Teil 2

Alle Tasten-, Zeichen- und

Hier lesen Sie, wie man eine vollständige Tabelle der ASCII-Codes vom VC 20 und Commodore 64 erstellt. Außerdem werden Sie erfahren, daß die beiden Systeme wesentlich mehr Funktionstasten bieten als Sie bisher vielleicht angenommen haben.

Im ersten Teil habe ich Ihnen gezeigt, wie alle Tasten des VC 20 beziehungsweise des C 64 in einer Matrix angeordnet sind. Sobald eine Taste gedrückt wird, steht eine spezielle, nur dieser einen Taste zugeteilte Code-Zahl im Register 37152 des VC 20. Das entsprechende Register des C 64 ist 56320.

Ich habe auch erklärt, wie die Code-Zahlen zustande kommen. In einem kleinen Demonstrationsprogramm haben wir dann durch Abfragen dieses Registers mit PEEK bestimmte Programmschritte mit Tastendruck gesteuert.

Ein kurzer Rückblick

Wir haben auch herausgefunden, daß das Betriebssystem des Computers bei Verwendung von Basic eine Abfrage von nur acht Tasten zuläßt. Die Abfrage aller Tasten, auch mehrerer Tasten gleichzeitig, mit einem Programm in Maschinencode habe ich Ihnen für den Schluß versprochen.

Wir sind aber noch einen Schritt weitergegangen und haben herausgefunden, daß diese Code-Zahlen der 64 Tasten umgerechnet und für Zeichen- und Steuertasten getrennt

```
505       REM *** Programm 5 ******************
506       REM DER VOLLSTÄNDIGE ASCII-CODE *
507       REM ********************************
510       PRINT CHR$(147)
520       FOR I=0 TO 255
530       IF I<0 THEN I=255
540       PRINT:PRINT:PRINT:PRINT:PRINT
550       PRINT"_________"
560       PRINT
570       PRINT I;"[" CHR$(I);"]","... AAA"
580       PRINT
590       PRINT "_________"
600       Z=PEEK(203)

                610       IF Z=64 THEN 600
                620       IF Z=39 THEN POKE 36879,11:GOTO 600
                630       IF Z=47 THEN POKE 36879,27:GOTO 600
VC 20           640       IF Z=55 THEN POKE 36879,155: GOTO 600
                650       IF Z=61 THEN I=I-2

                610       IF Z=64 THEN 600
                620       IF Z=4 THEN POKE 53280,3: POKE 53281,0: GOTO 600
                630       IF Z=5 THEN POKE 53280,3: POKE 53281,1:GOTO 600
C 64            640       IF Z=6 THEN POKE 53280,3: POKE 53281,10: GOTO 600
                650       IF Z=43 THEN I=I-2

660       FOR T=1 TO 100:NEXT T
670       PRINT CHR$(147)
680       NEXT I
690       GOTO 520
```

Bild 4: Programm zur Bestimmung des Kompletten ASCII-Codes

Steuercodes

in die Speicherzellen 203 und 653 gebracht werden.

Mit dem folgenden kleinen Programm haben wir dann diese beiden Speicherzellen abgefragt und eine Tabelle angefertigt.
Tippen Sie ein:
100 PRINT PEEK(203),PEEK(653)
200 GOTO 100

Für diejenigen Leser, die Teil 1 nicht gelesen oder keine Tabelle angefertigt haben, habe ich diesmal die Tabelle dabei. Der Grund, daß die Zahlen für die beiden Computer, trotz gleicher Tastatur, verschieden sind, liegt darin, daß die elektrische Anordnung der Tasten, in einer 8 x 8-Matrix (die ich das letzte Mal für den VC 20 gezeigt habe), beim C 64 anders sind.

```
 5 REM***VC-20   PROGRAMM 1**********
 6 REM TASTENABFRAGE IN 203 UND 653 *
 7 REM*******************************
 10 PRINT CHR$(147)
 20 A=PEEK(203)
 30 B=PEEK(653)
 40 IF A=39 AND B=0 THEN POKE 36879,126
 50 IF A=39 AND B=1 THEN POKE 36879,45
 60 IF A=47 AND B=4 THEN POKE 36879,25
 70 IF A=53 AND B=0 THEN POKE 36879,27
 80 GOTO 20


 105 REM*** VC-20 PROGRAMM 2 **********
 106 REM UMRECHNUNG DES TASTEN-CODES *
 107 REM              IN ASCII-CODE   *
 108 REM*******************************
 110 A=PEEK(203)
 120 B=PEEK(653)
 130 IF B=0 THEN Z=60510
 140 IF B=1 THEN Z=60575
 150 IF B=2 THEN Z=60640
 160 IF B=4 THEN Z=60835
 170 C=PEEK(Z+A)
 200 PRINT A;B;C
 210 FOR T=1 TO 200:NEXT
 220 GOTO 110


 305 REM***VC-20 PROGRAM 3 **********
 306 REM   ABFRAGE DES ASCII-CODES    *
 307 REM   AUS DEM TASTATURPUFFER     *
 308 REM*******************************
 310 PRINT CHR$(147)
 320 POKE 198,0
 330 A=PEEK(631)
 340 PRINT A
 350 GOTO 320

READY.
```

