er auf die Tastenabfrage, also auf $C00D zeigt. Genaueres zu diesem Thema können Sie beispielsweise im 64'er Sonderheft 4/85 nachlesen.

Die Veränderung des IRQ-Vektors wird in den Zeilen 1470 bis 1550 vorgenommen. Bevor aber der Vektor verändert werden kann, ist ein SEI(set-interrupt-disable-flag)-Befehl nötig. Dieser Befehl setzt das IRQ (interrupt-request)-Flag. Das bewirkt, daß der Prozessor keinen Interrupt mehr annimmt. Es ist ja eigentlich einleuchtend, daß kein Interrupt stattfinden darf, wenn man gerade die IRQ-Sprungadresse verändert. Das wäre so, als ob Sie auf einer Kreuzung ein Umleitungsschild aufstellen wollen, wenn dort gerade ein Lastzug fährt. Sie müssen vorher schon den Verkehr anhalten. Die Umstellung des IRQ-Vektors geschieht in diesem Programm durch SYS 49152. Dadurch wird die ganze Hardcopy-Routine initialisiert. Von nun an springt der Prozessor bei jedem Interrupt in die Tastenabfrage ab Zeile 1570. Dort wird der Inhalt der Speicherzelle $C5 geladen und mit »4« verglichen. »4« ist der Tastencode der F1-Taste. Fällt der Vergleich negativ aus, erfolgt sofort ein Sprung zur Adresse $EA31, der Systeminterrupt-Routine des C 64 (Zeile 1640). Fällt der Vergleich dagegen positiv aus (F1 gedrückt), verzweigt die Abfrage-Routine ins eigentliche Hardcopy-Programm ab Zeile 1700.

Start auf Tastendruck

Am Anfang der Hardcopy-Routine wird als erstes die gesamte Zeropage (Speicherzellen 0-255) zwischengespeichert. Dieses Verfahren ist zwar alles andere als elegant, erspart aber sehr viel Denkarbeit. Der Zeitbedarf dafür ist so gering, daß er bei einer Hardcopy-Routine vernachlässigt werden kann.

Nach dem »Retten« der Zeropage wird der Druckerkanal eröffnet und der Drucker normiert (Zeile 1670 bis 2000). Der Druckerkanal hat die Filenummer 126 und die Sekundäradresse 1. Die Geräteadresse ist 4. Die File-Nummer wurde so hoch gewählt, damit es keine Schwierigkeiten mit eventuell schon vorhandenen offenen Dateien gibt. Wer verwendet in Programmen schon die File-Nummer 126? Achtung: Eine File-Nummer größer als 127 sendet ein Linefeed (Zeilenvorschub) »CHR$(10)« nach jedem Carriage Return.

Ist der Druckerkanal offen, wird der Zeilenabstand des Druckers so eingestellt, daß alle Zeilen dicht aneinander liegen. Beim Epson RX/FX geschieht das mit der Steuersequenz CHR$(27)"3"CHR$(24). Mit der PRINT-Routine wird ein Zeichen an das, durch die CHKOUT-Routine festgelegte, Gerät geschickt. In diesem Fall eben an den Drucker.

Ab Zeile 2040 wird die Anfangsadresse des Bildschirms geholt und in VIRAM und VIRAM+1 übergeben. Das Programm muß ja wissen, wo die zu druckenden Daten stehen. Nach dieser Vorarbeit steht der eigentlichen Hardcopy-Routine nichts mehr im Weg.

In Zeile 2140 wird das x-Register mit der Zeilenzahl (25) des Bildschirms geladen. Danach wird die STOP-Taste abgefragt, um die Hardcopy vorzeitig beenden zu können. Die Abfrage findet übrigens zu Beginn jeder neu zu druckenden Bildschirmzeile statt. Ist die STOP-Taste gedrückt, wird die Hardcopy-Routine beendet und die normale Interrupt-Routine abgearbeitet. Wie die Hardcopy-Routine beendet wird, soll später erklärt werden. Vorläufig wird erstmal gedruckt!

