ALLE TASTEN-, ZEICHEN- U

Das ist der dritte Teil einer Serie über die Abfragemethoden für Tasten und ihre ver und den C 64 gleichermaßen. Wo Unterschiede auftreten, sind die

Ich habe im zweiten Teil dieser Reihe versucht, Sie zum Experimentieren mit den ASCII-Codes anzuregen. Heute habe ich für Sie die vollständige Liste aller 255 Codewerte vorbereitet und zwar in einer Art, die sicher einiger Erklärungen bedarf.

ASCII-Codes, die von Commodore verschwiegen werden

Ich habe nämlich noch ein paar zusätzliche Überraschungen parat, auf die man nur durch Zufall kommt oder durch Studium des Betriebssystems oder aber, wenn man den Aufsatz von G. Urbanczyk in Computer persönlich vom 19.10.1983, Seite 76, gelesen hat.

Tippen Sie bitte das Programm Nummer 3 aus dem 64'er Mai-Heft, Seite 107, ein, nämlich eine der drei Versionen zur Tastaturpuffer-Abfrage. (Ich verwende unten die GET-Version).

```
310 PRINT CHR$(147)
320 GET A$
330 IF A$= " " THEN 320
340 PRINT ASC(A$)
350 GOTO 320
```

Auf beiden, VC 20 und C 64 erhalten Sie nach RUN 310 und Drücken der RETURN-Taste (natürlich) den ASCII-Code 82. Wenn Sie zuerst die CTRL-Taste drücken und halten und dann erst das R drücken, dürfte eigentlich nichts passieren, denn die CTRL-Taste gilt ja angeblich nur für die Farben. Ja, denkste! Wir erhalten nämlich die Zahl 18. Ein Blick in die ASCII-Tabelle zeigt uns für 18 die Funktion »REVERSE-ON«.

Versuchen Sie dasselbe mit CTRL und der ←-Taste. Wir erhalten die 6, und nicht 95, wie es eigentlich sein sollte.

Für den VC 20 ist das alles. Aber immerhin, wir haben sozusagen noch zwei zusätzliche Funktionstasten gefunden.

Beim C 64 aber geht es erst richtig los:

Der Versuch wird Ihnen zeigen, daß alle Buchstaben, von A bis Z, zusammen mit CTRL gedrückt, einen anderen ASCII-Wert, nämlich 1 bis 26, ergeben, als allein gedrückt.

Desweiteren biete ich Ihnen noch:

CTRL — ↑ = 30
CTRL — = = 31
CTRL — £ = 28
CTRL — : = 27
CTRL — ; = 29

Das heißt aber, daß einige ASCII-Codezahlen zwei Bedeutungen haben. Oder umgekehrt, zwei verschiedene Tasten (kombiniert) haben denselben ASCII-Code.

Schwierigkeiten dadurch, daß einige ASCII-Werte zwei Bedeutungen haben, gibt es deswegen nicht, weil die Kombination mit CTRL nicht PRINT-bar ist (PRINT CHR$(19) schickt immer den Cursor »home«, mit dem »S« passiert gar nichts).

Andersherum kann es allerdings vorkommen, daß eine Tastenabfrage, zum Beispiel
GET A$:IF A$ = CHR$(19) THEN ..... sowohl auf die Taste »HOME« als auch auf »CTRL-S« reagiert. Da ist sicher etwas Vorsicht angebracht. Aber ein Blick in meine ASCII-Tabelle zeigt Ihnen ja die Doppeldeutigkeiten.

An dieser Stelle erwarte ich eigentlich einige Einsprüche, wie: »Wozu das alles, die acht Funktionstasten, oder gar erweitert auf 32, reichen doch völlig aus!« Für den Hausbeziehungsweise Spielgebrauch ist das sicher richtig. Aber bei professioneller Software, welche benutzerfreundlich aufgebaut ist, kann es oft gar nicht genügend Funktionstasten — besonders solche, die eine optische Buchstabenbeziehung zu der Abfrage haben sollen — geben. Wenn in einem Programm gefragt wird, ob Sie »LOADen« oder »SAVEn« wollen, ist CTRL-L oder CTRL-S halt klarer, als f-1 oder f-3.

Ich finde es schade, daß diese großartige Möglichkeit nur auf dem C 64 gegeben ist. Hier zeigt sich deutlich, daß dieser Computer doch professioneller ist als der VC 20.

