

**A**nfänger kennen die Müllabfuhr, die Garbage Collection, meistens noch nicht, da Ihre Programme noch kurz, die Anzahl der verwendeten Strings gering und der freie Speicherplatz groß ist.

Fortgeschrittene lernen sie spätestens dann kennen, wenn Sie ein umfangreiches und vielseitiges Programm geschrieben haben (mit vielen Strings und vor allem Stringarrays) und dann erstaunt feststellen, daß der Computer plötzlich alle Arbeiten unterbricht, um nach Sekunden oder Minuten weiter zu machen, als ob nichts geschehen wäre.

**Das Problem mit dem Müll**

In dieser Zeit, die im Extremfall eine Stunde übersteigen kann, war die besagte Garbage Collection am Werk. Dabei läuft etwa Folgendes ab (ausführlich beschrieben im 64'er, Ausgabe 1/85 und 2/85):

Da der Computer jedesmal, wenn eine Stringvariable neu definiert wird, den Text des Strings (wenn er nicht im Programm steht oder die Länge Null hat) in freie Speicherplätze schreibt, den Zeiger auf diesen Text ändert und den alten Text einfach stehenläßt, ist irgendwann kein Speicherplatz mehr frei. Dann muß das Betriebssystem die alten, ungültig gewordenen Texte entfernen, beziehungsweise die Aktuellen wieder zusammenschieben, bevor es weitergeht.

Dieses »Müllaufsammeln« (englisch: garbage collection) macht nun der C 64 nach einer relativ umständlichen Methode. Deren Zeitbedarf wächst ungefähr quadratisch mit der Anzahl der definierten Stringvariablen. Das heißt, bei einem Programm mit 1000 definierten Strings braucht die Garbage Collection etwa 10000mal so lang wie bei zehn definierten Strings.

# Weg mit dem Müll

**Der Feind aller Datenverarbeitungsprogramme ist die Beseitigung alter, überflüssiger Stringinhalte. Sie zwingt dem Anwender oft unzumutbare Wartezeiten auf. Dieses Programm befreit Sie davon endgültig.**

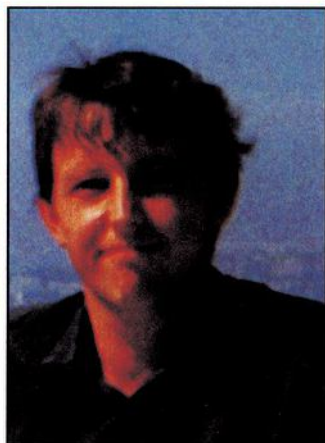


**Die Lösung: eine neue Routine**

Eine Grundregel des Programmierens lautet: Geschwindigkeit ist umgekehrt proportional zum verwendeten Speicherplatz. Ist man also knapp mit Speicherplatz, dann braucht man komplizierte und langsame Verfahren. Und so kann man mit einem einzigen Durchlauf die Überreste der Variablen beseitigen: Zuerst werden alle Stringvariablen und -arrays untersucht, und die gültigen Texte, sofern sie nicht im

Programm stehen oder die Länge Null haben, hintereinander in einen freien Speicherplatz geschrieben. Dieser befindet sich natürlich im RAM unterhalb des ROMs. Gleichzeitig korrigiert man die Textzeiger auf den späteren, richtigen Speicherbereich. Zuletzt überträgt man den gesamten Block gültiger Stringinhalte in den normalen Basic-Bereich und korrigiert die Basic-Zeiger auf die aktuellen Adressen.

(W. Meierhofer/og)



**Lebenslauf**

Ich wurde am 30.10.55 in Regensburg geboren und gehöre somit nicht mehr zu den jüngeren Computerfans. Bis zum Abitur an einem neusprachlichen Gymnasium beschäftigte ich mich hauptsächlich mit Modellbau, bis sich meine technische Ader durch Freunde auf Elektrotechnik, speziell Elektroakustik, umleiten ließ.

Dies setzte sich in einem Studium der Nachrichtentechnik an der Hochschule der Bundeswehr in München fort. Dort stieß ich auch

zum erstenmal auf Mikroprozessoren (den guten alten 8080), die mir bis dahin nicht ganz geheuer waren.

Durch meine (technische) Tätigkeit als Zeitsoldat bei der Bundeswehr und durch sich steigernde Einkäufe von Heimcomputern konnte ich mein Hobby zum Leidwesen meiner Frau stetig ausbauen.

Am C 64 fasziniert mich immer noch, daß man aus einem intelligent konstruierten Heimcomputer eigentlich großartige Effekte »herausholen« kann.

(W. Meierhofer)

# Weg mit dem Müll

## Superschnelle Garbage Collection auf dem C 64. Keine überflüssigen Wartezeiten mehr bei umfangreichen String- und Arrayoperationen.

Die hier vorliegende neue Garbage-Collection hat folgende Vorteile:

- a) Ausführen der Garbage Collection im 10tel Sekundenbereich
- b) Aufruf von Basic aus möglich
- c) Einbindung ins Betriebssystem möglich
- d) Obwohl, wie viele bereits erkannt haben werden, der Bereich unter dem Betriebssystem- und Interpreter-ROM benutzt wird, kann der Anwender das Betriebssystem nach seinen Erfordernissen ändern (zum Beispiel Tastaturtabellen-Änderungen) und unter bestimmten Voraussetzungen sogar hochauflösende Grafik in diesem RAM benutzen.