Damit der richtige Zeichensatz aufs Papier kommt, wird ab Zeile 2390 bei jeder neuen Zeile ein Prüfprogramm (ab Zeile 3820, CHRTEST) aufgerufen, das den momentanen Schriftmodus überprüft. Groß-/Klein oder Groß-/Grafik-Zeichensatz. Die Startadresse des Zeichensatzes im Character-ROM wird dann in ZROM, ZROM+1 abgelegt. Der Groß/Klein-Zeichensatz hat die Startadresse $D800, der Groß/Grafik-Zeichensatz steht ab Adresse $D000. Welcher Zeichensatz gewählt ist, steht in Speicherzelle $D018 des Videochips. Ergibt der Inhalt von $D018 AND 2 das Ergebnis 2, ist der Groß/Grafik-Modus eingeschaltet. In Zeile 3890 finden Sie den BIT-Befehl $2C. Dieser Befehl vergleicht normalerweise den Akkuinhalt mit der angegebenen 2-Byte-Adresse und verändert entsprechend das Z-Flag. Diese Funktion ist aber hier nur Mittel zum Zweck. In diesem Fall soll der Prozessor nur die auf $2C folgenden 2 Bytes ignorieren. Genauer: Wird der Akku mit dem High-Byte der Startadresse des Zeichensatzes 1 geladen (Zeile 3870), wird der 2-Byte-Ladebefehl LDA # >CHRGEN2 (Startadresse Zeichensatz #2) einfach ignoriert. Man erspart sich dadurch eine zusätzliche Abfrage und einen Branch-Befehl.

Nachdem der aktuelle Zeichensatz bekannt ist, wird der Drucker in den Bitmustermodus (Grafikmodus) geschaltet. Dazu werden aus der Tabelle GRAFIK sieben Steuercodes geholt und zeichenweise an den Drucker geschickt. Die Steuersequenz ist 24, 13, 27, "*", 4, 64, 1. »1« und »64« ist das High- beziehungsweise das Low-Byte von 320, der Anzahl der Grafikpunkte des C 64 in einer Zeile. Der Code »13« bewirkt nur einen Wagenrücklauf (Carriage Return), damit der Druckkopf am Zeilenanfang steht. CHR$(24) löst einen Drucker-Reset aus.

In zwei Schleifen wird nun der ganze Bildschirm »abgetastet« und die einzelnen Zeichen am Bildschirm in Bitmusterdaten übersetzt. Den Anfang der übergeordneten »Zeilenschleife« lernten Sie bereits kennen: LDX #25, STOP-Taste abfragen, Druckerzeile initialisieren (in Grafikmodus schalten). Die untergeordnete »Spaltenschleife« (Zeilen 2530 bis 2650) holt nacheinander den Bildschirmcode der Zeichen einer Zeile. Der Zeichencode wird in der Speicherzelle ZEICHEN abgelegt. Als Zählvariable für diese untergeordnete Schleife, die von 0 bis 39 zählt, wird das y-Register hergenommen. Der Abschnitt von Zeile 2250 bis 2330 überprüft, ob in einer Zeile überhaupt etwas steht. Wenn nicht, wird ein Zeilenvorschub ausgelöst und die nächste Zeile »abgetastet«.

Ist ein Zeichen-Code ermittelt, wird mit JRS AUSWERT in

Zeile 2620 ein Unterprogramm aufgerufen, das aus dem Zeichensatz-ROM die Punktmuster der Zeichen liest. Dazu werden nacheinander die acht senkrechten Punktspalten eines Zeichens aus den Daten des Zeichensatz-ROMs zusammengesetzt. Jede fertige Punktspalte wird an den Drucker geschickt. Aus acht dieser Spalten entsteht dann ein Zeichen auf dem Papier, aus 320 solcher Spalten eine ganze Druckzeile. Das Lesen des Zeichensatz-ROMs und der Zusammenbau der Punktspalten ist für den Anfänger in Maschinensprache nicht ganz einfach zu verstehen und soll deshalb kurz erklärt werden.

Das Unterprogramm AUSWERT beginnt ab Zeile 3060 mit drei PHA (push akku)- und zwei Transferbefehlen, die den Inhalt des Akkus und des x-,y-Registers auf den Prozessorstack legen. Das ist nötig, da im Unterprogramm der Akku und das x- und y-Register verändert werden. Danach wird in den Zeilen 3160 bis 3290 die Startadresse des momentanen Zeichens im Zeichensatz-ROM berechnet. Die Startadresse berechnet Anfangsadresse + Offset (Speicherstelle im Zeichensatz-ROM). Offset = Bildschirm-Code x 8.