Bild 2: Die im Text erklärten Programme für den VC 20

```
5 REM***VC-20   PROGRAMM 1**********
6 REM TASTENABFRAGE IN 203 UND 653 *
7 REM*******************************
10 PRINT CHR$(147)
20 A=PEEK(203)
30 B=PEEK(653)
40 IF A=39 AND B=0 THEN POKE 36879,126
50 IF A=39 AND B=1 THEN POKE 36879,45
60 IF A=47 AND B=4 THEN POKE 36879,25
70 IF A=53 AND B=0 THEN POKE 36879,27
80 GOTO 20

105 REM*** VC-20 PROGRAMM 2 **********
106 REM UMRECHNUNG DES TASTEN-CODES *
107 REM              IN ASCII-CODE   *
108 REM*******************************
110 A=PEEK(203)
120 B=PEEK(653)
130 IF B=0 THEN Z=60510
140 IF B=1 THEN Z=60575
150 IF B=2 THEN Z=60640
160 IF B=4 THEN Z=60835
170 C=PEEK(Z+A)
200 PRINT A;B;C
210 FOR T=1 TO 200:NEXT
220 GOTO 110

305 REM***VC-20 PROGRAM 3 **********
306 REM   ABFRAGE DES ASCII-CODES    *
307 REM   AUS DEM TASTATURPUFFER     *
308 REM*******************************
310 PRINT CHR$(147)
320 POKE 198,0
330 A=PEEK(631)
340 PRINT A
350 GOTO 320

READY.
```

Bild 3: Die im Text erklärten Programme für den C 64

TASTE	VC-20		C-64	
	203	653	203	653
nichts	64	0	64	0
f-1	39	0	4	0
f-3	47	0	5	0
f-5	55	0	6	0
f-7	63	0	3	0
A	17	0	10	0
B	35	0	28	0
C	34	0	20	0
D	18	0	18	0
E	49	0	14	0
F	42	0	21	0
G	19	0	26	0
H	43	0	29	0
I	12	0	33	0
J	20	0	34	0
K	44	0	37	0
L	21	0	42	0
M	36	0	36	0
N	28	0	39	0
O	52	0	38	0
P	13	0	41	0
Q	48	0	62	0
R	10	0	17	0
S	41	0	13	0
T	50	0	22	0
U	51	0	30	0
V	27	0	31	0
W	9	0	9	0
X	26	0	23	0
Y	11	0	25	0
Z	33	0	12	0
1	0	0	56	0
2	56	0	59	0
3	1	0	8	0
4	57	0	11	0
5	2	0	16	0
6	58	0	19	0
7	3	0	24	0
8	59	0	27	0
9	4	0	32	0
0	60	0	35	0
+	5	0	40	0
-	61	0	43	0
*	14	0	49	0
/	30	0	55	0
=	46	0	53	0
↑	54	0	54	0
←	8	0	57	0
.	37	0	44	0
:	45	0	45	0
,	29	0	47	0
;	22	0	50	0
£	6	0	48	0
@	53	0	46	0
CRSR←	23	0	2	0
CRSR↑	31	0	7	0
DEL	7	0	0	0
HOME	62	0	51	0
STOP	24	0	63	0
RETURN	15	0	1	0
SPACE	32	0	60	0
SHIFT	64	1	64	1
C=	64	2	64	2
CTRL	64	4	64	4
SHIFT u. C=	64	3	64	3
SHIFT u.CTRL	64	5	64	5
C= u. CTRL	64	6	64	6
SHIFT u. C= u.CTRL	64	7	64	7

Bild 1: Tabelle des Tastencodes in den Register 203 und 653

Im Gedenken an diesen Spruch meines Latein-Lehrers zeige ich noch einmal die Anwendung dieser Code-Zahlen in dem folgendem Programm 1. Zeilen in ( ), wie gesagt, sind für den C 64, aber nur dort, wo er sich vom VC 20 unterscheidet.

