Tips & Tricks für Profis

Neben Beiträgen über Listschutz bringen wir zwei erstaunliche Programme, die den Rasterzeilen-Interrupt ausnutzen. Weiterhin eine Joystick-Bastelei und ein Programm zur Zahleneingabe mit dem Joystick.

Gibt es den perfekten Listschutz? Sicherlich nicht. Das Programm »H.I.D.E.« von Frank Hund (siehe weiter unten) stellt die zur Zeit wohl beste Lösung dar. Aber gibt es noch bessere Möglichkeiten? Wo liegen die Grenzen eines Listschutzes (ohne Autostart)? Hat der Basic-Interpreter außer der <SHIFT+L>-Macke noch andere Programmierfehler? Kennen Sie als ausgefuchster C 64-Profi noch eine andere Methode? Wenn ja, schicken Sie uns diese! Der beste Profi-Listschutz wird unter Tips & Tricks veröffentlicht. (tr)

Fehler im LIST-Befehl

Wie Sie sicher wissen, ist folgende Programmzeile gegen den LIST-Befehl insofern geschützt, als beim Listen dieser Zeile anstelle des geSHIFTeten L ein »SYNTAX ERROR« auftritt.

```
10 REM <SHIFT+L>
```

Dieser Listschutz ist einer der einfachsten und hat entsprechend oft — vor allem bei Einsteigern — Verwendung gefunden. Um ein auf diese Weise geschütztes Programm problemlos listen zu können, ist folgender Einzeiler (Befehlsabkürzungen verwenden, Leerzeichen nicht eintippen) vor dem Listen einzugeben:

```
POKE 95,0:POKE 96, 160:POKE 90,0:POKE 91,192:POKE 88,0:POKE 89,192:SYS 41919:POKE 1,54:POKE 42816,144
```

Dann erscheint statt der »SYNTAX ERROR«-Meldung der Basic-Befehl END. Als Profis wollen wir uns aber nicht damit zufriedengeben, den Listschutz zu umgehen (wofür das Programm »LIST-KNACKER — Und er LISTet doch!« aus dem 64'er-Sonderheft 2/86 »Tips & Tricks« viel besser geeignet ist), sondern vor allem klären, warum dieser Listschutz funktioniert und wie er behoben wird.

Schlußteil der Befehlswörter-Tabelle:

```
:a146  4e c4 4f d2 be bd bc 53   nDoR...s
:a14e  47 ce 49 4e d4 41 42 d3   gNinTabS
:a156  55 53 d2 46 52 c5 50 4f   usRfrEpo
:a15e  d3 53 51 d2 52 4e c4 4c   SsqRrnDl
:a166  4f c7 45 58 d0 43 4f d3   oGexPcoS
:a16e  53 49 ce 54 41 ce 41 54   siNtaNat
:a176  ce 50 45 45 cb 4c 45 ce   NpeeKleN
:a17e  53 54 52 a4 56 41 cc 41   str.vaLa
:a186  53 c3 43 48 52 a4 4c 45   sCchr.le
:a18e  46 54 a4 52 49 47 48 54   ft.right
:a196  a4 4d 49 44 a4 47 cf 00   .mid.go.
.d a738 a747                          Null-Byte
,a738   b9 9e a0   lda   a09e,    y; Zeichen
,a73b   30 b2      bmi   a6ef      ; holen und
,a73d   20 47 ab   jsr   ab47      ; ausgeben
,a740   d0 f5      bne   a737      ←
***Hier endet »LIST« und beginnt »FOR«
,a742   a9 80      lda   #80
,a744   85 10      sta   10
,a746   20 a5 a9   jsr   a9a5      ; ruf »LET«-
                                   Routine auf
```

Diese Teile des Basic-Interpreters sind für den Fehler verantwortlich

Verantwortlich ist ein kleiner Fehler in der LIST-Routine, den wir mit Hilfe des SMON genau aufzeigen wollen. Mit »M A09E A19E« können Sie die Tabelle der Befehlswörter, welche der Computer zur Decodierung der Tokens in Klartext einsetzt, ausgeben lassen, mit »D A717 A741« die Umwandlungsroutine, der das Token im Akkumulator übergeben wird. Zunächst wird aus dem Token ermittelt, ab welcher Position in der Tabelle ab $A09E das benötigte Befehlswort steht. Betrachten wir nun das Bild 1, eine Hardcopy von einem Bildschirm, auf dem die verantwortlichen Teile des ROMs stehen.