Wie schon erwähnt, wird ab Zeile 2530 der »Spaltenschleife« der Bildschirmcode eines jeden Zeichens am Bildschirm geholt (LDA (VRAM,Y)) und in ZEICHEN gespeichert. Der Inhalt von ZEICHEN wird in Zeile 3170 in die Speicherzelle ADRESSE kopiert. Dann wird durch drei ASL-Befehle der Inhalt Speicherzelle ADRESSE und ADRESSE+1 (Grundwert immer »0«) mit 8 multipliziert. Ein ASL (arithmical shift left) verdoppelt den Inhalt einer Speicherzelle. Tritt dabei ein Übertrag auf (Ergebnis > 255) wird das Carry-Flag gesetzt. Das Carry-Flag wird in der Speicherzelle ADRESSE+1 berücksichtigt. Ein ROL-Befehl schiebt das Carry-Bit »von rechts in die Speicherzelle ADR+1«. Fand kein Übertrag statt, schiebt der ROL-Befehl einfach eine »0« nach. Der ROL-Befehl hat die gleiche Wirkung, wie eine Verdopplung eines Speicherzelleninhalts mit ASL, nur wird anschließend noch das Carry-Bit addiert. Nach der Multiplikation mit 8 steht in ADRESSE und ADRESSE+1 in Low-/High-Bytedarstellung der Offset. Um die absolute Adresse zu erhalten, wird ab Zeile 3110 der Offset zur Anfangsadresse des Zeichensatz-ROMs addiert. Es steht dann die absolute Zeichenadresse in ADRESSE (Low-Byte) und ADRESSE+1 (High-Byte).

Der Clear-Carry (CLC)-Befehl ist grundsätzlich vor jeder Addition notwendig, um ein eventuell gesetztes Carry-Flag zu löschen. Nur so kann ein Übertrag sicher festgestellt werden.

Ist die Anfangsadresse eines Bildschirmzeichens im Zeichensatz-ROM bekannt, fängt aber die Arbeit erst richtig an.

Ein Bildschirmzeichen setzt sich aus einer 8x8-Punktmatrix zusammen. Die Punktmatrix wird beim C 64 aus acht 8-Bit-Zahlen zusammengesetzt:

Adresse		Adresse	
$D000	. . . . .	$D005	. . . . .
$D001	. . . . .	$D006	. . . . .
$D002	. . . . .	$D007	. . . . .
$D003	. . . . .	$D008	. . . . .
$D004	. . . . .	$D009	. . . . .

Mit einer 8-Bit-Zahl pro Punktzeile. Dabei bestimmt die erste Zahl im ROM das Punktmuster der obersten Punktzeile eines Zeichens. Stellen Sie sich eine 8-Bit-Zahl einmal in Binärform vor. Eine »1« ist dann ein gesetzter Punkt und eine »0« kein Punkt. So könnte die Zahl 0000 1000 (dez. 8) einen i-Punkt wiedergeben. Wie Sie sehen, sind die Bildschirmzeichen aus Punktzeilen aufgebaut; im Gegensatz zu den Druckerzeichen, die aus Punktspalten zusammengesetzt werden. Die beiden Formate bringen ein Problem mit sich, denn die Bitmusterdaten des Bildschirms müssen in die des Druckers übersetzt werden. Diese Konvertierung wird in den Zeilen 3330 bis 3660 vorgenommen.

Doch bevor man die Bitmusterdaten konvertiert, muß erstmal eine Punktzeile aus dem Zeichensatz-ROM gelesen werden. Wie Sie vielleicht wissen, liegt das Zeichensatz-ROM im $D000-Bereich, dem kompliziertesten Speicherteil des C 64. Im Bereich von $D000 bis $DFFF arbeiten nämlich noch alle I/O-Bausteine. Man spricht in diesem Zusammenhang von Speicherebenen, die sich einen Speicherbereich teilen. Damit der Zeichensatz gelesen werden kann, muß also noch die richtige Speicherebene selektiert werden. Doch dazu später mehr. Zuerst soll die Konvertierung der Bitmusterdaten geklärt werden.

