# Tips und Tricks für den VC 20

**Nützliche ROM- und RAM-Adressen für den VC 20 wurden schon mehrfach veröffentlicht. Dieser Artikel bringt nun Beispiele, wie diese auch sinnvoll eingesetzt werden können. Denn was nützt das beste ROM-Adressenlisting mit den tollsten Erläuterungen, wenn man keinen Anwendungszweck findet?**

Beginnen wir mit der Tonerzeugung. Mancher VC 20-Besitzer hat das möglicherweise schon so oft gelesen, daß er in der Nacht von den Adressen träumt. Tongenerator 1 bis 3: 36874 bis 36876, Generator für weißes Rauschen: 36877, Lautstärke: 36878. Doch die wenigsten wissen, daß man mit der Lautstärke auch den Klang eines Tones entscheidend beeinflussen kann, indem man ihn langsam auf- oder abschwellen läßt.

In der Praxis sieht das so aus, daß man einen bestimmten Ton wählt, dann die Lautstärke langsam aufdreht, eine Weile hält und schließlich wieder langsam abdreht. Wenn man den Ton sofort voll aufdreht und gleich danach langsam abschwellen läßt, klingt es, als ob man auf einem Instrument einen Ton anschlägt, der dann langsam ausklingt. Listing 1 zeigt ein Beispielprogramm mit einer bekannten Melodie.

Die bekannteste Nutzungsmöglichkeit der Lautstärke dürfte wohl die Explosion sein. Man wählt ein tiefes Rauschen und läßt es langsam leiser werden. Das so entstehende Explosionsgeräusch kann sehr gut für Spiele verwendet werden. Das kleine Programm in Listing 2 erzeugt zum Beispiel bei jedem Tastendruck ein anderes Explosionsgeräusch.

## Farbige Anleitung

Da wir gerade von Spielen sprechen: Zu einem Spiel gehört auch eine Spielanleitung (auf dem Bildschirm). Und wenn man da einmal etwas ganz besonderes haben will, gibt es die Möglichkeit, innerhalb eines Buchstabens mehrere Farben zu verwenden. Wie geht das? In der Speicheradresse 646 steht die momentan benutzte Farbe. Zulässig sind Werte von 0 bis 7. Addiert man zu einem dieser Werte 8, so wird auf Vierfarbenmodus umgeschaltet und es entstehen bunte, verzerrte Buchstaben.

Diese lassen sich zwar schlechter lesen, ergeben aber zum Beispiel bei Spielprogrammen einen interessanten Effekt (für Textverarbeitung sind sie weniger geeignet).

## Save-Schutz

Die Buchstaben setzen sich aus vier Farben zusammen: Aus Hintergrundfarbe, Rahmenfarbe, Hilfsfarbe und der gewählten Zeichenfarbe aus der Adresse 646. Die Hilfsfarbe wird mit den Bits 4 bis 7 aus 36878 definiert. Es ist unbedingt notwendig, daß die Hintergrundfarbe nur einmal dabei vorkommt. Ist nämlich eine dieser Farben gleich der Hintergrundfarbe, so entstehen Lücken in den Buchstaben, und der Text wird so stark verstümmelt, daß er sich nicht mehr lesen läßt. Das Rücksetzen erfolgt einfach durch Umschalten der Zeichenfarbe mit der CTRL-Taste.

Doch nun zu einem anderen Thema: Im VC 20 gibt es einen Vektor, der jedesmal bei der Ausführung des SAVE-Befehls benutzt wird. Er steht in 818/819 und zeigt auf die SAVE-Routine des VC 20. Durch Ändern dieses Vektors kann man die Ausführung des SAVE-Befehls und damit eventuell das unerlaubte Kopieren eines Programms, verhindern. Zum Beispiel kann dieser Vektor durch POKE 818,116:POKE 819,196 auf $C474 gesetzt werden. Nach Eingabe des SAVE-Befehls erscheint dann immer direkt READY, ohne daß etwas passiert. Eine andere Möglichkeit ist, eine Fehlermeldung auszugeben. Dies geschieht durch POKE 818,53:POKE 819,196, wodurch der Zeiger auf $C435 gesetzt wird. Nach der Eingabe von SAVE erscheint »OUT OF MEMORY ERROR«.

Doch die wirksamste Möglichkeit wird wohl die sein, das Programm durch den SAVE-Befehl zu löschen, indem man den SAVE-Vektor auf $FD22 setzt. Dies geschieht durch POKE 818,34: POKE 819,253. Nach Eingabe des SAVE-Befehls erscheint die Einschaltmeldung, und das Programm ist gelöscht.

