```
7250 : 69 16 ca d0 fb 8d 1a 5f  e4
7258 : d8 68 29 0f c9 0a 90 02  c4
7260 : 69 05 f8 6d 1a 5f d8 48  c8
7268 : 4a 4a 4a 4a 09 30 c9 30  4d
7270 : d0 02 a9 20 8d c4 5f 68  fd
7278 : 29 0f 09 30 8d c5 5f a9  49
7280 : 00 8d c6 5f a9 c0 a2 5f  ce
7288 : 4c d1 70 08 8e c1 5f 8c  67
7290 : c2 5f a2 01 8e c0 5f e8  09
7298 : 8e c3 5f 28 f0 04 a0 00  96
72a0 : f0 02 a0 04 a2 00 8e 19  d1
72a8 : 5f 98 aa bd 87 6b ae 19  76
72b0 : 5f 9d c4 5f c8 e8 e0 04  5a
72b8 : d0 ec a9 00 8d c8 5f a9  59
72c0 : c0 a2 5f 4c d1 70 4a 90  1e
72c8 : 04 a2 2e d0 02 a2 00 0a  0c
72d0 : 18 69 10 a8 60 20 d8 72  05
72d8 : a0 80 a2 12 98 20 ce 62  2e
72e0 : a2 1f a9 00 20 ce 62 a2  c3
72e8 : 1e 20 ce 62 88 10 eb a2  14
72f0 : 19 a9 80 0d 30 5f 20 ce  bc
72f8 : 62 60 a9 05 a2 73 20 d1  7f
7300 : 70 20 80 69 60 01 1f 0b  6e
7308 : 20 20 20 20 20 20 20 20  08
7310 : 20 20 20 20 20 20 20 20  10
7318 : 20 20 01 1f 0c 20 20 52  54
7320 : 55 4e 44 45 20 20 42 45  ed
7328 : 45 4e 44 45 54 20 20 01  17
7330 : 1f 0d 20 20 20 20 20 20  a6
7338 : 20 20 20 20 20 20 20 20  38
7340 : 20 20 20 20 00 01 16 0c  f5
7348 : 02 47 41 4d 45 20 4f 56  27
7350 : 45 52 00 ad 3d 5f 18 69  76
7358 : 07 6d 12 d0 29 0f 8d 3d  70
7360 : 5f aa bd 74 73 f0 ec 8a  9a
7368 : 0a a8 b9 85 73 aa b9 84  62
7370 : 73 4c 40 70 01 01 01 01  46
7378 : 01 01 01 01 01 01 00 00  72
7380 : 00 00 00 00 56 69 a4 73  ab
7388 : e5 73 58 74 99 74 b2 74  bc
7390 : fd 74 20 75 75 75 ed 75  24
7398 : 00 00 00 00 00 00 00 00  99
73a0 : 00 00 00 00 b9 00 28 f0  bf
73a8 : 00 b9 00 28 1e 03 a9 01  2d
73b0 : 28 1e 03 d7 00 46 f0 00  99
73b8 : b9 00 28 5a 01 d7 00 46  22
73c0 : 5a 01 c7 01 46 5a 01 b9  5c
73c8 : 00 82 1e 03 d7 00 a0 f0  d3
73d0 : 00 a9 01 28 5a 01 a9 01  41
73d8 : 82 1e 03 b9 00 82 f0 00  39
73e0 : ff ff ff ff ff c8 00 46  b1
73e8 : 32 00 c8 00 46 14 02 b4  c3
73f0 : 00 5a 19 01 b4 00 73 14  c5
73f8 : 03 c8 00 87 32 00 0e 01  ae
7400 : 73 14 02 fa 00 5f 14 03  af
7408 : c8 00 5f 32 00 c8 00 5f  f3
7410 : 14 02 2c 01 46 2d 01 2c  7b
7418 : 01 73 4b 00 5e 01 5a 2d  57
7420 : 01 90 01 46 14 02 a4 01  58
7428 : 78 28 00 c2 01 6e 0a 03  be
7430 : ae 01 6e 0a 02 ae 01 6e  b2
7438 : 14 00 ae 01 6e 14 00 a4  e9
7440 : 01 78 0f 03 b3 01 87 14  2b
7448 : 00 e0 01 6e 19 01 ea 01  0e