Zur Konvertierung der Zeilenwerte in Spaltenwerte liest man nacheinander acht mal die Zeilenwerte des Zeichensatz-ROMs und vergleicht die Werte mit einer »Bit-Maske«. Die acht Bit-Masken sind:

Maske bin.	Wertigkeit
1000 0000	128
0100 0000	64
0010 0000	32
0001 0000	16
0000 1000	8
0000 0100	4
0000 0010	2
0000 0001	1

Beim ersten Durchgang hat die Bitmaske die Wertigkeit 128 (Zeile 3360). Mit dieser Maske werden nun nacheinander alle acht Punktzeilen eines Bildschirmzeichens AND-verknüpft. Ist beispielsweise in der Zeilenzahl das achte Bit gesetzt (Zahl >127), ist das Ergebnis der Verknüpfung 128, also größer 0. Liefert die Verknüpfung »0«, wird der nächste der acht Zeilenwerte mit der Maske verglichen. Ist das Ergebnis einer Verknüpfung größer als 0, muß an dieser Stelle eine Druckernadel anschlagen.

Bei einem 8-Nadeldrucker haben die Nadeln die binären Werte 1,2,4,8,16,32,64,128. Die unterste Nadel hat die Wertigkeit 1, die oberste die Wertigkeit 128. Ausnahmen bestätigen aber auch hier die Regel (Seikosha 550A: Die untere Nadel hat die Wertigkeit 128). Damit der Drucker die richtige Nadel anschlägt, muß die Wertigkeit der Nadel, oder die Summe der Wertigkeiten, dem Drucker geschickt werden. Dazu ordnet man den acht Punktzeilen eines Bildschirmzeichens die binären Wertigkeiten 1 bis 128 zu; entsprechend der Nadelanordung des Druckers. Bei jedem positiv ausgefallenen Maskenvergleich, addiert man die entsprechenden Nadelwertigkeiten. Wurden alle acht Zeichenzeilen auf diese Weise »abgetastet«, wird die Summe an den Drucker geschickt, der daraufhin eine senkrechte Punktreihe druckt.

Um die nächste Punktspalte drucken zu können, dividiert man die Maske mit »2«. Am einfachsten mit einem LSR (logical shift right)-Befehl. Dieser Befehl verschiebt die »1« in der »Binärmaske« um eine Stelle nach links, was eben einer Division mit 2 gleichkommt. Nun vergleicht man wieder die acht Zeilenwerte mit der neuen Maske und addiert die Nadelwertigkeiten. Nach acht Vergleichen ist der Druckercode für die zweite Druckspalte addiert und kann zu Papier gebracht werden. Dieses Spielchen wiederholt man, bis alle acht Punktspalten eines Zeichens gedruckt sind. Für ein einziges Zeichen sind insgesamt 64 Vergleiche nötig.

Als Basic-Programm wäre die Konvertierung der Bitmusterdaten des C 64 in die eines Druckers viel zu langsam. Es sind schließlich 64x40x25=64000 Konvertierungen pro Bildschirm nötig. Ebenso ist die »Bit-Schieberei« mit LSR, OR und AND in Basic ein Problem für sich.

Die Speicherebenenumschaltung auf das Zeichensatz-ROM findet in Zeile 3450 statt, nachdem zuvor das Interrupt-

Flag gesetzt wurde. Denn solange Speicherzelle 1 den Wert $33 enthält, dürfen keine I/O-Operationen des Prozessors, wie Tastenabfrage und Cursor-Blinken, erfolgen. Der Prozessor würde unweigerlich abstürzen, wenn er auf das Zeichensatz-ROM zugreift, anstelle auf einer der beiden CIAs oder dem VIC-Chip.

In Zeile 3480 wird das Zeichensatz-ROM ausgelesen und in Zeile 5335 mit der aktuellen Maske verglichen. Danach wird

