Folge 5

6.9.89

Hallo Spectrum-User!

Diesmal muß mein Beitrag aus Zeitgründen leider etwas Kürzer ausfallen als das letzte Mal! Zur Erinnerung: es geht um Einzelheiten des BETA-Disk-Betriebssystems Version 5.03.

Zuerst ein paar Hinweise zum Kopieren von Disketten. Mir ist neulich beim Anlegen einer Kopie der 'Utility-Disk' aufgefallen, daß nach Verwendung des Befehls 'COPY "b:*","a:"' eins der kopierten Programme nicht ordentlich lief! Da mit diesem Befehl die ganze Disk in einem Rutsch von Laufwerk "a" nach "b" überspielt wird und ich dies schon häufig für Sicherheitskopien ohne jede Schwierigkeit gemacht habe, hatte ich eigentlich keine Probleme erwartet.

Die Schwierigkeiten entstanden anscheinend durch unterschiedliche Formatierung der beiden Disketten! Die 'Utility Disk' war als 80 Tracks single sided und die Zieldisk 80 Tracks double sided formatiert! Wieso dies nur bei einigen Programmen zu Problemen führt, habe ich bisher noch nicht ergründet. Darum nur Disks mit gleichem Format verwenden hierbei!

Aber es klappte dann mit dem Befehl 'COPY b', der Kopier-Routine für komplette Disks mit nur einem Laufwerk. Das erfordert zwar etwas mehr Hantieren mit den beiden Disketten, aber da alles menugesteuert läuft, geht's trotzdem schnell und zuverlässig!

Noch eine Anmerkung zum Umgang mit dem Utility-Programm 'doctor'! Als erster Menu-Punkt sollte nach dem Laden oder Diskwechsei immer >(8) Change disk< angewählt werden, damit das korrekte Disk-Format erkannt wird.

Sonst kann's bei anderen Menu-Punkten zu Fehlfunktionen kommen, wenn die zu untersuchende Disk ein von der Utility-Disk abweichendes Format hat. Eventuelle Fehlfunktionen sind darauf zurückzuführen, daß die meisten Menu-Punkte per MC-Programm über einzelne, nummerierte Funktionen des TRDOS aufgerufen werden, die allerdings oft gewisse Vorarbeit voraussetzen (Register setzen, Aufruf vorbereitender Funktionen usw.)! Dazu ein anderes Mal mehr!

Nun will ich noch erläutern, wie das TRDOS herausfindet, wohin zur Ausführung eines Befehls gesprungen werden muß! Wie bekannt sein dürfte, sind im 48er-Modus die Befehlsworte nicht voll ausgeschrieben im Programm abgelegt, sondern als platzsparende 1-Byte-Tokens. Das Befehlswort in voller Länge erscheint also nur im Listing bzw. im Editorbereich auf dem Bildschirm.

Wie schon in Folge 1 angedeutet, ist die Eingabe im TRDOS in Klein- und Großschrift oder gemischt (auch mit Leerzeichen) möglich. Das dies mit einem Trick im 48er-Modus geht, habe ich damals schon erwähnt. Die kompletten Routinen des TRDOS dafür zu beschreiben, ist etwas zu umfangreich.

Nur soviel dazu: die Tabelle mit den möglichen Befehlsworten ist von $30FD-$31C7 zu finden. Hierin wird durch Vergleichen mit dem eingegebenen Befehl ermittelt, das wievielte Befehlswort in der Tabelle dem gesuchten entspricht.

Diese Zahl wird im C-Reg. festgehalten falls das Wort gefunden wurde! Dann kann mit Hilfe der Token-Tabelle von S31C8-S31F2 nach dem Schema 'Tabellenanfangsadr. + C-Reg.-Inhalt' das entsprechende Token ermittelt werden. Die Vergleichs- und Suchroutinen dazu stehen im Bereich von $3032-$30E0?

In beiden Modi findet man zum Schluß jedenfalls immer die Token-Form vor, da nur diese vom Rechner weiterverarbeitet werden kann! Wie's dann weitergeht, soll nun für die Direkteingabe gezeigt werden. Dabei steht das Token an der Adresse, auf die die Systemvariable ELINE (23641) zeigt.

Die Auswerteroutine beginnt bei Adresse $02EF. Hier landet man immer, wenn nach dem Einschalten oder dem 'RANDOMIZE USR 15616' der SPECTRUM eine Eingabe erwartet!

```
02EF            0010            org    $02EF
02EF CD9F1D     0020            CALL $1D9F          ;Fuehrt zu RST $20-Aufruf und damit zu ROM-Adr. $0D6E!
                0030 ;Diese ROM-Routine loescht den Editor-Bereich des Screens und erwartet
                0040 ;dann die Befehlseingabe.
02F2 21CB02     0050            LD     HL,$02CB     ;Diese Adresse wird spaeter als Sprungadresse gebraucht
02F5 221A5D     0060            LD     ($5B1A),HL   ;(bei $0200) und im TRDOS-Puffer deponiert.
                0070 ;
                0080 ;Adr.$02CB bewirkt bei Direktoodus u.a. Rueckkehr zur Eingabeschleife!
                0090;
02F8 AF         0100            XOR    A            ;A-Reg. auf 0 setzen
02F9 320F5D     0110            LD     ($590F),A    ;und TRDOS-Systenvariable ' Fehlernummer' loeschen.
02FC 2A595C     0120            LD     HL,($5C59)   ;HL-Reg. Bit Adr. aus ELINE  laden! Zeigt auf dasToken!
02FF E5         0130            PUSH   HL           ;Diese Adresse auf den Stapel.
0300 11205D     0140            LD     DE,$5D20
0303 CDB002     0150            CALL   $02B0        ;= 3 Zeichen ab dieser Adr.nach Adr. ab $5D20 laden!
0306 E1         0160            POP    HL           ;Adresse von Stapel zurueckholen.
0307 22115D     0170            LD     ($5D11),HL   ;und in $5D11 laden als zu bearbeitende Adresse!
030A 7E         0180            LD     A,(HL)       ;Zeichen aus Adr. auf die HL zeigt, ins A-Reg. bringen
030B 47         0190            LD     B,A          ;und in B-Reg. retten. A bzw B sollten nun das Token enthalten
030C E680       0200            AND    $80          ;Bit 0-6 ausblenden, = testen ob Token-Nr. >= 128 ist.
030E 78         0210            LD     A,B          ;A-Reg. mit urspruenglichen    Wert laden (=Token!
030F 2809       0220            JR     Z.L031A      ;Token-Code <128? (= Befehle *,4.8) Dann sofort nach $031A!
0311 FEFE       0230            CP     $FE          ;Wenn nicht, auf $FE pruefen (= RETURN)!
0313 2805       0240            JR Z.L031A          ;Ja? Dann auch nach $031A zum Vergleich und Adressensuche!
0315 F5         0250            PUSH AF             ;Bei allen anderen Tokens dieses erst mal auf Stapel retten.
0316 CDC83D     0260            CALL   $3DC8        ;Wahrscheinlich Routine zur Laufwerksvorbereitung aufrufen!
0319 F1         0270            POP    AF           ;Wert des Token zurueck holen vom Stapel.
031A 21F32F     0280    L031A LD       HL,$2FF3     ;HL-Reg. mit Anfangsadr. der Tokentabelle laden!
031D 2B         0290            DEC    HL           Diese um 1 vermindern (= $2FF2).
031E 0E00       0300            LD     C,$00        ;C-Reg. loeschen
0320 OC         0310    L0320 INC      C            ;und um 1 erhoehen. Dient als Zaehler fuer Zaehlschleife!
0321 57         0320            LD     D,A          ;Nun Token-Code ins D-Reg. retten
0322 3E15       0330            LD     A,$15        ;und A-Reg. Bit 21 (= Anzahl der Tokens) laden.
0324 B9         0340            CP     C            ;Sind alle Tokens geprueft? Dann ist C auf 22 hochgezaehlt!
0325 DAD301     0350            JP     C,$01D3      Dann ist C-Flag gesetzt und kein gueltiges Token gefunden!
                0360 ;
                0370 ;Ist dies der Fall, wird nach $01D3 gesprungen. Damit geht's nach
                0380 ;einigen Vorbereitungen bei Direktmodus in die Eingabeschleife zurueck
                0390 ;ueber die in $5D1A gerettete Adresse ($02CB)! Bei Abarbeitung eines
                0400 ;BASIC-Prog. ginge es dagegen ueber Adr.$0201 zun Programmabbruch!
                0410 ;
0328 7A         0420            LB     A,D          ;Sonst Code des Token wieder ins A-Reg. zurueckladen
0329 23         0430            INC    HL           ;und Adresse der Token-Tabelle um 1 erhoehen!
032ABE          0440            CP     (HL)         ;Entspricht A-Reg.-Inhalt dem Token-Code in der Tabelle?
032B 20F3       0450            JR     NZ.L0320     ;Wenn nicht, weitersuchen!
032D FEFE       0460            CP     $FE          ;Falls ja, war der Token-Code  $FE? (= RETURN)
032F C44A29     0470            CALL   NZ,$294A     ;Nein? Dann Puffer erweitern! (siehe Folge  4)
0332 3E09       0480            LD     A,$09        ;A-Reg. mit 9 laden, da (RETURN) 9. Befehl in Tokenliste???
0334 32065D     0490            LD     ($5D06),A    ;und in TRDOS-Puffer ablegen.
0337 AF         0500            XOR    A            ;A-Reg. loeschen (= 0)
0338 320F5D     0510            LD     ($5D0F),A    ;und die folgenden TRDOS-Variablen damit laden!
033B 32D65C     0520            LD     ($5CD6),A
033E 32105D     0530            LD     ($5D10),A
0341 213B5C     0540            LD     HL,$5C3B     ;Adr. von FLAGS in HL (=Flags zur BASIC-Systensteuerung)
0344 CBBE       0550            RES    7,(HL)       ;Bit  7 von       FLAGS auf 0! (=Syntaxpruefung anmerken)
0346 0600       0560            LD     B,$00        ;B-Reg.loeschen
0348 210830     0570            LD     HL,$3008     ;und HL mit Anfang der Einsprungadressentabelle laden!
034B 0B         0580            DEC    C            ;C-Reg.-Inhalt um 1 vermindern!
                0590 ;
                0600 ;Dies ist noetig, da zu Beginn die Zaehlschleife fuer die Token-Nr.
                0610 ;mit 1 begann (siehe Adr. $0320). Nun muss jedoch ab 0 gezaehlt werden
                0620 ;wie in Folgenden klar werden duerfte!
                0630 ;
034C CB21       0640            SLA C               ;Wert aus C-Reg. arithmetisch links schieben!
                0650 ;
                0660 ;Dadurch wird der Wert verdoppelt. Notwendig, weil jede der nun zu
                0670 ;ermittelnden Adressen 2 Bytes beansprucht!
                0680 ;
034E 09         0690            ADD    HL,BC        ;Nun ermittelten Wert zu Tab.-Anfangsadr. addieren
034F 5E         0700            LD     E,(HL)       ;und 1. Byte aus der gefunden Adr. ins E-Reg.umladen.
0350 23         0710            INC    HL           ;HL auf naechste Adr. und 2. Byte anschliessend in D laden!
0351 56         0720            LD     D,(HL)       ;DE enthaelt nun die Adr. der Routine zun jeweiligen Token!
0352 EB         0730            EX     DE,HL        ;Reg.-Inhalte vertauschen!
                0740 ;
                0750 ;Nun steht in HL die gefundene Einsprungadresse!
                0760 ;
0353 E5         0770            PUSH   HL           ;Diese auf den Stapel retten.
0354 115903     0780            LD     DE,L0359     ;$0359 dient nachher als Rueckesprungadresse,
0357 D5         0790            PUSH   BE           ;daher auch diese auf den Stapel ablegen!
0358 E9         0800            JP     HL           ;Nun zur gefundenen Adr. springen und Syntaxcheck durchfuehren!
                0810 ;
                0820 ;Vor der eigentlichen Befehlsausfuehrung wird von SPECTRUM  immer ein
                0830 ;Syntaxcheck durchgefuehrt, wenn Bit 7 von FLAGS geloescht ist! Danach
                0840 ;geht's zurueck zur letzten Adr. von Stapel, also hier $0359!
                0850 ;
0359 213B5C     0860    L0359 LD       HL,$5C3B     ;Adresse von FLAGS ins HL-Reg. laden
035C CBFE       0870            SET    7,(HL)       ;und Bit 7 von FLA5S auf 1! (= Befehl ausfuehren anmerken)
035E El         0880            POP    HL           ;Nun Einsprungadresse von Stapel zurueckholen
035F E9         0890            JP     HL           ;und diese nun zur endgueltigen Ausfuehrung anspringen!

#          61A8    L031A    031A    L0320     0320     L0359     0359
```

