sem Programm eine »heile« Hardcopy erstellen. Wenn diese Hardcopy-Routine Bilder doppelt groß drucken kann (Hi-Eddi!), hat man erstmalig die Möglichkeit, einen LoRes-Bildschirm im Riesenformat auf Papier zu bringen.

Zweites Beispiel:

Wer schon einmal ein schönes Grafikbild mit Hi-Eddi erstellt hat, weiß, wieviel Arbeit das macht. Oft kann man die grobe, geometrische Struktur mit einem Maskengenerator (64'er, Ausgabe 8/85) erstellen. Da man sein Werk jetzt in Hi-Res übertragen kann, ist für die Feinarbeit mit Hi-Eddi schon gute Vorarbeit geleistet.

Drittens: Man kann sogar ein LoRes-Bild, das mit einem anderen Zeichensatz erstellt wurde, weiterverarbeiten! Der Konvertierungsroutine muß nur die Basisadresse des neuen Zeichensatzes mitgeteilt werden. Das High-Byte wird in die Speicherstelle 828+55 geschrieben, das Low-Byte muß $00 sein, braucht also nicht extra geändert zu werden.

Das Basic-Programm (Listing 8) ist äquivalent zu dem Maschinen. Interessant ist der Zeitvergleich: In Basic dauert die Hi-Res-Übertragung 2:13 min, in Maschinensprache 0,4 s! Die Basic-Version ist daher weniger zum Abtippen, sondern vielmehr zum leichten Verstehen des Algorithmus gedacht.

(Martin Boels/tr)

```
programm : lores zu hires  033c 03ba
-------------------------------------

033c : a9 00 8d 0e dc a9 33 85  fd
0344 : 01 a9 04 85 f8 a9 00 85  b3
034c : f7 85 f9 a9 20 85 fa a2  19
0354 : 04 a0 00 8a 48 98 48 a9  b7
035c : 00 85 fc b1 f7 85 fb 06  3c
0364 : fb 26 fc 06 fb 26 fc 06  63
036c : fb 26 fc 18 a5 fc 69 d0  46
0374 : 85 fc a2 00 a0 07 b1 fb  21
037c : 91 f9 88 10 f9 18 a5 f9  19
0384 : 69 08 85 f9 90 02 e6 fa  3c
038c : 68 a8 68 aa c8 d0 c4 e6  ac
0394 : f8 ca d0 bd a9 37 85 01  49
039c : a9 01 8d 0e dc 20 fd ae  0f
03a4 : 20 d4 e1 a9 20 85 fa a9  49
03ac : 00 85 f9 aa a9 f9 a0 40  b0
03b4 : 20 d8 ff 60 01 08 00 00  9d
```

Listing 7. »LoRes zu HiRes«. Verwandelt den Textbildschirm in eine HiRes-Grafik

```
10 POKE 56334,0:POKE 1,51                          <109>
20 FOR I=1024 TO 2023:BA=53248+PEEK(I)*8           <072>
30 FOR J=0 TO 7:POKE 8192+Z,PEEK(BA+J):Z=Z
   +1:NEXT J,I                                     <207>
40 POKE 1,55:POKE 56334,1                          <175>
```

Listing 8. Der grundlegende Algorithmus von Listing 7

Tips & Tricks zum C 128

Hauptsächlich neue POKEs und interessante Programmiertricks möchten wir Ihnen diesmal anbieten. Auch die Nutzung des kompletten deutschen Zeichensatzes im C 64-Modus ist nicht zu verachten.

Der C 128 ist im Kommen. Sein hervorragendes Basic macht ihn zu einem immer beliebteren Aufsteigercomputer. Aber gerade seine (im Vergleich zum C 64) komplexe, interne Struktur mit unter anderem drei Mikroprozessoren macht den C 128 zu einer Fundgrube für die Tips & Tricks-Ecke. Übrigens: Wenn Sie etwas Neues herausgefunden haben, schreiben Sie uns! Jeder brauchbare Trick wird veröffentlicht!

Programmieren in Z80-Assembler

Einige frischgebackene C 128-Besitzer möchten jetzt auch einmal in Z80-Assembler programmieren. Um den Z80-Mikroprozessor einzuschalten, muß das Bit 0 in Speicherstelle $fd505 auf 1 gesetzt werden. Doch wo beginnt der Prozessor seine Arbeit und was ist sonst noch zu beachten?

Zunächst müssen die System-IRQs gesperrt werden. Weiterhin sollte man sich in Bank 0 befinden (dort wird später das Z80-ROM von $0000 bis $1000 eingeblendet). Nachdem dies geschehen ist, kann man endlich das Bit 0 in $fd505 auf »High« legen. Nach dem Abfallen des Systemtaktes der 8502-CPU beginnt nun endlich der Z80 seine Arbeit ab der Adresse $0ffed. Hinter dem dort stehenden »NOP« findet sie einen Sprungbefehl nach $0008 in das nun eingeschaltene ROM. Das dort stehende Programm bootet normalerweise die CP/M-System-Diskette.

