Tips & Tricks zum C 16

Auch ein sehr guter Basic-Dialekt, wie der des C 16, kann durch sinnvolle POKEs noch ergänzt werden. Wir bringen deshalb unter anderem eine Liste mit den nützlichsten Adressen.

Für die Neulinge unter den C 16/C 116-Fans noch eine wichtige Mitteilung: Wenn in dieser Rubrik vom C 16 gesprochen wird, so ist damit der C 116 ebenso gemeint. Diese beiden Computer unterscheiden sich nur durch die geänderte Tastatur und das Gehäuse. Ihr »Innenleben« ist hundertprozentig identisch.

Und noch ein heißer Tip: Ende Juli '86 erscheint unser Soderheft 8/86, Thema: »C 16«! Eine geballte Ladung von Grundlagen, Tips & Tricks und vielen Listings zum Abtippen erwartet Sie.

Vorsicht bei PRINT

Es gilt allgemein als bekannt, daß das Basic des C 64 aufwärtskompatibel zu dem des C 16 ist. Ich habe aber eine Differenz festgestellt, die sich auf die Bildschirmbehandlung bezieht. Als Beispiel soll folgendes kleines Programm dienen:

```
10 print" {CLR, 40 SPACE}"
20 print"Da bin ich!"
```

Auf dem C 64 befindet sich also der Cursor nach Ausführung der ersten Programmzeile in der zweiten, beim C 16 jedoch in der dritten Bildschirmzeile, was bei Verwendung eines Basic-Programms für beide Computer zur Zerstörung des Bildschirmaufbaus führen kann.

Beim C 64 wird demnach der Cursor beim Erreichen der letzten Bildschirmspalte schon in die nächste Zeile positioniert, beim C 16 bleibt der Cursor jedoch noch in der gleichen Zeile.

Dies läßt sich durch folgende kleine Programme beweisen:

```
a) C 64:   10  print" {CLR, 40 SPACE}"
           20  x=peek(211):y=peek(214)
           30  printx,y
b) C 16:   10  print" {CLR, 40 SPACE}"
           20  x=peek(202):y=peek(205)
           30  printx,y
```

In den Adressen 211 beziehungsweise 202 steht die Spaltenposition des Cursors (0-39), in 214 beziehungsweise 205 die Zeilenposition (0-24).

Ergebnis:
a) C 64: x=0 y=2
b) C 16: x=0 y=1

(Hermann-Josef Rottkemper/tr)

POKEs, PEEKs und SYS-Befehle

205,0-24	bewirkt Cursor-Down, je nach Wert (0-24)
775,128	bewirkt Listschutz, es werden nur die Zeilennummern ausgegeben. Normalwert ist 139.
775,252	bewirkt einen Reset bei einem List-Versuch. Normalwert ist 139.
19,0	Input-Fragezeichen aus
19,1	Input-Fragezeichen ein
194,1	Reverse on
194,0	Reverse off
239,0	löscht Tastaturpuffer. Nützlich zum Beispiel bei GET oder GETKEY.
1525,0:1526,0:1527,0	Bewirkt Saveschutz. (Der C 16 simuliert den Save- und Verify-Vorgang und alles sieht aus, als wäre es in Ordnung...) Normalwerte: alle 255
65286,peek(65286)and239	Schaltet Bildschirm ab. Programme werden jetzt bis zu 6 Prozent schneller bearbeitet. Nützlich bei Sortierprogrammen oder wenn in einer Dateiverwaltung ein Datensatz gesucht wird.
65286,peek(65286)or16	Schaltet Bildschirm wieder ein
print peek(56)x256+peek(55)—(peek(44)x256+peek(43))	Ergibt freien Speicherplatz, der zur Verfügung steht. (Es wird jedoch keine printfre(0)-Abfrage simuliert.)
print peek(56)x256+peek(55)	Ergibt Endadresse des Basic-Speichers + 1
print peek(44)x256+peek(43)	Ergibt Startadresse des Basic-Speichers
SYS 65526 oder SYS 65529	bewirkt Reset
SYS 32768 (oder SYS 128x256)	bewirkt ebenfalls einen Reset. Es wird jedoch nur ein Basic-Programm gelöscht. Sind die Farben (Hintergrund, Vordergrund, Zeichenfarbe) geändert, so werden sie beibehalten. Die F-Tasten behalten ebenfalls ihre Belegung bei.
SYS 65499	TI$ wird auf "000000" gestellt.

(Zoltan Djapjas/tr)

Tips & Tricks-Mischmasch

So kann ein Programm von einem Programm aus (von Datasette) nachgeladen und gestartet werden:

```
a$="load"+chr$(13)+"run"+chr$(13)
for a=1 to len(a$): poke 1318+a,asc(mid$(a$,a,1)): next: poke
239,a: end
```

Dieses Programm nutzt den »programmierbaren Direktmodus«: Die Zeichen des Strings a$ werden in den Tastaturpuffer geschrieben. Dann wird in die Adresse 239 die Anzahl der Zeichen geschrieben (Adresse 239 enthält Anzahl der Zeichen im Tastaturpuffer).