### Einschränkungen:

a) Da als Hilfsspeicherbereich das RAM unter dem ROM (2 x 8 KByte) verwendet wird, sollte die Zeichenzahl aller definierten, aktuellen Stringtexte (ohne Mülltexte) nicht größer als zirka 16 000 Byte sein. Bei einer größeren Zahl kann es zu Textfehlern kommen, da dann der Speicherbereich von \$FFFF bis \$E000 und \$BFFF bis \$A000 nicht mehr ausreicht (Bereich von oben nach unten benutzt) und damit der normale Stringtextbereich von \$A000 nach unten zu mit überschrieben wird. In der Regel sind aber Texte, die in diesem überschriebenen Bereich stehen, längst im Hilfsbereich als eine der ersten gerettet worden, so daß man auch mehr als zirka 16 000 Zeichen verwenden kann.

Auch in großen Programmen werden meist wesentlich weniger Strings benutzt. Wird der Bereich unterhalb von \$A000 vor Basic geschützt und mit Maschinenprogrammen belegt, darf die Zahl von zirka 16 000 nicht überschritten werden.

b) Das RAM unter dem Interpreter-ROM (\$A000 bis \$BFFF) kann frei benutzt werden, wenn die Gesamtzeichenzahl aller aktuellen Strings nicht über zirka 8 000 liegt und die Garbage Collection nicht ins Betriebssystem eingebunden wird.

Tippen Sie zuerst Listing 1 mit dem MSE ein, und speichern Sie es.

a) Anwendung in »normalen« Basic-Programmen

Einzige Voraussetzung ist, daß sich das Programm mit der schnellen Routine im Speicher befindet. Es kann jederzeit durch SYS 50944 (zum Beispiel in regelmäßig durchlaufenden Programmteilen oder während des Einlesens von sequentiellen Dateien) aufgerufen werden.

b) Einbindung ins Betriebssystem

Die Garbage Collection kann auch fest ins Betriebssystem eingebunden werden, das heißt wenn kein Speicherplatz mehr vorhanden ist, wird sie automatisch anstelle der alten ausgeführt. Selbstverständlich ist sie auch noch einzeln mit SYS 50944 aufrufbar.

Die Einbindung veranlaßt der Befehl SYS 51400.

Damit läuft das Betriebssystem und der Interpreter im RAM (Inhalt der Speicherstelle 1 ist 53 anstatt der üblichen 55).

## Die Anwendung

Soll das Betriebssystem (Tastaturtabelle oder ähnlich) noch mehr geändert werden, sind die entsprechenden POKes in Maschinensprache ab Speicherstelle \$C9B2 anzufügen und mit RTS abzuschließen (siehe Listing 2, Source-Code).

(An diese Routine, die immer wieder die geänderten Bytes für den automatischen Aufruf der neuen Garbage Collection ins Betriebssystem schreibt, hängt man dadurch die eigenen Änderungen an.)

