MZ-80B - intern Nr. 15

Dieses MZ-80B-intern stellt Ihnen eine Fülle von Manipulationsmöglichkeiten für die BASIC-Interpreter BASIC SB-5510 und BASIC SB-6510 vor. Die einzelnen Programme werden am sinnvollsten als Unterprogramme benutzt. Da z. T. erheblich in den Interpretern 'herumgepfuscht' wird, sollten die Programme für Ihren Anwendungsfall auf jeden Fall ausgetestet werden. Dabei sollte besonderes Augenmerk auf das Verhalten in Fehlerfällen gerichtet werden, wenn Möglichkeit sollten Fehler durch ON ERROR GOTO abgefangen werden (z. B. keine Diskette eingelegt bei DIR/SAVE u. ä.). In solchen Fällen ist unbedingt dafür zu sorgen, daß die jeweils hinter den USR-Befehlen folgenden POKE-Befehle ausgeführt werden, um den Interpreter wiederherzustellen.

Zuerst ein Programm, daß die Verwendung von Direktbefehlen in Programmen erlaubt. Vor und hinter dem Direktbefehl stehen vier POKE-Befehle, damit ist es schon getan. Im Beispiel wurde der KLIST-Befehl benutzt, an seine Stelle können die anderen Direktbefehle treten.

BASIC SB-5510

```
100 POKE $1AF1,$C9
110 POKE $1276,$C3
120 POKE $1277,$7C
130 POKE $1278,$1B
140 KLIST
150 POKE $1AF1,$E5
160 POKE $1276,$ED
170 POKE $1277,$7B
180 POKE $1278,$0A
```

BASIC SB-6510

```
100 POKE $1C10,$C9
110 POKE $1280,$C3
120 POKE $1281,$9E
130 POKE $1282,$1C
140 KLIST
150 POKE $1C10,$E5
160 POKE $1280,$ED
170 POKE $1281,$7B
180 POKE $1282,$0A
```

Schwierigkeiten macht hier nur der DIR-Befehl in BASIC SB-6510, wenn mehr als 17 Directory-Einträge angezeigt werden, wird die Ausgabe ja unterbrochen und die Tastatur abgefragt, was mit obiger Methode zum Programmabsturz führt. Nur für den DIR-Befehl deshalb das folgende Programm. Es gestattet die Fortsetzung der Directory-Ausgabe nur mit CR, andere Tasten werden zwar angenommen, bewirken aber nichts.

```
100 POKE $1C10,$C9: POKE $1280,$C3: POKE $1281,$9E: POKE $1282,$1C
110 POKE $1312,$A3: POKE $1317,$C9
120 DIR FD2
130 POKE $1312,$9A: POKE $1317,$C8
140 POKE $1C10,$E5: POKE $1280,$ED: POKE $1281,$7B: POKE $1282,$0A
```

Beide Interpreter unterstützen zwar das Laden von Maschinenprogrammen auch direkt aus dem Programm heraus (hierzu muß der Bereich, den das zu ladende Programm einnimmt, mit LIMIT vom Hauptspeicher abgetrennt werden, dann folgt der normale LOAD-Befehl), aber das programmgesteuerte Abspeichern von Maschinenprogrammen ist nicht möglich. Dies machen die beiden folgenden Programme verfügbar. Das erste benötigt vier Parameter, die Variable AN ist mit dem Anfang, EN mit dem Ende, ST mit der Startadresse und NA$ mit dem Programmnamen zu laden. Danach besorgt das Programm die Aufzeichnung auf Kassette. Es läuft unter beiden Interpretern.

```
100 POKE $10C0,1
110 POKE $10D4,AN-INT(AN/256)*256
120 POKE $10D5,INT(AN/256)
130 POKE $10D6,ST-INT(ST/256)*256
140 POKE $10D7,INT(ST/256)
150 L=EN-AN
160 POKE $10D2,L-INT(L/256)*256
170 POKE $10D3,INT(L/256)
180 FOR I=1 TO LEN(NA$)
190 POKE 4288+I,ASC(MID$(NA$,I,1))
200 NEXT I
210 POKE 4288+I,13
220 POKE $01BD,$C9
230 USR($01B5)
240 POKE $01BD,$C3
```

Das zweite Programm benötigt drei Parameter, gegenüber dem vorherigen entfällt ST. Die Angabe einer Startadresse ist also hier nicht möglich. Es zeichnet ein Maschinenprogramm auf Diskette auf. Der SAVE-Befehl kann der normalen Syntax entsprechen, also mit oder ohne Laufwerksangabe, Diskettennummer usw.

```
100 L=EN-AN
110 POKE $4225,$C3
120 POKE $4226,$39
130 POKE $4227,$42
140 POKE $423D,$01
150 POKE $4244,$21
160 POKE $4245,AN-INT(AN/256)*256
170 POKE $4246,INT(AN/256)
180 POKE $4247,$22
190 POKE $4248,$8B
200 POKE $4249,$4D
210 POKE $424A,$21
220 POKE $424B,L-INT(L/256)*256
230 POKE $424C,INT(L/256)
240 POKE $426B,AN-INT(AN/256)*256
250 POKE $426C,INT(AN/256)
260 SAVEFD2 NA$
270 POKE $4225,$CD
280 POKE $4226,$10
290 POKE $4227,$1C
300 POKE $423D,$02
310 POKE $4244,$11
320 POKE $4245,$5C
330 POKE $4246,$67
340 POKE $4247,$2A
350 POKE $4248,$4E
360 POKE $4249,$65
370 POKE $424A,$AF
380 POKE $424B,$ED
390 POKE $424C,$52
400 POKE $426B,$5C
410 POKE $426C,$67
```