Hier folgt nun die zum Vergleich mit dem Token erforderliche Tabelle mit den im TRDOS erlaubten Tokens und daran anschließend die Tabelle mit den jeweiligen Einsprungadressen. Bemerkenswert an der Token-Liste ist vielleicht noch, daß darin auch 3 im normalen SOS unbekannte Befehle enthalten sind (siehe dazu auch Folge 1, aus Info 6/89, Seite 3):

1.) '*' = Befehl zum Umstellen auf ein anderes Hauptlaufwerk
2.) '4' = Laufwerk auf 40 Tracks umstellen, wenn möglich
3.) '8' = Laufwerk auf 40 Tracks umstellen, wenn möglich

Im übrigen werden ja die normalen Tokens verwendet im TRDOS!

```
2FF3          0010        org $2FF3    ;Anfangsadr der Token-Tabelle
              0020 ;
              0030 ;    Token-Code      entsprechender Befehl
              0040 ;
2FF3          0050        DB   $CF       ;CAT
2FF4          0060        DB   $2A       ;#
2FF5          0070        DB   $80       ;FORMAT
2FF6          0080        DB   $B1       ;MOVE
2FF7          0090        DB   $E6       ;NEW
2FF8          0100        DB   $D2       ;ERASE
2FF9          0110        DB   $EF       ;LOAD
2FFA          0120        DB   $F8       ;SAVE
2FFB          0130        DB   $FE       ;RETURN
2FFC          0140        DB   $BE       ;PEEK
2FF0          0150        DB   $F4       ;POKE
2FFE          0160        DB   $D5       ;MERGE
2FFF          0170        DB   $F7       ;RUN
3000          0180        DB   $D3       ;OPEN#
3001          0190        DB    D4       ;CLOSE#
3002          0200        DB   $FF       ;COPY
3003          0210        DB   $34       ;40
3004          0220        DB   $EC       ;G0T0
3005          0230        DB   $38       ;80
3006          0240        DB   $F0       ;LIST
3007          0250        DB   $06       ;VERIFY
              0260 ;
              0270 ;Ab $3008 folgt dann die Tabelle mit den zugehoerigen
              0280 ;Einsprungadressen in LOW/HIGH-Notation!
              0290 ;
              0300 ; Adresse (l/h)    ;Befehl     ;Adr.(h/D
              0310 ;
3008          0320        DB   $33,$04    ;CAT        ;$0433
300A          0330        DB   $18,$10    ;*          ;$1018
300C          0340        DB   $C2,$1E    ;FORNAT     ;$1EC2
300E          0350        DB   $AB,$16    ;MOVE       ;$16AB
3010          0360        DB   $3A,$05    ;NEW        ;$053A
3012          0370        DB   $87,$07    ;ERASE      ;$0787
3014          0380        DB   $15,$18    ;LOAD       ;$1815
3016          0390        DB   $DO,$1A    ;SAVE .     ;$1AD0
3018          0400        DB   $FB,$1C    ;RETURN     ;$1CFB
301A          0410        DB   $A5,$19    ;PEEK       ;$19A5
301C          0420        DB   $A9,$19    ;POKE       ;$19A9
301E          0430        DB   $B1,$19    ;MERGE      ;$19B1
3020          0440        DB   $4D,$1D    ;RUN        ;$1D4D
3022          0450        DB   $82,$21    ;OPEN#      ;$2182
3024          0460        DB   $56,$26    ;CLOSE#     ;$2656
3026          0470        DB   $90,$06    ;COPY       ;$0690
3028          0480        DB   $97,$29    ;40         ;$2997
302A          0490        DB   $A1,$2D    ;GOTO       ;$2DA1
302C          0500        DB   $AE,$29    ;80         ;$29AE
302E          0510        DB   $CE,$11    ;LIST       ;$11CE
3030          0520        DB   $10,$18    ;VERIFY     ;$1810

#     61A8
```

Das war's dann mal wieder für heute! Bis demnächst!

MfG
Wilhelm

Folge 6

8.10.89

Hallo Spectrum-User!

Wie auch in den letzten Monaten, so will ich auch diesmal wieder einige Details des BETA-Disk-Betriebssystems beschreiben. Heute geht es erst noch mal um 2 sehr oft benutzte Routinen des TRDOS! Nächstes mal soll dann eine Beschreibung folgen, wie man die BETA-Disk-Befehle aus MC-Programmen heraus aufrufen kann. Die Informationen des Handbuchs sind gerade zu diesem Punkt nicht besonders gut. Das TRDOS beinhaltet nämlich spezielle Funktionen dafür, deren Existens leider gänzlich verschwiegen wird!

Mit besonderem Interesse habe ich im Oktober-Info (10/89) den Artikel einer Beta-Disc-Nutzerin gelesen! Es gibt also doch noch User, die sich mal dazu aufraffen können, etwas für die im Allgemeinen vernachlässigten BETA-Disk-User zu tun. Bitte weiter so!

Nun ein Hinweis zum von der Userin geschilderten Problem mit dem doppelten Zeilenvorschub beim Listing-Ausdruck: Da wird vermutlich ein Vorschub (Line-feed) vom Treiberprogramm aus softwaremäßig ausgelöst und ein weiterer hardwaremäßig vom Drucker. Der Zweite erzeugt dann zwangsläufig die unerwünschte Leerzeile. Um Abhilfe zu schaffen, muß meiner Meinung nach im Drucker mit Hilfe der DIP-Schalter der automatische Zeilenvorschub auf 'OFF' geschaltet werden (dazu bitte im Druckerhandbuch nachsehen, welche Schalter in Frage kommen)!

Bei meinem Drucker werden die entsprechenden Schalter 'Auto feed on/off' genannt, bei anderen Druckern findet man auch mal die Bezeichnung 'CR-Funktion'! Bei meinem Kempston-Centronics- Interface gibt es einen speziellen Befehl, um softwaremäßig den automatischen Zeilenvorschub an- oder abzustellen. Vielleicht existiert dieser auch beim SPECTRUM + 2A? Dann erübrigt sich das manuelle Umschalten natürlich!

Nun zum heutigen Thema: Zuerst nehmen wir uns eine wichtige Routine (Aufrufadr. $0405) vor, die von nahezu allen TRDOS-Befehlen verwendet wird und zwar jedesmal, wenn ein Zugriff auf den Organisationssektor notwendig ist! Dabei wird dann auch das jeweilige Diskformat ermittelt und die entsprechenden TRDOS-Systemvariablen eingestellt.

Wozu der Org.-Sektor gebraucht wird und was er für wichtige Informationen enthält, habe ich schon in Folge 3 beschrieben, übrigens handelt es sich bei der Routine ab Adr. $0405 um eine der oben erwähnten Funktionen. Diese haben alle Nummern von 0 - 24, wobei unsere die Funktion 24 ist. Sie wird auch vom Disk-Doktor-Programm unter dem Namen 'Change disk' aufgerufen, was ja auch schon einiges über die Aufgabe aussagt!

Wie aus dem Assemblerlisting ersichtlich, beginnt die Beschreibung schon bei Adresse $03E8, da diese Teile von der Funktion 24 mitbenutzt werden. Wenn der Einsprung jedoch direkt bei dieser Adresse erfolgt, wird nur der Katalog-Anfang (Trk O/Sek 0) geladen. Dagegen kann beim Aufruf über Adresse $03E9 nach Übergabe der gewünschten Sektor-Nummer ans A-Register jeder Sektor von Track 0 in den erweiterten Puffer eingelesen werden!

Das Gleiche kann auch über Adresse $03EC erledigt werden, wobei allerdings die Sektornummer vorher in die TRDOS-Systemvariable $5CCC geschrieben wird. Auch diese Möglichkeiten werden vom TRDOS des öfteren benutzt, gewöhnlich, um Katalog-Sektoren zu laden. Auch eine schon früher ausführlich erklärte Routine (in Folge 4) wird hier verwendet. Es handelt sich um die unvermeidliche Puffererweiterung (Aufrufadr. $294A).

Außerdem werden noch zwei der Funktionen aufgerufen. Einmal ist dies die in Zeile 470 indirekt aufgerufene Funktion 13, die ich gleich noch näher beschreibe. Dann wird noch in Zeile 250 Funktion 5 aufgerufen, die einen vorher angegebenen Sektor von Disk einliest! Diese im Detail zu beschreiben ist mir leider bisher noch nicht möglich, da ich noch nicht alle entscheidenden Routinen durchschaue! Außerdem werden eine Vielzahl von Unterprogrammen benutzt, so daß alles sehr lang und unübersicht1 ich ist! Hat sich vielleicht jemand mit diesen Routinen schon mal näher beschäftigt (eventuell auch bei den älteren BETA-Versionen)? Dann bitte melden!