Bei $A738 wird dann Zeichen für Zeichen das Befehlswort ausgegeben, da dort die Position in der Tabelle bereits im Y-Register steht. Wie man am Hex-Dump der Befehlswörter-Tabelle sieht, ist im letzten Zeichen eines Kommandos Bit 7 gesetzt. Diese Endmarkierung wird von $A73B erkannt, andernfalls wird das Zeichen bei $A73D ausgegeben. Nun kommt die kritische Stelle. Normalerweise ist das auszugebende Zeichen nicht CHR$(0) und somit das Z-Flag nach Ausgabe des Zeichens gelöscht, wodurch der BNE-Befehl bei $A740 einem JMP gleichkommt (mehr über diesen und andere Programmiertricks in »Effektives Programmieren in Assembler«, 64'er-Assembler-Sonderheft 8/85, Seite 74).

Bei $CC, dem Code, der durch das geSHIFTete L erzeugt wird, gibt es allerdings eine Ausnahme: nach der Umwandlung des Tokens in den Offset steht in der Tabelle bei $A738 im Y-Register der Wert $FF (dies können Sie mit dem Trace-Befehl des SMON nachvollziehen, indem Sie in den Akku den Wert $CC schreiben, von $A717 bis $A738 TRACEn lassen und dabei den Wert des Y-Registers beobachten), wodurch der Inhalt von $A19D ausgelesen wird. Wie das Hex-Dump zeigt (siehe Bild), steht dort ein Null-Byte!

Die Folgen ergeben sich ganz logisch: bei $A740 ist das Z-Flag gesetzt, da der Akku den Wert 0 hatte, und es erfolgt keine Verzweigung, was aber zur fehlerfreien Funktion unbedingt notwendig wäre. Statt dessen wird die im Speicher direkt hinter der »LIST«-Routine liegende »FOR«-Routine abgearbeitet (ab $A742), die ihrerseits die Parameter des »FOR«-Befehls einzuholen versucht (bei $A746 wird der »LET«-Befehl zur Definition der Schleifenvariablen aufgerufen). Da wir aber nach »LIST« keine »FOR«-Parameter eingeben können, vermißt der Interpreter solche: SYNTAX ERROR!

Nun wissen wir, wie es zu dieser Fehlermeldung kommt, die einen Abbruch des Listens bewirkt. Mit Hilfe des SMON fanden wir heraus, wie man diese Genauigkeit der LIST-Routine behebt. Wenn das Interpreter-ROM in das an gleicher Adresse liegende RAM kopiert wird, kann man den BNE-Befehl bei $A740 in einen »BCC« umwandeln. Dieser Verzweigungsbefehl funktioniert einwandfrei, da bei der Bildschirmausgabe ($A73D JSR $AB47) kein I/O-Fehler vorkommt und von daher das C-Flag immer gelöscht ist.

Genau diese Änderungen nimmt der oben vorgestellte Einzeiler vor, der im übrigen durch <RUN/STOP RESTORE> wieder aufgehoben werden kann. Dadurch wird nämlich wieder das alte (und fehlerhafte) Basic-ROM eingeschaltet.

Anmerkung der Redaktion: Unser Floppyspeeder »64'er-DOS« bringt bei einem List-Versuch des genannten Schutzes weder »Syntax Error« noch »REMEND«, sondern ein <SHIFT+L>, wie es ja auch richtig sein sollte.

(Florian Müller/tr)

<RESTORE> — Die Prügeltaste

Bei allen (zumindest uns bekannten) C 64-Computern braucht man bei der <RESTORE>-Taste einen sehr harten Anschlag, um überhaupt eine Reaktion zu erhalten. Da dieser durchaus störende Umstand nicht etwa auf einem Fehler in der Tastaturmechanik beruht, haben wir uns einmal den zuständigen Teil des Schaltplanes angesehen. Demnach befindet sich in der Restore-Leitung ein Kondensator C 38 (zu finden an der vorderen, linken Ecke der Computerplatine), der von Commodore mit 51 pF eindeutig zu klein dimensioniert

ist, um das nachfolgende Timer-IC zu triggern. Nach Austausch dieses Kondensators gegen einen einfachen 10-nF Keramiktyp ist die <RESTORE>-Taste ebenso leichtgängig wie jede andere Taste auch. (Arne Wohlfart/tr)