Fortsetzung auf Seite 177

```
1000 open2,8,2,"hardcopy.obj,p,w"
1010 sys9*4096
1020 ;*******************************
1030 ;*      lowscreenhardcopy      *
1040 ;*            incl.            *
1050 ;*   grafik- und reverszeichen  *
1060 ;*     epson/wiesemann & komp.  *
1070 ;*         version 1.2          *
1080 ;*    harald meyer 21.05.1985   *
1090 ;*******************************
1100 .opt o2  ; code auf disk
1110 *= $c000 ; startadresse
1120 ; routinen des betriebssytems
1130 ;*********************************
1140 ;
1150 open    = $ffc0 ; file oeffnen
1160 setnam  = $ffbd ; filenamen setzen
1170 setfls  = $ffba ; fileparameter
1180 print   = $ffd2 ; zeichen ausgeben
1190 clrch   = $ffcc ; bildsch.-ausgabe
1200 chkout  = $ffc9 ; ausgabegeraet
1210 close   = $ffc3 ; file schliessen
1220 stop    = $ffe1 ; stopvektor
1230 irqend  = $ea31 ; kernal-irq-rout.
1240 lovideo = $d018 ; videoram lo
1250 chrgen1 = $d000 ; 1.zeichensatz
1260 chrgen2 = $d800 ; 2.zeichensatz
1270 ;
1280 ; benoetigte speicherzellen
1290 ; *****************************
1300 ;
1310 cursor  = 204   ; cursor aus/an
1320 irqvek=$0314;irq-vektor
1330 taste   = $c5   ; letzte taste
1340 f1      = 4     ; f1-matrixnummer
1350 cr      = 13    ; carriage return
1360 esc     = 27    ; escape
1370 adresse = $f8   ; zeichenadresse
1380 zeichen = $d6   ; zeichencode
1390 viram   = $15   ; zeichenadresse
1400 zrom    = $f9   ; zeichengenerator
1410 maske   = $9d   ; bit-abfrage
1420 ;
1430 ; interrupt initialisieren
1440 ; *****************************
1450 ;
1460 sei             ; irq verhindern
1470 lda #<start     ; irq vektor auf
1480 ldy #>start     ; dieses programm
1490 sta irqvek
1500 sty irqvek+1
1510 cli             ; irq freigeben
1520 rts             ; init. ende
1530 ;
1540 ;
1550 ; f1 gedrueckt, dann start
1560 ; *****************************
1570 ;
1580 start lda taste ; tastenabfrage
1590 cmp #f1         ; taste = f1 "?"
1600 bne 11          ; nein, dann ende
1610 lda #00         ; tastendruck
1620 sta taste       ; loeschen
1630 jsr hdcopy      ; programmbeginn
1640 11  jmp irqend  ; zum kernal-irq
1650 ;
1660 ;
1670 ; zeropage retten
1680 ; *****************************
1690 ;
1700 hdcopy ldx #$ff ; byte 255-0
1710 12  lda 0,x     ; laden
1720 sta memory,x    ; und speichern
1730 dex             ; naechstes byte
1740 bne 12          ; x=0, dann ende
1750 lda #01
1760 sta cursor      ;ausschalten
1770 ;
1780 ;
1790 ; druckerfile oeffen und
1800 ; zeilenabstand initialisieren
1810 ; *****************************
1820 lda #126     ; filenummer
1830 ldx #4       ; geraeteadresse
1840 ldy #1       ; sekundaeradresse
1850 jsr setfls   ; parameter setzen
1860 lda #0       ; kein filenamen
1870 jsr setnam   ; namen setzen
1880 jsr open     ; file oeffnen
1890 ldx #126     ; alle
1900 jsr chkout   ; ausgaben auf #126
1910 ;
1920 ; drucker auf einzeiligen abstand
1930 lda #esc     ; esc-sequenz-beginn
1940 jsr print    ; chr$(27) an drucker
1950 lda #"3"     ; "3"+chr$(24)
1960 jsr print
1970 lda #24
1980 jsr print    ; an drucker
1990 ;
2000 ; zeiger auf videoram
2010 ; *****************************
2020 ;
2030 videoram lda #$00
2040 ldy $288     ; videoramadressen
2050 sta viram    ; uebergeben
2060 sty viram+1
2070 ;
2080 ;
2090 ; zeichen vom screen holen,
2100 ; charaktergenerator lesen und
2110 ; zeichenmatrix zusammensetzen
2120 ; *****************************
2130 ;
2140 ldx #25      ; zeilenanzahl
2150 ;
2160 ; ausgabeschleife
2170 ; *****************************
2180 ;
2190 ausg jsr stop; stoptaste abfragen
2200 beq hdende   ; gedrueckt,dann ende
2210 ;
2220 ; ist zeile leer "?"
2230 ; *****************************
2240 ;
2250 ldy #39
2260 19 lda (viram),y
2270 cmp #32
2280 bne 110
2290 dey
2300 bpl 19
2310 lda #cr
2320 jsr print
2330 jmp 111
2340 ;
2350 ; chrgen-adresse holen
2360 ; gross/klein oder gross/grafic
2370 ; *****************************
2380 ;
2390 110 jsr chrtest ;schriftmodus-test
2400 ;
2410 ; druckzeile vorbereiten
2420 ; *****************************
2430 ;
2440 ldy #7            ; 6 codes
2450 13  lda grafik,y  ; esc-sequenz