Leider ist es mit diesen zwei POKE-Befehlen nicht getan. Denn durch gleichzeitiges Drücken der STOP- und RESTORE-Taste werden sie sofort wieder rückgängig gemacht. Es muß also zusätzlich noch die RESTORE-Taste abgestellt werden. Das geht einfach mit POKE 37150,2. Danach hat die RESTORE-Taste keine Funktion mehr. Wenn nötig, kann sie durch POKE 37150,130 wieder eingeschaltet werden. Dieser POKE-Befehl schreibt in das Interrupt-Enable-Register des VIA 6522 # 1 des VC 20, der für den NMI zuständig ist.

Doch nun wollen wir keine Tasten außer Betrieb setzen, sondern welche abfragen. Oft steht man vor dem Problem, mehrere Funktionen gleichzeitig zu steuern. Leider kann man in Basic immer nur eine Taste auf einmal abfragen, weitere gedrückte Tasten werden normalerweise nicht erkannt.

## Zehn Tasten gleichzeitig abfragen

Aber glücklicherweise gibt es beim VC 20 eine Möglichkeit, sieben Tasten auf einmal abzufragen. Die Bits 1 bis 7 der Speicherstelle 145 entsprechen folgenden Tasten: Der linken Shift-Taste, X, V, N, ≠, ? und Cursor up/down. Die Taste ist gedrückt, wenn das entsprechende Bit gelöscht ist.

Nun läßt sich das ganze mit der Speicherzelle 653 noch erweitern. Bit 0 entspricht den Shift-Tasten, Bit 1 der Commodore-Taste und Bit 2 der CTRL-Taste. Im Gegensatz zu den sieben anderen Tasten sind hier jedoch die entsprechenden Bits bei Tastendruck gesetzt.

Der Computer unterscheidet übrigens zwischen der rechten und der linken Shift-Taste. Mit der Abfrage PEEK (145) AND 2 OR PEEK (653) AND 1 kann man zwischen beiden Shift-Tasten unterscheiden. Sie ergibt normalerweise 2, beim Drücken der linken Shift-Taste 1 und beim Drücken der rechten Shift-Taste 3. Shift lock entspricht der linken Shift-Taste.

(Thomas Gruber/ev)

```
1 REM TON-DEMO
2 REM ========
3 REM
4 REM
5 DIM T(5)
10 FOR X=1 TO 5:READ T(X):NEXT
15 READ T,D:POKE 36875,T(T):POKE 36876,T(T)
20 IF D=-1 THEN END
25 FOR X=15 TO 0 STEP -1
30 POKE 36878,X
35 FOR Y=1 TO 3*D:NEXT Y
40 NEXT X
45 GOTO 15
50 REM
55 REM
60 DATA 195,201,207,209,215
65 DATA 5,1,3,1,3,6,4,1,2,1,2,6
70 DATA 1,1,2,1,3,1,4,1,5,1,5,1,5,6
75 DATA 5,1,3,1,3,6,4,1,2,1,2,6
80 DATA 1,1,3,1,5,1,5,1,1,6,0,1
85 DATA 2,1,2,1,2,1,2,1,2,1,3,1,4,6
90 DATA 3,1,3,1,3,1,3,1,3,1,4,1,5,6
95 DATA 5,1,3,1,3,6,4,1,2,1,2,6
100 DATA 1,1,3,1,5,1,5,1,1,6
105 DATA 0,-1

READY.
```

**Listing 1. Ton-Demo**

```
1 REM EXPLOSIONEN
2 REM ===========
3 REM
4 REM
10 G=36877:L=36878:X=RND(1)*100+130
15 FOR I=15 TO 0 STEP -1
20 POKE G,X+I:POKE L,I
25 FOR D=1 TO 240-X:NEXT D
30 NEXT I
35 POKE G,0:POKE L,0
40 GET A$:IF A$="" THEN 40
45 RUN

READY.
```

**Listing 2. Explosionen**

# Veränderungen am 64-Basic

Eine der merkwürdigsten Effekte am Commodore 64 ist, wenn man unmittelbar nach dem Einschalten den Speicher abfragt, also eingibt

PRINT FRE(0)

Man erhält nämlich eine negative Zahl als Antwort. Man kann dies korrigieren, indem man bei einer negativen Antwort den Wert 65536 addiert. Im folgenden soll diese Merkwürdigkeit etwas näher beleuchtet und ein Weg zur Korrektur dargestellt werden.

Die Funktion FRE(0) liefert den noch verfügbaren, also freien Speicher. Dazu wird der Speicher aufgeräumt (Garbage-Collection), das heißt, die nicht mehr benötigten Strings werden entfernt und der gesamte String-Bereich folglich geordnet. Der freie Bereich ergibt sich dann als: Untere Grenze der Strings minus obere Grenze der Arrays.

Der Inhalt der folgenden Adressen muß also voneinander abgezogen werden

33/34 Untergrenze Strings
31/32 Ende Arrays
also
Inhalt von 33 minus Inhalt von 31
Ergebnis nach 63
Inhalt von 34 minus Inhalt von 32
Ergebnis nach 62.

