Alle Tasten-, Zeichen- und Steuercodes

4. Teil und Schluß

In dem letzten Teil dieses Kurses finden Sie eine Zusammenfassung der Methoden, wie man in Basic Tasten abfragen kann.

Wir haben insgesamt vier Methoden kennengelernt und verwendet, um das Drücken einer Taste im Programm abzufragen:

1. Tastencode in Speicherzellen 203/653

```
10 A=PEEK(203)
20 B=PEEK(653)
30 IF A = ZEICHENCODE AND
     B=STEUERCODE THEN
     AKTION
```

2. ASCII-Code im Tastaturpuffer

```
100 POKE 198,0
110 A=PEEK(631)
120 IF A=ASCII-CODE THEN AK-
    TION
```

3. Abfrage des ASCII-Codes mit GET/INPUT

```
200 GET A$
210 IF A$<>CHR$(ASCII-CODE)
    THEN AKTION 1
220 AKTION 2
```

4. Abfrage des Gänsefuß-Modus mit GET/INPUT

```
300 GET A$
310 IF A$ »ZEICHEN« THEN AK-
    TION 1
320 AKTION 2
```

Diese vier Methoden haben alle eins gemeinsam:

Sie können immer nur eine einzelne Taste abfragen. Zwei oder gar mehrere Tasten gleichzeitig oder kurz hintereinander gedrückt ergeben keine sinnvollen Resultate.

Jetzt möchte ich Sie an meine allererste Darstellung im Aprilheft des »64'er«, Seite 115, erinnern, nämlich wie die Tasten elektrisch angeordnet sind und wie das Betriebssystem sie abfragt. Ich möchte das hier noch einmal darstellen, erstens weil es eine gute Überleitung bildet zu meiner Methode der Vielfach-Tastenabfrage, zweitens weil meine damalige Darstellung nicht vollständig war.

In Bild 1 ist noch einmal die VC 20-Tastenanordnung dargestellt, in Bild 2 diejenige des C 64. Trotz des Unterschiedes der elektrischen Anordnung ist bei beiden Computern die Abfragemethode identisch, nur die dafür benötigten Register haben unterschiedliche Adressen.

Zur Erinnerung:

Das Betriebssystem wählt der Reihe nach die senkrechten Spalten dadurch an, daß es die Zahl, die am Fuße jeder Spalte steht, in das Spalten-Register 37152 (beziehungsweise 56320) schreibt. Diese Zahl ergibt in dualer Darstellung eine 0 an dieser Stelle.

Für jede Taste, die in der angewählten Spalte gedrückt ist, wird eine 0 in das andere Register 37153 (56321) geschrieben. Diese Dualzahl ergibt einen Dezimalwert, welcher aus dem Register herausPEEKbar ist.

Das, was das Betriebssystem macht, machen wir ihm nach, zuerst für den VC 20:

```
10 POKE 37152,247
20 PRINT PEEK (37152);PEEK(37153)
30 GOTO 10
```

und für den C 64:

```
10 POKE 56320,127
20 PRINT PEEK(56320);PEEK(56321)
30 GOTO 10
```

In Zeile 10 ist noch ein weiterer Unterschied zwischen den beiden Computern zu sehen. Beim VC 20 habe ich die Spaltenzahl 247 gewählt, weil in dieser Spalte die STOP-Taste liegt. Diese Spalte ist nämlich, wie wir in Teil 1 ja schon festgestellt haben, die einzige Spalte, die wir mit Basic abfragen können. Bei POKEn der anderen sieben Spaltenzahlen in Zeile 10 wirft uns die 60mal in der Sekunde stattfindende Überprüfung der STOP-Taste aus dem Programm.

Beim C 64 ist das die Spalte 127, wie es uns ein Blick auf das Bild 2 zeigt.

Ich freue mich übrigens, daß ein aufmerksamer Leser aus Wien diesen Unterschied sofort bemerkt und mich darauf aufmerksam gemacht hat. Aber ich hatte damals nur einen VC 20 zur Verfügung, und mangels eigener Erprobung ist es mir nicht aufgefallen. Das hat sich übrigens jetzt geändert.