Musik am User-Port

Dieses Programm gibt einen Ton in der gewünschten Frequenz am User-Port aus. Als Vorbereitung nimmt man einen Lautsprecher und verbindet ihn mit den Anschlüssen GND und Pb6. Dann gibt man folgendes Programm ein:

```
10 POKE 56590,7
20 INPUT"Hertz :";he
30 h=985248.4/he:hi=INT(h/256):lo=h-hi*256
40 POKE 56580,lo:POKE 56581,hi
```

Erklärung:

In Zeile 10 wird der Timer A des CIA2 gestartet und ein Unterlauf an Pb6 des User-Ports ausgegeben. In Zeile 20 wird die Frequenz des Tons eingegeben. Zeile 30 rechnet die Frequenz mit Hilfe der Taktfrequenz (985248,4 kHz) in Timerwerte um. In Zeile 40 werden diese Timerwerte in die Zählregister von Timer A gePOKEt.

Zur Umrechnung der Frequenz:

Nehmen wir an, wir wollen alle 60stel Sekunde einen Interrupt (also einen Timerunterlauf) verursachen, wie dies zum Beispiel beim Systeminterrupt der Fall ist. Dann berechnen wir die Timerwerte nach folgender Rechnung:

Timerwerte = Taktfrequenz * 1/t, dabei ist t die Anzahl der Unterläufe. Für unser Beispiel würde das bedeuten:

Timerwerte = Taktfrequenz * 1/60 bei einer Taktfrequenz von 985248.4 gibt das als Timerwert 16420.8. Diese Zahl wird in LSB und MSB zerteilt und in die Zählregister gePOKEt. Nun wird alle 60stel Sekunde ein Unterlauf erzeugt. Dieser Unterlauf wird am User-Port ausgegeben (an Pb6) und erzeugt im Lautsprecher ein Knacken. Aufgrund der hohen Frequenz erscheint dieses Knacken als gleichmäßiger Ton.

Zum Bereich der Frequenz sei gesagt, daß er von 16 Hz bis zirka 20000 Hz reicht. Nach oben hin wird die Tonausgabe immer ungenauer, da die Stufung der Zählregister in 255er-Schritten zu ungenau ist.

Zur Ergänzung sei gesagt, daß diese Tonausgabe an einem weiteren Pin des User-Ports möglich ist: Es ist dies der Pin Pb7. Dabei müssen nur die entsprechenden Register von Timer B des CIA2 angesteuert und ein zweiter Lautsprecher angeschlossen werden. (Dirk Trossen/tr)

Tempo für die RS232-Schnittstelle

Wie inzwischen bekannt, lassen sich alle Parameter der internen RS232-Schnittstelle durch ein entsprechendes Bit-Muster im Steuerregister (665) und Kontrollregister (666) einstellen. Die Übergabe der Parameter erfolgt beim Öffnen der Schnittstelle mit einem OPEN-Befehl unter der Gerätenummer 2.

Möchte man nun in einem laufenden Programm die Bedingungen Parität, Anzahl der Stopp-Bits und Datenwortlänge ändern, so kann dies durch Verändern der entsprechenden Bitwerte im Steuer- und Kontrollregister jederzeit geschehen. Will man jedoch die Übertragungsgeschwindigkeit in Bit pro Sekunde wechseln und setzt die korrespondierenden Bits neu, so stellt man fest, daß dies ohne Wirkung bleibt. Eine andere Geschwindigkeit läßt sich nur bei einem erneuten Initialisieren mit OPEN... einstellen. Dieser Befehl löscht aber gleichzeitig alle Variablen, so daß dann sowieso das Programm abgebrochen wird.

Eine Untersuchung der ROM-Routinen brachte die Ursache zutage. Beim Eröffnen der Schnittstelle werden die Speicherzellen $299 und $29A mit Konstanten versehen, die je nach eingestellten Bit pro Sekunde aus einer Tabelle entnommen werden. Diese Konstanten steuern den Timer im 6522 und damit die Taktrate. Bei der Bearbeitung der Schnittstelle werden jetzt nur noch die Speicherzellen $299 und $29A ausgewertet, daher bleibt eine nachträgliche Änderung im Steuerregister wirkungslos.

Das Problem läßt sich dadurch lösen, indem man nachträglich noch einmal den Teil der OPEN-Routine anspringt, wo die Konstanten der Übertragungsgeschwindigkeit zugeordnet werden. Eine geeignete Einsprungstelle ist die Adresse 62499 ($F423). Man ändert also die entsprechenden Bits im Steuerregister und fügt anschließend an: SYS 62499.