Dies macht man am besten durch ein Monitorprogramm, mit dem man dann auch die gesamte Neufassung seiner »persönlichen« Garbage Collection/Betriebssystemversion speichern kann.

```

programm : garbage64          c700 c9b5
-----
c700 : 78 48 98 48 8a 48 a5 01 4f
c708 : 8d 84 c9 a9 2e 8d e7 07 7c
c710 : a9 09 8d e7 db a0 14 b9 25
c718 : 44 00 99 86 c9 88 d0 f7 a7
c720 : a9 37 85 01 a5 2d 85 45 4b
c728 : a5 2e 85 46 a9 00 85 47 4e
c730 : a9 00 85 48 a9 00 85 4b 8b
c738 : 85 4c 8d 85 c9 a5 37 85 a9
c740 : 4d a5 38 85 4e a5 45 c5 d1
c748 : 2f d0 06 a5 46 c5 30 f0 4b
c750 : 4e a0 00 b1 45 0a b0 39 fe
c758 : c8 b1 45 0a 90 33 18 a9 e2
c760 : 02 65 45 85 45 90 02 e6 c6
c768 : 46 a0 00 b1 45 f0 14 a0 a2
c770 : 02 b1 45 c5 34 90 0c d0 ee
c778 : 07 88 b1 45 c5 33 90 03 17
c780 : 20 25 c9 18 a9 05 65 45 8b
c788 : 85 45 90 02 e6 46 4c 45 71
c790 : c7 18 a9 07 65 45 85 45 d0
c798 : 90 02 e6 46 4c 45 c7 a5 05
c7a0 : 45 c5 31 d0 09 a5 46 c5 91
c7a8 : 32 d0 03 4c 32 c8 a0 00 79
c7b0 : b1 45 0a b0 68 c8 b1 45 bb
c7b8 : 0a 90 62 a0 03 b1 45 48 1a
c7c0 : 88 18 b1 45 65 45 85 4f 9f
c7c8 : 68 65 46 85 50 a0 04 b1 a3
c7d0 : 45 a8 a9 05 18 65 45 85 41
c7d8 : 45 90 02 e6 46 a9 02 18 ad
c7e0 : 65 45 85 45 90 02 e6 46 33
c7e8 : 88 d0 f2 a5 45 c5 4f d0 ab
c7f0 : 06 a5 46 c5 50 f0 a8 b1 a6

c7f8 : 45 f0 14 a0 02 b1 45 c5 1d
c800 : 34 90 0c d0 07 88 b1 45 9f
c808 : c5 33 90 03 20 25 c9 18 6e
c810 : a9 03 65 45 85 45 90 02 06
c818 : e6 46 4c eb c7 a0 03 b1 a3
c820 : 45 48 88 b1 45 18 65 45 17
c828 : 85 45 68 65 46 85 46 4c 59
c830 : 9f c7 a5 47 c9 00 d0 09 f7
c838 : a5 48 c9 00 d0 03 4c 09 dc
c840 : c9 a9 35 85 01 a5 4d 85 59
c848 : 33 a5 4e 85 34 ad 85 c9 ec
c850 : d0 16 a5 47 85 51 a5 48 88
c858 : 85 52 a9 ff 85 45 a9 ff 9a
c860 : 85 46 20 91 c8 4c bb c8 b2
c868 : a5 47 85 51 a5 48 85 52 94
c870 : a9 ff 85 45 a9 bf 85 46 5e
c878 : 20 91 c8 a5 4b 85 51 a5 b9
c880 : 4c 85 52 a9 ff 85 45 a9 ed
c888 : ff 85 46 20 91 c8 4c bb e7
c890 : c8 a0 00 b1 51 91 4d a5 01
c898 : 45 c5 51 d0 08 a5 46 c5 81
c8a0 : 52 d0 02 f0 0f e6 4d d0 fb
c8a8 : 02 e6 4e e6 51 d0 e4 e6 8b
c8b0 : 52 4c 93 c8 e6 4d d0 02 46
c8b8 : e6 4e 60 ad 84 c9 c9 37 bf
c8c0 : f0 47 20 d5 c8 4c 09 c9 bd
c8c8 : a9 01 8d 85 c9 20 d5 c8 8c
c8d0 : a9 35 85 01 60 a9 37 85 d1
c8d8 : 01 a9 00 85 45 a9 e0 85 8f
c8e0 : 46 20 f8 c8 ad 85 c9 f0 9d
c8e8 : 0b a9 00 85 45 a9 a0 85 a8

c8f0 : 46 20 f8 c8 20 a0 c9 60 8c
c8f8 : a2 20 a0 00 b1 45 91 45 e9
c900 : c8 d0 f9 e6 46 ca d0 f4 73
c908 : 60 a0 14 b9 86 c9 99 44 9a
c910 : 00 88 d0 f7 a9 20 8d e7 29
c918 : 07 ad 84 c9 85 01 68 aa a7
c920 : 68 a8 68 58 60 a0 02 b1 78
c928 : 45 85 4a 88 b1 45 85 49 c1
c930 : 88 b1 45 85 53 a8 a5 47 32
c938 : 38 e5 53 85 47 b0 26 c6 08
c940 : 48 a5 48 c9 df d0 1e a9 f6
c948 : 01 8d 85 c9 98 18 65 47 19
c950 : 85 4b 90 02 e6 48 a5 48 b7
c958 : 85 4c a9 00 85 47 a9 c0 28
c960 : 85 48 4c 36 c9 88 b1 49 1d
c968 : 91 47 88 c0 ff d0 f7 a0 7f
c970 : 01 a5 4d 38 e5 53 85 4d 48
c978 : 91 45 b0 02 c6 4e c8 a5 65
c980 : 4e 91 45 60 ea ea ea 00 a6
c988 : 00 00 00 00 00 00 00 00 89
c990 : 00 00 00 00 00 00 00 00 91
c998 : 00 00 00 00 00 00 00 00 99
c9a0 : a9 20 8d 26 b5 a9 00 8d 45
c9a8 : 27 b5 a9 c7 8d 28 b5 a9 51
c9b0 : 60 8d 29 b5 60 00 ff 00 de
    
```

Listing 1. »Garbage 64« reduziert die Zeiten für die Stringmüllbeseitigung auf ein Minimum.

Normaler Inhalt		Beispiel für Änderungen	
\$C9B2 \$60 RTS		\$C9B2 \$A9 LDA # \$FF	
		\$C9B4 \$8D STA \$A000	
		\$C9B7 \$60 RTS	

Speichert den Wert 255 (\$FF) an die Speicher Stelle 40960 (\$A000). Dieses Beispiel ergibt natürlich keinen Sinn.

Mit Grundkenntnissen im Maschinensprache ist es kein Problem, die Erweiterung herzustellen, zumal ja meist mehr als Grundkenntnisse notwendig sind, um das Betriebssystem zu ändern.