7450 : 6e 0a 02 ff ff ff ff ff  44
7458 : 40 01 64 3c 03 40 01 64  b8
7460 : 3c 02 04 01 a0 78 00 46  19
7468 : 00 14 78 00 46 00 14 3c  be
7470 : 01 50 00 5a 64 00 be 00  26
7478 : 14 3c 01 c2 01 14 78 00  d6
7480 : c2 01 14 3c 01 3a 02 14  62
7488 : 3c 01 3a 02 14 3c 01 cc  d4
7490 : 01 5a 64 00 ff ff ff ff  d7
7498 : ff 25 01 14 a0 02 5a 01  72
74a0 : 14 a0 03 25 01 64 50 02  e2
74a8 : 5a 01 64 50 03 ff ff ff  d6
74b0 : ff ff 00 00 62 ff 00 ff  d5
74b8 : 00 62 37 00 00 00 66 ff  51
74c0 : 00 ff 00 66 37 00 4a 01  2b
74c8 : 62 ff 00 49 02 62 37 00  63
74d0 : 4a 01 66 ff 00 49 02 66  53
74d8 : 37 00 18 01 5a 50 00 18  8e
74e0 : 01 6e 50 00 50 00 0a 3c  d2
74e8 : 02 30 02 0a 3c 03 14 00  f0
74f0 : 82 3c 03 6c 02 82 3c 02  08
74f8 : ff ff ff ff ff ff 02 00  32  6b
7500 : c6 00 00 00 96 c8 00 c8  08
7508 : 00 32 64 01 b7 01 32 c6  34
7510 : 00 b7 01 96 c8 00 b7 01  6c
7518 : 32 64 01 ff ff ff ff ff  bd
7520 : 36 01 14 a0 01 4a 01 14  7f
7528 : a0 01 36 01 14 14 00 36  45
7530 : 01 b4 14 00 18 01 14 14  93
7538 : 00 18 01 14 50 01 54 01  68
7540 : 14 14 00 54 01 14 50 01  dd
7548 : 2c 01 14 50 01 68 01 14  83
7550 : 50 01 68 01 64 46 03 54  88
7558 : 01 64 50 03 18 01 64 46  a8
7560 : 02 2c 01 64 50 02 d2 00  a6
7568 : aa 0a 03 ae 01 aa 0a 02  40
7570 : ff ff ff ff ff c8 00 28  05
7578 : 73 00 45 01 28 73 00 c8  0d
7580 : 00 a0 73 00 45 01 a0 73  73
7588 : 00 c8 00 00 a0 01 b8 01  e3
7590 : 28 a0 01 00 00 6e a0 00  3f
7598 : e0 01 5a a0 00 a0 00 14  d1
75a0 : 5a 01 e0 01 5a 5a 01 14  78
75a8 : 00 14 8c 00 e0 01 b4 8c  d7
75b0 : 00 00 00 28 8c 00 14 00  cf
75b8 : 3c 8c 00 00 00 50 8c 00  ef
75c0 : 00 00 50 8c 00 f4 01 a0  53
75c8 : 8c 00 e0 01 8c 8c 00 f4  c4
75d0 : 01 78 8c 00 a0 00 6e 3c  6d
75d8 : 02 c8 00 64 50 02 08 02  04
75e0 : 14 50 02 1c 02 1e 3c 02  27
75e8 : ff ff ff ff ff 28 00 5a  dd
75f0 : ff 00 27 01 5a ff 00 26  cb
75f8 : 02 5a 33 00 28 00 6e ff  30
7600 : 00 27 01 6e ff 00 26 02  3f
7608 : 6e 33 00 28 00 14 ff 00  b6
7610 : 27 01 14 ff 00 26 02 14  1e
7618 : 32 00 28 00 b4 ff 00 27  ee
7620 : 01 b4 ff 00 26 02 b4 33  27
7628 : 00 28 00 6e 46 01 28 00  17
7630 : 14 46 01 58 02 14 46 01  8f
7638 : 58 02 6e 46 01 ff ff ff  06
7640 : ff ff 00 00 00 00 00 00  3f
```