(Dieter Laues/tr)

EOR in Basic

Wir haben eine Möglichkeit gefunden, in Basic ein »exclusiv-oder« zu simulieren. Dazu muß im Programm einfach die Zeile »DEF FNa(e)=((NOTx)ANDy)OR((NOTy)ANDx)« erscheinen. Dann setzt man die Variablen x und y auf die zu verknüpfenden Werte und ruft die Funktion mit einem beliebigen Parameter für e auf. Die Idee kam aus einem Digitalelektronikbuch, in dem eine EXOR-Schaltung durch Oder-, Nicht- und Und-Gatter aufgebaut wurde. (Jens Vöcker/tr)

»Epson-Support« ohne Line-Feed

Im Sonderheft 6/86 ist ein Grafik-Programm abgedruckt, (»Epson-Support«), in dessen Beschreibung angegeben wird, daß Auto-Line-Feed abgeschaltet werden muß. Ich habe das Programm abgetippt, und sann sofort auf Abhilfe. Das Ergebnis lautet: POKE50326,32 und ALF kann weiter eingeschaltet bleiben. (Alfred Jäger/tr)

Die Musik von »Elite«

Ich habe einen Tip für alle diejenigen, die bei dem Spiel »Elite« die Musik vermissen. Es gibt bei der englischen Version von Elite nur ein Lied (von Johann Strauß), dagegen bei der deutschen Version noch ein Lied zusätzlich (die Anfangsmusik). Wenn man das Spiel mit <INST/DEL> angehalten hat, kann man mit der Taste <M> lediglich das Start- und Andockrauschen ein-/ausschalten. Was nicht in der Beschreibung steht, ist, daß man mit der Taste <X> in einen Modus kommt, indem man die Musik im Hintergrund ein-/ausschalten kann, bestimmen kann, ob neben der Musik andere Geräusche zugelassen sind oder nicht, und (bei der deutschen Version) sogar zwischen zwei Liedern wählen kann.

Folgende Tasten haben obige Wirkung, wenn Elite angehalten wurde:

<X> = Modus ein-/ausschalten
<C> = Musik möglich/nicht möglich (beim nächsten Docken/Neustart wird die Musik automatisch ein-/ausgeschaltet)
<M> = Musik sofort ein-/ausschalten
<B> = Geräusche während Musik ein-/ausschalten
<E> = (nur deutsche Version) Wahl zwischen zwei Liedern

Vielleicht weiß jemand, welche Bedeutung die <D>-Taste hat, denn auch bei ihr hört man das typische Klickgeräusch.

(Torsten Tiedemann/tr)

Der Bildschirm als liniertes Blatt

Es ist oft recht ärgerlich, wenn man am Bildschirm des C 64 arbeitet und, wenn Rahmen und Hintergrund gleichfarbig sind, nicht weiß, wo die letzte Zeile ist und einem auf einmal die ganze Sache nach oben wegscrollt. Für solche und ähnliche Fälle ist das Programm »Liniertes Blatt« (Listing 1) wie geschaffen. Durch eine geniale Raster-Interrupt-Routine erreichen Sie nämlich dasselbe wie wenn Sie mit Filzstift auf der Mattscheibe am Ende jeder Zeile eine Linie ziehen würden. Eingegeben wird das äußerst kurze Programm mit dem MSE und gestartet mit SYS49206. Es liegt von $C000 bis $C05a im Speicher. Die Hintergrundfarbe muß von jetzt an nach 49175 gePOKEt werden; die Farbe der Linien ist in 49163 festgelegt.

(Günter Burgstaller/tr)

Kursivschrift am Bildschirm

Kursivschrift oder »Italics« gehört heutzutage schon zur Standardausrüstung eines durchschnittlichen Druckers. Mit dem gleichnamigen Programm (Listing 2) kann der C 64-Besitzer diese Schriftart auch auf dem Bildschirm genießen. Eine VIC-Interrupt-Routine setzt ohne Veränderung des Zeichensatzes diesen Effekt für Sie in Szene. Eingegeben wird wie gewohnt mit dem MSE, die Inbetriebnahme erfolgt durch SYS49197. Das Prográmmchen belegt die Speicherstellen $C000 bis $C051. (Günter Burgstaller/tr)