Nun endlich das kommentierte Listing:

```
03E8            0010            org $03E8
                0020 ;
03C0            0030 L03C0 equ $03C0            ;Adr. enthaelt: JP $01D3 (= Befehlabarbeitung beenden)
                0040 ;
                0030 ;Bei $03E8 erfolgt Einsprung, wenn nur Trk 0/Sek 0 gelesen werden soll
                0060 ;
03E8 AF         0070            XOR A           ;A-Reg. loeschen
                0080 ;
                0090 ;Falls ein anderer Sektor von Track 0 gelesen werden soll, muss der
                0100 ;Einsprung bei $03E9 erfolgen. Dazu muss die Sektornummer vorher
                0110 ;ins A-Reg. geladen werden.
                0120 ;
03E9 32CC5C 0130                LD ($5CCC),A ;Systemvar. fuer zu ladenden Track auf 0 stellen!
                0140 ;
                0150 ;$5CCC und $5CCD = Zaehler  fuer zu ladenden Sektor bzw. Track
                0160 ;
03EC ED5BCC5C   0170            LD    DE,($5CCC)    ; Track- und Sektornummer nun ins DE-Reg. laden
03F0 1600       0180            LD    D,$00         ;und D-Reg. auf 0, da hier nur Track 0 geladen werden soll!
03F2 CD4A29     0190 L03F2      CALL   $294A        ;Puffererweiterungsroutine aufrufen (siehe Folge 4)!
03F5 21255D     0200            LD    HL,$5D25      ;Adr. des normalen Pufferbereich-Endes ins HL-Reg.
                0210 ;
                0220 ;Nach $5D25 soll nun der gewuenschte Sektor von Disk kopiert werden!
                0230 ;
03F8 0601       0240            LD    B,$01         ;und B-Reg. mit Anzahl der zu ladenden Sektoren beschreiben!
03FA C33D1E     0250            JP $1E3D            ;Nun Funktion 5 aufrufen (= Sektor von Disk einlesen)!
                0260 ;
                0270 ;Eine ausfuehrliche Beschreibung der Funktion wuerde zu lang geraten.
                0280;Es gibt bei der BETA-Version 5.03 insgesamt 24 Funktionen, die aus
                0290 ;MC-Programmen heraus aufgerufen werden. Aber dazu demnaechst mehr!
                0300 ;
03FD CB4A29     0310 L03FD   CALL $294A             ;Puffererweiterungsroutine aufrufen (siehe Folge 4)!
0400 110800     0320            LD DE,$0008         ;D-Reg. mit Track-, E-Reg. mit Sektornummer laden.
                0330 ;
                0340 ;In DE steht nun Trk -(0) und Sek.-Nr. (8) des Organisationssektors!
                0350 ;
0403 18ED       0360            JR L03F2            ;Zum Laden des Sektors von Disk in den erweiterten Puffer!
                0370 ;
0405 CDFD03     0380            CALL L03FD          ;Hier beginnt die Funktion 24 (=Change disk)!
                0390 ;
                0400 ;Nachdem der Org.-Sektor in den Puffer geladen ist, hier weiter:
                0410 ;
0408 3A0C5E     0420            LD    A,($5E0C)     ;Sektorenzahl/Trk aus Org.-Sektorkopie ins A-Reg. laden!
040B FE10       0430            CP    $10           ;Steht dort der Wert 16, wie beim BETA-System erforderlich?
040D 2806       0440            JR    Z,L0415       ;Ja? Dann ist es o.k.! Weiter bei $0415.
040F 21E229     0450            LD    HL,$29E2      ;Nein? Dann HL mit Anfangsadr. von Text >Disk  Error< laden
0412 DF         0460            RST    $18          ;und diesen per RST $18-Routine ausgeben!
0413 18AB       0470            JR    L03C0         ;Ueber diese Adr. erfolgt ein Sprung nach $01D3 = Beenden!
                0480 ;
                0490 ;Nachdem die Vorbereitungen beendet sind, wird nun das eigentliche
                0500 ;>Change disk< vorgenommen!
                0510;
0415 CD113E     0520 L0415     CALL $3E11           ;=>Diskformat beim gewaehlten Laufwerks ermitteln!
                0530 ;
                0540 ;HL zeigt nun fuer Seite 0 der Disk auf eine der Systemvariablen
                0550 ;$5CC8-$5CCB! Als Defaultwert steht da bei 40 Trk > $00, 80 Trk > $80.
                0560 ;Sonst  moegliche Werte: $02, $81, $83, je nach geladener Diskversion!
                0570 ;
0418 CB86       0580            RES    0,(HL)       ;Bit 0 und 1 bei der ermittelten Systemvariablen auf 0!
041A CB8E       0590            RES    1,(HL)       ;Damit fruehere Werte loeschen!
041C 3A085E     0600            LD    A,($5E08)     ;Nun Wert fuer Disk-Format aus Org.-Sektorkopie holen.
                0610 ;
                0620 ;Folgende  Werte kommen in Frage:
                0630 ;bei     80  Tracks  /  single sided  ->   $18     =    00011000
                0640 ;      " 80      "         double  •     ->   $16     =    00010110
                0650 ;      " 40      "         single  •     ->   $19     =    00011001
                0660 ;      " 40      "         double  •     ->   $17     =    00010111
                0670 ;
041F CB47       0680            BIT    0,A          ;Bit 0 von Disk-Formatwert gesetzt? Dann Z-Flag nicht gesetzt!
0421 2002       0690            JR    NZ,L0425      ;Dann ist es eine Disk mit 40 Tracks! Weiter bei $0425!
0423 CBC6       0700            SET    0,(HL)       ;Sonst Bit 0 setzen, da 80 Tracks.
0425 CB5F       0710 L0425      BIT    3,A          ;Bit 3 von Formatwert gesetzt? Dann Z-Flag nicht gesetzt!
0427 C0         0720            RET    NZ           ;Deshalb zurueck, da 'single sided' formatierte Disk!
0428 CBCE       0730            SET    1,(HL)       ;Andernfalls Bit 1 setzen fuer 'double sided',
042A C9         0740            RET                 ;und zurueck!
                0750 ;
                0760 ;Aus den Default-Werten $00 / $80 in $5CC8 - $5CCB ist nun geworden:
                0770 ;
                0780 ;bei     40  Tracks  /  single   sided     ->        $00 =     00000000
                0790 ;      " 40      "         double       "     ->        $02 =     00000010
                0800 ;      " 80      "         single       "     ->        $81 =     10000001
                0810 ;      " 80      "         double       "     ->        $83 =     10000011
```

Die folgende Routine wird auch häufig gebraucht, hauptsächlich nämlich beim Abbruch eines BASIC-Programms wegen Fehler bei den TRDOS-Befehlen oder wenn ein Direktaufruf auf einen Fehler stößt. Im ersten Fall wird ins BASIC zurückgekehrt und eine der normalen Fehlermeldungen ausgegeben und damit das laufende Programm unterbrochen. Beim Direktaufruf erscheint dagegen eine der TRDOS-Fehlermeldungen und es wird in die Eingabeschleife des TRDOS (Beschreibung siehe Folge 5) zur Neueingabe oder Korrektur zurückgekehrt! Dazu nun das Listing:

```
01D3              0010          org     $01D3          ;Routine zur Rueckkehr ins BASIC oder TRDOS bei Fehlern!
                  0020 ;
01D3 210000       0030          LD      HL,$0000       ;HL-Reg. loeschen
01D6 22F85C       0040          LD      ($5CF8),HL     ;und $5CF8 damit laden.
                  0050 ;
                  0060 ;$5CF8 dient als Merker, ob Puffererweiterung noch bestehen ($FF) oder
                  0070 ;gelöscht ($00) werden soll! (Vergleiche auch Folge 4,Seite 1)
                  0080 ;
01D9 CDE520       0090          CALL $20E5           ;Da 0, Puffererweiterung loeschen!
01DC CD631D       0100          CALL $1D63           ;Fehler-Nr. in $5D0F pruefen!
                  0110 ;
                  0120 ;Falls $FF (=o.k.), Workspace und Calc.-Stack per ROM-Routine loeschen
                  0130 ;
01DF 21175D       0140          LD      HL,$5D17       ;Merkeradr. fuer zu loeschenden Screenbereich ins HL-Reg.
01E2 36AA         0150          LD      (HL),$AA       ;und diese mit $AA laden (=nur untere 2 Zeilen loeschen)!
01E4 211F5D       0160          LD      HL,$5D1F       ;Adresse der Systemvariablen $5D1F ins HL-Reg. laden.
                  0170 ;
                  0180 ;$5D1F enthaelt nur bei MERGE oder Funktionsaufruf per MC-Prog. $FF,
                  0190 ;da dann die Floatingpoint-Routine uebergangen werden kann! Sonst: $00
                  0200 ;
01E7 7E           0210          LD      A,(HL)         ;Deren Inhalt nach A!
01E8 B7           0220          OR      A              ;Enthaelt A-Reg. 0? Dann ist Zero-Flag gesetzt!
01E9 3600         0230          LD      (HL),$00       (Jetzt $5D1F mit 0 laden
01EB 2006         0240          JR      NZ,L01F3       ;Falls A-Reg. <> 0 war, weiter bei 001F3!
01ED CD1C1E       0250          CALL    $1E1C          ;ROM-Rout., ab $11A7 per RST $20-Befehl aufrufen!
                  0260 ;
                  0270 ;Holt die versteckten Floatingpointzahlen einer BASIC-Zeile zurueck!
                  0280 ;
01F0 CD1202       0290          CALL    $0212          ;Aktuelles bzw  naechstes Zeichen auf $0D (= ENTER) testen!
01F3 ED7B1C5D     0300 L01F3    LD  SP,<$5D1C          ;Alte Stapeladresse ins SP-Reg. laden.
01F7 2A1A5D       0310          LD      HL,($5D1A)     ;Enthaelt die jeweilige Ruecksprungadresse?
                  0320 ;
                  0330 ;Bei Abarbeiten eines BASIC-Prog. steht hier $0201. bei Direktaufruf
                  0340 ;aber $02CB! Dadurch ist unterschiedliche Fehlerbehandlung moeglich!
                  0350 ;
01FA ED460F5D     0360          LD      BC,($5D0F)     ;Adresse fuer Fehlernummer ins BC-Reg.
01FE 0600         0370          LD      B,$00          ;und B-Reg. loeschen! C-Reg. enthaelt jeweilige Fehlernuner.
0200 E9           0380          JP      HL             ;Nun nach $0201 oder $02CB springen!
                  0390 ;
                  0400 ;Bei BASIC-Prog. wird bei Fehler abgebrochen  mit Fehlermeldung des
                  0410 ;SOS, sonst der naechste Programmschritt abgearbeitet!
                  0420 ;Bei Direktaufruf wird bei Fehler dagegen die Fehlermeldung des TRDOS
                  0430 ;ausgegeben und in jeden Fall zur Korrektur oder naechsten Befehls-
                  0440 ;eingabe in die Eingaberoutine zurueckgekehrt!
                  0450 ;
                  460 ;================================================================================

                  0470 ;
                  0480 ;Weiter in Programm oder BASIC-Fehlermeldung ausgeben!
                  0490 ;
0201 CD3202       0500          CALL L0232           ;Alten Wert aus ERR SP-Adr. zurueckholen.
0204 FDCB007E     0510          BIT 7,(IY+$00)       ;ERR NR auf gesetztes 7. Bit testen!
                  0520 ;
                  0530 ;ERR NR enthaelt Meldecode - 1, das heisst, bei Fehlermeldung 0 -> 255
                  0540 ;Dann ist 7. Bit gesetzt (= 1), da kein Fehler! Z-Flag ist deshalb 0,
                  0550 ;also nicht gesetzt!
                  0560 ;
0208 C0           0570          RET     NZ             ;Kein Fehler? Dann weiter im Programm.
0209 11C25C       0580          LD      DE,$5CC2       ;Sonst DE-Reg. mit RAM-Adr.$5CC2 (enthaelt RET)
020C ED7B3D5C     0590          LD      SP,($5C3D)     ;und Stapelzeiger auf Adr. aus ERR SP setzen!
0210 D5           0600          PUSH DE              ;Nun Adr. $5CC2 auf den Stapel legen
0211 C9           0610          RET                  ;und diese anspringen! Entsprechende Fehlermeldung ausgeben.
                  0620 ;
                  0630 ;Bewirkt ein RET zur Fehlerrueckspungadr. auf die (ERR SP), zeigt.
                  0640 ;
                  0650 ;================================================================================
                  0660 ;
                  0670 ;Ende einer Programmzeile suchen!
                  0680 ;
                  0690 ;
0212 CD8C1D       0700  L0212   CALL $1D8C           ;Per RST $20 zu ROM-Adr. $0018. Holt aktuelles Zeichen!
0215 FEOD         0710          CP      $0D            ;Ist es der Code fuer ENTER 1= Zeilenende)?
0217 C8           0720          RET     Z              ;Dann zurueck!
0218 CD2A1E       0730          CALL $1E2A           ;Sonst per RST $20 zu ROM-Adr $0020! Holt naechstes Zeichen
021B 18F5         0740          JR      L0212          ;und weitersuchen, bis Zeilenende gefunden wird!
                  0750 ;
                  0760 ;================================================================================
                  0770 ;
```

```
                0780 ;Hier wird der alte Wert von ERR SP des BASICs gerettet (er weist auf
                0790 ;die Adr., die angibt, wohin bei Fehlermeldung gesprungen werden soll!
                0800 ;und die Fehlerrueckspungadr. nach $3D16 'verbogen'fuer's TRDOS!
                0810;
021D 2A3D5C     0820        LD    HL,($5C3D)    ;Alte Adr. aus ERR SP ins HL-Reg. laden
0220 22135D     0830        LD    ($5D13),HL    ;und in Adr. $5513 retten!
0223 2A1C5D     0840        LD    HL,($5D1C)    ;Zwischengespeicherte Stapeladr. ins HL-Reg. laden
0226 2B         0850        DEC   HL
022? 2B         0860        DEC   HL            ;und diese um  2 erniedrigen!
0228 223D5C     0870        LD    ($5C35D,HL    ;Diesen neuen Wert in Systemvariable ERR SP schreiben!
0225 11163D     0880        LD    DE,$3D16      ;Adr. der Fehlerbehandlungsroutine des TRDOS in DE laden.
                0890 ;
                0900 ;Fehler bei Direktaufruf des TRDOS werden ueber diese Adr. bearbeitet!
                0910 ;
022E 73         0920        LD    (HL),E        ;Ins L-Reg. der Adr. aus ERR SP  das Low-Byte  ($16) bringen,
022F 23         0930        INC   HL            ;HL auf naechste Adr. erhoehen
0230 72         0940        LD    (HL),D        ;Und darin das High-Byte  ($3D)   laden.
0231 C9         0950        RET                 ;Zurueck!
                0960 ;
                0970 ;=============================================================================
                0980 ;
                0990 ;Diese Routine wird nur bei TRDOS-Aufruf aus BASIC gebraucht und
                1000 ;holt die alte Fehlerrueckspungadr. zurueck an Stelle von $3516!
0232 2A135D     1010 L0232  LD HL,($5D13)   ;Alten geretteten  Wert (des BASIC) aus ERR SP in HL!
0235 223D5C     1020        LD ($5C3D),HL   ;zurueckschreibenin Systeavariable ERR SP!
0238 C9         1030           RET          ;Zurueck
                1040 ;
```