Listing 1. »Vectors« (Schluß)

Tips & Tricks für Einsteiger

Diesmal zeigen wir Ihnen, wie man Programme nachlädt, das C 64-Basic schneller macht, Listings einfärbt und außerdem natürlich viele weitere nützliche Programmiertips zum C 64.

Es ist allgemein bekannt, daß das C 64-Basic nicht gerade das schnellste ist. Durch einen kleinen Trick läßt es sich aber um zirka 6 Prozent beschleunigen. Durch »POKE 53265,PEEK(53265) AND 239« wird der Bildschirm und damit auch der Video-Prozessor abgeschaltet. Dieser hat nämlich die unangenehme Eigenschaft, bei jedem Speicherzugriff (»welche Zeichen müssen nun auf dem Bildschirm dargestellt werden?«) den Prozessor kurzerhand anzuhalten. Mit dem oben genannten POKE wird nun der Video-Prozessor an dieser unliebsamen Angewohnheit gehindert. Ab jetzt werden PRINT-Anweisungen zwar ausgeführt, sind aber auf dem Monitor oder Fernseher nicht mehr sichtbar. Erst durch »POKE 53265, PEEK(53265) OR 16« ist der Bildschirminhalt wieder sichtbar. Sinnvoll ist dieses Verfahren zum Beispiel bei komplizierten mathematischen Berechnungen. Achtung: Wenn das Programm mit einer Fehlermeldung aussteigt, ist diese natürlich auch nicht mehr sichtbar. Man sollte also während der Aus-Phase zum Beispiel alle fünf Sekunden einen Signalton geben oder die Bildschirmfarbe (»POKE 53280,Farbe«) ändern. So hat man immer die Gewähr, daß das Programm noch läuft.

(Michael Rauh/tr)

Farbenspiel

Ich habe eine kleine Routine geschrieben, die sich sehr gut in eigene Programme einbauen läßt. Diese Routine läßt den Rahmen in allen Farben aufblinken:

```
10  S = 49152 : REM Das ist die Startadresse
20  FOR A = S TO S + 10 : READ X : POKE A,K : NEXT
30  DATA 238,32,208,173,141,2,201,1,208,246,96
```

Mit der <CTRL>-Taste wird die Routine abgebrochen und im Basic-Programm fortgefahren. Zum Basic-Lader: Mit der Variable S kann man den Start der Routine verändern, Aufruf geschieht über SYS S. Wenn man den zweiten DATA-Wert (32) in 33 ändert, blinkt der Bildschirm. Durch Änderung der 1 (viertletzte Zahl) in 2 oder 4 wird eine andere Taste zum Abbruch ausgewählt.

(Stefan Pohl/tr)

Der Mini-Autostart

Wer zu faul ist, jedesmal »RUN« einzutippen, wenn ein Programm von der Floppy geladen wurde, kann jetzt aufatmen: Ein kleiner Trick macht dies automatisch.

Wie Sie vielleicht wissen, bewirkt ein <SHIFT-RUN/STOP>-Tastendruck ein Laden des nächsten Programms von Datasette mit automatischem RUN danach. Diesen Umstand können sich die Floppy-Besitzer zunutze machen.

Wenn Sie ein Programm laden möchten, tippen Sie ganz normal »LOAD"Programmname",8« ein und setzen dahinter noch einen Doppelpunkt. Drücken Sie nun <SHIFT-RUN/STOP>. Auf dem Bildschirm erscheint hinter dem Doppelpunkt noch ein LOAD-Befehl (dieser wird vom C 64 jedoch ignoriert). Wenn das Programm fertig geladen ist, führt der Computer ein RUN aus.