Der Super-Listschutz

Es handelt sich hier um eine ganz besondere Art von Listschutz, welche das Auflisten zwar nicht verhindert, aber den unbefugten Eindringling mit den blanken Zeilennummern konfrontiert. Nehmen wir an, Sie hätten ein Basic-Programm geschrieben und wollen es ganz oder nur bestimmte Teile (zum Beispiel Algorithmen) vor neugierigen Blicken schützen. Sie laden dazu Ihr Programm und H.I.D.E. (Listing 3) und starten dieses mit SYS49152. Achtung: Ihr Programm muß 100prozentig in den Basic-Zeilen untergebracht sein. Maschinenprogramme, die nur eine Basic-Kopfzeile haben, können nicht bearbeitet werden. Es ist übrigens egal, ob Sie H.I.D.E. vor oder nach Ihrem Programm laden, da bei seinem Start die Basic-Endezeiger wieder gerichtet werden und so ein »OUT OF MEMORY ERROR« verhindert wird. Nach kurzer Zeit hat das Maschinenprogramm seine geheimnisvolle Arbeit getan, und Sie dürfen jetzt versuchen, Ihr Basic-Programm zu LISTen.

Beispiel:
(vorher)
```
10 A$="NIZAGAM RE'46"
20 FOR T=LEN(A$) TO 1 STEP -1
30 PRINT MID$(A$,T,1);:REM GEHEIMER ALGORITHMUS
40 NEXT T
SYS 49152
```
(nachher)
```
10
20
30
40
```

Es ist verblüffend, nicht? Sie können natürlich auch Zeilen stehenlassen. Sie teilen das dem Programm mit, indem Sie jeder Zeile, die später noch sichtbar sein soll, einen Klammeraffen (»@«) voransetzen.

Beispiel:
```
50 @REM COPYRIGHT BY 64'er 1986
```
wird nicht verändert. Nur der Klammeraffe wird gelöscht. So lassen sich Bemerkungen, die den Neugierigen empfangen sollen (zum Beispiel GET OUT OF HERE!) leicht anbringen. Noch besser sieht's aus, wenn die Zeile nicht angesprungen wird, und so das REM entfallen kann!

Eine Einschränkung: Die erste Zeile darf nicht stehenbleiben! Lassen Sie notfalls eine REM-Zeile verschwinden.

Ihr Programm können Sie jetzt ganz normal speichern. Nach dem Laden muß nichts mehr eingegeben werden, wie bei anderen Listschutz-Versuchen üblich, die meist ziemlich sinnlos sind, wenn der Unbefugte erst einen POKE eingeben muß, um abgewiesen zu werden.

Wenn Sie nun auf eine Erklärung des Phänomens gewartet haben, muß ich Sie enttäuschen, denn was ist schon ein Listschutz, wenn er überall bekannt ist? Versuchen Sie, es selber herauszufinden. Wie Sie sicher merken werden, kann es je nach Programmlänge etwas länger dauern, bis H.I.D.E. fertig ist. H.I.D.E. informiert Sie regelmäßig, welche Zeile er gerade bearbeitet. So sehen Sie auch, daß er bei längeren Programmen relativ lange für eine Zeile braucht.

```
Name : liniertes blatt   c000 c05a
-----------------------------------
c000 : ea ea ea ea ea a9 01 8d  92
c008 : 19 d0 a9 00 8d 21 d0 a2  5e
c010 : 06 ea ea ca d0 fb a9 06  3f
c018 : 8d 21 d0 a5 02 c9 f9 f0  57
c020 : 0b 18 69 08 85 02 8d 12  55
c028 : d0 4c 81 ea a9 39 85 02  5a
c030 : 8d 12 d0 4c 31 ea 78 a9  24
c038 : 39 85 02 8d 12 d0 ad 11  e7
c040 : d0 29 7f 8d 11 d0 a9 01  77
c048 : 8d 0d dc 8d 1a d0 a9 00  13
c050 : a2 c0 8d 14 03 8e 15 03  37
c058 : 58 60 cc 0f 0c 00 0f 33  59
```