Die Wiederholung ist die Mutter der Weisheit

Natürlich wähle ich wieder die Funktionstasten und am besten auch noch eine andere Tastenkombination. Zweck der kleinen Demonstration soll das Umschalten auf verschiedene Bildschirmrahmen- und Hintergrundfarben sein. Ich schlage vor, Sie nehmen wie üblich die Zeitschrift zum Computer und lesen tippend weiter. Es folgt nun das Programm 1 zur Tastaturabfrage.

```
10 PRINT CHR$(147)
```

Diese Befehlsfolge löscht den Bildschirm. Die Code-Zahlen der Funktion CHR$ werde ich noch erklären.

Die Zeilen 20 und 30 erleichtern die Tipperei und machen übrigens das Programm ein bißchen schneller. Sie ordnen den Variablen A und B den Inhalt der Speicherzellen 203 und 653 — die wir ja abfragen wollen — zu.

```
20 A=PEEK(203)
30 B=PEEK(653)
```

Jetzt geht's los mit der Fragerei. Die Taste f1, mit der wir die Farbkombination Blau/Gelb schalten wollen, hat folgende Code-Zahlen:

```
40 IF A=39 AND B=0 THEN
POKE 36879,126
(40 IF A=4 AND B=0 THEN POKE
53280,6:POKE 53281,7 )
```

f2 ist dieselbe Taste, aber geSHIFTet (B=1). Die Farben sollen jetzt Rot/Grün sein:

```
50 IF A=39 AND B=1 THEN
POKE 36879,45
(50 IF A=4 AND B=1 THEN
POKE 53280,5: POKE 53281,2)
60 IF A=47 AND B=4 THEN
POKE 36879,25
(60 IF A=5 AND B=4 THEN
POKE 53280,1: POKE 53281,1)
```

Zeile 60 bestimmt ebenfalls eine Funktionstaste und zwar f3. Allerdings habe ich sie willkürlich mit der CTRL-Taste kombiniert, nicht, um Sie zu verwirren, sondern um zu zeigen, daß wir mit den f-Tasten mehr als acht Funktionen festlegen können, nämlich 32! (Vier f-Tasten mal acht Steuertasten-Kombinationen).'

Mit dem Klammeraffen »@« schalten wir die Farben in den Normalzustand zurück.

```
70 IF A=53 AND B=0 THEN
POKE 36879,27
(70 IF A=46 AND B=0 THEN
POKE 53280,3: POKE 53281,1)
```

Zeile 80 läßt das Programm im Kreis laufen, so daß die Funktionstasten beliebig oft ausprobiert werden können. S 80 GOT020

Funktionstasten im Überfluß

So, jetzt kommt der absolute Hit dieser Methode!

Ich habe gerade vorher gesagt, daß wir nicht acht, sondern 32 mögliche Funktionstasten haben, nämlich durch Verwendung der vier f-Tasten mit den acht Steuertasten-Codes in Speicherzellen 653.

Dasselbe gilt für jede andere Taste natürlich auch!

Der Computer benützt allerdings einige davon, zum Beispiel die 1 (SHIFT) für die Zeichen über den Zahlen beziehungweise rechts unten auf den Tasten, die 2 (C=) für die Zeichen links unten auf den Tasten und die 4 (CTRL) für die Farben.

Wir können daher mit den vom Computer benutzten Zahlen 3, 5, 6 und 7 aus Zelle 653 in Kombination mit allen 55 Tasten eine riesige Anzahl verschiedener »Funktionstasten« erfinden und sie in unseren Programmen einsetzen.

Der Computer geht bei der Abfrage noch einen Schritt weiter

Aber auch die 4 der CTRL-Taste, die ja nur die obere Reihe der Tasten beeinflußt, hat einen praktischen Wert, meiner Meinung nach sogar einen sehr großen, da sie ja nur einen einzigen Tastendruck erfordert und nicht eine Kombination.

Die CTRL-Taste kombiniert mit den 35 Tasten der unteren drei Reihen der Tastatur gibt uns mehr »Funktionstasten« als wir wahrscheinlich jemals brauchen werden.