(Bernd Roggendorf/tr)

Schablonen-Trick

Jeder, der einen Drucker besitzt, kennt folgendes Problem: Ein Druckprogramm für zum Beispiel ein Formular oder für Etiketten soll geschrieben werden. Zuweilen ist es recht mühsam, die richtige Druckposition zu finden. Mit folgendem Trick geht es etwas leichter:

Auf einem Bogen Transparent-Papier druckt man zunächst eine Seite voll Zeichen oder Ziffern. Anschließend legt man diese »Schablone« über das zu bearbeitende Formular und markiert sich die relevanten Ausschnitte, also die eigentlichen Druckpositionen. Durch Abzählen kann man nun auf der Schablone relativ leicht die Druckspalte und -zeile herausfinden. Kopiert man sich diese Schablone auf Overhead-Folie (im Schreibwarengeschäft erhältlich), so läßt sich diese, mit Hilfe eines wasserlöslichen Stiftes, beliebig oft wieder verwenden. (Norbert J. Peter/tr)

SAVE — mal etwas anders

Vor einigen Monaten las ich in einer Computerzeitschrift den verzweifelten Brief eines Lesers, der aus Basic-Programmen unbedingt gewisse Teile herausspeichern wollte. Nach einigen Stunden Arbeit entstand das Programm CLIP (Listing 1), das diese Aufgabe löst.

Das Programm »CLIP« mit LOAD"CLIP",8,1 laden und danach »NEW« eingeben. Von nun an können Teile aus Basic-Programmen auf Diskette herausgespeichert werden. Es gibt vier Varianten, um den Befehl anzuwenden:

1. SYS 828,»NAME«,AZ
2. SYS 828,»NAME«,AZ –
3. SYS 828,»NAME«,AZ – EZ
4. SYS 828,»NAME«, – EZ

AZ = Anfangszeile, EZ = Endzeile

Die Parameterverarbeitung entspricht dem List-Befehl. Der Programmteil wird unter »NAME« auf Diskette gespeichert. Dieser Befehl erspart das zeitaufwendige und unkomfortable Löschen von Programmzeilen, um am Ende dieser Prozedur nur die gewünschten Zeilen zu erhalten, die man speichern will.

Zur Funktionsweise wäre lediglich zu sagen, daß aufgrund der angegebenen Zeilennummern die Adressen der Zeilen im Speicher berechnet werden und danach dieser Speicherbereich auf Diskette gespeichert wird.

Die Geräteadresse kann durch POKE 186, (gewünschte Geräteadresse) geändert werden.

(Hermann Schinagl/tr)

```
name : clip                 033c 03cb
-------------------------------------
033c : 20 fd ae 20 57 e2 a9 08  4e
0344 : 85 b8 20 fd ae f0 7d 90  77
034c : 4b c9 ab d0 77 20 73 00  c7
0354 : 20 c2 03 a5 2b a6 2c 85  ee
035c : c1 86 c2 a0 05 b1 5f f0  62
0364 : 08 e6 5f d0 f8 e6 60 d0  bb
036c : f4 a6 60 a5 5f 69 07 90  ff
0374 : 01 e8 85 ae 86 af a0 06  95
037c : a2 00 b1 5f 48 8a 91 5f  54
0384 : c8 b1 5f 48 8a 91 5f 20  f9
038c : fa f5 a0 07 68 91 5f 88  2b
0394 : 68 91 5f 60 20 c2 03 a5  18
039c : 5f a6 60 85 c1 86 c2 20  b3
03a4 : 79 00 f0 b7 c9 ab d0 1c  c6
03ac : 20 73 00 f0 06 20 c2 03  16
03b4 : 4c 5f 03 a5 2d a6 2e 85  f1
03bc : ae 86 af 4c fa f5 20 6b  d9
03c4 : a9 4c 13 a6 4c 08 af 00  f1
```

Listing 1. Mit »CLIP« können Sie einzelne Teile eines Programmes speichern

Unterdrückte Fehlermeldung

Manchmal ist es in einem Programm sinnvoll, eine Fehlermeldung zu unterdrücken. Ein Beispiel: Sie möchten eine mathematische Funktion darstellen und benötigen dazu eine Zahlenreihe von —10 bis 10. Die Funktion selbst lautet: y = 1/x. An der Stelle x = 0 ist die Funktion nicht definiert, was sich durch einen »DIVISION BY ZERO ERROR« zeigt. Probieren Sie einmal aus: »10 FOR X = —10 TO 10 : PRINT X; : NEXT«. Die genannte Fehlermeldung erscheint bei x = 0. Geben Sie nun ein: »POKE 768,61« und die FOR-NEXT-Schleife. Die Fehlermeldung bleibt aus. Sie sparen sich in Ihren Programmen so eine IF-Abfrage auf x = 0. Mit »POKE 768,139« lassen Sie Fehlermeldungen wieder zu. Achtung: Die POKEs funktionieren nur innerhalb eines Programms. (Kai Vorhauser/tr)