Ich habe hier noch 4 unmittelbar folgende kleine Unterprogramme mit kommentiert, da 3 davon ja auch direkt von der 'Rückkehr-Routine' ($01D3-$0200) mitverwendet werden! Aber auch das Unterprogramm zum Verbiegen der Fehlerrueckspungadresse ($021D-$0231) ist sehr wichtig, da dadurch erst die neue Fehlerbehandlung im TRDOS ermöglicht wird! Sie wird bei jedem Aufruf des TRDOS angesprungen und deshalb will ich auch dazu noch die entscheidenden Routinen beschreiben. Vorher noch 2 Anmerkungen, die vielleicht das Verständnis der Routinen erleichtern:

Bei der Systemvariablen ERR SP handelt es sich um einen indirekten Zeiger. Die darin abgelegte Adr. gibt an, wohin beim Auftreten eines Fehlers gesprungen werden soll um eine Fehlermeldung auszugeben. Durch verändern dieser Adr. kann eine neue Fehlerbehandlung erfolgen. Genau dies machte das Programm von $021D-$0231!

Das SP-Register zeigt gewöhnlich auf die letzte auf den Stapel abgelegte Adresse. Dabei handelt es sich meist um Rücksprungadressen oder mit PUSH XX auf den Stapel zwischengespeicherte Adressen. Das SP-Register wird dabei jedesmal um 2 erniedrigt (DECrementiert), weil jede Adresse 2 Bytes beansprucht. Bei RET oder POP XX wird der Stapelzeiger dagegen um 2 erhöht (INCrementiert), da der Stapel von 'oben' nach 'unten' aufgebaut wird!

```
3D16            0010        org $3D16           ;Alle Fehlermeldungen laufen ueber diese Adr. im TRDOS!
                0020 ;
3D16   00       0030        NOP
3D17   C3692F   0040        JP    $2F69         ;Weiter bei $2F69.
                0060 ;=================================================================

2F69            0010        org    $2F69        ;Stapelzeiger neu setzen!
                0020 ;
2F69 2A1C5D     0030        LD    HL,($551C)    ;Geretteten Stapelzeigerwert in HL laden!
                0040 ;
                0050 ;Dies ist die nach Ausfuehrung des RANDOMIZE USR 15616 (oder 15619)
                0060 ;gerade aktuelle Stackadresse!
                0070 ;
2F6C 2B         0080        DEC   HL            ;Um 2 erniedrigen! Damit duerfte er wieder auf Stapeladresse
2F65 2B         0090        DEC   HL            ;zeigen, die $3D16 enthaelt.
2F6E F9         0100        LD    SP,HL         ;Diesen Wert als neuen Stapelzeiger festlegen
2F6F C32F1D     0110        JP    $1D2F         ;und hier weiter!
                0120 ;
                0130 ;=================================================================

1D1A            0010        org $1D1A           ;Fehler in BASIC-Zeile oder Direktaufruf?
                0020 ;
1D1A FDCB007E   0030 L1D1A  BIT   7,(IY+$00)    ;ERR NR auf gesetztes 7. Bit testen!
1D1E 2805       0040        JR    Z,L1D25       ;Bit 7 nicht gesetzt? Dann nach $1D25, da Fehler!
1020 3E0B       0050        LD    A,$0B         ;Meldecode 11 in A. Da diese um 1 kleiner ist als Fehlernr.
1D22 323A5C     0060        LD    ($5C3A),A     ;steht nun in ERR NR Fehlernr. fuer 'Nonsence in BASIC'
1025 3C         0070 L1D25  INC    A            ;Fehlernr. + 1! Damit ist genannte Fehlernr. eingestellt.
1B26 21B229     0080        LD    HL,$29B2      ;Textanfangsadr. von*ERROR* ins HL-Reg. laden.
1D29 CDC303     0090 L1D29  CALL S03C3           ;Zur Fehlermeldungsausgabe oder Programmabbruch!
1D2C C3D301     0100        JP    $01D3         ;Zur Rueckkehr ins BASIC oder TRDOS!
                0110 ;
                0120 ;=================================================================
```

```
                0130;
                0140 ;Meldecode testen!
                0150 ;
1D2F 3A3A5C     0160        LD    A,($5C3A)     ;Fehlernummer ins A-Reg.
1D32 21CA27     0170        LD    HL,$27CA      ;und Textanfangsadr. von *BREAK* ins HL-Reg. laden.
1D35 FE14       0180        CP    $14           ;Meldecode 20? (Entspricht Fehlernr. 21 = BREAK  into Programm)!
1D37 28F0       0190        JR    Z.L1929       ;Dann Fehlermeldungsausgabe oder Abbruch des Programms!
1D39 FE0C       0200        CP    $0C           ;Meldecode 12? (Entspricht Fehlernr. 13 = BREAK-Cont repeats)!
1D3B 28EC       0210        JR    Z,L1D29       ;Dann Fehlermeldungsausgabe oder Abbruch des Programms!
1D3D 21D227     0220        LD    HL,$27D2      ;Textanfangsadr. von 'Out of RAM' in HL laden.
1D40  FE03      0230        CP    $03           ;Meldecode 3? (Entspricht Fehlernr. 4 = Out of Memory)!
1D42 28E5       0240        JR    Z,L1D29       ;Dann Fehlermeldungsausgabe oder Abbruch des Programms!
1D44 21DD27     0250        LD    HL,$27DD      ;Textanfangsadr. von 'Array not found' in HL laden.
1D47 FE01        0260       CP $01              ;Meldecode 1? (Entspricht Fehlernr. 2 = Variable not found)!
1D49 28DE       0270        JR    Z,L1D29       ;Dann Fehlermeldungsausgabe oder Abbruch des Programms!
1D4B 18CD       0280        JR    L1D1A         ;Bei anderen Fehlern nach $1D1A!
                0290 ;
                0300 ;=================================================================

03C3            0010        org   $03C3         ;TRDOS-Fehlermeldungausgabe  oder Programmabbruch?
                0020 ;
03C3 F5         0030        PUSH   AF           ;Fehlernummer retten.
03C4 3A0E5D     0040        LD     A,($5D0E)    ;Wert aus $5D0E ins A-Reg.!
03C7 FEFE       0050        CP     $FE          ;Ist es der Code $FE? (Dieser wird bei Autoboot geladen)!
                0060 ;
                0070 ;Dann Fehlermeldung wie bei Direktaufruf behandeln!
                0080 ;
03C9 2002       0090        JR     NZ,L03CD     ;Nein? Dann weiter bei $03CD!
03CB F1         0100        POP    AF           ;Sonst Fehlernummer zurueckholen
03CC C9         0110        RET                 ;und zurueck nach Adr. $1D2C zur Rueckkehr  ins BASIC!
                0120 ;
                0130 ;Fehlerbehandlung bei Direktaufruf des TRDOS!
                0140 ;
03CD Fl         0150 L03CD  POP    AF           ;Fehlernummer zurueckholen
03CE 320F5D     0160        LD     ($5D0F),A    ;und in TRDOS-Systemvariable fuer Fehlernr. retten.
03D1 3A155D     0170        LD     A,($5D15)    ;TRDOS-Systemvar. fuer Aufruf-Art ins A-Reg. laden
03D4 B7         0180        OR     A            ;und pruefen ob es 0 ist, d.h. es war ein Direktaufruf des TRDOS!
03D5 CC0727     0190        CALL   Z,$2707      ;Dann Text der jeweiligen Fehlermeldung ausgeben!
03D8 C9         0200        RET                 ;Sonst oder danach wieder nach Adr. $1D2C!
```

Das waren diesmal reichlich Listings, aber ich hoffe, daß sie zum besseren Verständnis des BETA 5.03-Betriebssystem wieder etwas beigetragen haben. Übrigens mochte ich nochmals darauf hinweisen, daß alle Angaben ohne Gewähr für die absolute Richtigkeit gemacht werden. Ich bemühe mich jedoch, alles so gut wie eben möglich zu überprüfen und wo Zweifel geblieben sind, merke ich es an! Aber bei so einer komplexen Sache wie der TRDOS-Kommentierung sind Fehler eben nicht ganz auszuschließen.