Die vollständige ASCII-Tabelle

So, jetzt können Sie meine ASCII-Tabelle erst richtig interpretieren (Tabelle 1).

Leere Kästchen haben keine Bedeutung für die betreffende Codezahl.

```
5 REM************************************
6 REM***** SPIEL MIT FINKELN **********
7 REM************************************
10 PRINT CHR$(147)
20 F=0
30 R=65
40 FOR T=1 TO 600:NEXT T
50 A=INT(RND(0)*7)+65
60 IF R>71 THEN 400
70 PRINT CHR$(A);
80 FOR T=1 TO 1000:NEXT T
100 GET A$
110 IF A$<>"■" THEN 300
120 PRINT"■";
200 IF A=R THEN F=F+1
210 GOTO 50
300 IF A<>R THEN F=F+1
310 R=R+1
320 GOTO 50
400 PRINT"■■■DAS SPIEL IST ZU ENDE";"■
"F"FEHLER"
```

Programm 2
Spiel mit »Finkeln« für C 64

ND STEUERCODES TEIL 3

schiedenen Codes. Alle Angaben gelten für den VC 20 Werte für den Commodore 64 in Klammern gesetzt.

Jeweils zwei Zeichen nebeneinander mit derselben Codezahl stellen die beiden Zeichensätze dar, in die mit C=SHIFT (Commodore-Taste) umgeschaltet werden kann. Wo nur ein Zeichen steht, ist es in beiden Zeichensätzen identisch.

Die Funktionen der Codezahlen 129 und 149 bis 155 gelten nur für den C 64. Interessant ist übrigens, daß die 4. und 7. Spalte identisch ist, ebenso die 6. und 8. Spalte (außer dem Zeichen für 255).

Ich möchte jetzt gern die Szene wechseln, ohne aber den ASCII-Code aus den Augen zu verlieren. Wir haben den ASCII-Code bisher verwendet, um Tasten abzufragen oder Funktionen auszuführen. PRINT CHR$(66) druckt zum Beispiel den Buchstaben B auf den Bildschirm.

Welche Methoden kennen Sie noch, mit denen das gleiche erzielt werden kann?

Die erste, die jeder aus dem Handbuch lernt, ist
PRINT "B".
Die komplizierteste ist:
POKE 7680,2: POKE 38400,7
(POKE 1024,2: POKE 55296,7)

Diese beiden Vorgehensweisen wollen wir uns näher anschauen und prüfen, ob wir sie in Analogie zu dem ASCII-Code für Tastenabfragen einsetzen können.

Es ist sicher viel bequemer, längere Buchstabenreihen oder gar Texte zwischen Gänsefüße gestellt einzutippen, als eine Serie von CHR$-Werten, ganz abgesehen vom erforderlichen Speicherplatz.

Nicht ganz so bequem ist der Gänsefuß-Modus bei Steuerzeichen, wie zum Beispiel Cursor-Bewegungen, besonders, wenn man diese herbeiführen will, aber statt dessen die reversen Darstellungen auf dem Bildschirm erzeugt.

Der Gänsefuß-Modus

Geben Sie es zu. Sie haben deswegen auch schon herzhaft geflucht. Auch jeder Redakteur bittet um Listings mit CHR$-Darstellung anstelle der reversen Zeichen, die bei der Druckwiedergabe oft zu Schwierigkeiten führen.

Jetzt wissen Sie, warum ich bei meinen Prográmmchen immer PRINT CHR$(147) statt PRINT" " verwende.

Genauso austauschbar wie bei PRINT ist der ASCII-Code mit dem Gänsefuß-Modus bei der Tastenabfrage.
Statt:
10 GET A$
20 IF A$<>CHR$(65) THEN 10
30 PRINT CHR$(88)
können wir schreiben:
10 GET A$
20 IF A$<>"A" THEN 10
30 PRINT "X"

Beide Programme sind gleichwertig. Nach RUN rührt sich gar nichts. Erst, wenn die A-Taste gedrückt wird (Zeile 20), druckt Zeile 30 den Buchstaben X.

In Zeile 20 können wir natürlich statt des A jeden beliebigen Buchstaben, Zahl oder Zeichen nehmen.