Reset-Schutz für Basic-Programme

Dieses Programm (Listing 2) schützt Ihre Basic-Programme vor unerlaubtem Zugriff. Einmal aktiviert, verhindert es a) Abbruch des Programmlaufs mit RUN/STOP-RESTORE, b) Listen des Programms und c) Auslösen eines Resets.

Wird ein Reset-Taster betätigt, führt das Programm automatisch einen »RUN 0«-Befehl aus. Das zu schützende Basic-Listing muß daher eine Zeile 0 besitzen (zum Beispiel »0 REM«). Wird nach Aktivieren der Routine ein Programm gelistet, so erscheinen nur wirre Zeichen auf dem Bildschirm. Trotzdem funktioniert es nach einem »RUN« einwandfrei. Um einen automatischen Basic-Programm-Start auch bei Drücken von <RUN/STOP-RESTORE> auszulösen, müssen Sie noch folgende POKEs eingeben: »POKE 792,226 : POKE 793,252« (diese POKEs bewirken, daß bei der genannten Tastenkombination ein Reset und damit ein Programmstart ausgelöst wird). Wenn Sie das Listing mit dem MSE abgetippt haben, können Sie es bei Bedarf absolut (also mit »,8,1«) laden und mit »SYS 49152« starten (danach »NEW« eingeben!). Der Reset-Schutz ist so lange aktiv, bis Sie Ihren C 64 ausschalten. (Martin Legarth/tr)

```
name : reset                c000 c04e
-----------------------------------
c000 : a9 31 a2 c0 8d 00 80 8e  fa
c008 : 01 80 8e 03 80 a9 45 8d  d3
c010 : 02 80 a9 c3 a2 c2 a0 cd  93
c018 : 8d 04 80 8e 05 80 8c 06  2c
c020 : 80 a9 38 a2 30 8d 07 80  64
c028 : 8e 08 80 a9 ea 8d 28 03  d1
c030 : 60 a9 00 8d 03 08 8d 04  c5
c038 : 08 20 a3 fd 20 8e a6 20  4a
c040 : 5e a6 4c 71 a8 68 a8 68  74
c048 : aa 68 40 aa 68 40 00 00  14
```

Listing 2. Ein neunzigprozentiger Reset-Schutz

Buntes Listing

Wenn Sie ein Basic-Listing vor sich haben, werden Sie bemerken, wie schwierig es sein kann, zusammengehörende Unterprogramme zu erkennen. Die Idee ist es nun, diese Unterprogramme einfach mit der gleichen Farbe aufzulisten. Dies können Sie mit Hilfe von »künstlichen Steuerzeichen« (siehe Serie in früheren 64'er-Ausgaben) umständlich oder auch mit List COLOR einfach erreichen.

```
10  DATA 72,201,143,208,11,200,177,95,201,32,240,3,141,
    134,2,136,104,76,26,167
20  FOR I=49152 TO 49171 : READ A : POKE I,A : NEXT
30  POKE 774,0 : POKE 775,192
```

Starten Sie hierzu das kleine Programm. Geben Sie hinter einem REM-Befehl direkt ein Zeichen ein, so wird dieses als Farbcode interpretiert. Wenn kein Zeichen hinter dem REM folgt, so wird das folgende Listing schwarz gefärbt. Wenn Sie zwischen REM und dem nächsten Zeichen ein Leerzeichen eingeben, wird in der bisherigen Farbe weitergelistet. (Arno Gölzer/tr)