Bis demnächst!

MfG
Wilhelm

Folge 7

29.10.89

Hallo Spectrum-User!

Heute mal erst etwas für alle, die den Assembler/Disassembler SYS verwenden! Mich störte, daß der Assembler 'MICASS' keine deutschen Sonderzeichen kennt und deshalb habe ich dies nun geändert! Ich verwende die 64 Zeichen/Zeile-Ausführung und da liegt zumindest bei der Version 2.2 der Zeichensatz ab Adr. 56626 (wurde mit dem File >"1NSTL.SCRd" CODE 56626,384< geladen)!

Im Gegensatz zum normalen SPECTRUM-Zeichensatz enthält hier ein 8*8 Raster immer gleich 2 neue Zeichen, sodaß sich der Platzbedarf halbiert. Die Sonderzeichen werden am Besten auf die Codes 91-93 und 123-126 gelegt, da dies auch bei den meisten Druckern mit deutschem Zeichensatz so ist und vielen Usern auch vom deutschen TASWORD II her geläufig sein dürfte.

Wie nun die Adr. des 1. zu ändernden Zeichens, der eckigen Klammer finden? Ganz einfach: das erste Zeichen im MICASS-Zeichensatz ist >!<, also Code 33, die eckige Klammer Code 91! Also 91-33=58, da 2 Zeichen pro Raster: 58/2=29! Weil jedes Raster 8 Bytes belegt nun 29*8=232, dies zur Anfangsadr. addieren und schon ist die Adresse des ersten zu ändernden Rasters gefunden: 56626+232=56858!

Hier steht nun allerdings das >Z< und die gesuchte Klammer an deren Stelle ein >Ä< eingebaut werden soll. Binär sieht das nach Änderung so aus, wobei eine 1 einem Punkt im Zeichen entspricht wie unschwer zu erkennen ist:

Adr.	Dual	Hex	Adr.	Dual	Hex	Adr.	Dual	Hex
56858	11101010	$EA	56866	10101010	$AA			
56859	00100100	$24	56867	01000000	$40			
56860	00101010	$2A	56868	10101010	$AA			
56861	01001010	$4A	56869	10101010	$AA			
56862	10001110	$8E	56870	10101010	$AA			
56863	10001010	$8A	56871	10101010	$AA			
56864	11101010	$EA	56872	01001110	$4E			
56865	00000000	$00	56873	00000000	$00			

Nun sind die Zeichen >A, Ö, Ü< fertig! Die Restlichen haben nun analog zur obigen Berechnung nun bei der Adresse 56986 zu beginnen:

56986	00001010	$0A	56994	10101010	$AA	57002	01001110	$4E
56987	00000000	$00	56995	00000000	$00	57003	10101110	$AE
56988	11101100	$EC	56996	01000000	$40	57004	10101110	$AE
56989	00100010	$22	56997	10101010	$AA	57005	11001110	$CE
56990	01001110	$4E	56998	10101010	$AA	57006	10101110	$AE
56991	10001010	$8A	56999	10101010	$AA	57007	10101110	$AE
56992	11101110	$EE	57000	01001110	$4E	57008	11001110	$CE
56993	00000000	$00	57001	00000000	$00	57009	10000000	$80

Somit sind die Zeichen >ä, ö, ü und ß< auch fertig! Wer das bei seinem MICASS-Programm nachvollziehen will, ruft am Besten mit 'mon.' den Disassembler auf, geht dort in den Monitor-Modus und gibt ab der genannten Startadr. die Hex-Werte ein! Wer's mit POKE machen will, kann sich die Hex-Zahlen anhand beigelegter Umrechnungstabelle umwandeln! Nicht vergessen, das Programm in der neuen Version dann zu Speichern! Die Zeichen sind bei MICASS übrigens im SHIFT-Modus zu erreichen!

Jetzt endlich wieder zur BETA-Version 5.03! Diesmal soll es also wie versprochen um die Funktionen gehen. Wie schon angedeutet, gibt es 24 davon und sie dienen dazu, verschiedene Befehle aus MC-Programmen aufzurufen.

Das Handbuch beschreibt eine Methode, bei der im Grunde die BASIC-Befehlssequenz in MC umgesetzt und dann mit Hilfe der Systemvariablen CHADD angesprochen wird. Diese Methode ist nicht sehr elegant und bei Zahlen im Befehl (wie Startadr., CODE-Länge usw.) treten Schwierigkeiten auf, und zwar, weil die Zahlen in einer REM-Zeile (wie sie ja auch hier nachgebildet wird) als ASCII-Codes abgelegt sind, bei der Abarbeitung hier aber ins Fließkommazahlenformat gebracht werden.

Leider wird dies anschließend nicht automatisch rückgängig gemacht, wie beim normalen TRDOS-Aufruf (siehe Folge 6, ab Adr. $01E4), sodaß spätestens beim 2. Aufruf Probleme auftreten! Die Existenz der Funktionen werden leider ganz verschwiegen und es war sehr mühsam, deren Bedeutung im Einzelnen zu ergründen! Eine große Hilfe beim Entschlüsseln bot jedoch der MC-Teil des Disk-Doctorprogramms auf der Utility-Disk, der auch die Funktionen nutzt! Nach dem Disassemblieren ließen sich schon mal 5 Funktionen und deren Handhabung leicht identifizieren.

Die Schlüsselrolle spielt Adr. $3D13 (15635) im TRDOS-Initiallsierungstei1 (Folge 2), die Einsprungadresse, über die alle Funktionen aufgerufen werden. Grundsätzlich hat die gewünschte Funktionsnummer im C-Register zu stehen, bevor der Aufruf vorgenommen wird.