Programm 3. ASCII-Code Wandler

```
3 REM*****************
4 REM**CODE-WANDLER**
5 REM*****************
10 PRINT CHR$(147)
20 INPUT "■ASCII-CODE";ACII
30 IF ACII=0 THEN END
40 IF ACII AND 128 THEN BILD=ACII AND 127 OR 64:GOTO 80
50 IF NOT ACII AND 64 THEN BILD=ACII:GOTO 80
60 IF ACII AND 32 THEN BILD=ACII AND 95:GOTO 80
70 BILD=ACII AND 63
80 PRINT TAB(5) "BILDSCH.CODE:"BILD
90 GOTO 20
```

LISTen Sie einfach die 2. Version der drei Zeilen, fahren mit dem Cursor auf das A und verändern Sie es Ihren Wünschen entsprechend. Wie ich Sie einschätze, machen Sie das sicher auch mit den Funktionstasten.

Nein? Dann machen Sie es mal. Sie haben nämlich Pech, so geht es nicht. Aber es geht, wenn Sie sich mit Absicht in die Lage begeben, die wie vorhin beschrieben, Flüche auslöst. Fahren Sie mit dem Cursor auf den 1. Gänsefuß, tippen Sie ihn noch einmal ein und drücken Sie dann eine Funktionstaste. Siehe da, es erscheint ein reverses Zeichen. Mit RETURN wird es »fixiert«, nach RUN wird das X erst mit der verwendeten Funktionstaste ausgelöst.

Der Trick besteht also darin, durch eine ungerade Anzahl von Gänsefuß-Eingaben diesen Modus herbeizuführen. Es geht ebenso durch Drücken der INSERT(INST)-Taste, allerdings nur für soviele Zeichen, wie oft sie gedrückt worden ist.

Im Gänsefuß-Modus erscheinen alle Steuer- und Funktionstasten in reverser Darstellung

Sie haben oben ein reverses Zeichen für die Funktionstasten erhalten. Die Zeichen für die Farben und Cursorbewegungen, also alle »gängigen« Funktionen, kennen Sie inzwischen sicher schon. Aber alle Steuer- und Funktionstasten?

Es gibt zwei Möglichkeiten, diese Zeichen zu finden:
Die 1. Methode verwendet entweder ganz primitiv im Direkt-Modus den Befehl: PRINT " mit nachfolgendem Drücken der Steuer- oder Funktionstaste oder sehr elegant den Dreizeiler
10 GET A$: IF A$=""THEN 10
20 PRINT CHR$(34) A$ CHR$(34) ASC(A$)
30 GOTO 10

ALLE TASTEN-, ZEICHEN- UND STEUER-CODES

Auch hier ist ein kleiner Pfiff drin. In Zeile 20 wird zuerst ein Gänsefuß (34) gedrückt, wodurch wir in dem danach benannten Modus sind. Das nachfolgende A$ erscheint im Fall einer Steuertaste als reverses Zeichen, nach dem abschließenden 2. Gänsefuß gibt uns die ASC-Funktion noch den ASCII-Code der gedrückten Taste.

Mit Gänsefüßen statt CHR$ hätten wir lediglich die beiden Zeichen A und $ auf den Bildschirm gedruckt. Mit dieser Methode erhalten Sie zum Beispiel für:
ASCII-Code 17 = Cursor Down .....
ASCII-Code 147 = CLR .....

Die 2. Methode ist viel einfacher. Schauen Sie in meine ASCII-Tabelle (Tabelle 1). Sie ist in Spalten zu je 32 Zeichen angeordnet. Die Steuerzeichen und die Farben stehen alle in Spalte 1 und 5.

Da finden Sie zum Beispiel über der Codezahl 17 die Funktion »Cursor-Down«. Wenn Sie jetzt in die 3. Spalte waagrecht rübergehen — also den Wert um 64 erhöhen — steht da das Q . Oder: In Spalte 5 ist der Taste CLR die Codezahl 147 zugeordnet. Zwei Spalten weiter (64 höher) steht das .

Wenn Sie die Ergebnisse der 1. Methode oben mit den durch Spaltenhüpfen gefundenen Zeichen vergleichen, sehen Sie, daß es dieselben Zeichen sind, halt nur reversiert.