Mehrere handelsübliche Zusatzmodule und -programme verwenden diese Methode, zum Beispiel auch die »Programmierhilfe« von Commodore. Verwenden Sie's doch auch! Das Kochrezept dazu steht oben in den Zeilen 50 bis 80.

Um das Ergebnis eines Tastendrucks weiter zu verarbeiten, wäre dem Computer die Abfrage zweier Speicherzellen zu langsam. Außerdem entsprechen diese Tastencodes keiner internationalen Norm, was in Verbindung mit anderen Geräten sehr lästig wäre.

Es gibt den international anerkannten ASCII-Code (American Standard Code for Information Interchange), der ursprünglich für die Zeichenübertragung von Fernschreibern erfunden wurde. Er besteht aus Dualzahlen mit einer Länge von 8 Bit oder falls Sie Dualzahlen nicht kennen (eine erste Einführung ins Dualsystem steht im Grafikkurs) aus Zahlen von 0 bis 255.

In diesen ASCII-Code wandelt der Computer nun die oben verwendeten Codezahlen der Tasten um. Die Umwandlung ist denkbar einfach.

Umrechnung des Tastatur-Codes in den ASCII-Code

Im nicht löschbaren Speicher (ROM) des Betriebssystems stehen vier Tabellen mit Zahlen.

Wenn man nun die Code-Zahl einer Taste zur Anfangsadresse der Tabellen addiert, erhält man eine Adresse, in der die ASCII-Codezahl gespeichert ist. Einfach, nicht wahr?

Probieren geht über studieren. Die Tabelle der ungeSHIFTeten Zeichen, also der Zahlen und Großbuchstaben, beginnt beim VC 20 ab Speicherzeile 60510, beim C 64 ab 60289.

Nehmen wir das »G«, sein Tatencode aus der Tabelle ist 19 (26).

Zu 60510 (60289) dazugezählt ergibt das 60529 (60315). Nun wollen wir mal nachschauen, was in dieser Zelle steht.

Geben Sie »direkt«, (das heißt ohne Zeilenzahl) ein:

```
für VC 20: PRINT PEEK (60529)
für C 64: PRINT PEEK (60315)
```

Das Resultat ist 71. Ein Blick auf die ASCII-Liste des Handbuches (oder in jede andere ASCII-Tabelle) zeigt uns die Richtigkeit dieser Aktion. Der ASCII-Code des Zeichens »G« ist 71.

Um alle ASCII-Codes abfragen zu können, schreiben wir wieder ein kleines Programm.

```
110 A=PEEK (203)
120 Z=60510
(120 Z=60289)
170 C=PEEK(Z+A)
```

Bitte verwenden Sie meine Zeilennummern, ich möchte nämlich später noch andere Zeilen einschieben.

Diese drei Zeilen sollten Ihnen klar sein. Zur Erinnerung: 60510 ist die Anfangsadresse der Code-Tabelle beim VC 20. Die 64er müssen also ihre eigene Adresse (60289) nehmen.

Als Ergebnis wollen wir noch den Tastaturcode (Zeile 110) und den ASCII-Code (Zeile 170) nebeneinander ausdrucken:

```
200 PRINT A;C
220 GOTO 110
```

Damit die beiden Zahlenbänder nicht zu schnell laufen, fügen wir noch eine Verzögerungsschleife ein.

```
210 FOR T=1 TO 200: NEXT
```

Dieser Programmablauf, der mit RUN 110 gestartet wird, reagiert jetzt auf jeden Tastendruck, links mit dem Tastencode, rechts mit ASCII. Worauf er aber nicht reagiert, sind geSHIFTete Zeichen, solche mit der Commodore-Taste »C« und die Farben (mit CTRL).

Sie wissen, warum? Natürlich, denn wir fragen ja nur die Speicherzellen 203 ab und nicht zusätzlich auch 653.

Jetzt muß ich noch schnell erwähnen, daß auch der Computer diese Zellen abfragt.

Wo steht was?