ASCII-Code in Bildschirmcode umwandeln

Wie wandelt man am einfachsten ein Zeichen vom ASCII-Code (PRINT CHR$(...)) in den Bildschirmcode (POKE...) um? Die wirklich einfachste und genialste Lösung besteht darin, das entsprechende Zeichen auf den Bildschirm zu bringen und dann mit PEEK den Code direkt aus dem Bildschirmspeicher auszulesen. Also:

```
10 PRINT"(HOME)"CHR$(ASCII-Code):PRINT PEEK(1024)
```

Umgekehrt geht es natürlich ebenso einfach:

```
10 POKE 1024,Bildschirmcode:PRINT"(HOME)":OPEN1,3:INPUT#
1,A$:PRINT ASC(A$+CHR$(0)):CLOSE1
```

Und weil wir gerade bei der Bildschirmverwaltung sind: Um in einem Programm die nächste PRINT-Anweisung zu positionieren, gibt es zwei Möglichkeiten:

1. Man verwendet eine Kolonne von Cursor-Steuerzeichen, oder
2. zwei POKEs und einen SYS-Befehl:

```
POKE211,Spalte:POKE214,Spalte:SYS58640
```

(Hans Ippisch/tr)

Das geheimnisvolle »READY.«

Bestimmt ist Ihnen das auch schon passiert: Sie waren mit dem Cursor in der Zeile, in der gerade eine »READY.«-Meldung stand und haben <RETURN> gedrückt — und sich über den daraufhin ausgegebenen »OUT OF DATA ERROR« gewundert. Warum ist das so?

Schalten Sie Ihren C 64 einmal aus und wieder ein. Fahren Sie mit dem Cursor eine Zeile nach oben, so daß er auf der »READY.«-Meldung steht, und drücken RETURN. Es erscheint die beschriebene Fehlermeldung. Geben Sie nun einmal eine Basic-DATA-Zeile ein, zum Beispiel »10 DATA 123«. Fahren Sie jetzt mit dem Cursor wieder auf das Wort »READY« und drücken RETURN. Was passiert? Richtig, die Fehlermeldung bleibt aus. Tippen Sie jetzt »PRINT Y.« (plus ⟨RETURN⟩-Taste natürlich). Sie werden erstaunt sein, die Zahl 123 auf dem Bildschirm zu finden. Ahnen Sie, warum manchmal ein »OUT OF DATA ERROR« erscheint, wenn Sie »READY.« eingeben?

Der Computer interpretiert Ihre »READY.«-Eingabe nämlich als den Basic-Befehl READ. Er versucht, in die Variable »Y.« eine Zahl aus einer DATA-Zeile einzulesen. Da aber keine DATA-Zeile vorhanden war, meldet er einen »OUT OF DATA ERROR«. Erstaunlich, nicht? (tr)

Programme nachladen

Wenn man von einem Basic-Programm aus ein zweites Programm (meist eine Maschinenroutine) nachladen möchte, so ist dies gar nicht einmal so schwierig. Man muß nur ein paar Punkte beachten.

Der wichtigste wäre, daß der C 64 nach Beenden des Ladevorgangs einen »GOTO erste Programmzeile« ausführt. Wenn Sie zu Beginn eines Basic-Programms also eine Maschinenroutine (zum Beispiel Listing 2) durch »10 LOAD "XYZ",8« nachladen möchten, würde sich der C 64 in einer Endlosschleife »aufhängen«. Wir müssen uns beim ersten Durchlauf der Zeile 10 merken, daß das Programm jetzt schon geladen ist. Am einfachsten geschieht dies durch »10 IF A = 0 THEN A = 1 : LOAD "XYZ",8«. Beim ersten Programmstart mit »RUN« werden alle Variablen, also auch A, auf Null gesetzt. Die IF-Bedingung ist daher erfüllt; der LOAD-Befehl wird ausgeführt. Danach beginnt der C 64 wieder bei Zeile 10, die Variable A hat jetzt jedoch den Wert 1 — der LOAD-Befehl wird übersprungen und im Programm fortgefahren. Möchte man mehrere Programme nachladen, geschieht dies auf dieselbe Weise:

```
10  IF A=0 THEN A=1 : LOAD ''P1'',8
20  IF A=1 THEN A=2 : LOAD ''P2'',8
30  IF A=2 THEN A=3 : LOAD ''P3'',8
```

und so weiter.