### Übersicht:

	Keine Einbindung	Einbindung ins Betriebssystem
Aufruf	SYS 50944	Automatisch und mit SYS 50944 zusätzlich möglich.
Speicherkonfiguration	Normal (Inhalt von 1 ist 55)	Inhalt von 1 ist 53 ROM läuft im RAM Einbindung ins Betriebssystem mit SYS 51398 Betriebssystemänderungen mit aufnehmbar
Verwendbarkeit RAM von \$A000-\$BFFF (40960-49151)	ja, wenn Gesamtzeichenzahl aller aktuellen Strings geringer als zirka 8000 (8 KByte)	nein
Verwendbarkeit RAM von \$E000-\$FFFF (57344-65535)	nein	nein
Speicherplatz unterhalb \$A000 für Maschinenprogramme	frei verwendbar, wenn Gesamtzeichenzahl aller aktuellen Strings kleiner als 16 KByte ist	

Eine Besonderheit am Schluß: Um eine Kontrolle zu haben, wann und wie lange die Garbage Collection läuft, wird während der Ausführung ganz unten rechts am Bildschirm ein kleiner, brauner Punkt angezeigt.

Will man dieses Zeichen ändern, so POKEt man nach 50956 den Bildschirmcode des gewünschten Zeichens, nach 50961 die Farbe.

POKE 50956, 1: POKE 50961, 7 bewirkt, daß dort während des Ablaufs ein gelbes A steht.

Mit POKE 50956, 32 wird die Anzeige abgestellt.

## Programmbeschreibung für Maschinisten

### Aufruf

Prinzipiell kann das Programm, wie oben erwähnt, jederzeit mit SYS 50944 (beziehungsweise mit JSR \$C700) aufgerufen werden.

Der Trick, es ins Betriebssystem einzubinden (SYS 51400), besteht darin, das ROM (Betriebssystem und Interpreter) ins RAM zu kopieren und anstelle der normalen Garbage Collection einen Sprung in »Garbage 64« unterzubringen. Dies muß natürlich nach jeder Garbage Collection immer wieder automatisch geschehen, da »Garbage 64« das RAM unter dem ROM beschreibt, um die gültigen Stringtexte zwischenspeichern (siehe weiter unten).

An die Routine, die diese Änderung vornimmt (am Ende von »Garbage 64« ab \$C9A0), kann man eigene, zusätzliche Befehle anfügen, die zum Beispiel die Tastatortabellen beeinflussen, so daß der Interpreter beziehungsweise das Betriebssystem bei der Einbindung von »Garbage 64« durch SYS 51400 anwenderspezifisch mit geändert und diese Änderung nach jeder Garbage Collection erhalten wird.

```

10 SYS9*4096
20 .OPT P,00
30 *= $C700
100 ; *** GARBAGE COLLECTION ***
101 ;
102 ; DEFINITION DER HILFSZEIGER
103 LAZEI =#45;LAUFZEIGER D.DESKRIPTOREN
104 ROMZEI =#47;LAUFZEIGER UNTER ROM
105 STRIZEI =#49;HILFSZEIGER F. STRING
106 MEZEI =#4B;LETZTER PLATZ OBERES ROM
107 NEUZEI =#4D;STRINGBEREICHSLAUFZEIGER
108 FELDEND =#4F;HILFSZEIGER ARRAYENDE
109 STRISPEI =#51;HILFSZEIGER
110 SUBTRAH =#53;STRINGLAENGEZWISCHENSPE.
111 ;
112 ;
113 ; VORBELEGUNG DER SPEICHERPLAETZE
114 VROMZEI =#0000;START VON ROMZEI
116 VGRNZEI =#DF00;GRENZE+ FUER ROMZEI
117 VJUMPZEI =#C000
118 ;
119 ; 1. VORBEREITUNG -----
120 ;
121 ANFANG SEI; INTERRUPTS VERHINDERN
122 PHA; PROZESSORREGISTER RETTEN
123 TYA
124 PHA
125 TXA
126 PHA
127 LDA 1; SPEICHERKONFIGURATION RETTEN
128 STA ZAHL
129 LDA #46; KONTROLLANZEIGE BILDSCHIRM
130 STA 2023
131 LDA #9
132 STA 56295
133 LDY #20; BENDETIKTE SPEICHER RETTEN
134 RET LDA #44, Y
135 STA SAVE, Y
136 DEY
137 BNE RET
138 LDA #55; AUF ROM UMSCHALTEN
139 STA 1
140 LDA 45; VARIABLENSTART NACH LAZEI
141 STA LAZEI
142 LDA 46
143 STA LAZEI+1
144 ;
145 LDA #<VROMZEI; $FFFF+1 NACH ROMZEI
146 STA ROMZEI
147 LDA #>VROMZEI
148 STA ROMZEI+1
149 ;
150 LDA #0; MEZEI VORBELEGEN
151 STA MEZEI
152 STA MEZEI+1
153 STA ZWEI
154 ;
155 LDA 55; DURCHL. ZEIGER F. STRINGBEREICH
156 STA NEUZEI
157 LDA 56
158 STA NEUZEI+1
159 ;
160 ;
161 ; 2. VARIABLENBEREICH -----
162 ;
163 VOVORN LDA LAZEI; SCHON ENDE VARIABLE
164 CMP 47
165 BNE WEI1
166 LDA LAZEI+1
167 CMP 48
168 BEQ FELDER
169 ;
170 WEI1 LDY #0; STRINGVARIABLE J/N
171 LDA (LAZEI), Y
172 ASL
173 BCS LA7; KEINE STRINGVARIABLE
174 INY
175 LDA (LAZEI), Y
176 ASL
177 BCC LA7; KEINE STRINGVARIABLE
178 ;
179 CLC; LAZEI=LAZEI+2; STRINGVARIABLE
180 LDA #2
181 ADC LAZEI
182 STA LAZEI
183 BCC WEI2
184 INC LAZEI+1
185 ;
186 ;
187 WEI2 LDY #0
188 LDA (LAZEI), Y; LEERSTRING J/N
189 BEQ LA5
190 ;
191 LDY #2; DESCRIPTOR IN STR. BEREICH J/N
192 LDA (LAZEI), Y
193 CMP 52
194 BCC LA5; HB 52 > ALS DESCRIPTOR HB
195 BNE WEI3; HB 52 <> HB DESKRIPTOR=OK
196 DEY
197 LDA (LAZEI), Y
198 CMP 51
199 BCC LA5; LB 51 > LB DES, HB 52 =HB DES
200 ;
201 WEI3 JSR ABSPEI
202 ;
203 LA5 CLC; LAZEI UM 5 ERHOEHEN
204 LDA #5
205 ADC LAZEI
206 STA LAZEI
207 BCC L1
208 INC LAZEI+1
209 LI JMP VOVORN
210 ;
211 LA7 CLC; LAZEI UM 7 ERHOEHEN
212 LDA #7
213 ADC LAZEI
214 STA LAZEI
215 BCC L2
216 INC LAZEI+1
217 L2 JMP VOVORN
218 ;
219 ;
220 ;
221 ; 3. ARRAYBEREICH -----
222 ;
223 FELDER LDA LAZEI; ARRAYBEREICHENDE J/N
224 CMP 49
225 BNE WEI11
226 LDA LAZEI+1
227 CMP 50
228 BNE WEI11
229 JMP RAMUN
230 ;
231 WEI11 LDY #0; STRINGFELD J/N
232 LDA (LAZEI), Y
233 ASL
234 BCS LASTRL; KEIN STRINGFELD
235 INY
236 LDA (LAZEI), Y
237 ASL
238 BCC LASTRL; KEIN STRINGFELD
239 ;