Erinnern Sie sich, ich habe oben gesagt, daß im ROM Tabellen stehen:

(1) ab 60510 (60289) für normale Zeichen
(2) ab 60575 (60354) für Zeichen mit SHIFT
(3) ab 60640 (60419) für Zeichen mit C=
(4) ab 60835 (64632) für Zeichen (Farben) mit CTRL

Das bauen wir jetzt in das Programm ein:

Zeile 120 ändern wir ab. Sie fragt jetzt die Speicherzelle 653 nach den Steuertasten ab. In den Zeilen 130 bis 160 springen wir auf die vier Tabellenanfänge. Der Ausdruck in Zeile 200 schließlich wird mit der Codezahl aus 653 erweitert, nämlich B.

```
120 B=PEEK(653)
130 IF B=0 THEN Z=60510 (60289)
140 IF B=1 THEN Z=60757 (60354)
150 IF B=2 THEN Z=60640 (60419)
160 IF B=4 THEN Z=60835 (64632)
200 PRINT A;B;C
```

Die Reihenfolge der Zahlenbänder auf dem Bildschirm — entsprechend der Zeile 200 — ist jetzt von links Zeichentaste, Steuertaste, ASCII-Code.

Der ASCII-Code wird im Tastatur-Puffer abgelegt

Ehrlich gesagt, das Verfahren der Tastaturabfrage in dem vorherigen Programm ist immer noch recht kompliziert. Eine direkte Abfrage des ASCII-Codes einer Taste wäre viel besser.

Und in der Tat, der Computer bietet sie uns. Er bringt nämlich jeden ASCII-Wert in einen Speicher zur Zwischenlagerung, bis er von einem Programmschritt gebraucht wird.

Dieser Speicher heißt »Tastatur-Puffer« und liegt von Speicherzelle 631 bis einschließlich 640.

Es können also maximal zehn Werte gespeichert werden. Das erste Zeichen steht immer in 631, alle anderen werden der Reihe nach in die folgenden Zellen gebracht.

Als erstes wird das Zeichen aus 631 ausgelesen und alle anderen rücken nach.

Wenn das Programm die ASCII-Werte aus dem Tastaturpuffer auslesen kann, dann können wir das natürlich auch. Das folgende kleine Programm soll es beweisen.

```
310 PRINT CHR$(147)
330 A=PEEK(631)
```

Sie sehen, wir wollen ganz einfach in Zelle 631 des Tastaturpuffers nachschauen, welcher ASCII-Wert nach Drücken einer Taste dort steht (Zeile 330). In Zeile 340 drucken wir den Wert aus und springen dann zum PEEK-Befehl zurück.

```
340 PRINT A
350 GOTO 330
```

Wenn Sie dieses Programm mit RUN 310 laufen lassen, werden Sie merken, daß es nicht geht, ich will sagen; noch nicht geht. Den Grund dafür habe ich kurz vorher schon angedeutet.

Wo sind die Computer-Detektive? Haben Sie's gemerkt?

Nun, ich habe erklärt, daß die ASCII-Werte der gedrückten Tasten der Reihe nach im Tastaturpuffer gespeichert werden und dann, wenn ein Wert aus 631 ausgelesen wird, nachrücken. Das ist der springende Punkt: durch PEEKen lesen wir nicht aus, wir schauen nur nach!

Es gibt zwei Lösungen für dieses Problem:

1) Wir verwenden einen Befehl, der den Wert herausholt; das ist GET oder INPUT.

2) Wir gaukeln dem Computer vor, daß der Tastaturpuffer nur aus einer einzigen Speicherzelle besteht.

Die Methode 2 ist exotischer, deshalb zeige ich Sie Ihnen zuerst. Es gibt eine Speicherzelle 198. In dieser Zelle steht eine Zahl, die angibt, wieviele Zeichen im Tastaturpuffer Platz haben. Normalerweise steht da eine 10 (schauen Sie nach).

Diese Zelle kann mit kleineren Zahlen gePOKEt werden, auch mit einer 0. Die 0 löscht sozusagen den Puffer. Das machen wir jetzt in unserem Programm vor dem PEEKen:

```
320 POKE 198,0
350 GOTO 320
```

Jetzt läuft's.

Die Zeile 340 gibt uns also den ASCII-Code der gerade gedrückten Taste auf dem Bildschirm aus, zum Beispiel die Zahl 84 für das »T«, 163 für das »—« (T mit C=) und so weiter.

Unter Verwendung von GET würde unser Programm so aussehen:

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

Neu ist die Verwendung der Funktion ASC. Sie bildet den ASCII-Wert des Zeichens A$.

Für diejenigen, die es noch nicht kennen: Zeile 330 nach dem GET ist erforderlich, da der GET-Befehl nicht auf das Drücken einer Taste wartet, sondern gleich weiterläuft. Solange aber keine Taste gedrückt ist, geht die Schleife nach 320 zurück.