KOMBINATIONEN		1	2	3	4	5	6	7	8
VC 20	C 64								
	CTRL — A	001	033 !	065 A	097 ♠	129 BLK	161	193 ♠	225
	CTRL — B	002	034 "	066 B	098	130	162	194	226
	CTRL — C	003 STOP	035 #	067 C	099	131 LOAD RUN	163	195	227
	CTRL — D	004	036 $	068 D	100	132	164	196	228
	CTRL — E	005 WHT	037 %	069 E	101	133 (F1)	165	197	229
	CTRL — F								
CTRL — ←	CTRL — ←	006	038 &	070 F	102	134 (F3)	166	198	230
	CTRL — G	007	039 '	071 G	103	135 (F5)	167	199	231
	CTRL — H	008 LOCK	040 (	072 H	104	136 (F7)	168	200	232
	CTRL — I	009 UNLOCK	041)	073 I	105	137 (F2)	169	201	233
	CTRL — J	010	042 *	074 J	106	138 (F4)	170	202	234
	CTRL — K	011	043 +	075 K	107	139 (F6)	171	203	235
	CTRL — L	012	044 ,	076 L	108	140 (F8)	172	204	236
	CTRL — M	013 CAR RET	045 -	077 M	109	141 SHFT RET	173	205	237
	CTRL — N	014 TEXT	046 .	078 N	110	142 GRAF	174	206	238
	CTRL — O	015	047 /	079 O	111	143	175	207	239
	CTRL — P	016	048 0	080 P	112	144 BLK	176	208	240
	CTRL — Q	017 CRSR DOWN	049 1	081 Q	113	145 CRSR UP	177	209	241
CTRL — R	CTRL — R	018 RVS ON	050 2	082 R	114	146 RVS OFF	178	210	242
	CTRL — S	019 CRSR HOME	051 3	083 S	115	147 CLR SCRN	179	211	243
	CTRL — T	020 DEL	052 4	084 T	116	148 INS	180	212	244
	CTRL — U	021	053 5	085 U	117	149 BRN	181	213	245
	CTRL — V	022	054 6	086 V	118	150 LT RED	182	214	246
	CTRL — W	023	055 7	087 W	119	151 DARK GREY	183	215	247
	CTRL — X	024	056 8	088 X	120	152 MED GREY	184	216	248
	CTRL — Y	025	057 9	089 Y	121	153 LT GRN	185	217	249
	CTRL — Z	026	058 :	090 Z	122	154 LT BLU	186 ✓	218	250 ✓
	CTRL — :	027	059 ;	091	123	155 LT GREY	187	219	251
	CTRL — £	028 RED	060 <	092 £	124	156 PURP	188	220	252
	CTRL —]	029 CRSR RT	061 =	093]	125	157 CRSR LEFT	189	221	253
	CTRL — ↑	030 GRN	062 >	094 ↑	126	158 YEL	190	222	254
	CTRL — =	031 BLU	063 ?	095 ←	127	159 CYAN	191	223	255