Meist müssen aber noch weitere Parameter übergeben werden in bestimmte Register und sonstige Vorbereitungen getroffen werden. Wie im Listing in Folge 2 aus Info 8/89 zu sehen, verzweigt das TRDOS nach Aufruf der genannten Adresse per 'JR $3CFD' und gelangt so zum: 'JP $283C'! Genau hier beginnt die Suche nach der Funktionsnummer und Startadresse und letztlich deren Aufruf! Das folgende Programm benutzt auch einige der im letzten Info beschriebenen Routinen, also eventuell dort noch mal nachschauen. Nun das Listing mit zugehöriger Tabelle:

```
283C             0010          org $283C          ;Beginn der Funktionsnummern-Suchroutine!
                 0020;
283C F5          0030          PUSH  AF           ;A-Reg. enthält hier Kanalnr., Laufwerksnr. usw.?
283D C5          0040          PUSH  BC           ;C-Reg. enthält hier Funktionsnr.! AF- u. BC-Reg. retten!
283E EB53045D    0050          LD    ($5D04),DE   ;TRDOS-Systemvar. $5D04 und $5D02 als Ablage für DE-
2842 22025D      0060          LD    ($5D02),HL   ;und HL-Reg.-Inhalt benutzen.
2845 CDF120      0070          CALL  $20F1        ;IF 1 vorhanden? Ja? Dann Parameter setzen, sonst weiter!
2848 3EFF        0080          LD    A,$FF        ;A-Reg. mit $FF laden und
284A 32155D      0090          LD    ($5D15),A    ;dies als Merker für TRDOS-Aufruf aus BASIC in $5B15 und
284D 321F5D      0100          LD    ( $5D1F),A   ;hier ebenso ablegen! (Siehe dazu Folge 6 ab Adr. $01E4)
2850 C1          0110          POP   BC           ;BC-Reg. und
2851 F1          0120          POP   AF           ;AF-Reg. vom Stapel zurück holen.
2852 210102      0130          LD    HL,$0201     ;Rucksprungadr. für TRDOS-Aufruf aus BASIC in HL laden
2855 221A5D      0140          LD    ($5D1A),HL   ;und in TRDOS-Systemvariable $5D1A retten!
2858 210000      0150          LD    HL,$0000     ;Nun HL-Reg. löschen,
285B 39          0160          ADD   HL,SP        ;zur aktuellen Stapeladresse addieren und
285C 221C5D      0170          LD    ($5D1C),HL   ;das Ergebnis als alte Stapeladr. in $5B1C schreiben!
285F 2B          0180          DEC   HL
2860 2B          0190          DEC   HL           ;Jetzt diese um2 vermindern
2861 F9          0200          LD    SP,HL        ;und als neue aktuelle Stapeladresse festlegen!
2862 F5          0210          PUSH  AF           ;AF-Reg. wieder auf den Stapel retten.
2863 CD1D02      0220          CALL  $0210        ;ERR SP anpassen und Error-Routine nach $3D16 'verbiegen'!
2866 218C28      0230          LD    HL,L288C     ;HL mit Anfang der Funktions- und Sprungadr.-Tabelle laden
2869 7E          0240 L2869    LD    A,(HL)       ;und Wert daraus ins A-Reg. bringen.
286A B9          0250          CP    C            ;Entspricht er der gesuchten Funktionsnummer  im C-Reg.'
286B 2012        0260          JR    NZ,L287F     ;Nein? Dann weitersuchen!
286D Fl          0270          POP   AF           ;Gefunden? Dann erst mal AF-Reg. zurückholen
286E 23          0280          INC   HL           ;und HL-Reg. auf nächste Adresse einstellen.
286F 5E          0290          LD    E,(HL)       ;Deren Inhalt (=Low-Byte) ins E-Reg.,
2870 23          0300          INC   HL           ;HL noch eine Adr. weiter zählen und deren Inhalt ins D-Reg.
2871  56         0310          LD    D,(HL)       ;DE enthält nun die Startadr. der gesuchten Funktion!
2872 21D301      0320          LD    HL,$01D3     ;HL mit Adr. der Rückkehrfunktion laden,
2875 E5          0330          PUSH  HL           ;diese auf den Stapel ablegen
2876 B5          0340          PUSH  DE           ;und darunter die Funktions-Startadr.!
2877 2A025D      0350          LD    HL,($5D02)   ;Nun die zu Anfang in den TRDOS-Systemvar. deponierten
287A EB5B045D    0360          LD    DE,($5D04)   ;Werte in die ursprünglichen Reg. zurückschreiben!
287E C9          0370          RET                ;Dies RET führt zur zuletzt auf den Stapel gelegten Adresse!
                 0380 ;
                 0390 ;Dort liegt hier die Startadr. der gewählten Funktion.Diese wird nun
                 0400 ;ausgeführt, danach geht's nach $01D3 und über $0201 zurück ins
                 0410 ;BASIC-Prog., wenn alles stimmt, sonst zu einer BASIC-Fehlermeldung!
                 0420 ;
287F FEFF        0430   L287F CP     $FF          ;War der Code aus derTabelle etwa $FF (=Tab.- Ende!?
2881 2004        0440         JR     NZ,L2887     ;Nein? Dann weiter bei $2887.
2883     F1      0450         POP    AF           ;Sonst erst mal das AF-Reg. vom Stapel holen
2884     C3D301  0460         JP     $01D3        ;und zurück ins BASIC - Prog., da Funktionsnr. nicht gefunden!
2887     23      0470   L2887 INC    HL           ;HL-Reg. insgesamt um 3 erhöhen. Zeigt dann auf nächste
2888     23      0480         INC    HL           ;Funktionsnr. in der Tabelle, da jede Nummer mit der dazu
2889     23      0490         INC    HL           ;gehörenden Startadresse 3 Bytes belegt!
288A 18DD        0500         JR     L2869        ;Nächste Funktionsnr.aus Tabelle vergleichen mit Wert in C!
                 0510 ;
                 0520 ;Hier beginnt die Tabelle der Funktionsnummern und Startadressen.
                 0530;
288C             0540 L288C   DB     $00,$98,$3B      (Funktionnr.: 0   ;Startadresse:   $3D98
288F             0550         DB     $01,$CB,$3D   ;       "      1   ;       "        $3DCB
2892             0560         DB     $02,$63,$3E   ;       "      2   ;       "        $3E63
2895             0570         DB     $03,$02,$3F   ;       "      3   ;       "        $3F06
289B             0590         DB     $05,$3D,$1E   ;       "      5   ;       "        $1D3D
289E             0600         DB     $06,$4D,$1E   ;       "      6   ;       "        $1E4D
28A1             0610         DB     $07,$D8,$28   ;       "      7   ;       "        $28D8
28A4             0620         DB     $08,$5C,$16   ;       "      8   ;       "        $165C
28A7             0630         DB     $09,$64,$16   ;       "      9   ;       "        $1664
28AA             0640         DB     $0A,$F0,$1C   ;       "      10  ;       "        $1CF0
28AD             0650         DB     $0B,$FB,$28   ;       "      11  ;       "        $28FB
28B0             0660         DB     $0C,$F2,$28   ;       "      12  ;       "        $28F2
28B3             0670         DB     $0D,$D3,$01   ;       "      13  ;       "        $01D3
28B6             0680         DB     $0E,$0F,$29   ;       "      14  ;       "        $290F
28B9             0690         DB     $0F,$D3,$01   ;       "      15  ;       "        $01D3
28BC             0700         DB     $10,$D3,$01   ;       "      16  ;       "        $01D3
28BF             0710         DB     $11,$D3,$01   ;       "      17  ;       "        $01D3
28C2             0720         DB     $12,$26,$29   ;       "      18  ;       "        $2926
28C5             0730         DB     $13,$E0,$28   ;       "      19  ;       "        $28E0
28C8             0740         DB     $14,$E3,$28   ;       "      20  ;       "        $28E3
28CB             0750         DB     $15,$39,$27   ;       "      21  ;       "        $2739
28CE             0760         DB     $16,$EB,$1F   ;       "      22  ;       "        $1FEB
28D1             0770         DB     $17,$F6,$1F   ;       "      23  ;       "        $1FF6
28D4             0780         DB     $18,$05,$04   ;       "      24  ;       "        $0405
28D7             0790         DB     $FF           ;Ende der Tabelle!
```

So, jetzt ist hoffentlich klar, wie die gesuchte Funktion und deren Startadresse gefunden wird vom TRDOS! Bleibt nur noch die Frage, was die einzelnen Funktionen bewirken und welche Vorbereitungen jeweils nötig sind! Zuerst eine Übersicht mit Kurzbeschreibung (soweit bekannt), Einzelheiten folgen später!

Nr.	**Adr.**	**Beschreibung**
0	$3D98	BREAK-Taste abfragen
1	$3DCB	Laufwerksnummer wählen
2	$3E63	Laufwerk auf eine ausgewählte Spur einstellen
3	$3F02	Eine gewünschte Sektornummer in $5CFF ab legen
4	$3F06	Pufferende bzw Ladeadresse in $5D00 ablegen
5	$1E3D	Sektor(en) von Disk ins RAM einlesen
6	$1E4D	Sektor(en) von RAM auf Disk schreiben
7	$28D8	CAT ausgeben auf Screen oder Drucker
8	$165C	Ermittelten Namen und Daten aus CAT-Kopie in Puffer kopieren
9	$1664	Ermittelten Namen und Daten aus Puffer in CAT-Kopie kopieren
10	$1CFO	Name und Typ aus Puffer in CAT-Kopie suchen
11	$28FB	Bytes saven
12	$28F2	BASIC-Programm komplett saven
13	$01D3	Rückkehr ins BASIC oder TRDOS (Hier könnten,wie auch bei Nr. 15, 16 und 17 weitere Funktionsadr. ins EPROM eingebaut werden)
14	$290F	Bytes laden
15	$01D3	Rückkehr ins BASIC oder TRDOS
16	$01D3	" " " "
17	$01D3	" " " "
18	$2926	File löschen
19	$28E0	Prog.-Parameter von im HL-Reg. angegebenen Adresse nach $5CDD - $5CEC (in Puffer) kopieren
20	$28E3	Prog.-Parameter von $5CDD - $5CEC (aus Puffer) zur im HL-Reg. angegebenen Adresse Kopieren
21	$2739	Verify Track
22	$1FEB	Seite 0 einer Disk anwählen
23	$1FF6	Seite 1 einer Disk anwählen
24	$0405	Change disk = Diskformat ermitteln, eventuell Puffer erweitern und Organisations-Sektor laden

Wie kann man die Funktionen nun in der Praxis nutzen? Dazu ein ganz einfaches Beispiel, wie sich der Katalog aufrufen läßt (per Funktion 7)! Was muß dabei jedoch berücksichtigt werden?

Bekanntlich ist eine Grundvoraussetzung für alle TRDOS-Operationen das Vorhandensein des schon mehrfach erwähnten Puffers (112 Zeichen ab $5CB6). Gewöhnlich kann man wohl voraussetzen, daß er schon vorhanden ist, wenn einmal das TRDOS aufgerufen wurde, aber vielleicht wurde er auch inzwischen gelöscht (z.B. durch ein ’NEW’ im BASIC)!

Deshalb ist es in jedem Fall sinnvoll, zuerst prüfen zu lassen, ob der Puffer schon existiert und ihn gegebenenfalls aufzubauen. Dazu reicht ein einfacher Befehl: ’CALL $3D21’! Was sich dann abspielt, läßt sich in Folge 2 (Info 8/89) meiner Serie verfolgen. Speziell für die CAT-Ausgabe muß noch angegeben werden, ob die Darstellung auf dem Bildschirm (Kanal 2) oder Drucker (Kanal 3) erfolgen soll! Dazu wird die gewünschte Kanalnummer ins A-Reg. geschrieben.

So sieht unsere kleine Routine nun aus:

```
CALL $3D21
LD A,2
LD C,7
CALL $3D13
```

Wenn man nun den CAT oder andere Befehle per MC aufruft, würde immer Laufwerk "A" angesprochen. Aber auch das läßt sich ja mit Funktion 1 ändern, falls mehrere Disk-Laufwerke vorhanden sind! Wer nur eins hat, für den sind die folgenden Routinen natürlich momentan nicht nutzbar.

Diesmal muß die Laufwerksnummer ins A-Reg. übergeben werden (0-3)! Also fügen wir folgendes hinter dem CALL $3D21 ein, damit z.B. Laufwerk "B” angesprochen wird:

```
LD A, 1
LD C, 1
CALL $3D13
```

Wenn man das Programm nun aufruft, geschieht etwas unerwartetes! Der CAT einer Disk im Laufwerk "B" wird zwar gelistet, aber auf dem Bildschirm wird hartnäckig behauptet, es sei Laufwerk ”A", was natürlich nicht stimmt!

Wo liegt der Fehler? Grundsätz1ich wird die aktuelle Drive-Nummer bei Laufwerkswechsel sofort in der TRDOS-Systemvariablen $5CF6 abgelegt, aber ausgerechnet beim CAT-Befehl muß sie auch in $5CF9 zwischengespeichert werden. Das Warum soll uns hier nicht weiter interessieren, die vollständige Routine hat jedenfalls so auszusehen:

```
EA60            0010        org 60000          ;Startadresse beliebig!
                0020 ;
EA60 CD213D     0030        CALL $3021         ;Puffer vorhanden? Sonst installieren!
                0040 ;
EA63 3E01       0050        LD   A,1           ;A-Reg. Bit Nummer für Laufwerk 'A' laden ('B'=1, 'C'=2, 'D'=3)
EA65 0E01       0060        LD   C,1           ;Nummer der Funktion 'Laufwerk wählen' ins C-Reg. laden
EA67 CD133D     0070        CALL  $3D13        ;und diese ausführen!
                0080 ;
EA6A 3AF65C     0090        LD   A,($5CF6)     ;Laufwerksnummer aus TRDOS-Systemvar. ins A-Reg. bringen
EA6D 32F95C     0100        LD   ($5CF9),A     ;und spez. für CAT-Befehl in $5CF9 ablegen!
EA70 3E02       0110        LD   A,2           ;Nun Kanalnummer ins A-Reg. laden (2 - Screen, 3 = Drucker)
EA72 0E07       0120        LD   C,7           ;und Nummer der Funktion 'CAT  ausgeben' ins C-Reg.!
EA74 CD133D     0130        CALL  $3D13        ;Funktion ausführen
EA77 C9         0140        RET                ;und zurück ins BASIC!
```

Uber RAMTOP in einem beliebigen Bereich abgelegt und mit RANDOMIZE USR xxxxx aufgerufen, erledigt die kurze Routine ihre Aufgabe prompt: den Katalog einer Disk in Laufwerk ”B” zu lesen! Zwei wichtige Funktionen sind damit schon mal abgehakt!