2460 jsr print         ; senden
2470 dey               ; naechster code
2480 bne 13            ; fertig "?"
2490 ;
2500 ; neue zeile beginnen
2510 ; *****************************
2520 ;
2530 ldy #0  ; zeile von neuem
2540 ;
2550 ; zeile abarbeiten, 2. schleife
2560 ; zeichen holen und im up bearbei.
2570 ; *****************************
2580 ; zeichen v. bildsch. holen
2590 15  lda #0    ; zeichen von
2600 lda (viram),y; bildschirm holen
2610 sta zeichen   ; merken
2620 jsr auswert   ; bearbeiten
2630 iny           ; spalte erhoehen
2640 cpy #40       ; zeilenende "?"
2650 bne 15        ; bildsch.-ende "?"
2660 ;
2670 ; neue zeile vorbereiten
2680 ; *****************************
2690 ;
2700 111 lda #40; 40 spalten
2710 clc    ; zeiger auf naechste zeile
2720 adc viram     ; neue zeile setzen
2730 sta viram
2740 bcc 112
2750 inc viram + 1
2760 112 dex       ; zeilen erniedrigen
2770 bne ausg      ; screen zu ende "?"
2780 hdende lda #13; zum ende cr an
2790 jsr print     ; drucker senden
2800 ;
2810 ; fertig, dann file schliessen
2820 ; *****************************
2830 ;
2840 lda #126 ; filenummer
2850 jsr close; druckerdatei schliessen
2860 jsr clrch; cmd auf screen
2870 ;
2880 ;
2890 ; zeropage wiederherstellen
2900 ; *****************************
2910 ;
2920 ldx #$ff        ; zeropage
2930 14  lda memory,x; wieder
2940 sta $00,x       ; herstellen
2950 dex
2960 bne 14
2970 ;
2980 rts             ; hardcopy ende
2990 ;
3000 ;
3010 ;
3020 ;
3030 ; charaktergenerator lesen
3040 ; *****************************
3050 ;
3060 auswert pha;  register retten
3070 txa
3080 pha
3090 tya
3100 pha
3110 ;
3120 ; adresse im charakterram berech.
3130 ; = zeichencode * 8
3140 ; *****************************
3150 ;
3160 lda zeichen; zeichencode laden
3170 sta adresse; adresse im zeichen-
3180 lda #0     ; rom feststellen
3190 sta adresse+1
3200 asl adresse; zeichencode
3210 rol adresse+1
3220 asl adresse
3230 rol adresse+1
3240 asl adresse
3250 rol adresse+1 ; mal 8
3260 lda adresse+1
3270 clc           ; und
3280 adc zrom+1    ; romadresse
3290 sta adresse+1 ; addieren
3300 ;
3310 ; charaktergen. lesen und zeichen-
3320 ; matrix fuer drucker aufbereiten
3330 ; *****************************
3340 ;
3350 ldx $01        ; speicherselekt
3360 lda #%10000000 ; bit-maske
3370 sta maske      ; speichern
3380 114 lda #$00   ; code
3390 pha            ; loeschen
3400 ldy #7         ; 8 bytes abfragen
3410 ;
3420 ; chr-ram einschalten & byte lesen
3430 ; *****************************
3440 ;
3450 115 sei     ; irq sperren
3460 lda #01:and #251      ; zeichen-rom
3470 sta $01     ; selektieren
3480 lda (adresse),y ; code holen
3490 and maske       ; maskenvergleich
3500 ;
3510 ; chr-rom wieder einschalten
3520 ; *****************************
3530 ;
3540 stx $01     ; norm speicherkonf.
3550 cli         ; wieder herstellen
3560 beq 16
3570 pla         ; code laden und
3580 clc         ; code laden und
3590 adc nwert,y ; bitwert zu code add.
3600 pha         ; und merken
3610 16  dey     ; naechstes byte
3620 bpl 115     ; 8 bits gelesen "?"
3630 pla         ; ja, dann
3640 jsr print   ; an drucker
3650 lsr maske   ; maske erhoehen
3660 bcc 114     ; naechste punktzeile
3670 ;
3680 ; register wieder holen
3690 ; *****************************
3700 ;
3710 pla ; register wieder holen
3720 tay
3730 pla
3740 tax
3750 pla
3760 rts
3770 ;
3780 ; anfangsadresse des charakterrams
3790 ; feststellen und merken
3800 ; *****************************
3810 ;
3820 chrtest lda #$00
3830 sta zrom     ; zeichensatzadr. lo
3840 lda lovideo  ; zeichensatz-page
3850 and #%00000010
3860 bne 116
3870 lda #>chrgen1          ;$d000
3880 sta zrom + 1
3890 .byt $2c
3900 116 lda #>chrgen2      ;$d800
3910 sta zrom + 1
3920 rts            ; zum hauptprogramm
3930 ;
3940 ;
3950 ;
3960 ;tabelle 320 bitmusterdaten
3970 grafik .byt 0,$01,$40
3980 ;
3990 ;epson auf einzelnadelansteuerung
4000 .byt 4,"*",esc,cr,24
4010 ;
4020 ;tabelle fuer bit-wertigkeiten
4030 nwert .byt 128,64,32,16,8,4,2,1
4040 ;
4050 ;zeichensatz startadressen
4060 memory .byt 0
4070 .end
4080 sys49152

ready.
```