An dieser Stelle kurz eine Erklärung zum Unterschied LOAD und RUN bei Maschinenprogrammen. LOAD läd das angegebene Maschinenprogramm nur in den Speicher und ignoriert die Startadresse, die Abarbeitung des BASIC-Programms wird fortgesetzt. Falls gewünscht, muß das Maschinenprogramm mit USR gestartet werden. RUN dagegen läd ein Maschinenprogramm und führt es sofort von der bei der Aufzeichnung angegeben Startadresse an aus. Es ist durchaus möglich, Maschinenprogramme von Diskette (nur) durch RUN in vom Interpreter belegte Bereiche zu laden, da dieser das Maschinenprogramm erst direkt vor der Ausführung an die richtige Stelle kopiert.

Um den Komfort noch auf die Spitze zu treiben, kann bei der Aufzeichnung auf Kassette auch noch die Meldung WRITING ... mit POKE $05CF,$C9 (alt $CD) unterdrücken. Damit entfällt auch FOUND ... und LOADING. Beim LOAD-Befehl allerdings kommen diese Meldungen aus dem Interpreter und müssen in BASIC SB-5510 mit POKE $3D34,$C9 (alt $CD) und in BASIC SB-6510 mit POKE $4F61,$C9 (alt $CD) unwirksam gemacht werden. Schließlich kann man mit folgendem Programm auch noch ermitteln, ob und wie eine Kassette im Laufwerk liegt.

```
10 INP§ 225,A: A=INT(A/16)-INT(A/64)¥4
20 IF A=0 THEN PRINT "Kassette ist eingelegt, kein Schreibschutz vorhanden."
30 IF A=1 THEN PRINT "Kassette ist eingelegt, Schreibschutz ist gesetzt."
40 IF A=2 THEN PRINT "Kassettenfachklappe ist nicht richtig geschlossen."
50 IF A=3 THEN PRINT "Keine Kassette eingelegt."
```

Zum Schluß zu den verblüffendsten Programmen. Die beiden folgenden ermöglichen die übernahme einer beliebigen BASIC-Zeile des gerade laufenden Programms in die Variable Z$. Dazu muß in NR die gewünschte Zeilennummer stehen. Existiert die Zeile nicht, enthält Z$ nur CR.

BASIC SB-5510

```
100 POKE $4B73,NR-INT(NR/256)¥256
110 POKE $4B74,INT(NR/256)
120 POKE $4B75,NR-INT(NR/256)¥256
130 POKE $4B76,INT(NR/256)
140 POKE $1CE1,$C8
150 POKE $1D03,$C8
160 POKE $49CD,$0D
170 USR($1CD9)
180 POKE $1CE1,$28
190 POKE $1D03,$11
200 Z$="": ZE=18893
210 Z$=Z$+CHR$(PEEK(ZE))
220 IF PEEK(ZE)<>13 THEN ZE=ZE+1: GOTO 210
```

BASIC SB-6510

```
100 POKE $61B3,NR-INT(NR/256)¥256
110 POKE $61B4,INT(NR/256)
120 POKE $61B5,NR-INT(NR/256)¥256
130 POKE $61B6,INT(NR/256)
140 POKE $1E15,$C8
150 POKE $1E37,$C8
160 POKE $600D,$0D
170 USR($1E0D)
180 POKE $1E15,$28
190 POKE $1E37,$11
200 Z$="": ZE=24589
210 Z$=Z$+CHR$(PEEK(ZE))
220 IF PEEK(ZE)<>13 THEN ZE=ZE+1: GOTO 210
```

Die beiden letzten Programme schließlich stellen praktisch das Gegenteil zum vorherigen Programm dar. Sie schreiben die in Z$ enthaltene BASIC-Zeile in das laufende Programm. Dabei existieren alle Möglichkeiten der normalen Programmeingabe, enthält also Z$ nur eine Zeilennummer, wird die entsprechende Zeile gelöscht, enthält Z$ eine Zeile mit einer existierenden Nummer, so wird diese durch die in Z$ ersetzt, und enthält Z$ eine neue Zeile, so wird diese eingefügt. Fehlt die Zeilennummer, wird die Zeile ignoriert. Diese beiden Programmen dürfen nur und ausschließlich auf ihnen folgende Zeilennummern angewendet werden, ansonsten kommt es zum Programmabsturz! Aus diesem Grund sollten sie als Unterprogramm möglichst am Anfang Ihres Programms stehen.

BASIC SB-5510

```
100 FOR I=1 TO LEN(Z$)
110 POKE 18892+I,ASC(MID$(Z$,I,1))
120 NEXT I
130 POKE 18892+I,13
140 POKE $12ED,$C9
150 POKE $1325,$C9
160 POKE $12EC,$14
170 USR($12E0)
180 POKE $12ED,$21
190 POKE $1325,$C3
200 POKE $12EC,$0E
```

BASIC SB-6510

```
100 FOR I=1 TO LEN(Z$)
110 POKE 24588+I,ASC(MID$(Z$,I,1))
120 NEXT I
130 POKE 24588+I,13
140 POKE $132C,$C9
150 POKE $1365,$C9
160 POKE $132B,$15
170 USR($131F)
180 POKE $132C,$21
190 POKE $1365,$C3
200 POKE $132B,$0F
```

***