Mit INPUT geht's noch kürzer, nur ist die Bedienung etwas umständlicher.

```
310 PRINT CHR$(147)
320 INPUT A$
340 PRINT ASC(A$)
350 GOTO 320
```

Der Umstand liegt daran, daß INPUT im Gegensatz zu GET auf einen Tastendruck wartet, der zusätzlich mit RETURN abgeschlossen werden muß.

Nach diesen Erklärungen und Versuchen müßten Sie eigentlich in der Lage sein, Programm 1 so umzuschreiben, daß statt der Abfrage der Speicherzellen 203 und 653 der Tastaturpuffer abgefragt wird. Ich schlage Ihnen vor, daß Sie das jetzt selbst ausprobieren, sozusagen als Hausaufgabe. Die Lösung, das Programm Nummer 4, ist an anderer Stelle in dieser Ausgabe versteckt.

Das einzige, was ich Ihnen verraten will, wissen Sie eigentlich schon, nämlich, daß Sie sich die ASCII-

Werte für die f-Tasten und den Klammeraffen »@« besorgen müssen. Aber wozu haben Sie Programm 3 gleich in drei Versionen?

Ich hoffe, daß Sie nach dieser Übung einsehen, daß Sie eine vollständige Liste aller ASCII-Codezahlen und ihrer Bedeutung unbedingt brauchen. Halt, werden Sie jetzt sagen, die Liste steht ja in jedem Handbuch — sogar in dem von Commodore.

Das stimmt, nur sind die meisten Listen nicht komplett.

Erinnern Sie sich? Ich habe vorher mal gesagt, daß der ASCII-Code die Werte von 0 bis 255 hat. Diese werden von den Commodore-Computern in nicht immer ganz der Norm entsprechender Weise für alle möglichen Zeichen und Sonderfunktionen verwendet, wie zum Beispiel die Farben, die Sonderzeichen auf den Tasten, Zeichenumschaltung und so weiter.

Auch die Funktion »Bildschirm löschen und Cursor auf HOME-Position« (das heißt die CLR/HOME-Taste) ist dabei — mit dem Codewert 147. Merken Sie was? Schauen Sie mal die jeweiligen 10er-Zeilen der Programme bisher an!

Es lohnt sich also schon, alle Code-Werte und die dazugehörigen Zeichen und Funktionen anzusehen.

Ihnen die Liste einfach abzudrucken wäre zu simpel. Sie sollen ja durch Experimentieren Ihren Computer besser kennenlernen. Ich liefere Ihnen die Versuchsanordnung dazu.

Vorher aber brauchen wir noch ein Hilfsmittel, welches uns erlaubt, aus einem ASCII-Wert das Zeichen beziehungsweise die Funktion zu ermitteln. Es ist die Umkehrung der in Programm 3 verwendeten ASC-Funktion. Sie kommt ebenfalls aus Basic und heißt CHR$(x).

Dieser Befehl liefert uns das Zeichen oder die Funktion des ASCII-Codes x.

Der Befehl PRINT CHR$(x) bringt Zeichen auf den Bildschirm. (Mit dem Befehl PRINT# a,CHR$ wird das Zeichen an ein beliebiges, mit der Nummer a bezeichnetes Peripheriegerät gebracht. Doch das will ich hier nicht weiter verfolgen.)

Jetzt aber zurück zu dem Hilfsmittel

Ergänzen Sie bitte in Programm 3 die Zeile 340 auf:

```
340 PRINT A, CHR$(A)
```

Jetzt druckt das Programm nach Start mit RUN 310 neben dem ASCII-Code auch das Zeichen, welches natürlich mit der gedrückten Taste identisch ist, auf den Bildschirm. Funktionen kann man allerdings nicht ausdrucken, sondern nur ihre Auswirkungen feststellen.

Doch nun zum Kochrezept. Der entscheidende Teil steht in Zeile 570.

```
570 PRINT I;''['']CHR$(I)'']'';...
''AAA''
```

I ist die ASCII-Codezahl, die in einer FOR-NEXT-Schleife von 0 bis 255 hochgezählt wird. Die beiden Klammern [und] stehen in Anführungszeichen, damit sie ausgedruckt werden. Zwischen ihnen soll das zum Wert I gehörige Zeichen stehen.

In den Fällen, wo der ASCII-Code nicht ein Zeichen, sondern eine Funktion bedeutet, bleibt die Klammer leer. Aber der Code wirkt sich durch die Form PRINT CHR$(I) auf die 3 As aus (die Punkte ... stellen drei Leertasten dar). Zum Beispiel erscheinen sie nach I=28 in roter Farbe, bei I=17 (Cursor Down) eine Zeile tiefer.