Quelltext der Hardcopy-Routine

Und wo lassen Sie reparieren?

In unserer Reparaturumfrage wollten wir wissen, was Sie im Falle eines Falles mit Ihrem defekten Gerät machen. Die zahlreiche Teilnahme zeigte, daß sich viele C 64-Besitzer über diese Frage Gedanken gemacht haben — hier sind einige Ergebnisse der Umfrage.

Computer gehören mit zu den zuverlässigsten elektronischen Geräten, die man überhaupt kaufen kann. Trotzdem kann es natürlich vorkommen, daß durch statische Entladung, Blitzschlag oder Fehlbedienung ein Computer nicht das tut, was er soll. Nicht ganz unschuldig sind auch manche Basteleien, die schon einigen Computern einen überdurchschnittlichen Alterungsprozeß beschert haben. In solchen Fällen ist es immer gut, wenn man einen schnellen und leistungsfähigen Reparatur-Service im Adreßbüchlein stehen hat, denn gerade bei einem Computer können falsche Reparaturen viel Schaden anrichten.

Ziel dieser Umfrage war es, festzustellen, wie der Reparatur-Service noch verbessert werden könnte. Am häufigsten wurde dabei der Wunsch geäußert, so kurz wie möglich auf den Computer (oder das Peripheriegerät) verzichten zu müssen.

Gleichzeitig sollte ein Reparaturservice natürlich preiswert sein. Viele Leser betrachten den Service auch in zunehmendem Maße als Kriterium für die Auswahl des Computers, sei es beim Erstkauf oder bei der Anschaffung eines leistungsfähigeren Systems. Als Ideallösung ist das früher übliche, aber bei steigender Verbreitung der Heim- und Personal Computer etwas in Vergessenheit geratene Austauschgerät anzusehen.

Wir sind sicher, daß Ihre Meinung nicht unberücksichtigt bleibt, denn ein guter Service ist der halbe Verkauf und gehört eigentlich zum Computer wie das Handbuch. Wir werden auf die entsprechenden Stellen einwirken, um eine Verbesserung der Reparatursituation in Deutschland zu erwirken, damit Sie in Zukunft nicht länger als unbedingt notwendig auf Ihr Gerät zu warten brauchen. Möglicherweise schaffen wir es sogar, den guten Brauch des Austauschgeräts wieder ins Gespräch zu bringen, denn nur so läßt sich Computer-Ausfallzeit vermeiden.

Für Ihre rege Mitarbeit gab es auch einiges zu gewinnen. Der Hauptpreis ist ein nagelneuer C 128, die Preise 2 bis 11 sind je ein Anwenderprogramm von Commodore. Der Gewinner des C 128 ist Herr Wilfried Christen aus Dattenberg — herzlichen Glückwunsch!

Je ein Commodore Magic Desk haben gewonnen: Harald Pinnow, Torsten Stracke, Andreas Urban, Sören von Otte. Je ein Calc Result von Commodore geht an Jürg Brik, Farine Robin. Die Gewinner der beiden Datenmanager sind Manfred Müller und Thomas Trenkwaldor. Die Programme Text 64 mit Adress 64 gehen je einmal an Erwin Wahsenbruck und Werner Braun. Ihnen allen herzlichen Glückwunsch und viel Freude mit Ihren neuen Programmen.

Alle Gewinner werden schriftlich benachrichtigt. Wir danken der Firma Commodore Büromaschinen GmbH für die Preise. (aw)

Die Computer und die Programme sind vom Umtausch ausgeschlossen.