Listing 1. »Liniertes Blatt«

```
Name : kursivschrift     c000 c051
-----------------------------------
c000 : ea ea ea ea ea a9 01 8d  92
c008 : 19 d0 ad 16 d0 49 01 8d  2e
c010 : 16 d0 a5 02 c9 f6 f0 0b  66
c018 : 18 69 04 85 02 8d 12 d0  0d
c020 : 4c 81 ea a9 32 85 02 8d  8f
c028 : 12 d0 4c 31 ea 78 a9 32  59
c030 : 85 02 8d 12 d0 ad 11 d0  bc
c038 : 29 7f 8d 11 d0 a9 01 8d  20
c040 : 0d dc 8d 1a d0 a9 00 a2  02
c048 : c0 8d 14 03 8e 15 03 58  83
c050 : 60 c0 8d 14 03 8e 15 03  f5
```

Listing 2. »Kursivschrift«

```
Name : h.i.d.e.          c000 c19d
-----------------------------------
c000 : 20 33 a5 a5 22 18 69 02  64
c008 : 85 2d a5 23 69 00 85 2e  fb
c010 : a5 2b 85 fb a5 2c 85 fc  f7
c018 : a9 00 8d 98 c1 8d 99 c1  aa
c020 : 20 2d c1 a0 00 b1 fb 8d  f4
c028 : 9a c1 c8 b1 fb 8d 9b c1  29
c030 : d0 08 ad 9a c1 d0 03 4c  0a
c038 : c5 c0 20 ac c0 a0 04 b1  7f
c040 : fb c9 40 f0 42 a0 00 18  a7
c048 : ad 98 c1 69 06 8d 98 c1  92
c050 : ad 99 c1 69 00 8d 99 c1  be
c058 : 20 93 c0 18 a5 fb 69 04  5d
c060 : 85 fb a5 fc 69 00 85 fc  92
c068 : 4c d9 c0 a0 00 98 91 fb  e8
c070 : c8 a9 3a 91 fb c8 c0 06  e3
c078 : d0 f9 ad 9a c1 85 fb ad  97
c080 : 9b c1 85 fc 4c 23 c0 a9  31
c088 : 20 91 fb a0 00 20 93 c0  55
c090 : 4c 7a c0 18 ad 9a c1 6d  de
c098 : 98 c1 8d 9a c1 91 fb ad  bb
c0a0 : 9b c1 6d 99 c1 8d 9b c1  25
c0a8 : c8 91 fb 60 a2 00 bd 8d  80
c0b0 : c1 20 d2 ff e8 c9 00 d0  b4
c0b8 : f5 a0 02 b1 fb aa c8 b1  50
c0c0 : fb 20 cd bd 60 18 a5 2b  aa
c0c8 : 69 05 85 fb a5 2c 69 00  f6
c0d0 : 85 fc a0 00 a9 41 91 fb  de
c0d8 : 60 a5 2d 85 fd a5 2e 85  d8
c0e0 : fe 18 a5 2d 69 06 85 2d  31
c0e8 : a5 2e 69 00 85 2e c9 a0  31
c0f0 : f0 73 20 22 c1 a0 00 b1  6b
c0f8 : fd a0 06 91 fd c6 fd a5  52
c100 : fd c9 ff d0 02 c6 fe a5  99
c108 : fd c5 fb d0 e8 a5 fe c5  44
c110 : fc d0 e2 20 19 c1 4c 6b  d9
c118 : c0 e6 fb a5 fb d0 02 e6  1b
c120 : fc 60 c6 fb a5 fb c9 ff  df
c128 : d0 02 c6 fc 60 a2 00 bd  e1
c130 : 3b c1 20 d2 ff e8 c9 00  1d
c138 : d0 f5 60 11 12 48 2e 49  ec
c140 : 2e 44 2e 45 2e 20 56 33  68
c148 : 2e 32 92 0d 4d 4f 4d 45  e5
c150 : 4e 54 20 42 49 54 54 45  2c
c158 : 20 21 0d 11 5a 45 49 4c  fc
c160 : 45 20 3a 20 00 a2 00 bd  d9
c168 : 75 c1 20 d2 ff e8 c9 00  8f
c170 : d0 f5 4c c5 c0 0d 12 53  6a
c178 : 4f 52 52 59 2c 20 50 52  5a
c180 : 47 20 5a 55 20 4c 41 4e  1f
c188 : 47 2e 92 11 00 0d 91 1d  96
c190 : 1d 1d 1d 1d 1d 1d 1d 00  56
c198 : 00 00 00 00 0d d0 ce e2  f1
```