```

Listing 2. Quellcode zu »Garbage 64«

```

240 ;
241 ;
242 ;STRINGFELD
243 LDY #3;ERRECHNEN FELDENDE
244 LDA (LAZEI),Y
245 PHA
246 DEY
247 CLC
248 LDA (LAZEI),Y
249 ADC LAZEI
250 STA FELDEND
251 PLA
252 ADC LAZEI+1
253 STA FELDEND+1
254 ;
255 LDY #4;ANZAHL DER DIMENSIONEN NACH Y
256 LDA (LAZEI),Y
257 TAY
258 ;
259 LDA #5;LAZEI AUF 1. DESCRIPTOR
260 CLC;LAZEI UM 5 ERHOEHEN
261 ADC LAZEI
262 STA LAZEI
263 BCC WEI14
264 INC LAZEI+1
265 ;
266 ;
267 WEI14 LDA #2;LAZEI + DIMENS.ANZAHL*2
268 CLC
269 ADC LAZEI
270 STA LAZEI
271 BCC WEI15
272 INC LAZEI+1
273 WEI15 DEY
274 BNE WEI14
275 ;
276 VUVURN LDA LAZEI;FELDENDE J/N
277 CMP FELDEND
278 BNE WEI16
279 LDA LAZEI+1
280 CMP FELDEND+1
281 BEQ FELDER
282 ;
283 WEI16 LDA (LAZEI),Y;STRING LEER J/N
284 BEQ LA3
285 ;
286 LDY #2;DESCRIPTOR IM STR.BEREICH J/N
287 LDA (LAZEI),Y
288 CMP 52
289 BCC LA3
290 BNE WEI17
291 DEY
292 LDA (LAZEI),Y
293 CMP 51
294 BCC LA3
295 WEI17 JSR ABSPEI
296 ;
297 LA3 CLC;LAZEI UM 3 ERHOEHEN
298 LDA #3
299 ADC LAZEI
300 STA LAZEI
301 BCC WEI18
302 INC LAZEI+1
303 WEI18 JMP VUVURN
304 ;
305 LASTRL LDY #3;LAZEI + ARRAYLAENGE
306 LDA (LAZEI),Y
307 PHA
308 DEY
309 LDA (LAZEI),Y
310 CLC
311 ADC LAZEI
312 STA LAZEI
313 PLA
314 ADC LAZEI+1
315 STA LAZEI+1
316 JMP FELDER
317 ;
318 ;
319 ;
320 ;
321 ;4. RAM UNTER ROM NACH VARIABLE ----
322 ;
323 RAMUN LDA ROMZEI;WAREN STRINGS DA
324 CMP #<VROMZEI
325 BNE RUMUN
326 LDA ROMZEI+1
327 CMP #>VROMZEI
328 BNE RUMUN
329 JMP ENDE
330 ;
331 RUMUN LDA #53;AUF ROM SCHALTEN
332 STA 1
333 LDA NEUZEI;STRINGBEGINNSZEIGER NEU
334 STA 51
335 LDA NEUZEI+1
336 STA 52
337 ;
338 LDA ZWEI;WAR UNTERES RAM IN USE J/N
339 BNE DOPP
340 ;
341 LDA ROMZEI;NUR 1.BEREICH ROMZEI-FFFF
342 STA STRISPEI;SUBROUTINEVORBELEGUNG
343 LDA ROMZEI+1
344 STA STRISPEI+1
345 LDA #<VROMZEI-1
346 STA LAZEI
347 LDA #>VROMZEI-1
348 STA LAZEI+1
349 ;
350 JSR SPEISTRI;UMSPEICHERUNGSRoutine
351 ;
352 JMP ROMNEU
353 ;
354 ;
355 DOPP LDA ROMZEI;1.UND 2. BENUTZT
356 STA STRISPEI;SUBROUTINEVORBELEGUNG
357 LDA ROMZEI+1
358 STA STRISPEI+1
359 LDA #<VJUMPZEI-1
360 STA LAZEI
361 LDA #>VJUMPZEI-1
362 STA LAZEI+1
363 ;
364 JSR SPEISTRI;UMSPEICHERROUTINE
365 ;
366 LDA MEZEI;SUBROUTINEVORBELEGUNG
367 STA STRISPEI
368 LDA MEZEI+1
369 STA STRISPEI+1
370 LDA #<VROMZEI-1
371 STA LAZEI
372 LDA #>VROMZEI-1
373 STA LAZEI+1
374 ;
375 JSR SPEISTRI;UMSPEICHERROUTINE
376 JMP ROMNEU
377 ;
378 ;
379 ;
380 ;
381 SPEISTRI LDY #0;KOPIER V STRISPEI
382 ;- LAZEI NACH NEUZEI AUFWAERTS
383 WIDA LDA (STRISPEI),Y
384 STA (NEUZEI),Y
385 LDA LAZEI
386 CMP STRISPEI
387 BNE W21
388 LDA LAZEI+1
389 CMP STRISPEI+1
390 BNE W21
391 BEQ W24
392 W21 INC NEUZEI
393 BNE W22
394 INC NEUZEI+1
395 W22 INC STRISPEI
396 BNE WIDA
397 INC STRISPEI+1