Allerdings haben wir hier ohne es zu merken, gleich noch eine ganze Reihe weiterer Funktionen mitbenutzt, u.a. Funktion 0,5,22 bzw. 23, 24 usw.! Nur werden sie im TRDOS nicht über die Funktionsnummern, sondern direkt per CALL-Befehl angesprungen. Jedenfalls nehmen die Funktionen eine Schlüsselstellung im TRDOS ein. Wer obiges Programm testen möchte, jedoch keinen Assembler besitzt, kann sich mit dem folgendem Programm behelfen und damit die Werte in den Speicher poken! Der Code ist praktisch überall im RAM lauffähig.

```
 10 CLEAR 59999
 20 RESTORE 90
 30 LET start=60000
 40 FOR n=l TO 24
 50 READ a
 60 POKE start,a
 70 LET start=start+l
 80 NEXT n
 90 DATA 205,33,61,62,1,14,1,205,19,61
100 DATA 58,246,92,50,249,92
110 PATA 62,2,14,7,205,19,61,201
120 STOP
```

Ein RANDOMIZE USR 60000 setzt hier alles in Gang! Wer einen anderen Kanal oder Laufwerk einstellen will, muß die DATA-Zeilen entsprechend anpassen.

Für diesen Monat muß ich nun Schluß machen! Wenn meine Zeit reicht, dann nächstes Mal mehr zu weiteren Funktionen!

MfG
Wilhelm

Dez.	Hex.	Dual
0	0000	00000000
1	0001	00000001
2	0002	00000010
3	0003	00000011
4	0004	00000100
5	0005	00000101
6	0006	00000110
7	0007	00000111
8	0008	00001000
9	0009	00001001
10	000A	00001010
11	000B	00001011
12	000C	00001100
13	000D	00001101
14	000E	00001110
15	000F	00001111
16	0010	00010000
17	0011	00010001
18	0012	00010010
19	0013	00010011
20	0014	00010100
21	0015	00010101
22	0016	00010110
23	0017	00010111
24	0018	00011000
25	0019	00011001
26	001A	00011010
27	001B	00011011
28	001C	00011100
29	001D	00011101
30	001E	00011110
31	001F	00011111
32	0020	00100000
33	0021	00100001
34	0022	00100010
35	0023	00100011
36	0024	00100100
37	0025	00100101
38	0026	00100110
39	0027	00100111
40	0028	00101000
41	0029	00101001
42	002A	00101010
43	002B	00101011
44	002C	00101100
45	002D	00101101
46	002E	00101110
47	002F	00101111
48	0030	00110000
49	0031	00110001
50	0032	00110010
51	0033	00110011
52	0034	00110100
53	0035	00110101
54	0036	00110110
55	0037	00110111
56	0038	00111000
57	0039	00111001
58	003A	00111010
59	003B	00111011
60	003C	00111100
61	003D	00111101
62	003E	00111110
63	003F	00111111
64	0040	01000000
65	0041	01000001
66	0042	01000010
67	0043	01000011
68	0044	01000100
69	0045	01000101
70	0046	01000110
71	0047	01000111
72	0048	01001000
73	0049	01001001
74	004A	01001010
75	004B	01001011
76	004C	01001100
77	004D	01001101
78	004E	01001110
79	004F	01001111
80	0050	01010000
81	0051	01010001
82	0052	01010010
83	0053	01010011
84	0054	01010100
85	0055	01010101
86	0056	01010110
87	0057	01010111
88	0058	01011000
89	0059	01011001
90	005A	01011010
91	005B	01011011
92	005C	01011100
93	005D	01011101
94	005E	01011110
95	005F	01011111
96	0060	01100000
97	0061	01100001
98	0062	01100010
99	0063	01100011
100	0064	01100100
101	0065	01100101
102	0066	01100110
103	0067	01100111
104	0068	01101000
105	0069	01101001
106	006A	01101010
107	006B	01101011
108	006C	01101100
109	006D	01101101
110	006E	01101110
111	006F	01101111
112	0070	01110000
113	0071	01110001
114	0072	01110010
115	0073	01110011
116	0074	01110100
117	0075	01110101
118	0076	01110110
119	0077	01110111
120	0078	01111000
121	0079	01111001
122	007A	01111010
123	007B	01111011
124	007C	01111100
125	007D	01111101
126	007E	01111110
127	007F	01111111
128	0080	10000000
129	0081	10000001
130	0082	10000010
131	0083	10000011
132	0084	10000100
133	0085	10000101
134	0086	10000110
135	0087	10000111
136	0088	10001000
137	0089	10001001
138	008A	10001010
139	008B	10001011
140	008C	10001100
141	008D	10001101
142	008E	10001110
143	008F	10001111
144	0090	10010000
145	0091	10010001
146	0092	10010010
147	0093	10010011
148	0094	10010100
149	0095	10010101
150	0096	10010110
151	0097	10010111
152	0098	10011000
153	0099	10011001
154	009A	10011010
155	009B	10011011
156	009C	10011100
157	009D	10011101
158	009E	10011110
159	009F	10011111
160	00A0	10100000
161	00A1	10100001
162	00A2	10100010
163	00A3	10100011
164	00A4	10100100
165	00A5	10100101
166	00A6	10100110
167	00A7	10100111
168	00A8	10101000
169	00A9	10101001
170	00AA	10101010
171	00AB	10101011
172	00AC	10101100
173	00AD	10101101
174	00AE	10101110
175	00AF	10101111
176	00B0	10110000
177	00B1	10110001
178	00B2	10110010
179	00B3	10110011
180	00B4	10110100
181	00B5	10110101
182	00B6	10110110
183	00B7	10110111
184	00B8	10111000
185	00B9	10111001
186	00BA	10111010
187	00BB	10111011
188	00BC	10111100
189	00BD	10111101
190	00BE	10111110
191	00BF	10111111
192	00C0	11000000
193	00C1	11000001
194	00C2	11000010
195	00C3	11000011
196	00C4	11000100
197	00C5	11000101
198	00C6	11000110
199	00C7	11000111
200	00C8	11001000
201	00C9	11001001
202	00CA	11001010
203	00CB	11001011
204	00CC	11001100
205	00CD	11001101
206	00CE	11001110
207	00CF	11001111
208	00D0	11010000
209	00D1	11010001
210	00D2	11010010
211	00D3	11010011
212	00D4	11010100
213	00D5	11010101
214	00D6	11010110
215	00D7	11010111
216	00D8	11011000
217	00D9	11011001
218	00DA	11011010
219	00DB	11011011
220	00DC	11011100
221	00DD	11011101
222	00DE	11011110
223	00DF	11011111
224	00E0	11100000
225	00E1	11100001
226	00E2	11100010
227	00E3	11100011
228	00E4	11100100
229	00E5	11100101
230	00E6	11100110
231	00E7	11100111
232	00E8	11101000
233	00E9	11101001
234	00EA	11101010
235	00EB	11101011
236	00EC	11101100
237	00ED	11101101
238	00EE	11101110
239	00EF	11101111
240	00F0	11110000
241	00F1	11110001
242	00F2	11110010
243	00F3	11110011
244	00F4	11110100
245	00F5	11110101
246	00F6	11110110
247	00F7	11110111
248	00F8	11111000
249	00F9	11111001
250	00FA	11111010
251	00FB	11111011
252	00FC	11111100
253	00FD	11111101
254	00FE	11111110
255	00FF	11111111

Folge 8

11.12.89

Hallo Spectrum-User!

Auch diesmal zuerst ein Hinweis für SYS-User (Assembler/Disassemblers, Vers. 2.2)! Wer dies tolle Programm mit Hilfe der Anpassungsroutine 'INSTAL.DRV' zur Zusammenarbeit mit dem BETA- Disk-Interface überredet hat, wird zumindest bei den Versionen 5.xx auf eine kleine 'Macke' stoßen! Denn jedes mal, wenn ein Disk-Befehl (z.B. *savet "Filename") mit ENTER übergeben wird, meldet sich das TRDOS, aber gespeichert auf Disk wird nichts. Also RETURN gedrückt und schon meldet sich der Assembler zurück.

Falls man nun mit 'ret.' ins BASIC geht und sich das Programm anschaut, erkennt man, daß eine Zeile 5 entstanden ist, die einen unvo11ständigen SAVE-Befehl in BETA-Syntax enthält!

5 REM : SAVE "Filename" CODE 32768,xxxxx

Man kann ja nun einfach das erforder1iche 'RANDOMIZE USR 15619' vor dem 'REM' einfügen und mit 'RUN' anschließend das Saven erzwingen, aber das ist ja wohl nicht der Weisheit letzter Schluß. Bei mir kam noch erschwerend hinzu, daß ich auf keinen Fall vergessen durfte mein ISO-Monitor-ROM abzuschalten vor Disk-Befehlen aus dem Assembler heraus. Andernfalls hängte sich der SPECTRUM wohl aus Verzweiflung über soviel Vergesslichkeit auf und die Arbeit von Stunden war hin!

Also habe ich mir die 'INSTL.DRVc'-Routine, die auch das Auswahl-Menü für die diversen Laufwerksanpassungen erzeugt, näher angesehen. Dabei konnte ich feststellen, daß bei Wahl des Menü-Punktes 3 (=BETA) ein 87 Bytes langer Block aus dieser Routine von Adresse 25654 an nach 58849 in den Assembler kopiert wird (hier beginnen auch die Anpassungsroutinen für alle übrigen Speichermedien!).

Da es sich dabei offensichtlich um die eigentliche Anpassung handelt, war klar, daß ab Adresse 58849 die Suche nach dem Schwachpunkt einsetzen mußte! Und ganz schnell war er auch gefunden: bei Adresse 58873 steht nämlich 'CD 03 3C', was bekanntlich den Befehl: CALL $3C03 ergibt! Dies ist meines Wissens die Aufrufadresse für die BETA-Version 3.xx, bei Version 5.xx landet man damit jedoch bei einem $FF im TRDOS (vergleiche Folge 2 aus Info 8/89).

Das bewirkt im TRDOS aber nur ein Einschalten des Interrupts mit anschließendem 'ret'. Das ist hier aber ein Irrweg. Daher ändert man am Besten den Wert in Adresse 58875 um in $3D (mit ' POKE 58875,61) und schon hat man die korrekte Aufrufadresse $3D03 = 15619! Dann sind alle genannten Probleme behoben und jeder Diskbefehl wird prompt ausgeführt! Wer sich das Listing im Info 8 genau ansieht, wird feststellen, daß ein Ändern des Inhalts von Adr. 58874 nach $04 auch zum Ziel führt, den dann erreicht der CALL-Befehl ein 'JR $3C09', was über 'JP $3D03' endlich auch das Richtige bewirkt!