Um in eigene Programme zu gelangen, muß man vor dem Einschalten den Sprungbefehl verbiegen. Dazu schreibt man in die Speicherstelle $0ffee ein $c3 (entspricht dem »JP«-Befehl) und in die folgenden beiden Stellen die Startadresse im Low/High-Format.

Listing 1 zeigt ein Programm, das die oben genannten Punkte, außer dem Verändern des Sprungbefehls, für Sie ausführt. Nach dem Abarbeiten des Z80-Programms sollte man nach $0ffe0 springen. Dort wird die 8502-CPU reaktiviert (Näheres in Listing 2). Beide Programme stehen sofort nach Einschalten des Computers zur Verfügung.

Zum Schluß möchte ich noch auf einen Fehler im Betriebssystem hinweisen: Betätigt man, während der 40-Zeichen-Bildschirm gescrollt wird, die ASCII/DIN-Taste, so erscheint sehr oft in etwa der Mitte des Bildschirms ein Zeichensalat. Disassembliert ergeben die ASCII-Werte ein Programm, das auch in der erweiterten Zeropage zu finden ist.

(Frank Probst/tr)

»PRINT AT« oder »CHAR«?

Zunächst möchte ich zu dem Artikel »Entdeckungsreise durch den C 128« (64'er, Ausgabe 12/85) Stellung nehmen: Ein Simulieren des »PRINT AT«-Befehls, wie in diesem Artikel beschrieben, ist nicht nötig: Er ist im Basic 7.0 im Prinzip enthalten. Er wird mit dem »CHAR«-Befehl angesprochen, der nicht nur im Grafik-Modus wirksam ist. Format: char, Spalte, Zeile, Beispiel:

```
10 char,5,5,"hallo" : char,15,15,"....",1
15 char,20,20 : input "eingabe";a
```

Der eingebaute Sprite-Editor des C 128 besitzt außer den im Handbuch aufgeführten Anweisungen noch den Befehl »COPY«. Drückt man in der SPRDEF-Ebene die Taste »C«, so erscheint unten die Frage »COPY FROM ?«. Es wird dann die Nummer des Sprite angegeben (1-8), dessen Bitmuster in das momentan bearbeitete Raster kopiert werden soll. So lassen sich an verschiedenen Sprites geringfügige Änderungen

leicht anbringen. Hat man den Sprite-Editor aufgerufen, haben die Funktionstasten F1 bis F8 folgende Bedeutung: F1 = C, F2 = A, F5 = A, F6 = RETURN, F7 = RETURN, F8 = M

Der eingebaute Maschinensprache-Monitor kann zusätzlich zu den im Handbuch beschriebenen Funktionen die vier möglichen Zahlensysteme (Hexadezimal, Dezimal, Binär, Oktal) ineinander umrechnen und anzeigen. Dazu gibt man, wenn man den Monitor aufgerufen hat, nur die umzuwandelnde Zahl mit ihrem entsprechenden Kennsymbol ein (»$«, »+«, »&«, »%« siehe Handbuch, Anhang C-3).

Im Kapitel »Organisation der Zeropage« des Handbuches ist die Anfangsadresse des Tastaturpuffers nicht erwähnt. Dieser liegt bei 842 (dezimal) in der Bank 0 und hat normalerweise die Länge 10 (Länge in Speicherstelle dez. 2592, Bank 0). Der Tastaturpuffer ist in Verbindung mit der Speicherstelle, die die Anzahl der Zeichen im Tastaturpuffer enthält (dez. 208, Bank 0), wichtig, um vom Programm aus Funktionen auszuführen, die sonst nur im Direktmodus möglich sind (zum Beispiel die Übernahme von neuen Programmzeilen). Dieser Programmiertrick ist schon vom C 64 und VC20 her bekannt.

Neben den im Handbuch auf Seite 4 bis 6 erwähnten CTRL-Funktionen gibt es im C 128-Modus noch folgende:

```
CTRL - S   verbietet Bildschirmaufrollen (entspricht »NO
           SCROLL«-Taste)
CTRL - Q   erlaubt Bildschirmaufrollen, löst »Cursor down« aus
CTRL - T   entspricht DELETE-Taste
CTRL - M   entspricht RETURN-Taste
CTRL - J   entspricht LINE FEED-Taste
CTRL - R   Revers on
```

(Frank Mülheims/tr)

Neue SYS' und POKEs

Mit »SYS 65341« läßt sich beim C 128 ein Reset erreichen. Mit »SYS 65357« kommt man vom C 128-Modus in den C 64-Modus, ohne Sicherheitsabfrage. Einen recht guten Programmschutz erreicht man mit folgenden POKEs:

```
POKE 802,0:POKE803, 224
```

In ein Programm eingebaut, lösen diese POKEs nach dem Drücken der RUN/STOP-Taste einen Reset aus. Wer Spaß an Bildschirmeffekten hat, sollte einmal folgendes Programm ausprobieren:

```
10 FOR I = 1 TO 15: SYS 51914: NEXT
20 FOR I = 1 TO 15: SYS 51900: NEXT
30 GOTO 10
```

(Andreas Ligendza/tr)