Listing 3. »H.I.D.E.«, der Super-Listschutz

```
60000 DIM W(6)                                    <155>
60010 PRINT"EINGABE:";                            <157>
60020 PRINT"000000{LEFT}";:A$="{DOWN,LEFT}
      ↑{UP,LEFT}":S=6                             <112>
60030 A=PEEK(56321)                               <042>
60040 IF A=254 AND W(S)<9 THEN W(S)=W(S)+1        <116>
60050 IF A=253 AND W(S)>0 THEN W(S)=W(S)-1        <125>
60060 IF A=251 AND S>1 THEN S=S-1:PRINT"{D
      OWN,SPACE,UP,2LEFT}";                       <224>
60070 IF A=247 AND S<6 THEN S=S+1:PRINT"{D
      OWN,SPACE,UP}";                             <141>
60080 IF A=239 THEN 60110                         <011>
60090 PRINT RIGHT$(STR$(W(S)),1)A$;               <057>
60100 GOTO 60030                                  <093>
60110 W=W(6)+W(5)*10+W(4)*100+W(3)*1000+W(
      2)*10000+W(1)*100000                        <010>
60120 PRINT:PRINT                                 <125>
60130 PRINT"WERT:"W                               <228>
60140 POKE 198,0                                  <101>
```

© 64'er

Listing 4. Verbesserte Zahleneingabe mit dem Joystick

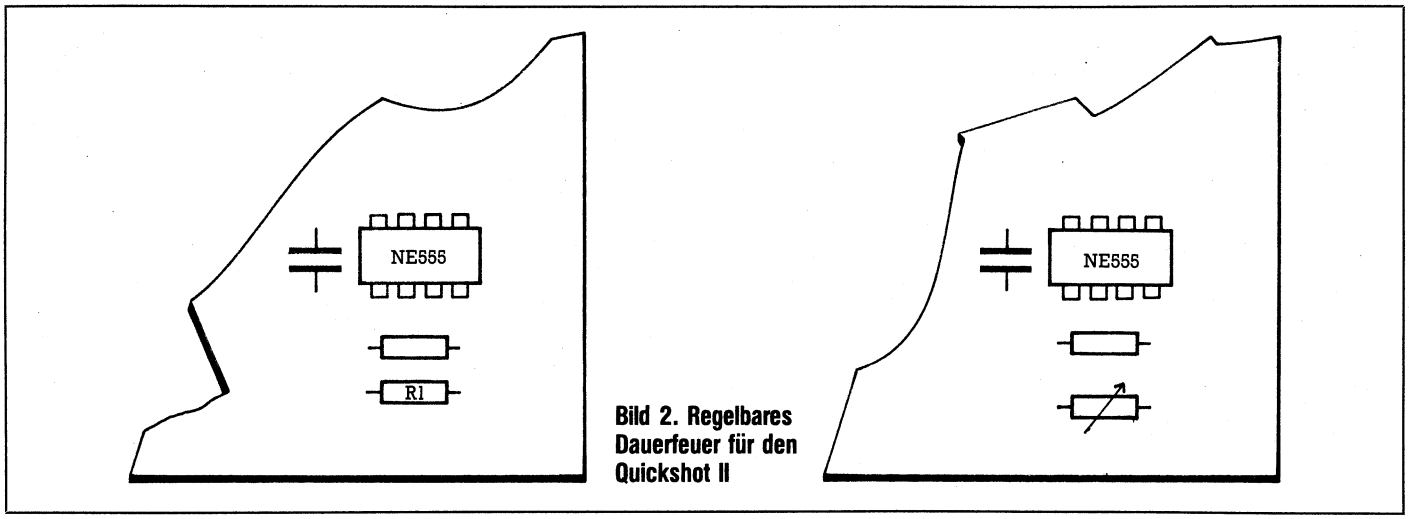

Bild 2. Regelbares Dauerfeuer für den Quickshot II

Nur so viel: es liegt daran, daß extrem viel im Speicher verschoben werden muß. Deshalb sollte man bei seinen Programmen wenn möglich folgendes beachten:
— Zur Einsparung von Speicherplatz lassen Sie alle REMs weg.
— Versuchen Sie so viel Programmtext wie möglich in eine Zeile zu packen. Man sieht es später ja nicht mehr.