Tabelle 1. ASCII-Code

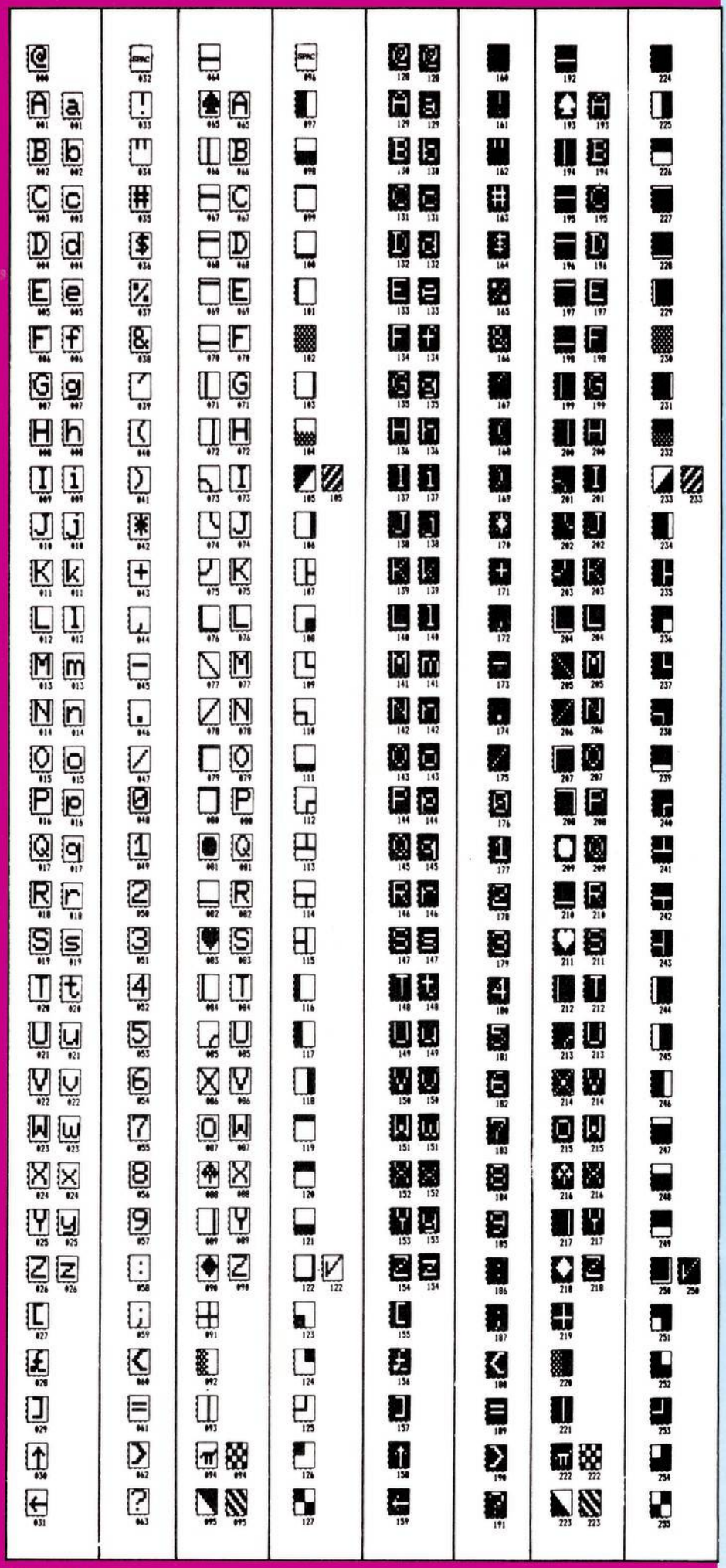

Tabelle 3. Bildschirm-Code

TEIL 3

Machen wir die Probe:

Mit Methode 1 erhalten wir für »Rot« das reverse Pfund-Zeichen . In der ASCII-Tabelle finden wir »Rot« unter 28. Zwei Spalten weiter, unter 28 + 64 = 92, steht dasselbe Zeichen.

Das gilt auch für alle CTRL-Kombinationen, nicht nur für die der Farben. Bei beiden Computern entspricht dem CTRL- das , beim C 64 erzeugt CTRL- ein . Alle Kombinationen der Buchstaben mit CTRL erzeugen diese Buchstaben in reverser Darstellung.

Um das in einer kleinen praktischen Anwendung zu verdeutlichen, schlage ich vor, dieselbe Aufgabenstellung, die in Heft 5/84 sowohl mit Tastencode-Abfrage (Programm Nummer 1 auf Seite 104/105) als auch mit Tastaturpuffer-Abfrage (Programm Nummer 4 auf Seite 135) gelöst wurde, noch einmal zu verwenden, jetzt aber die Gänsefuß-Methode einzusetzen.

Um beim Eintippen des Programms unten sicherzustellen, daß alles klappt, habe ich statt der reversen Zeichen die Tasten angegeben beziehungsweise umrahmt, die nach dem 1. Gänsefuß gedrückt werden müssen.

Das Programm schaltet, wie die beiden anderen Versionen auch, die Bildschirm-Farben mit f-1, f-2, f-3 und @ um.

Programm 1. Abfrage mit Gänsefuß

```
410 PRINT "SHIFT und CLR/HOME
"
420 GET A$
430 IF A$= "" THEN 420
440 IF A$= " f-1 " THEN POKE
36879,126
450 IF A$= " f-2 " THEN POKE
36879,45
460 IF A$= " f-3 " THEN POKE
36879,25
470 IF A$= "  " THEN POKE
36879,27
480 GOTO 420
```

Für den C 64 gelten in den Zeilen 440 bis 470 andere POKE-Adressen:

440 ..... POKE 53280,6:POKE 53281,7
450 ..... POKE 53280,5:POKE 53281,2
460 ..... POKE 53280,1:POKE 53281,1
470 ..... POKE 53280,3:POKE 53281,1

Ein letztes Problem bleibt uns noch. Wie schaffen wir es, daß wir im Gänsefuß-Modus auch Funktionen einsetzen können, die entweder keine eigene Taste haben (zum Beispiel 14 = Text, 8 = Lock) oder die beim Eintippen sofort die Funktion auslösen (zum Beispiel 13 = RETURN, 20 = DELete)?