Durch die END-Anweisung wird das Programm beendet und das Basic sucht nun in der Adresse 239, ob sich Zeichen im Tastaturpuffer befinden. Dort findet es die Länge des Strings a$, gibt dementsprechend viele Zeichen auf dem Bildschirm aus und führt diese direkt aus. Da es sich hier um die Befehle »LOAD« (+ RETURN) und »RUN« (+ RETURN) handelt, lädt das Basic nun das nächste Programm nach und startet es. Das vorher im Speicher befindliche Programm wird dabei natürlich gelöscht.

Mit dieser Methode können sich Basic-Programme auch selber verändern. Man kann etwas mit PRINT auf den Bildschirm schreiben (zum Beispiel eine neue Basic-Zeile) und in den Tastaturpuffer dann chr$(13) (=RETURN), je nach erforderlicher Anzahl schreiben.

Ich habe immer die Gewohnheit, die Funktionstasten des C 16 umzubelegen. Da ich nicht immer »KEY 1,"..."« eingeben möchte, habe ich nach einer anderen Methode gesucht:
1. Man könnte sich ein Basic-Programm schreiben, das die Funktionstasten selbständig belegt und sich dann löscht. Ein Nachteil wäre dann aber: Man kann dies nicht mehr tun, wenn sich schon ein Basic-Programm im Speicher befindet. Ein anderer Nachteil wäre die Länge des Programms. Deshalb mache ich das mit einer viel eleganteren Methode:
2. Man definiert die Funktionstasten einfach mit den KEY-

Befehlen um. Vorher geht man in den Monitor und gibt folgendes ein:
F 055D 05E6 00

Der Speicherbereich für die F-Tasten wäre somit gelöscht. Dann mit X den Monitor verlassen. Jetzt die F-Tasten umbelegen. Dann wieder in den Monitor. Geben Sie nun ein:
S "KEYDEF",01 (beziehungsweise 08 für Floppy), 055d, 05E6
...und drücken die Return-Taste.

Der Speicherbereich für die Funktionstasten wird nun gespeichert.

Nun kann man jederzeit die Funktionstasten schnell und bequem neu belegen. Dazu gibt man einfach ein:
load" ",1,1 (bei Disk: load"keydef",8,1)

(Zoltan Djapjas/tr)

Systemabsturz

Haben auch Sie Programme, die an immer anderen Stellen urplötzlich abstürzen, vielleicht sogar nachdem sie etliche Minuten vollkommen normal gelaufen sind?

Dann sollten Sie einmal nachsehen, ob Sie irgendwo in dem betreffenden Programm »DS« oder »DS$« zur Abfrage des Floppy-Fehlerkanals verwenden.

Wenn ja, haben Sie gute Chancen, das Programm jetzt doch noch zum Laufen zu bringen.

Fragen Sie doch den Fehlerkanal einmal so ab, wie es ein jeder C 64-Benutzer machen würde:

```
OPEN 15,8,15
INPUT #15,X,X$,T,S
CLOSE 15
```

Wenn Sie alle Abfragen im Programm dementsprechend ändern (eventuell ohne OPEN beziehungsweise CLOSE), würde es mich nicht wundern, wenn Ihr Programm auf einmal fehlerfrei läuft.

Bei Verwendung von »DS« beziehungsweise »DS$« gerät nämlich anscheinend die Speicherverwaltung durcheinander.

Dies kann sich sehr verschieden bemerkbar machen.
1. Unerklärlicher Absturz an immer anderen Programmstellen
2. In irgendwelchen Programmzeilen steht urplötzlich Schrott
3. Ein »OUT OF MEMORY ERROR IN LOOP«

Der Fehler läßt sich recht einfach demonstrieren.

```
10 PRINT DS$,FRE(0)
RUN
```

Dies löst mit nahezu 100 Prozent Sicherheit einen »OUT OF MEMORY ERROR IN LOOP« aus.

Fehlerfrei funktioniert dagegen die folgende Version:

```
10  OPEN 15,8,15:INPUT #15,DE,DE$,T,S
20  PRINT DE;DE$;T;S;FRE(0)
30  CLOSE 15
```

Fragt man FRE(0) nicht ab, so tritt in beiden Fällen kein Fehler auf.

Bei Verwendung von DS beziehungsweise DS$ kann man aber zumindest bei umfangreicheren Programmen sicher sein, daß es irgendwann abstürzt.

Ich kann nur empfehlen, DS und DS$ nicht zu verwenden.

(B. Kardel/tr)

Anmerkung der Redaktion:

Der beschriebene Fehler trat bei unserem C 16 nicht auf. Aus Leserreaktionen wissen wir jedoch, daß einige Computer durchaus dieses Verhalten zeigen können. Da taucht natürlich die Frage auf, ob es unterschiedliche Versionen des C 16/C 116 gibt. Es besteht auch die Möglichkeit, daß der Fehler durch die verwendete Floppy verursacht wird. Sollte Ihr Computer das gleiche Verhalten zeigen wie der des Autors, so schreiben Sie uns bitte mit Angabe der Seriennummer Ihres C 16/C 116 und Ihrer Floppy. (tr)