Das Ergebnis ist also vom Typ INTEGER. Allerdings arbeitet der Interpreter intern nicht mit Integer-Zahlen, sondern mit REAL-Zahlen (Gleitkommazahlen), das Ergebnis muß also gewandelt werden. Bei der Wandlung wird ein Bit (von 16, die für die Darstellung der Integer benötigt werden) als Vorzeichen betrachtet. Ist der Inhalt von Zelle 62 größer oder gleich $80 = 128, so wird die Zahl bei der Wandlung als negative Zahl angesehen. Der Inhalt von 62 wird dann größer oder gleich $80 sein, wenn der freie Bereich größer 32767 Byte ist.

Aus diesen Gründen taucht gelegentlich eine negative Zahl als freier Speicherplatz auf. Wie kann man dies ändern?

Im Interpreter gibt es vier Wandlungsroutinen.

1) Wandle REAL-Zahl nach INTEGER.
Dabei wird die Integer-Zahl mit 15 Bit + 1 Bit für das Vorzeichen verschlüsselt, darstellbar sind also als Integer ganze Zahlen von —32767 bis +32767.

2) Wandle INTEGER-Zahl nach REAL
Dies ist die Umkehrung von 1).

3) Wandle REAL nach INTEGER.
Dabei wird eine Integer-Zahl mit 16 Bit (kein Vorzeichen!) verschlüsselt, darstellbar sind also Zahlen von 0 bis 65535. Diese Routine wird für die interne Darstellung der Zeilennummern des Basic-Programms benötigt (Wandlung von ASCII nach REAL, von dort nach INTEGER).

4) Wandle INTEGER nach REAL.
Dies ist die Umkehrung von 3). Diese Routine wird zum Beispiel für die Ausgabe der Fehlermeldungen benötigt (...ERROR in (Zeile)).

Bei der FRE-Funktion müßte also nach der Berechnung des freien Platzes die Routine 4) aufgerufen werden. Aufgerufen wird aber die Routine 2). Im Interpreter steht

JMP $BC44 (an der Adresse $B39B).

Dort müßte aber eigentlich stehen

SEC
JMP $BC49

Die Frage ist jetzt nur noch, wie kann man dies ersetzen?

Dabei tauchen folgende Probleme auf

1) Der Interpreter wird im ROM gespeichert.
2) Man muß 3 Byte durch 4 Byte ersetzen.

Im Commodore 64 kann man bestimmte Bereiche, die durch ROM belegt sind, ausblenden und dafür RAM einblenden. Das folgende Programm kopiert den Interpreter vom ROM ins RAM

```
10 FOR I=10*16*256 TO 12*16*256-1
20 POKE I,PEEK(I)
30 NEXT I
```

Dabei wird ausgenutzt, daß der Computer zwar über PEEK(I) aus dem ROM liest, aber mit dem Befehl POKE ins RAM schreibt (dies ist intern gelöst).

Man müßte jetzt nur noch das RAM einblenden, vorher sollen aber noch die notwendigen Veränderungen vorgenommen werden.

So einfach kann man natürlich nicht drei Byte durch vier Byte ersetzen. Die Lösung besteht darin, die Routine in einen freien Bereich umzuleiten und dort die notwendigen Befehle einzufügen (sogenannte Patches). Also anstelle von

JMP $BC44 an der Adresse $B39B
sollte stehen
JMP $C000 in den freien Bereich.

Dies erreicht man durch

```
40 POKE 11*16*256+3*256+9*16+12 , 0
(ersetzt $44 = 44*16+4)
50 POKE 11*16*256+3*256+9*16+13 , 12*16
(ersetzt $BC = 11*16+12).
```

An der Stelle $C000 ff. solte dann stehen

SEC
JMP $BC49

Also wird wieder gePOKET

```
60 POKE 12*16*256     , 3*16+8  (=SEC)
70 POKE 12*16*256+1   , 4*16*12 (=JMP)
80 POKE 12*16*256+2   , 4*16+9  (=$49 Teil der Adresse)
90 POKE 12*16*256+3   , 11*16+12 (=$BC Teil der Adresse).
```

Jetzt muß man nur noch das RAM einschalten

```
100 POKE 1 , PEEK(1) AND 254 (Einschalten)
(  POKE 1 , PEEK(1) OR 1     (Ausschalten) ).
```

Im Speicherplatz 1 wird im Bit 0 (= Wertigkeit 1) hinterlegt, wo die Speicherplätze von $A000 bis $BFFF sind.

Und nun, lassen Sie Ihren Computer doch mal den Befehl

PRINT FRE(0)

ausführen. Wenn Sie keinen Fehler gemacht haben, wird er Ihnen diese Frage ab sofort korrekt beantworten.

Zusammenfassend kann man sagen, daß der beschriebene Effekt unschön und der Aufwand zur Änderung gering gewesen wäre. Der hier beschriebene Aufwand wurde dadurch groß, daß die Änderung nachträglich vorgenommen werden mußte.

(Dr. August Quint)