398 W23 JMP WIDA
399 ;
400 W24 INC NEUZEI
401 BNE W25
402 INC NEUZEI+1
403 W25 RTS
404 ;
405 ;
406 ;
407 ;5. ROM NEU INS RAM KOPIEREN -----
408 ;
409 ROMNEU LDA ZAHL;EINGEBUNDEN J/N
410 CMP #55
411 BEQ ENDE;LAEUFT IM ROM
412 JSR RUMNEU
413 JMP ENDE
414 ;
415 LDA #1;HIER EXTRAEINSPRUNG VON BASIC
416 STA ZWEI
417 JSR RUMNEU
418 LDA #53
419 STA 1
420 RTS
421 ;
422 RUMNEU LDA #55;AUF ROM UMSCHALTEN
423 STA 1
424 ;
425 LDA #*00
426 STA LAZEI
427 LDA #*E0
428 STA LAZEI+1
429 JSR COPI
430 ;
431 LDA ZWEI;UNT. ROM AUCH ZU KOPIEREN
432 BEQ END
433 LDA #*00
434 STA LAZEI
435 LDA #*A0
436 STA LAZEI+1
437 JSR COPI
438 END JSR EINBIND
439 RTS
440 ;
441 ;
442 COPI LDX #32;8K ROM IN RAM KOPIEREN
443 LDY #0
444 AGEIN LDA (LAZEI),Y
445 STA (LAZEI),Y
446 INY
447 BNE AGEIN
448 INC LAZEI+1
449 DEX
450 BNE AGEIN
451 RTS
452 ;
453 ;6. ENDE -----
454 ;
455 ENDE LDY #20;SPEICHER RUECKRETEN
456 ROT LDA SAVE,Y
457 STA #44,Y
458 DEY
459 BNE ROT
460 LDA #32;KONTROLLANZEIGE LOESCHEN
461 STA 2023
462 LDA ZAHL;ALTE SPEICHERKONFIGURATION
463 STA 1
464 PLA;PROZESSORINHALTE WIEDERHOLEN
465 TAX
466 PLA
467 TAY
468 PLA
469 CLI;INTERRUPTS WIEDER ERLAUBT
470 RTS
471 ;
472 ;
473 ;SUBROUTINEN -----
474 ;
475 ;UNTERPROGRAMM, DAS DEN STRING, AUF
476 ;DESSEN DESKRIPTOR LAZEI STEHT,UN-
477 ;TERHALB ROMZEI ABSPEICHERT UND DIE
478 ;POSITIONSANGABE DES DESKR. RELATIV
479 ;ZUM RICHT. STRINGBER. AKTUALISIERT
480 ;
481 ABSPEI LDY #2;STRINGADR. IN STRIZEI
482 LDA (LAZEI),Y;LAENGE IN Y UND STACK
483 STA STRIZEI+1
484 DEY
485 LDA (LAZEI),Y
486 STA STRIZEI
487 DEY
488 LDA (LAZEI),Y
489 STA SUBTRAH
490 TAY
491 ;
492 OGAIN LDA ROMZEI;ROMZEI-STRINGLAENGE
493 SEC
494 SBC SUBTRAH
495 STA ROMZEI
496 BCS ETZ
497 DEC ROMZEI+1
498 LDA ROMZEI+1
499 CMP #>VGRENZEI
500 BNE ETZ
501 ;
502 LDA #1;0.RAM VOLL,ROMZEI+YNACH MEZEI
503 STA ZWEI
504 TYA
505 CLC
506 ADC ROMZEI
507 STA MEZEI
508 BCC WTR
509 INC ROMZEI+1
510 WTR LDA ROMZEI+1
511 STA MEZEI+1
512 LDA #<VJUMPZEI;ROMZEI AUF C000
513 STA ROMZEI
514 LDA #>VJUMPZEI
515 STA ROMZEI+1
516 JMP OGAIN
517 ;
518 ETZ DEY;STRING UNTERS ROM SPEICHERN
519 NOML LDA (STRIZEI),Y
520 STA (ROMZEI),Y
521 DEY
522 CPY #255
523 BNE NOML
524 ;
525 ;
526 LDY #1;NEUZEI-STR.LAENGE=DESKRIPTOR
527 LDA NEUZEI
528 SEC
529 SBC SUBTRAH
530 STA NEUZEI
531 STA (LAZEI),Y
532 BCS WAIDR
533 DEC NEUZEI+1
534 WAIDR INY
535 LDA NEUZEI+1
536 STA (LAZEI),Y
537 RTS
538 ;
539 ZAHL NOP;MERKER FUER KONFIGURATION
540 ZWEI NOP;1 WENN MEHR 0.RAM USED
541 SAVE NOP
542 *= *+25
543 ;
544 ;VERAENDERUNG DES GARBAGEEINSPRUNGS
545 ;UND EVENTUELL DES BETRIEBSSYSTEMS
546 GAR =#B526
547 EINBIND LDA #*20;JSR ANFANG U. RTS
548 STA GAR
549 LDA #<ANFANG
550 STA GAR+1
551 LDA #>ANFANG
552 STA GAR+2
553 LDA #*60
554 STA GAR+3
555 RTS