Dies alles gilt jedoch nur für Maschinenprogramme! Möch-

te man ein Basic-Programm nachladen, so ist dies schon etwas komplizierter. Solange das nachgeladene Programm kürzer ist als das erste, genügt es, einen einfachen LOAD-Befehl einzusetzen (die IF-Abfrage kann entfallen, da das zweite Programm ja gleich gestartet wird). Wenn das nachgeladene Programm jedoch länger ist, werden sämtliche Variablen überschrieben.

Die allgemein günstigste Lösung soll hier kurz vorgestellt werden. Sie kann sowohl für Maschinen- als auch für Basic-Programme Verwendung finden.

Wenn Ihnen der Begriff »Tastaturpuffer« geläufig ist, können Sie diesen Absatz überspringen. Wenn nicht, geben Sie einmal folgende Zeile ein (ohne Zeilennummer), drücken <RETURN> und dann ein paar Tasten (bevor das »READY.« erscheint).

```
FOR I=1 TO 2000 : NEXT
```

Sie sehen, daß sich der C 64 Ihre Tastendrücke (bis zu zehn Stück) gemerkt hat. Er speichert sie in seinem »Tastaturpuffer«. Das Gute daran ist, daß wir durch ein paar gezielte POKE-Anweisungen in einem Basic-Programm diese Tastendrücke simulieren können. Das ist auch das Prinzip unserer Laderoutine:

Wir schreiben den LOAD-Befehl, der unser zweites Programm nachladen soll, auf den Bildschirm und machen dem C 64 vor, daß wir die <RETURN>-Taste gedrückt hätten. Dadurch wird der LOAD-Befehl dann ausgeführt. Also:

```
10  PRINT "{CLR,2DOWN}LOAD"CHR$(34)"XYZ"CHR$(34)",8
20  PRINT "{4DOWN} RUN{HOME}";
```

(Zur Erinnerung: Buchstaben innerhalb geschweifter Klammern dürfen Sie nicht ausschreiben, sondern müssen die entsprechenden Cursor-Tasten drücken.)

Was bedeuten die »CHR$(34)« in der Zeile 10? Wie Sie vielleicht wissen, ist es nicht möglich, ein Anführungszeichen innerhalb von Anführungszeichen zu schreiben (mit »PRINT """«?). Dieses benötigen wir aber für den LOAD-Befehl. Das CHR$(34) bewirkt nun, daß an der entsprechenden Bildschirmposition ein Anführungszeichen ausgegeben wird.

Wenn Sie das Programm mit RUN starten, erscheint in der dritten Zeile der LOAD-Befehl und etwas weiter unten ein »RUN«. Wenn Sie jetzt zweimal <RETURN> drücken würden und ein Programm mit dem Namen »XYZ« auf Diskette hätten, würde dieses geladen und gestartet. Da wir das Ganze aber programmgesteuert machen wollen, müssen wir die beiden RETURNs von Basic aus simulieren, also in den Tastaturpuffer schreiben. Dies geschieht über:

```
30 POKE 631,13 : POKE 632,13
```

Der Tastaturpuffer hat nämlich die Adressen 631 bis 640. »13« ist der ASCII-Code der <RETURN>-Taste (siehe auch entsprechenden Anhang im Commodore-Handbuch zu Ihrem C 64). Jetzt müssen wir dem C 64 nur noch mitteilen, daß im Tastaturpuffer noch zwei unbearbeitete Tastendrücke vorliegen: `40 POKE 198,2`

Und damit die Tastendrücke auch ausgeführt werden, muß noch eine END-Anweisung folgen: `50 END`

Natürlich können Sie über diesen Trick nach dem Ladevorgang auch erst eine Variable definieren oder einen POKE ausführen. Dazu müßten Sie einfach die PRINT-Anweisung in Zeile 20 ändern.