ALLE TASTEN-, ZEICHEN- UND STEUERCODES

Hier müssen wir eine Methode anwenden, die meine Kinder und ich »finkeln« getauft haben und zwar deswegen, weil wir sie zum ersten und einzigen Mal vom Commodore-Software-Spezialisten Andy Finkel im amerikanischen Handbuch gefunden haben.

Sein Trick besteht darin, daß er in einer ASCII-Tabelle das entsprechende Zeichen für die Funktion heraussucht und es in mehreren Schritten an seinen vorgesehenen Platz bringt.

Ich will Ihnen zeigen, was ich damit meine:

Bitte, versuchen Sie mit der Gänsefuß-Methode die DELete-Taste in eine PRINT-Anweisung zu bringen

10 PRINT " INST/DEL "

Sie werden es nicht schaffen, da die DEL-Taste, statt ein reverses Zeichen zu drucken, ihrer Funktion nachgeht und das vorherige Zeichen löscht.

Jetzt »finkeln« wir:

1. Schritt:

10 PRINT " " (mit RETURN abschließen)

2. Schritt:

Mit dem Cursor auf die Leerstelle zwischen den Gänsefüßen fahren.

3. Schritt:

Aus der ASCII-Tabelle das Zeichen der DEL-Taste holen (T).

4. Schritt:

Die reverse Darstellung mit CTRL-REV.ON einschalten (der Cursor bleibt auf seiner Stelle) und das T drücken, mit RETURN abschließen. Jetzt steht das Zeichen drin und das Programm läuft.

Um Ihnen den Schritt 3 für alle widerborstigen Funktionen zu erleichtern, habe ich sie alle in der Tabelle 2 zusammengefaßt.

Da ich hoffe, daß Sie in Zukunft fleißig finkeln werden, muß ich Sie noch über einen lästigen Nebeneffekt aufklären, der bei ein paar Finkeleien auftritt. Einige der Funktionen, nämlich RETURN, DELete (schon wieder) und das SHIFT-RETURN wirken nicht nur im Programmablauf wie vorgesehen, sondern auch beim LISTen, was lästig sein kann. (Allerdings ergeben sich dadurch auch ungeahnte Möglichkeiten — siehe Artikel »Synthetische Steuerzeichen«. Das geSHIFTete RETURN (ASCII-Code 14) ist sehr nützlich bei Platz- und Speichermangel. Sie können nämlich mit " ▚ " in einer langen Programmzeile den Cursor mit nur drei Zeichen auf den Anfang der nächsten Zeile bringen, mit CHR$(141) bräuchten Sie schon neun Zeichen, mit SPC(...) müssen Sie sehr genau die Cursorposition berechnen, mit einer entsprechenden Anzahl von »Cursor-Rechts«-Zeichen geht es auch nur mühsam. Also, nützlich ist SHIFT-RETURN durchaus!

Nur: Beim LISTen wird es auch ausgeführt und die Zeile, in der es steht, sieht recht blöd aus. Zusätzlich kann eine derart geLISTete Zeile nicht mehr geändert werden, sondern muß bei Verbesserungen völlig neu geschrieben und gefinkelt werden. Alles Gute hat seinen Preis!

Soviel sei zur Methode gesagt. Jetzt wollen wir zur Erholung und zur Übung ein kleines Spiel programmieren, in dem wir (fast) alles Gelernte auch anwenden.

Eine kleine Seltenheit ist bemerkenswert: Das Programm ist für VC 20 und C 64 identisch!

Die Spielaufgabe soll darin bestehen, die ersten sieben Buchstaben des Alphabets möglichst in der richtigen Reihenfolge auf den Bildschirm zu bringen.

BEDEUTUNG	ASCII-CODE	REVERSE DARSTELLUNG	FINKELN
LOCK (Sperre der Zeichensatz-Umschaltung	8	H	H
UNLOCK (Sperre aufheben)	9	I	I
RETURN	13	M	M
TEXT (2. Zeichensatz)	14	N	N
DEL (Zeichen löschen)	20	T	T
SHIFT RETURN (Cursor auf Anfang der nächsten Zeile)	141	▚	SHIFT M
GRAF (1. Zeichensatz)	142	▞	SHIFT N

Tabelle 2. Funktionen, die im Gänsefuß-Modus nur durch »Finkeln« eingetippt werden können

Das Finkel-System

Klingt einfach, aber die Buchstaben sollen in zufälliger Reihenfolge auftauchen. Zusätzlich hat der Spieler, falls der Buchstabe nicht der Reihenfolge entspricht, lediglich die Möglichkeit, ihn mit der DEL-Taste zurückzuweisen, wenn er schnell genug ist. Das Programm zählt die Felder und zeigt am Schluß das Ergebnis an.