Diese Tastenabfrage hat den großen Vorteil, daß mehrere Tasten gleichzeitig drück- und abfragbar sind. Jede gedrückte Taste erzeugt eine 0 im »Reihen-Register«. Mit dem kleinen Programm oben können Sie es ausprobieren. Drücken Sie alle Tasten der Spalte 247 (127), die STOP-Taste bitte als letzte! Der rechte Zahlenstreifen auf dem Bildschirm zeigt die 0.

Lassen Sie die STOP-Taste (die 1-Taste) los, und es erscheint die 1, bei Loslassen der linken SHIFT-Taste (der ←-Taste) zusätzlich erhalten wir die 3. Das ist die Dezimalzahl für 00000011, die beiden Einser entstehen durch die losgelassenen Tasten.

Jede mögliche Tastenkombination in einer Spalte hat ihre spezielle und abfragbare (!) Codezahl im Register 37153 (56321), insgesamt 256 Kombinationen. Ist das nichts?

Diese Methode der Mehrfach-Tastenabfrage haben wir bereits in der Ausgabe 4/84 im ersten Teil meines Aufsatzes erfolgreich eingesetzt.

Heute möchte ich diese Methode auf alle acht Spalten, also auf alle Tasten erweitern. Dazu müssen wir das oben erwähnte Hindernis, nämlich den Rausschmiß durch das Betriebssytem, überwinden. Dazu gibt es zwei Methoden. Methode 1 will ich nur kurz erwähnen — sie ist einen eigenen Aufsatz wert.

Man kann durch Beeinflussung der Speicherzellen 788 und 789 die Pause (Interrupt) zur Tastenabfrage des Betriebssytems künstlich verlängern und sie zur eigenen Abfrage benutzen.

Die zweite Methode ist einfacher. Wir schreiben das Programm oben (Zeilen 10 und 20) in Maschinensprache. Das läuft dann so schnell ab, daß es innerhalb zweier Unterbrechungen ausgeführt und somit vom Betriebssystem nicht gestört wird.

Da ich nicht annehme, daß alle Leser dieses Kurses in Maschinencode programmieren können, zeige ich es Ihnen einfach als Kochrezept. Es besteht aus einer Reihe von Zahlen, die über READ ... DATA in vorbestimmte Speicherplätze gePOKEt und von dort dann mit SYS gestartet werden.

Für den VC 20 gilt:
100 DATA 169, 254 ,141,32,145,173, 33,145,141,255,29,96

Für den C 64 gilt:
100 DATA 169, 254 ,141,0,220,173,1, 220,141,255,29,96

Ich habe einige Zahlen gekennzeichnet:
— Die zweite Zahl der DATA-Reihe in den Kästchen ist die Zahl der Spalte, die angewählt wird (ich habe 254 gewählt).
— Die 32 und 145 (0,220 beim C 64) ergeben die Registeradresse 37152 (56320), die DATA-Werte 33 und 145 (1,220) die Registeradresse 37153 (56321), die Werte 255,29 ergeben eine Speicherzelle 7679, auf die ich noch zurückkomme.

Die Registeradressen sind, wie immer bei Commodores 8-Bit-Computern, mit zwei Zahlen, das heißt mit zwei Bytes dargestellt.

REGEL:
LSB+256*MSB = Adresse
32 + 256* 145 = 37152
1 + 256* 220 = 56321
LSB = niederwertiges Byte
MSB = höherwertiges Byte

Diese 12 DATA-Zahlen, ich nenne sie X, werden eingelesen mit:
110 FOR K=1 TO 12
120 READ X
140 NEXT K

Damit diese Zahlen vom Basicprogramm nicht überschrieben oder gelöscht werden (was dasselbe ist), POKEn wir sie an das Ende des VC 20-Speichers ohne Erweiterung, nämlich ab Speicherzelle 7661. Für den C 64 geht es mit denselben Adressen natürlich allemal.
130 POKE 7660+K,X

Um ganz sicher zu gehen, daß mit dem Maschinenprogramm nichts passiert, schützen wir es, indem wir dem Computer vorgaukeln, daß sein für Basic zur Verfügung stehender Speicher bei Adresse 7659 aufhört. Diese Speichergrenze steht in den Speicherzellen 51/52 und 55/56, und sie kann natürlich durch POKEn anderer Zahlen willkürlich verändert werden.

Die entsprechenden Werte für die Grenze 7659 sind 235 und 29. Machen wir die Probe:
PRINT 235+256+29 (RETURN)
ergibt 7659.