```

Listing 2.  
Quellencode zu  
»Garbage 64«  
(Schluß)

READY.

### Programmteil Vorbereitung

»Garbage 64« schaltet zuerst Interrupts ab, rettet die Speicherkonfiguration und die Prozessorregister, und schreibt in die rechte untere Ecke des Bildschirms eine Kontrollanzeige. Dieses Zeichen kann, wie schon erwähnt, von Basic aus mittels POKE geändert werden.

Ferner werden die benötigten Speicherplätze in der Zeropa-ge ebenfalls gerettet und anschließend die benötigten Zeiger und Bytes vorbelegt.

### Programmteil Variablen

Nun durchsucht »Garbage 64« den Bereich der Variablen mittels des Zeigers »LAZEI« und überprüft in dieser Schleife laufend, ob das Ende des Variablenbereichs bereits erreicht wurde.

Wird eine Stringvariable gefunden, deren Länge nicht 0 ist und deren Text nicht außerhalb des Stringtextbereichs am Basic-Ende steht (zum Teil weisen die Zeiger auf Texte im Basic-Programm), so wird der dazugehörige Text im Unterprogramm ABSPEI (ab \$C925) anhand des Stringzeigers im RAM unter dem ROM, von \$FFFF beginnend abwärts, abgelegt.

Dabei wird der Textzeiger im Deskriptor so korrigiert, als ob der Stringtext im normalen Bereich (ab Basic-Ende abwärts) stehen würde.

Es wird überprüft, ob der zwischenspeichernde Stringtext den Bereich von \$E000 abwärts überschreiben würde, was natürlich einen Totalabsturz zur Folge hätte. In diesem Fall wird die letzte benutzte Position im »oberen« RAM in »MEZEI« gemerkt und ab \$BFFF abwärts weitergemacht. Sind mehr als zirka 8 KByte + 8 KByte gültige Stringtexte vorhanden, was so gut wie nie der Fall ist, wird der Bereich unterhalb \$A000 ebenfalls noch beschrieben, was aber keine Rolle spielt, solange in diesem Bereich keine Maschinenprogramme (die vor Basic geschützt wurden) stehen.

Die Möglichkeit, daß hier noch nicht gerettete, gültige Stringtexte stehen, ist vernachlässigbar und wäre von Basic aus nur durch absichtliche und äußerst umständliche Definitionsreihenfolgen erreichbar.

### Programmteil Arraybereich

Ist der Variablenbereich durchsucht, wird mit dem Arraybereich genauso vorgegangen. Jedes Array wird überprüft, ob

es ein Stringarray ist und jeder Stringdeskriptor wird untersucht, ob der Stringtext die Länge größer 0 hat und der Text im richtigen Bereich steht.