Der Tastaturpuffer bietet eine Fülle von Möglichkeiten. Wer sich erst einmal mit ihm angefreundet hat, wird ihn nicht mehr missen wollen. Beim Ausprobieren von Programmen, die mit dem Tastaturpuffer arbeiten, ist es ratsam, den Tastenzähler bei Adresse 198 so lange auf Null stehen zu lassen, bis auf dem Bildschirm die Positionierung der auszuführenden Befehle stimmt. Zu beachten wäre noch, daß Sie in den Tastaturpuffer nur ASCII-Werte POKEn dürfen (über PRINT ASC("Taste") oder Anhang im Handbuch herauszufinden).

(tr)

Tips & Tricks für Profis

Diesmal sollen vor allem die Schachspieler unter den C 64-Fans mit einem äußerst nützlichen Programm bedacht werden. Weiterhin bringen wir zwei Tips zum Thema »Basic-Tokens«, eine Disketten-Reformat-Routine als Dreizeiler (!) und viele weitere Aha-Erlebnisse für Profis.

In der letzten Ausgabe fragten wir Sie, warum der Einzeiler

```
10 FOR I=1 TO 20 : OPEN I,2 : NEXT
```

keinen »too many files«-, sondern einen »next without for«-Error zur Folge hat. Haben Sie es erraten?

Der Grund liegt in der Sekundäradresse 2 des OPEN-Befehls. Sie besagt: »RS232-Kanal öffnen«. Und wenn man jetzt noch weiß, daß nach einem solchen OPEN-Befehl automatisch sämtliche Variablen, Rücksprungadressen für GOSUBs und FOR-NEXT-Schleifen gelöscht werden, hat man die Lösung schon. Der C 64 reserviert sich am Ende des Basic-Speichers etwas Platz für einen RS232-Puffer. Und sicherheitshalber löscht er dabei den kompletten Variablenspeicher.

Also: Beim Schreiben von Programmen, die RS232-Routinen verwenden, immer darauf achten, daß der entsprechende OPEN-Befehl in der ersten Zeile des Programms steht!

(tr)

Tokens im Klartext

```
1 POKE769,177:FORI=1TO76:POKE73,255:POKE781,I:SYS42794:
PRINT,:NEXT:POKE769,227
```

Dieser Einzeiler gibt sämtliche Basic-Befehle auf dem Bildschirm aus.

Er benützt die Routine, die den Basic-Code in Klartext umwandelt ($A717). Diese Routine zieht von dem übergebenen Basic-Token 127 ab und schiebt das Ergebnis ins X-Register. Danach wird das Y-Register in der Speicherstelle $49 (73) gespeichert. Die Routine gibt nun den Basic-Befehl aus und springt nach $A6EF. Dort wird unter anderem das Y-Register mit dem Wert der Speicherstelle $49 (73) geladen und um eins erhöht. Ist das Ergebnis 0 wird zum Basic-Warmstart ($E386) verzweigt, dort wird über den Befehl »JMP ($0300)« nach $E38B gesprungen. Dabei steht in $0300 (768) der Wert $8B und in $0301 (769) der Wert $E3.

Der Einzeiler POKEt nun in einer FOR-NEXT-Schleife die Werte von 1 bis 76 (entspricht den Token -127) in die Speicherstelle 781. Der Wert dieser Speicherstelle wird beim SYS-Befehl in das X-Register geladen. Zusätzlich wird der Wert 255 in die Speicherstelle 73 ($49) gePOKEt und mit dem SYS 42794 nach $A72A gesprungen. Darauf wird der Basic-Befehl auf dem Bildschirm ausgegeben, das Y-Register mit dem Wert der Speicherstelle 73 ($49) geladen und um eins erhöht. Dadurch steht nun 0 im Y-Register und der Computer verzweigt zum Basic-Warmstart. Zuvor hat der Einzeiler das High-Byte des Warmstartvektors auf ein »RTS« umgebogen (statt $E38B auf $B18B). Deshalb wird anstatt zum Basic-Warmstart wieder zurück ins Basic-Programm gesprungen. Nach Beendigung der Schleife wird der Warmstartvektor auf den ursprünglichen Wert gebogen. Deshalb darf das Programm nicht unterbrochen werden!

(Mathias Kühlewein/tr)