Programmschutz-POKEs

Wie man beim C 64 über »POKE 808,225« die RUN/STOP- und die RESTORE-Taste abschalten kann, geht dies beim C 128 über POKE 808,98. Der große Nachteil ist, daß damit die interne Uhr (TIME,TI,TIME$,TI) unbrauchbar wird. Doch durch einen Reset und gleichzeitiges Drücken von RUN/STOP kann man in den Monitor gelangen, diesen verlassen, und ans Programm kommen, da es nicht gelöscht ist. Dies umgeht man über BANK 1:POKE 65528,3. Nach diesem POKE führt ein Reset nur zum Absturz. Dieser POKE funktioniert selbstverständlich auch ohne das Abschalten von RUN/STOP-RESTORE, aber beide POKEs zusammen ergeben einen recht guten Schutz. Wer auf TIME$ (TI$) und TIME (TI) nicht verzichten will, kann mit POKE792,51:POKE793,255 die Tastenkombination RUN/STOP-RESTORE abschalten, die RUN/STOP-Taste alleine funktioniert aber weiterhin. Die letzten POKEs eignen sich vor allem dann, wenn man ein Basic-Programm compiliert: nach dem Compilieren sind sie ohnehin gegen RUN/STOP (ohne RESTORE) geschützt, und durch die beiden letzten POKEs wird auch noch die Kombination RUN/STOP-RESTORE und/oder durch BANK1:POKE65528,3 der RESET verhindert.

(Florian Müller/tr)

Einfache Spritesteuerung

```
10 SPRDEF:SPRITE 1,1
20 J=(JOY(2)-1)*45:IFJ=-45THENMOVSPR1,0#0:GOTO20:
   ELSE:MOVSPR1,J#7:GOTO20
```

Dieser Einzeiler ermöglicht eine einfache (aber schnelle) Joystickabfrage. Statt »IF...THEN«-Anweisungen werden die Joystickrichtungen in Grad umgewandelt und später von einem »MOVSPR x,x#x«-Befehl verarbeitet.

Zum Programm:

In Zeile 10 wird das Sprite definiert (SPRDEF) und eingeschaltet.

In Zeile 20 wird in der Variablen J die Richtung des Joysticks —1*45 (360 Grad / 8 Richtungen = 45 Grad) abgelegt. Ist J=—45 (bei Nullstellung), wird das Sprite gestoppt und das Programm springt wieder nach Zeile 20.

Ist J nicht —45, bewegt sich das Sprite in Richtung J (mit »MOVSPR 1,j# Geschwindigkeit« (die Geschwindigkeit ist beliebig von 2-15 wählbar). Zum Schluß startet das Programm erneut von Zeile 20. (Thorsten Wanschura/tr)

Deutscher Zeichensatz im C 64-Modus

Wenn man im C 64-Modus die »CAPS-LOCK«-Taste drückt, stehen alle internationalen Zeichen zur Verfügung, die sonst nur vom C 128-Modus aus erreichbar sind. Dabei werden einige Grafikzeichen mit den internationalen Zeichen belegt. Die CHR$ Tabelle sieht nun folgendermaßen aus:

```
CHR$ (172) = é   CHR$ (187) = ä   CHR$ (174) = è   CHR$ (123) = Ä
CHR$ (177) = µ   CHR$ (188) = ö   CHR$ (178) = à   CHR$ (124) = Ö
CHR$ (179) = ù   CHR$ (189) = ü   CHR$ (180) = â   CHR$ (125) = Ü
CHR$ (181) = ê   CHR$ (190) = ß   CHR$ (182) = î   CHR$ (191) = ^
CHR$ (183) = ô   CHR$ (192) = à   CHR$ (184) = û   CHR$ (64)  = §
CHR$ (185) = |/  CHR$ (96)  = á   CHR$ (186) = Σ
```

(Carsten Schmidt/tr)

Listing 1:

```
$0FFD0    SEI           ;IRQ aus
$0FFD1    LDA #$3E      ;Wert für »I/O ein«
$0FFD3    STA $FF00     ;in Konf. Register
$0FFD6    LDA #$B0      ;Wert für »Z80 ein«
$0FFD8    STA $D505     ;in Mode Konf. Reg. Kurz: Einschalten
$0FFDB    NOP
$0FFDC    JMP $1100     ;muß auf eigene Routine verbogen werden

Wichtig: Die Z80-CPU beginnt ihre Arbeit bei $0FFED!
```

Listing 1. So wird die Z80-CPU eingeschaltet

Listing 2:

```
$0FFE0    DI            ;IRQ aus
$0FFE1    LD A,$3E      ;Wert für »I/O ein«
$0FFE3    LD $FF00,A    ;in Konf. Reg.
$0FFE6    LD BC,$D505   ;Mode Konfigurations Register
$0FFE9    LD A,$B1      Wert für »8502 ein«
$0FFEB    OUT (C),A     ;Einschalten
$0FFED    NOP
$0FFEE    RST $08       ;Weiter bei $0008

Wichtig: Die 8502-CPU beginnt ihre Arbeit bei $0FFDB!
```

Listing 2. Ein Z80-Programm, um die 8502-CPU wieder einzuschalten