Zu beachten ist, daß der Arraykopf abhängig von der Anzahl der Dimensionen verschieden lang ist, und so der Beginn der Deskriptoren errechnet werden muß.

Wenn ein Array kein Stringarray ist, wird es anhand der Längenangabe im Arraykopf (siehe dazu auch 64'er, Ausgabe 1/85 und 2/85) übersprungen.

### Programmteil RAM unter ROM nach Variable

Schließlich wird der Bereich der im RAM unter dem ROM abgelegten gültigen Stringtexte in den Bereich verlegt, in dem Stringtexte zu stehen haben, nämlich am Basic-Ende. Dabei wird zuerst der Bereich ab »ROMZEI« (der durch das RAM unter dem ROM »mitgelaufene« Zeiger) bis \$C000, falls dieser Bereich benutzt wurde und dann der Bereich von »MEZEI« bis \$FFFF (beziehungsweise von »ROMZEI« bis \$FFFF, wenn nur das obere RAM beschrieben worden war), in den Bereich von »NEUZEI« aufwärts übertragen. »NEUZEI« ist der Zeiger, der im »echten« Stringtextbereich mitgelaufen war und der zur Aktualisierung der neuen Zeiger in den Stringdeskriptoren dien- te. Er bezeichnet ebenfalls den neuen Beginn des Stringtextbereichs.

Das Unterprogramm »SPEISTRI« (ab \$C891) führt die eigentliche Umspeicherung durch, nachdem die oben erwähnten zwei Fälle festgestellt und die verwendeten Hilfszeiger (die jetzt allerdings eine neue Funktion haben) vorge- setzt wurden.

### Programmteil ROM neu ins RAM kopieren

Nun wird, wenn der Interpreter und das Betriebssystem vorher im ROM »gelaufen« waren und »Garbage 64« nicht eingebunden war, sofort zum Programmteil »ENDE« gesprungen.

Ansonsten wird das Betriebssystem wieder ins RAM kopiert, und, wenn das RAM unter dem Interpreter-ROM benutzt worden war (mehr als 8 KByte Stringtexte), der Interpreter ebenfalls.

In der Subroutine »EINBIND« wird »Garbage 64« wieder eingebunden und die vom Anwender ergänzten Änderungen wieder durchgeführt.

### Programmteil Ende

Hier sind die am Anfang geretteten Speicher wieder zu holen und die Kontrollanzeige zu löschen. Die vorher eingestellte Speicherkonfiguration wird aktiviert und mit RTS »Garbage 64« beendet.

## Testen Sie doch mal die »Müllabfuhr« im C 64

Um Ihnen den Geschwindigkeitsvorteil einmal vorzuführen, haben wir noch ein Demonstrationsprogramm (Listing 3) für Sie. Tippen Sie es zuerst ohne die Zeilen 1 bis 5 ein und starten Sie das Demo. Die Zahl hinter dem Kommentar »test ok« ist zunächst die normale Zeit der Schleifen. Wenn die Garbage Collection zuschlägt, sehen Sie das an der wesentlich längeren Zeit. Tippen Sie dann die fehlenden fünf Zeilen ein, und starten Sie das Programm erneut. Dazu muß »GARBAGE64« (Listing 1) auf Diskette verfügbar sein. Für Kassette ändern Sie bitte Zeile 2 in »... ,1,1«. Von der Garbage Collection merken Sie nun nichts mehr. Noch krasser ist der Vorgang beim folgenden Beispiel zu sehen:

```
DIM A$(9000):FOR I=0 TO 9000 : A$(I)="A" : NEXT I : PRINT FRE(0)
```

Mit der schnellen Routine ist das kein Problem, aber ohne sie . . . (W. Meierhofer/og)

```

0 REM DEMO GARBAGE COLLECTION <217>
1 IF A=1 THEN 3 <012>
2 A=1:LOAD"GARBAGE64",8,1 <072>
3 POKE 53280,0:POKE 53281,0 <131>
4 SYS 51400 <228>
5 POKE 50956,1:POKE 50961,7 <247>
10 DIM K(30) <006>
12 DIM LZ(200) <005>
14 DIM E$(400,1):E$(400,1)="OK" <210>
16 DIM N$(20):N$(20)="TEST" <158>
30 TI$="000000":F%=56 <201>
32 A$="12345" <058>
34 X=3 <075>
36 B$="ABCDEF"+"G" <004>
38 C$="TES"+"T" <085>
40 B$="" <073>
62 FOR X=1 TO 400:E$(X,0)="ABCDEFGH"+"9":N <197>
EXT
63 FOR X=1 TO 400:E$(X,0)="12345678"+"9":N <197>
EXT
85 PRINT A$:PRINT B$:PRINT C$:PRINT E$(1,0 <255>
):PRINT E$(400,0):PRINT N$(20):E$(400,1
):
90 PRINT TI/60:GOTO 30 <234>

```

© 64'er

Listing 3. Demoprogramm für »Garbage 64«. Die Zeitunterschiede werden deutlich sichtbar.