Fortsetzung von Seite 67

der ursprüngliche Inhalt von Speicherzelle 1 zurückgelassen und ein Interrupt wieder erlaubt.

Die Addition der Nadelwertigkeiten der Punktzeilen findet in Zeile 3590 mit einem ADC-Befehl statt. Die Wertigkeiten der Nadeln stehen in der Tabelle NWERT am Schluß des Programms. Nachdem alle acht Werte im Akku summiert sind, positive Maskenvergleiche vorausgesetzt, wird der Akkuinhalt an den Drucker geschickt. Der Befehl JSR PRINT schickt das Punktmuster der Druckzeichenspalte an den Drucker. Sind die acht Punktreihen gedruckt, werden sämtliche Register wieder hergestellt und die Unterroutine AUSWERT wird beendet. Der RTS-Befehl führt dann wieder in die Spaltenschleife zu Zeile 2510.

Sind auf diese Weise 40 Zeichen gedruckt, wird in der Zeilenschleife das y-Register wieder auf »0« gesetzt, der Drucker auf eine neue Druckzeile mit Bitmusterdaten eingestellt und eine neue Druckzeile abgearbeitet.

Ist der ganze Bildschirm ausgedruckt (oder die STOP-Taste gedrückt), wird ab Zeile 2980 das Programm beendet. Im einzelnen heißt das, daß der Druckerkanal geschlossen, die Zeropage zurückgeschrieben und die normale Interruptroutine bearbeitet wird.

Danach läuft Ihr Programm weiter als wenn nichts geschehen wäre. (hm)

```
 programm : hardcopy.obj  c000 c12f
-----------------------------------

c000 : 78 a9 0d a0 c0 8d 14 03  73
c008 : 8c 15 03 58 60 a5 c5 c9  c9
c010 : 04 d0 07 a9 00 85 c5 20  f7
c018 : 1d c0 4c 31 ea a2 ff b5  fe
c020 : 00 9d 2e c1 ca d0 f8 a9  1d
c028 : 01 85 cc a9 7e a2 04 a0  a2
c030 : 01 20 ba ff a9 00 20 bd  86
c038 : ff 20 c0 ff a2 7e 20 c9  aa
c040 : ff a9 1b 20 d2 ff a9 33  19
c048 : 20 d2 ff a9 18 20 d2 ff  d4
c050 : a9 00 ac 88 02 85 15 84  df
c058 : 16 a2 19 20 e1 ff f0 3f  6a
c060 : a0 27 b1 15 c9 20 d0 0b  9a
c068 : 88 10 f7 a9 0d 20 d2 ff  48
c070 : 4c 91 c0 20 09 c1 a0 07  e8
c078 : b9 1e c1 20 d2 ff 88 d0  a6
c080 : f7 a0 00 a9 00 b1 15 85  ea
c088 : d6 20 b7 c0 c8 c0 28 d0  49
c090 : f2 a9 28 18 65 15 85 15  a3
c098 : 90 02 e6 16 ca d0 bc a9  1f
c0a0 : 0d 20 d2 ff a9 7e 20 c3  08
c0a8 : ff 20 cc ff a2 ff bd 2e  68
c0b0 : c1 95 00 ca d0 f8 60 48  7c
c0b8 : 8a 48 98 48 a5 d6 85 f8  af
c0c0 : a9 00 85 f9 06 f8 26 f9  bf
c0c8 : 06 f8 26 f9 06 f8 26 f9  c8
c0d0 : a5 f9 18 65 fa 85 f9 a6  36
c0d8 : 01 a9 80 85 9d a9 00 48  36
c0e0 : a0 07 78 a9 01 29 fb 85  ab
c0e8 : 01 b1 f8 25 9d 86 01 58  67
c0f0 : f0 06 68 18 79 26 c1 48  61
c0f8 : 88 10 e7 68 20 d2 ff 46  b4
c100 : 9d 90 da 68 a8 68 aa 68  f2
c108 : 60 a9 00 85 f9 ad 18 d0  fd
c110 : 29 02 d0 05 a9 d0 85 fa  3c
c118 : 2c a9 d8 85 fa 60 00 01  b4
c120 : 40 04 2a 1b 0d 18 80 40  64
c128 : 20 10 08 04 02 01 00 07  ac
```

MSE-Listing der Hardcopy-Routine