Wir brauchen dazu:

— Einen Zufalls-Buchstaben-Erzeuger von A bis G (ASCII-Code 65 bis 71)
— einen Buchstaben-Drucker
— einen Buchstaben-Reihenfolgezähler
— eine Möglichkeit, die DEL-Taste zu drücken und damit den gedruckten Buchstaben rückgängig zu machen
— einen Fehlerzähler
— eine Prüfung, ob der letzte Buchstabe (71) erreicht ist.

Normalerweise müßte ich jetzt ein Flußdiagramm zeichnen und »strukturiert« vorgehen, so wie die ausgezeichnete Serie in diesem Heft lehrt. Man möge mir aber verzeihen, daß ich aus Erklärungsgründen in einzelnen Schritten vorgehe, welche uns erlauben, jederzeit Zwischenresultate mit Probeläufen zu überprüfen (Programm 2, siehe Listing).

Auf geht's!

TEIL 3

Den Buchstaben-Erzeuger und -drucker erhalten wir durch Zeile 50, welche für eine Variable A zufällige ASCII-Codes zwischen 65 und 71 erzeugt, sowie durch Zeile 70, die das Zeichen für den ASCII-Code ausdruckt.

```
50 A=INT(RND(0)*7)+65
70 PRINT CHR$(A);
```

Für weniger Versierte sei gesagt, daß RND(0) eine Zufallszahl zwischen 0 und 0,99 erzeugt, mit 7 multipliziert gibt das eine Zahl zwischen 0 bis 6,93. Die Funktion INT macht daraus eine ganze Zahl, zwischen 0 und 6, mit 65 addiert letztlich eine Zahl zwischen 65 und 71 — ASCII-Werte der Buchstaben A bis G.

Den Ausdruck der Buchstaben nebeneinander erreichen wir durch das Semikolon in Zeiel 70, die laufende Wiederholung durch einen Rücksprung in Zeile 320.

```
320 GOTO 50
```

Damit es nicht zu schnell geht, verzögern wir das Ganze mit einer Warteschleife in Zeile 80.

```
80 FOR T=1 TO 1000:NEXT T
```

Probieren Sie es mit RUN aus. Zeile 80 übrigens erlaubt Ihnen später den Schwierigkeitsgrad zu verändern.

Die geforderte richtige Reihenfolge der Buchstaben A, ich nenne sie hier R, setzen wir am Anfang auf 65 und erhöhen sie schrittweise um 1.

```
 30 R = 65
310 R = R+1
```

In Zeile 60 prüfen wir, ob die Endzahl 71 für das G überschritten ist. Wenn ja, springen wir auf Zeile 400, mit der wir das Spielende anzeigen.

```
 60 IF R > 71 THEN 400
400 PRINT " ■■■
SPIELENDE"
```

Bitte RUNnen Sie das Fragment wieder zur Probe.

Jetzt kommt die Beeinflussung der Reihenfolge mit der DEL-Taste. Wie gelernt fragen wir diese Taste mit einer GET-Schleife ab (Zeilen 100,110), ihre Lösch-Wirkung erreichen wir in Zeile 120 durch einen PRINT-Befehl (mit Semikolon!). Nach Drücken der DEL-Taste darf der Reihenfolge-Zähler der Zeile 310 natürlich nicht wirken, deshalb springen wir schon vorher aus der Zeile 210 zurück.

```
100 GET A$
110 IF A$<>" ■ " THEN 300
120 PRINT " ■ ";
210 GOTO 50
```

Sie sehen oben, daß ich für die Abfrage der DEL-Taste die Finkel-Methode vorschlage. Die anderen Methoden gehen natürlich auch.

Nach RUN springt das Programm auf die noch nicht existierende Zeile 300 (was prompt zur Fehlermeldung führt), es sei denn, Sie drücken rechtzeitig die DEL-Taste.