Das Hochzählen von I in Zeile 520 wollen wir aber ein bißchen beeinflussen, und das natürlich mit Drücken von Funktionstasten und solchen, die wir dazu verdonnern.

In Zeile 600 werden daher solche Tasten abgefragt, mit der »alten« Methode in Zelle 203. Das ist reine Willkür beziehungsweise Sentimentalität von mir, die »neue« Methode über Zelle 631 geht genauso gut.

Wenn in 203 eine 64 steht (keine Taste gedrückt), dann wartet das Programm durch Rücksprung auf die Zeile 600.

Falls wir die f1-Taste drücken, schaltet Zeile 620 den Hintergrund auf Schwarz und geht wieder in Wartestellung. Diese und die beiden anderen Farbumschaltungen mit f3 (Zeile 630) auf weiß und mit f5 auf hellorange (Zeile 640) sind dann sehr nützlich, wenn die Farbe der drei As gegen den Hintergrund nur schlecht oder überhaupt nicht lesbar sind.

Zeile 650 gibt uns die Möglichkeit, mit der Minus-Taste in der Liste um 1 zurückzuschalten. Zeile 530 erlaubt ein Zurückschalten über die 0 nach 25. Erst wenn irgendeine beliebige Taste gedrückt wird, springt das Programm auf Zeile 660, wo nach kurzer Zeitverzögerung der Bildschirm gelöscht (70) und I weitergezählt wird (680 und 690).

So können Sie sich bequem alle 256 Werte des ASCII-Codes und ihre Wirkung anschauen.

Ich möchte Sie hier noch auf folgende Codewerte aufmerksam machen, deren Funktion Sie in diesem Programm entweder nicht sehen könnnen oder deren Funktion den Ablauf stören:

Code	Funktionen
3	entspricht der ungeSHIFTeten STOP/RUN-Taste
8	setzt die Umschaltung (mit SHIFT und C=) auf den 2. Zeichensatz außer Betrieb (ausprobieren!)
9	hebt die Sperre wieder auf
13	entspricht der RETURN-Taste
14	schaltet den 2. Zeichensatz per Programm ein (ich empfehle, danach wieder auf den »normalen« Zeichensatz zurückzuschalten)
131	entspricht LOAD/RUN (geSHIFTete STOP/RUN-Taste)
133-140	Funktionstasten f1 bis f8
141	geSHIFTete RETURN-Taste
142	schaltet den 1. Zeichensatz ein (Umkehrung von 14)
146	REVERSE-OFF (Taste 0 mit CTRL)
160	geSHIFTete SPACE-Taste (ja, ja, das gibt es auch!)

Ich empfehle Ihnen, mit diesem Programm 5 zu experimentieren. Versuchen Sie, besonders die Funktionen zu identifizieren, es ist nicht schwer. Zusätzlich sollten Sie alle sinnvollen ASCII-Werte mit den Ihnen zur Verfügung stehenden ASCII-Listen vergleichen. Verbessern und vervollständigen Sie diese Listen. Sie gehören zu Ihrem wichtigsten Handwerkszeug.

Im nächsten und zugleich letzten Teil werde ich Ihnen die Zusammenhänge zwischen dem ASCII-Code, dem Bildschirm-Code und dem PRINTen in Anführungszeichen erklären, natürlich wieder mit Kochrezepten.

Und schließlich will ich mein Versprechen einlösen, endlich die Methode der Abfrage von mehreren gleichzeitig gedrückten Tasten zu zeigen.

Übrigens, wenn Sie Fragen haben, scheuen Sie sich nicht, an den Verlag beziehungsweise über den Verlag an mich zu schreiben. Ich versuche mein Bestes.