Sollte Ihr Programm zu lang sein, meldet sich H.I.D.E. und sagt es Ihnen. Normalerweise sind Programme bis 32 KByte möglich. Beachten Sie auch, daß Sie das Programm jetzt nicht mehr editieren können und die Zeilen gelöscht werden, wenn Sie mit dem Cursor auf die Zeilennummer gehen und <RETURN> drücken! (Frank Hund/tr)

Regelbares Dauerfeuer

Ich habe einen Quickshot II-Joystick, der mit einfachem Dauerfeuer ausgestattet ist. Durch das Auswechseln eines Widerstandes gegen ein Poti (Bild 2) kann das Dauerfeuer geregelt werden. Hardwareschäden sind während der vier Wochen Dauertest nicht aufgetreten (K. Pichlmair/tr)

Zahlen eingeben mit dem Joystick

In Ausgabe 6/86 auf Seite 77 wurde bereits ein solches Programm veröffentlicht. Es läßt sich aber wesentlich kürzer und übersichtlicher schreiben (siehe Listing 4). Der Joystick wird in Port 1 gesteckt. Der Feuerknopf beendet die Eingabe. Danach wird die eingegebene Zahl berechnet und in der Variablen W gespeichert.

Die Eingabe kann mit Hilfe von PRINT-Befehlen beliebig positioniert werden. Der POKE 198,0 in der letzten Zeile löscht den Tastaturpuffer, der wegen des Joysticks in Port 1 wirre Zeichen enthält. (Andreas Kaiser/tr)

Tip zu PrologicDOS

Über »POKE 781,133: SYS 58551« läßt sich aus einem Basic-Programm heraus das Disketteninhaltsverzeichnis auf den Bildschirm bringen (entspricht der <F1>-Taste). Das Ganze läßt sich natürlich auch in Maschinensprache schreiben: »LDX #$85: JSR $E4B7« (tr)

Reise durch den C128 (Teil 5)

Erneut hat die Redaktion einen Bericht ihres Korrespondenten erhalten. Es ist ihm gelungen, eine Karte der Speicherlandschaft zu entwerfen, die zwar noch einige weiße Flecken aufweist, aber dennoch die Orientierung erleichtert.

Als C 128-Benutzer stolpert man meistens zuerst einmal über den BANK-Befehl. Was es damit auf sich hat, bekommt man zwar in dem Moment heraus (oder im schlimmeren Fall nicht), wenn man mit PEEK oder POKE arbeitet und auch bei Verwendung des eingebauten Monitors. Aber welche Speicher steuert man eigentlich durch die diversen BANK-Kennzahlen an und wo befinden sich die Speicher? Bild 1 gibt zunächst einen Überblick:

Sie sehen darin unten vier große (64 KByte) RAM-Bereiche, von denen die untere durch BANK 0, die darüberliegende durch BANK 1 gekennzeichnet ist. Zwei weitere 64-KByte-Bereiche sollen demnächst als Speichererweiterungen zu erwerben sein. Diese vier Bereiche werden durch BANK n angesteuert, wobei n die BANK-Nummer ist, also von 0 bis 3 reicht. Alle vier haben zwei Adressenbereiche gemeinsam: Eine »Common Area« (das heißt auf deutsch »gemeinsamer Bereich«) von Speicherstelle 0 bis 1023, also 1 KByte lang. Jedes Lesen oder Schreiben in diesen Teil findet automatisch in BANK 0 statt. Man braucht sich hier daher um keine BANK-Umschaltung zu kümmern. Ein schmaler Bereich zwischen $FF00 (dezimal 65280) und $FF04 (dezimal 65284) gehört zur sogenannten MMU (das heißt »Memory Management Unit« = »Speicherorganisations-Einheit«), um die wir uns noch kümmern werden. Dieser kleine Bereich ist deshalb allen BANKs gemeinsam, weil damit die Umschaltung von einer Speicherkonfiguration zur anderen vollzogen wird, er ist gewissermaßen der Schalter dazu. Hätten wir aber eine Speicherzusammenstellung vorliegen, in der dieser Schalter nicht enthalten wäre, dann könnten wir nicht mehr zurück in andere Konfigurationen.

Zurück zur BANK 0: Direkt oberhalb der Common Area befinden sich der normale Textbildschirm und die Sprite-Zeiger, die vom Basic 7.0 benutzt werden. Zwischen den Adressen $0800 bis $1300 liegt wieder Systemspeicher vor: Eine Wanderung durch die Memory-Map wäre hier schon ganz schön lang! Ab $1300 bis $1C00 haben wir freien Speicherplatz zur Verfügung, den wir beispielsweise für Maschinenprogramme nutzen können. Von da an wird's mehrdeutig.