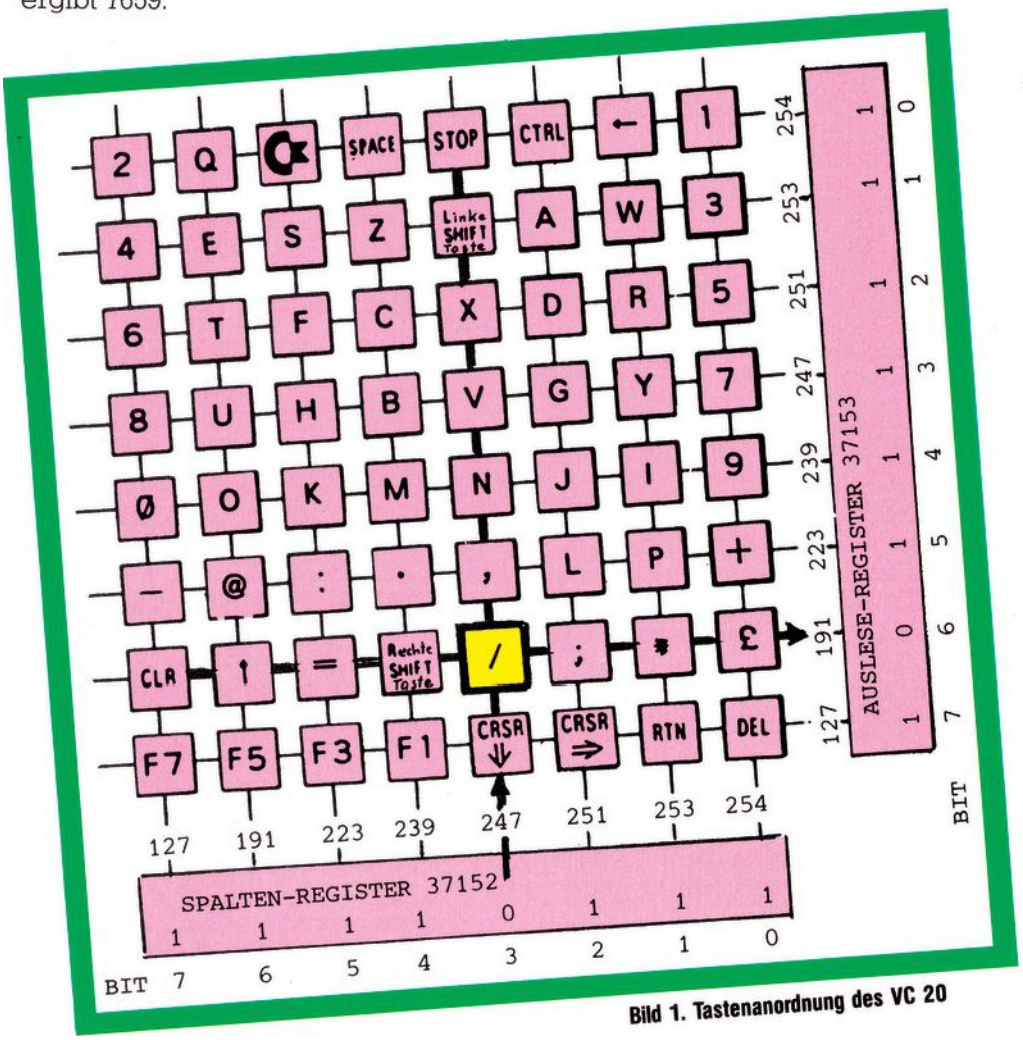

Bild 1. Tastenanordnung des VC 20

20 POKE 51,235
30 POKE 52,29
40 POKE 55,235
50 POKE 56,29

Falls Sie diese Speicherzellen und ihre Anwendung nicht kennen, bitte ich um Zuschrift, da ich es hier aus Platzgründen nicht näher erläutern kann.

So, nun ist das Maschinenprogramm gespeichert und gesichert. Wir starten das Programm mit SYS-Aufruf der 1. Adresse und PEEKen danach in das Register 37153 (56321), genau wie vorher.
150 SYS 7661
160 PRINT PEEK(37153): REM VC 20
160 PRINT PEEK(56321): REM C 64
170 GOTO 150.