In der Zeile 300 wollen wir prüfen, ob ein Fehler gemacht wurde, das heißt ob A mit der Reihenfolge R übereinstimmt. Im Fehlerfall wird die Fehlerzahl F um 1 erhöht. Vorher aber muß F auf 0 gesetzt werden.

```
 20 F=0
300 IF A<>R THEN F=F+1
```

Sie können jetzt schon das Spiel üben. Aber es fehlen noch ein paar Feinheiten.

```
 10 PRINT CHR$(147)
410 PRINT F "Fehler"
```

Zeile 10 ist klar, Zeile 410 druckt am Spielende die Fehlerzahl F aus.

Aber es gibt noch einen Fehler des Spielers, nämlich wenn er aus Versehen einen richtigen Buchstaben zurückweist. Deshalb fragen wir nach erfolgtem Drücken der DEL-Taste in den Zeilen 100 bis 120 nach, ob der Buchstabe tatsächlich falsch war. Wenn nicht, wird die Fehlerzahl F um 1 erhöht.

```
200 IF A=R THEN F=F+1
```

Damit uns nach RUN der erste Buchstabe nicht überrascht, verzögern wir sein Erscheinen mit

```
40 FOR T=1 TO 600: NEXT T
```

Zum Finkeln-Üben arrangieren wir die Anzeige des Spielendes und der Fehler etwas um. Alle Anweisungen sollen in nur einer Zeile stehen. Löschen Sie bitte die Zeile 410. In Zeile 400 wird gefinkelt und zwar mit dem Zeichen für SHIFT RETURN, welches laut Tabelle 2 mit SHIFT M erzeugt wird.

```
400 PRINT " ■■■■    DAS
SPIEL IST ZU ENDE";" ■■■■"
F "FEHLER".
```

Bei LIST und bei Ausdruck mit einem Drucker sehen die gefinkelten Zeilen 110, 120 und 400 natürlich kurios aus und wie gesagt, sie lassen sich bei einem Tippfehler nicht korrigieren, sondern müssen neu geschrieben werden.

Der Bildschirm-Code

Der Vollständigkeit halber will ich noch die letzte der vorher genannten drei Methoden, ein Zeichen auf den Bildschirm zu bringen, erwähnen, insbesondere, weil der dabei verwendete Bildschirm-Code (auch Video-Code genannt) oft zu Verwechslungen mit dem ASCII-Code führt.

Auf Anhieb ist es auch nicht einzusehen, warum Commodore einen anderen Code verwendet, wenn ein Zeichen direkt auf den Bildschirm — oder genauer gesagt in den Bildschirm-Speicher — gePOKEt werden soll.

Der Grund dafür liegt darin, daß dem ASCII-Code nicht nur Zeichen zugeordnet sind, sondern auch Farben und Funktionen. Außerdem sind im ASCII-Code die reversen Zeichen nicht enthalten, sondern müssen — wie Sie ja inzwischen wissen — jeweils umgeschaltet werden. Das alles ist für ein Betriebssystem viel zu kompliziert.

Es ist viel einfacher, im Festspeicher (ROM) alle Zeichen der zwei Zeichensätze fest zu verankern, von wo sie das Betriebssystem herausholen und auf den Bildschirm bringen kann.

Die Reihenfolge der Zeichen und ihr Code sehen Sie in der Tabelle 3. Sie ähnelt in mehreren Bereichen der ASCII-Reihenfolge, einige Spalten sind sogar identisch. Das macht eine Umrechnung — auch für das Betriebssystem — sehr einfach.

Folgende Blöcke der beiden Codearten entsprechen einander:

ASCII-CODE	BILDSCHIRM-CODE
0 - 31	entspricht keinem Zeichen
32 - 64	32 - 64
64 - 95	0 - 31
96 - 127	64 - 95
192 - 223	
128 - 159	entspricht keinem Zeichen
160 - 191	96 - 127
224 - 255	
entspricht keinem ASCII-Code	128 - 255

Ein Programm zur Umrechnung von ASCII-Code in Bildschirm-Code sieht dementsprechend aus wie in Programm 3 (VC 20 und C 64).

Dabei habe ich als Variable gewählt:

— ACII = ASCII-Code
— Bild = Bildschirm-Code

Im Hinblick darauf, daß unser Hauptthema die Abfrage der Tastatur ist, soll uns dieser Ausflug genügen.

(Dr. Helmut Hauck)