(Dr. Helmut Hauck)

```
7160 PRINT#1,"RANG  PKT.  KM   FAHRZEUG                TEAM";CHR$(27);"H"
7170 PRINT#2,"
--------"
7180 PRINT#1
7200 GOSUB10000
7220 FORY=XTO1STEP-1
7230 L=L+1
7240 IFY=KTHENPRINT#1,CHR$(27);"-";CHR$(1);:REM JEWEILGES TEAM WIRD UNTERSTRICHE
N
7250 IFL<10THENPRINT#1,"  ";L;:GOTO7270
7260 IFL<100THENPRINT#1," ";L;:GOTO7270
7270 PRINT#1," ";
7280 PRINT#1,G3(Y);KM(Y);" ";FT$(Y);" ";
7300 G=LEN(FT$(Y))+8:G=30-G
7310 FORT=1TOG:PRINT#1," ";:NEXTT
7320 PRINT#1,CHR$(15);TE$(Y);CHR$(18)
7350 :
7360 :
7370 IFY=KTHENPRINT#1,CHR$(27);"-";CHR$(0);
7380 NEXTY
7385 :
7810 PRINT#1,CHR$(27);"E"
7815 PRINT#1,"DIE RENNLEITUNG BEDANKT SICH FUER IHR FAIRES MITFAHREN !";CHR$(27)
;"F"
7820 REM PRINT#1,CHR$(12)
7840 CLOSE1:CLOSE2
7850 NEXTK
7900 RETURN:REM ENDE DES AUSDRUCKS PERSOENLICHE LISTEN
10000 REM SHELL-METZNER-SOTIERROUTINE
10010 :
10020 :
10030 J6=X
10040 J6=INT(J6/2)
10050 IF(J6=0)THEN10490
10060 J2=X-J6
10070 FORJ=1TOJ2
10080 I=J
10090 J3=I+J6
10100 IF(G3(I)<=G3(J3))THEN10160
10110 H1=G3(I):H2=KM(I):H3$=FT$(I):H4$=TE$(I)
10120 G3(I)=G3(J3):KM(I)=KM(J3):FT$(I)=FT$(J3):TE$(I)=TE$(J3)
10130 G3(J3)=H1:KM(J3)=H2:FT$(J3)=H3$:TE$(J3)=H4$
10140 I=I-J6
10150 IF(I>0)THEN10090
10160 NEXTJ
10170 GOTO10040
10490 RETURN:REM ENDE SHELL-SORT
30000 REM AUTOS DRUCKEN
30010 REM IN EPSON BIT IMAGE SCHREIBWEISE
30015 PRINT#1,CHR$(27);"F";
30025 FORT=1TO20
30030 PRINT#1,CHR$(27);"K";CHR$(22);CHR$(0);
30040 FORN=1TO22
30045 READD
30050 PRINT#1,CHR$(D);
30060 NEXTN
30065 RESTORE:NEXTT
30800 DATA24,56,56,56,60,126,158,156,152,248,248,152,152,152,156,158,126,60,56,2
4,0,0
30900 RESTORE:RETURN
40000 OPEN1,4,0:CMD1:LIST:PRINT#1
41000 PRINT"[SHIFT-CLR]";:END

READY.
```

Listing 2. Programmteil 2 (Schluß)

Programm zur Tastenabfrage aus dem Tastaturpuffer

```
405   REM **** C 64 PROGRAMM 4 *********
406   REM TASTENABFRAGE AUS T.-PUFFER *
407   REM*************************************
410   PRINT CHR$(147)
420   POKE 198,0
430   A=PEEK(631)
440   IF A=133 THEN POKE 53280,6: POKE 53281,7
450   IF A=137 THEN POKE 53280,5: POKE 53281,2
460   IF A=134 THEN POKE 53280,1: POKE 53281,1
470   IF A=64 THEN POKE 53280,3: POKE 53281,1
480   GOTO 420
```

```
405   REM***VC 20 PROGRAMM 4 *********
406   REM TASTENABFRAGE AUS T.-PUFFER *
407   REM *************************************
410   PRINT CHR$(147)
420   POKE 198,0
430   A=PEEK(631)
440   IF A=133 THEN POKE 36879,126
450   IF A=137 THEN POKE 36879,45
460   IF A=134 THEN POKE 36879,24
470   IF A=64 THEN POKE 36879,27
480   GOTO 420
```

Sie haben natürlich gemerkt, daß die Abfrage der f3-Taste kombiniert mit CTRL aus dem Tastaturpuffer nicht geht. Der Grund dafür ist, daß diese Kombination keinen eigenen ASCII-Code hat. Die so laut gepriesenen 32 Funktionstasten sind also nur bei einer Abfrage der Speicherzellen 203 und 253 möglich. In Zeile 460 des Programms 4 habe ich daher reumütig die f3-Taste allein verwendet.

Diese beiden Programme gehören als Lösung zu einer Aufgabe die im Beitrag »Alle Tasten-, Zeichen- und Steuercodes« gestellt wurde.