Nach RUN erzeugt der Rücksprung den schon obligatorischen Zahlenstreifen mit 255 für »keine Taste gedrückt«. Wenn wir die Taste 3 (RETURN beim C 64) drücken, dann muß entsprechend Bild 1 und 2 (Spalte 254) die Zahl 253 erscheinen. Das tut sie auch, aber immer wieder unterbrochen durch 255. Das kommt daher, daß das Auslesen des Registers 37153 (56321) in Basic doch schon zu langsam ist. Wir müssen daher die Funktion der Zeile 160 ebenfalls in das Maschinenprogramm einbauen, was ich in weiser Voraussicht in Zeile 100 schon vorbereitet habe. Der Trick ist nämlich dabei, daß der Inhalt des Registers vom Maschinenprogramm in eine vor dem bösen Basic geschützte Speicheradresse gebracht wird. Da wir ja einen gesicherten Bereich bereits haben, habe ich darauf die Zeile 7679 ausgewählt.

Bitte ändern Sie die Zeile 160 ab:
160 PRINT PEEK(7679)

Jetzt haben wir's geschafft.

Natürlich können wir mit diesem Programm alle acht Spalten abfragen. Wir brauchen bloß in Zeile 100 die Zahl, die hier im Kästchen steht, abändern, zum Beispiel in 253, und schon reagiert das Programm auf alle Tasten dieser Spalte.

Ich habe mir den Luxus erlaubt

und unser Programm von oben erweitert, indem ich der Reihe nach alle acht Spalten aufrufe (siehe Listing 1). Dazu habe ich in Zeile 100 die Spaltennummer auf 0 gesetzt, was eigentlich unnötig aber einleuchtend ist, POKE aber dann jeweils die Spaltenzahl (in den Zeilen 200, 300, 400 etc.) auf diesen Platz, der die Adresse 7662 hat.

Das Programm ist immer noch schnell genug, um innerhalb der zur Verfügung stehenden Zeit von 1/60 Sekunde alle Abfragen durchzuführen.

Ein Appell an alle 64er: Wenn Sie die Semikolons in den Zeilen 220, 320, 420 etc. (außer 920!!!) richtig plaziert haben, erscheinen die Zahlen für alle acht Spalten brav nebeneinander.

Beim VC 20 haben wir ein kleines Problem, hervorgerufen durch die kleinere Zeilenlänge, die uns eine Darstellung aller acht Zahlen nebeneinander nicht erlaubt.

Ich schlage deshalb vor, nach vier Zahlen ab Zeile 520 die Reihenfolge zu unterbrechen (Semikolon weglassen) und an den Anfang (Zeile 200) zurückzuspringen, es sei denn, die f-1-Taste wurde gedrückt (Zeile 560). Dann nämlich kommen die nächsten vier Zahlen dran — sogar in einer anderen Farbe. Eine weitere Tastenabfrage in Zeile 960, diesmal der .....Taste — ja schauen Sie halt in der ASCII-Tabelle nach, welche Taste dieses Zeichen hat (Methode des waagrechten Sprunges zwei Spalten weiter) — entscheidet zwischen Wiederholung der zweiten Gruppe oder Rücksprung auf die ersten vier Zahlen. Wie gesagt, die Verwendung der Semikolons an der richtigen Stelle ist wichtig.

Jetzt ist die Gelegenheit da zum Experimentieren:

— mehrere Tasten aus verschiedenen Spalten

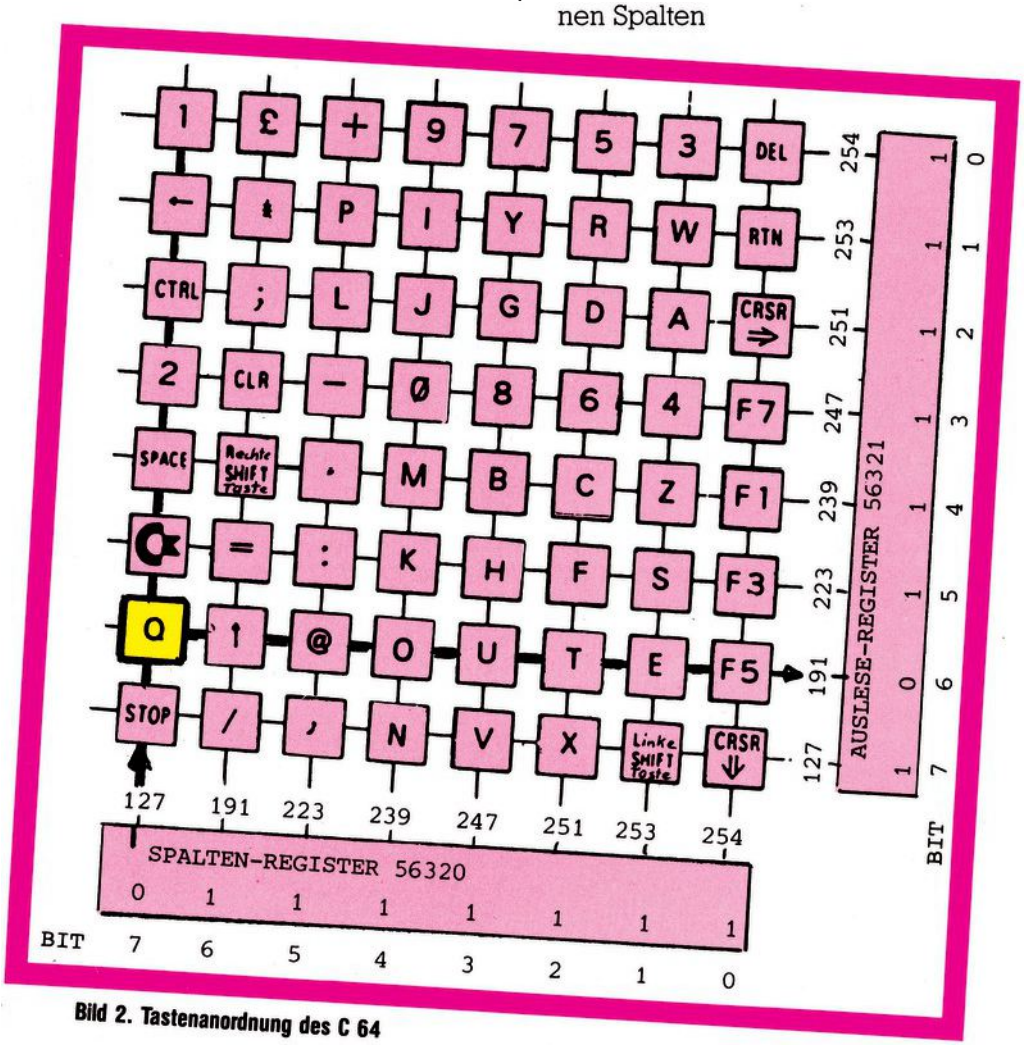

Bild 2. Tastenanordnung des C 64

— mehrere Tasten aus derselben Spalte

Ja, und irgendwann bleibt Ihnen das Programm mit BREAK IN .... stehen!!

Leider ist das schon wieder der STOP-Tasten-Effekt, den wir nicht ganz loswerden. Wenn nämlich Tasten der waagrechten Reihe 254 (beziehungsweise 127) gedrückt werden, interpretiert dies das Betriebssystem als STOP-Taste, die ja in dieser Reihe liegt, und stoppt das Programm.

Uns soll das aber nicht bedrücken, denn wir können in unserem Programm den Gebrauch dieser Tastenreihe einfach vermeiden. Für unsere Zwecke reichen alle andern Tasten allemal aus. Man muß diese Einschränkung nur berücksichtigen.

Zusammenfassend sei gesagt, daß Sie in einem Programm jetzt alle Kombinationen aller Tasten abfragen können mit

IF PEEK(7679)= ..... THEN .....

Sie müssen sich lediglich die entstehenden Dualzahlen im Register 37153 (56321) in Dezimalzahlen umrechnen oder sie vorher ausprobieren.

Ich will auch am Schluß meiner Darstellung der Gewohnheit treu bleiben und das Kochrezept in einem kleinen Gericht anwenden (Listing 2).

Ein Spiel für vier Personen

Um mir und Ihnen die Sache leichter zu machen, verwende ich möglichst viele Teile aus den vorherigen Programmen.

Spielidee:

— Jeder Spieler wählt eine von vier bestimmten Tasten.

— Auf dem Bildschirm wird ein beliebiger Buchstabe »angesagt«

— Der Bildschirm füllt sich langsam mit Zufalls-Buchstaben

— Sobald der »angesagte« Buchstabe erscheint, sollen die Spieler ihre Taste drücken

— Wer zuerst drückt, hat gewonnen!

Programmbeschreibung:

Zeile 10 bis 50

— schützt das Maschinenprogramm (wie vorher)

Zeile 100 bis 140

— liest das Kochrezept »Maschinenprogramm« ein

Zeile 215 und Zeile 530

— stellen einen besonderen Trick dar. Man nennt das eine »Flagge setzen« (set a flag). Die Abfrage der Tasten soll nämlich nur dann funktionieren, wenn ein Buchstabe (später VB genannt) mit dem anfangs »angesagten« Buchstaben BU übereinstimmt. Wenn VB = BU dann wird in eine »sichere« Speicherzelle (ich habe 830 gewählt) eine Zahl (hier 250) hineingePOKEt. Diese Zahl in 830 wird bei der Abfrage ebenfalls abgefragt. Bei Beginn des Spieles muß diese Flagge natürlich eingeholt, das heißt auf 0 gesetzt werden (Zeile 215).

Zeile 220 bis 270

— gibt Anweisungen und teilt den Spielern ihre Taste zu (von mir willkürlich ausgewählt)
Zeile 280/300
— löscht den Bildschirm und färbt alles schwarz
Zeile 310/320
— gibt Anweisungen
Zeile 330
— wählt per Zufallsgenerator (wie vorher) einen Buchstaben BU aus, zwischen 1 und 10 (A bis J in Bildschirm-Code!)
Zeile 340/350
— druckt diesen Buchstaben BU genau an das Ende des Textes der Zeile 320 (Bildschirmspeicher-Platz und Farbspeicher-Platz ausrechnen!)
Zeile 260/270 und 370/380
— schalten den Text weiter mit irgendeiner Taste

Listing 2. Ein Tastenspiel für vier Personen

```
0 REM*****************
1 REM** SPIEL FUER **
2 REM** 4 PERSONEN **
3 REM*****************
4 REM
5 REM
6 REM*****************
7 REM*** EINGABE ****
8 REM**MASCHINENCODE*
9 REM*****************
10 PRINT CHR$(147)
20 POKE 51,235
30 POKE 52,29
40 POKE 55,235
50 POKE 56,29
```

VC 20:
```
100 DATA 169,0,141,32,145,173,33,145,141,255,29,96
```

C 64:
```
100 DATA 169,0,141,0,220,173,1,220,141,255,29,96
```

```
110 FOR K=1 TO 12
120 READ X
130 POKE 7660+K,X
140 NEXT K
199 REM***************
200 REM*ANWEISUNGEN**
205 REM***************
210 PRINT CHR$(147)
215 POKE830,0
```

VC 20:
```
220 PRINT"■■SPIELER 1:(W)TASTE
230 PRINT"■■SPIELER 2:(Z)TASTE
240 PRINT"■■SPIELER 3:(DEL)TASTE
250 PRINT"■■SPIELER 4:(CRSR←)TASTE
```

C 64:
```
220 PRINT"■SPIELER 1 DRUECKT 'W'
230 PRINT"■SPIELER 2 DRUECKT 'COMMODORE'
240 PRINT"■SPIELER 3 DRUECKT 'INST/DEL'
250 PRINT"■SPIELER 4 DRUECKT 'CRSR←'
```

```
260 PRINT TAB(125)"■■SPACE■←"
270 GET A$:IF A$=""THEN 270
280 PRINT CHR$(147)
300 POKE 36879,8
310 PRINT"■■ER SIEHT ALS ERSTER
320 PRINT"■DEN BUCHSTABEN
330 BU = INT(RND(0)*10)+1
```

VC 20:
```
340 POKE 7808,BU
350 POKE 38528,7
360 PRINT"■■UND DRUECKT SEINE ■TASTE ?
```

C 64:
```
340 POKE 1224+17,BU
350 POKE 55496+17,7
360 PRINT"■■UND DRUECKT SEINE TASTE ?
```

```
370 PRINT TAB(165)"■■SPACE■←"
380 GET A$:IF A$="" THEN 380
390 PRINT CHR$(147)
400 FOR T=1 TO 800:NEXT T
```

Zeile 400
— verzögert das Erscheinen des 1. Buchstabens
Zeile 520
— wählt wieder per Zufallsgenerator einen Buchstaben, diesmal VB, aus (zwischen A und J)
Zeile 540
— wählt per Zufallsgenerator einen Platz C auf dem Bildschirm aus, auf den der Buchstabe VB gePOKEt wird

Beim VC 20 fülle ich den ganzen Bildschirm von Platz 0 bis 505. Beim C 64 wäre das der Bereich von 0 bis 1000. Damit aber den Spielern wegen der viel kleineren Buchstaben des C 64 die Augen nicht übergehen, habe ich die eigentliche Formel C = INT(RND(0)*1000) in Zeile 540 so abgeändert, daß die Buchstaben nur auf jeden 2. Platz gePOKEt werden können.
Zeile 550/560
— die Buchstaben VB werden in weiß gePOKEt
Zeile 610
— leiert die Tastenabfrage 15 mal hintereinander durch. Diese Zahl bestimmt auch die Geschwindigkeit, mit der die Buchstaben auf dem Bildschirm erscheinen. Ihre Verkleinerung erhöht die Schwierigkeit des Spieles
Zeile 620 bis 740
— fragt die in Zeile 220 bis 250 definierten Tasten und die Flagge in 830 ab. Beim VC 20 ist jede der gewählten Tasten in einer anderen Spalte, daher sind vier POKEs in 7662 notwendig. Beim C 64 liegen je zwei Tasten in einer Spalte, daher nur zwei POKEs in 7662. Da aber gleichzeitige Tastendrücke vorkommen können (in einer Spalte natürlich), fragen Zeile 660 und 710 diesen speziellen Fall ab.
Zeile 810 bis 840 beziehungsweise 850
— meldet den Sieger (Aussprung aus der Abfrage)

Zusammenfassung und Schluß

Diese »gleichzeitige« Mehrfach-Tastenabfrage ist genau genommen nicht ganz gleichzeitig.

Nichts in einem einzelnen Computer ist gleichzeitig! Alles erfolgt in einer Sequenz.

In dem Spielprogramm für vier Personen erfolgt die Tastenabfrage

Listing 1. In diesem Programm können alle Tasten abgefragt und angezeigt werden

```
490 REM***************
495 REM* BUCHSTABEN *
500 REM** WUERFELN **
510 REM***************
520 VB=INT(RND(0)*10)+1
530 IF VB=BU THEN POKE 830,250
```

VC 20:
```
540 C=INT(RND(0)*505)
550 POKE 7680+C,VB
560 POKE 38400+C,1
```

C 64:
```
540 C=INT(RND(1)*501)*2
550 POKE 1024+C,VB
560 POKE 55296+C,1
```

```
590 REM***************
595 REM** ABFRAGE ***
600 REM* DER TASTEN *
605 REM***************
610 FOR Z=1 TO 15
```

VC 20:
```
620 POKE 7662,253
630 SYS 7661
640 IF PEEK(7679)=253 AND PEEK(830)=250 THEN 810
650 POKE 7662,239
660 SYS 7661
670 IF PEEK(7679)=253 AND PEEK(830)=250 THEN 820
680 POKE 7662,254
690 SYS 7661
700 IF PEEK(7679)=127 AND PEEK(830)=250 THEN 830
710 POKE 7662,251
720 SYS 7661
730 IF PEEK(7679)=127 AND PEEK(830)=250 THEN 840
740 NEXT Z
```

C 64:
```
620 POKE 7662,127
630 SYS 7661
640 IF PEEK(7679)=253 AND PEEK(830)=250 THEN 810
650 IF PEEK(7679)=223 AND PEEK(830)=250 THEN 820
660 IF PEEK(7679)=221 AND PEEK(830)=250 THEN 850
670 POKE 7662,254
690 IF PEEK(7679)=254 AND PEEK(830)=250 THEN 830
700 IF PEEK(7679)=251 AND PEEK(830)=250 THEN 840
710 IF PEEK(7679)=250 AND PEEK(830)=250 THEN 850
720 NEXT Z
```

```
750 GOTO 500: REM (NEUER BUCHSTABE)
799 REM***************
800 REM** ERGEBNIS **
805 REM***************
810 PRINT"■■■SPIELER 1 IST SIEGER":GOTO 860
820 PRINT"■■■SPIELER 2 IST SIEGER":GOTO 860
830 PRINT"■■■SPIELER 3 IST SIEGER":GOTO860
840 PRINT"■■■SPIELER 4 IST SIEGER":GOTO 860
```

C 64:
```
850 PRINT"■■■ZWEI SPIELER WAREN GLEICHZEITIG...
```

```
860 PRINT"■■■NOCH EINMAL ?  (J/N)
870 GET A$:IF A$="J" THEN 200
880 IF A$="N" THEN END
890 GOTO 870
READY.
```

Ein Spiel für 4 Personen

```
4 REM **********************
5 REM   VIELFACHTASTEN
6 REM      ABFRAGE
7 REM **********************
10 PRINT CHR$(147)
20 POKE 51,235
30 POKE 52,29
40 POKE 55,235
50 POKE 56,29
100 DATA 169,254,141,0
101 DATA 220,173,1,220
102 DATA 141,255,29,96
105 REM --------VC20-----------
106 REM   100 DATA169,254,141,32
107 REM   101 DATA145,173,33,145
108 REM   102 DATA141,255,29,96
109 REM --------VC20-ENDE------
110 FOR K=1 TO 12
120 READ X
130 POKE 7660+K,X
140 NEXT K
200 POKE 7662,127
210 SYS 7661
220 PRINT " ■"PEEK(7679);
300 POKE 7662,191
310 SYS 7661
320 PRINT PEEK(7679);
400 POKE 7662,223
410 SYS 7661
420 PRINT PEEK(7679);
500 POKE 7662,239
510 SYS 7661
511 REM  ------VC20-------------
512 REM  520 PRINT PEEK(7679)
513 REM  550 GET A$
514 REM  560 IF A$<>"■"THEN200
515 REM  ------VC20-ENDE--------
520 PRINT PEEK(7679);
600 POKE 7662,247
610 SYS 7661
620 PRINT PEEK(7679);
700 POKE 7662,251
710 SYS 7661
720 PRINT PEEK(7679);
800 POKE 7662,253
810 SYS 7661
820 PRINT PEEK(7679);
900 POKE 7662,254
910 SYS 7661
920 PRINT PEEK(7679)
940 REM -------VC20-------------
950 REM  GET A$
960 REM  IF A$<>"■" THEN 600
970 REM ------VC20-ENDE--------
1000 GOTO 200

READY.
```

natürlich auch hintereinander, wodurch die zuerst abgefragten Tasten ihren Besitzern einen kleinen Vorteil gewähren, aber halt nur einen ganz winzigen!

Dieser Effekt wird durch das 15fache Durchlaufen der Abfrage und durch die Schnelligkeit des Maschinenprogramms gemildert. Das ganze Programm in Maschinensprache geschrieben würde natürlich durch die Geschwindigkeit auch die letzte Ungerechtigkeit ausmerzen.

So, liebe VC 20er und C 64er. Das war's.

Ich hoffe, Sie kennen jetzt die verschiedenen Codes und fühlen sich wohl bei der Tasten-Abfrage. Methoden und Kochrezepte haben Sie dazu ja jetzt genug.

Ich habe mich bemüht, Sie zum Experimentieren anzuregen, denn gerade diese spielhafte Auswirkung der Neugier unterscheidet den Amateur vom Profi und fördert das Verständnis der Zusammenhänge.

Ich habe mir am Anfang das Ziel gesetzt, allgemein verständlich und ohne Fachchinesisch zu schreiben.

Ob mir das gelungen ist, können nur Sie mir sagen. Falls etwas unklar geblieben ist und Sie Fragen haben, schicken Sie sie mir, ich werde Ihnen antworten. Und wenn es sogenannte »gute Fragen« sind, schreibe ich vielleicht darüber den nächsten Kurs.

(Dr. Helmuth Hauck/gk)

Literatur:
1. VIC REVEALED von N. Hampshire, Computabits Ltd., 1981
2. M. Bassman, S. Lederman in COMPUTE!S FIRST BOOK OF VIC COMPUTE!, Books Publ., 1982
3. VC 20 SPIELE-BUCH 1 von A. Dripke, Computer Life Verlag, 1983
4. VC-INTERN von Angerhausen und Englisch, Data Becker, 1983
5. 64-INTERN von Angerhausen et al., Data Becker, 1983
6. DAS VC-20-Buch von M. Hegenbarth, M. Schäfer, Markt & Technik Verlags, 1983
7. VIC-20-PROGRAMMERS REFERENCE GUIDE von A. Finkel et al., Howard W. Sams & Co, 1982