Das alles hätte meiner Meinung nach nicht sein brauchen, wenn beim Programmieren des BETA-Eproms an der Adresse $3C03 an Stelle des $FF (= RST $38) ein $00 (= NOP) eingefügt worden wäre. Da sieht man wieder was für unangenehme Folgen ein kleiner verborgener Fehler haben kann! Aber es lohnt sich gewöhnlich, den Dingen auf den Grund zu gehen, deshalb auch die lange Erläuterung dazu. Ob bei den Versionen 4.xx auch Probleme auftreten, weiß ich leider nicht, falls ja, sollte das Einbauen der richtigen Adresse nun keine Schwierigkeiten mehr bereiten.

Jetzt aber mehr zu den schon im letzten Info angesprochenen Funktionen des TRDOS 5.03 (zur Erinnerung: darüber lassen sich verschiedene Befehle aus MC—Programmen heraus einfach nutzten)! Wie schon gesagt, geht im TRDOS nichts ohne den Systemvariablen-Puffer, dessen Vorhandensein durch ein 'CALL $3D21' abgefragt und der dann gegebenfalls installiert wird.

Es reicht, wenn dies einmal vorm ersten Aufruf einer Funktion vorgenommen wird und wenn gewährleistet ist, daß der Puffer noch vom Laden eines Programms existiert usw., dann kann auch ganz auf Überprüfung verzichtet werden! Ein 'NEW' im BASIC löscht den Puffer jedoch.

In den folgenden Beispielen werden wir allerdings sehen, daß manchmal die Grundwerte (auch Defaultwerte genannt) im Puffer nicht ausreichen, um den Funktionsaufruf zum erfo1greichen Abschluß zu bringen. Dazu müssen dann bestimmte TRDOS-Systemvariab1en vorher mit neuen Werten geladen werden. Aber so einen Fall hatten wir ja auch schon beim Katalog-Laden im vorigen Artikel!

Nun ein Beispiel, wie ein komplettes BASIC-Programm mit Hilfe der Funktion 12 gespeichert werden kann:

```
             0010 ;FUNKTION 12 (= BASIC-PROGRAMH KOMPLETT SAVEN)
             0020 ;
EA60         0030      org 60000       ;Startadresse beliebig!
             0040 ;
EA60 CD213D  0050      CALL  $3021     ;TRDOS-Puffer vorhanden? Kein? Dann diesen installieren!
             0060 ;
EA63 3E00    0070      LD    A,0       ;Laufwerksnummer ins A-Register (A=0, B=l, C=2, 0=3)und
EA65 0E01    0080      LD    C, 1      ;C-Reg. mit Nummer der Funktion 'Laufwerksnummer wählen' laden!
EA67 CD133D  0090      CALL  $3D13     ;Nun diese Funktion aufrufen und ausführen!
             0100 ;
EA6A 217BEA  0110      LD   HL,name    ;Adresse des Namenanfangs in HL,
EA6D 11DD5C  0120      LD   DE,$5CDD   ;Adr. der TRDOS-Variablen für Programmnamen in DE
EA70 010900  0130      LD    BC,9      ;und BC mit 9 laden (Länge des Namens + Typ-Code =9 Zeichen)!
EA73 EDB0    0140      LDIR            ;Die 9 Zeichen jetzt nach $5CDD-$5CE5in den Puffer kopieren!
            0150 ;
EA75 0EOC    0160      LD   C,12       ;Nun Funktionsnr. 12 (= BASIC-Prog. saven) ins C-Reg. bringen
EA77 CD133D  0170      CALL $3D13      ;und auch diese Funktion ausführen!
EA7A C9     0180       RET             ;Danach zurück ins BASIC!
            0190 ;
            0200 ;
            0210 ;Nachfolgend sind 9 Zeichen für Prog.-Namen + Typ-Code reserviert!
            0220 ;
EA7B        0230 name  DM Filename
EA83        0240       DM B
```

Hier ist also vorm eigentlichen Saven noch der Aufruf der Funktion 1 (=Laufwerksnr. wählen) erforderlich, sonst erscheint die BASIC-Fehlermeldung 'Tape loading error'. Des weiteren muß das zu speichernde Programm ja auch einen Namen haben (immer 8 Zeichen lang, notfalls mit Leerzeichen auffüllen!) Dazu muß auch noch der Typ (hier immer 'B' wie BASIC!) angegeben werden. Ein LDIR-Befehl kopiert dies dann in den vorbereiteten Puffer!

Ganz ähnlich wird auch Funktion 18 (= Erase file) aufgerufen. Aber es gibt 2 wichtige Punkte zu beachten! Einmal kann hier natürlich eine Änderung des Programmtyps (B = BASIC, C = CODE, D = DATA oder # - Random access files) notwendig sein und außerdem muß vor Funktionsaufruf die TRDOS-Variable $5D06 mit dem Wert 9 geladen werden. Dieser dient dem System hier als Zählvariable (Name + Typ = 9 Zeichen!) während des Namens- und Typvergleichs mit den Eintragungen im Katalog! Hier nun die Beispielroutine:

```
             0010 ;FUNKTION 18 (= FILENAMEN IM KATALOG LÖSCHEN)
             0020 ;
EA60         0030     org 60000       ;Startadresse beliebig!
             0040 ;
EA60 CD213D  0050     CALL $3D21      ;TRDOS-Puffer vorhanden? Nein? Dann diesen installieren!
             0060 ;
EA63 3E00    0070     LD   A,0        ;Laufwerksnummer ins A-Register (A=0, B=1, C=2, D=3) und
EA65 0E01    0080     LD   C,1        ;C-Reg. mit Nummer der Funktion ' Laufwerksnummer wählen' laden!
EA67 CD133D  0090     CALL $3D13      ;Nun diese Funktion aufrufen und ausführen!
             0100 ;
EA6A 2180EA  0110     LD   HL,name    ;Adresse des Namenanfangs in HL,
EA6D 11DD5C  0120     LD   DE,$5CDD   ;Adr. der TRDOS-Variablen für Programmnamen in DE
EA70 010900  0130     LD   BC,9       ;und BC mit 9 laden (Länge des Namens + Typ-Code =9 Zeichen)!
EA73 EDB0    0140     LDIR            ;Die 9 Zeichen jetzt nach $5CDD-$5CE5 in den Puffer kopieren!
             0150 ;
EA75 3E09    0160     LD    A,9       ;A-Reg. mit 9 laden und diesen Wert in TRDOS-Variable $5606
EA77 32065D  0170     LD   ($5D06),A  ;ablegen als Zähler beim Namensvergleich im Katalog!
EA7A 0E12    0180     LD   C,18       ;Nun Funktionsnr. 18 (=Filenamen löschen ) ins C-Reg.bringen
EA7C CD133D  0190     CALL $3D13      ;und auch diese Funktion ausführen (= ERASE)!
EA7F C9      0200     RET             ;Danach zurück ins BASIC!
             0210 ;
             0220 ;
             0230 ;nachfolgend sind 9 Zeichen für Prog.-Namen + Typ-Code reserviert!
             0240 ;Als Typ-Code kommt hier je nach Programm B, C, D, oder # in Frage!
             0250 ;
EA80         0260 name   DM Filename
EA88         0270        DM B
```

Als nächstes folgt ein Beispiel für Funktion 11 (=Bytes saven)! Wieder muß das gewünschte Laufwerk eingestellt und der Programmname in den Puffer kopiert werden. Soweit kennen wir das ja schon! Neu kommt hinzu, daß die Anfangsadresse des CODE ins HL- und dessen Länge ins DE-Reg. gebracht werden muß, und zwar unmittelbar vorm Aufruf der Funktion 11! Jetzt das vollständige Programm dazu:

```
                0010 ;FUNKTION 11 (= CODE SAVEN)
                0020 ;
EA60            0030          org 60000          ;Startadresse beliebig!
                0040 ;
EA60 CD213D     0050          CALL S3D21         ;TRDOS-Puffer vorhanden? Nein? Dann diesen installieren!
                0060 ;
EA63 3E00       0070          LD A,0             ; A-Reg. mit Laufwerksnummer und
EA65 0E01       0080          LD C,1             ;C-Reg. mit Nummer der Funktion 'Laufwerksnummer wählen' laden!
EA67 CD133D     0090          CALL$3D13          ; Nun diese Funktion aufrufen und ausführen!
                0100 ;
EA6A 2181EA     0110          LD   HL,name       ;Adresse des Namenanfangs in HL,
EA6D 11DD5C     0120          LD   DE,$5CDD      ;Adr. der TRDOS-Variablen für Programmnamen in DE
EA70 010900     0130          LD   BC,9          ;und BC mit 9 laden (Länge des Namens + Typ-Code  =9 Zeichen)!
EÄ73 EDB0       0140          LDIR               ;Die 9 Zeichen jetzt nach $5CDD-S5CE5 in den Puffer kopieren!
                0150 ;
EA75 210040     0160          LD   HL,16384      ;Anfangsadr. des CODE's in HL (hier Screenanfang) und
EA78 11001B     0170          LD   DE,6912       ;CODE-Länge in DE (hier Länge eines Screens) laden!
EA7B 0E0B       0180          LD   C,11          ;Nun Funktionsnr. 11 (=Bytes saven) ins C-Reg. bringen
EA7D CD133D     0190          CALL  $3D13        ;und auch diese Funktion ausführen (=CODE laden)!
EA80 C9         0200          RET                ;Danach zurück ins BASIC!
                0210 ;
                0220 ;Nachfolgend sind 9 Zeichen für Prog.-Namen + Typ-Code reserviert!
                0230 ;Als Typ-Code kommt hier nur 'C' in Frage!
                0240 ;
EA81            0250 name DM       Codename
EA89            0260          DM   C
```

Nun kommt auch noch die Funktion 14 (=Bytes laden) dran. Damit kann man sich u.a. Programme mit Nachladern, wie man sie für Microdrives oft findet, für BETA-Disk umschreiben, was jedoch gewöhnlich eine genaue Analyse des ins Auge gefaßten Programms erfordert aber sehr interessant sein kann!

Wie im Listing zu sehen, müssen die beiden TRDOS-Variablen $5CF7 und $5D10 erst mal gelöscht werden. Ihre Bedeutung ist mir zur Zeit leider noch nicht ganz klar. Hier habe ich das Laufwerk A (=0) gewählt, bei einem anderen müßte noch eine Zeile 95 mit einem Befehl 'LD A,x' eingefügt werden, wobei x für die Laufwerksnummer steht!

Dann mußte auch die Funktion 24 (Change disk) aufgerufen werden, da unbedingt der Organisationssektor geladen, daraus u.a. das aktuelle Diskformat ermittelt und eventuell der Puffer erweitert werden muß. Dann wird auch noch die schon bekannte TRDOS-Variable $5D06 wieder mit 9 geladen. In Zeile 240 wird das A-Reg. hier gelöscht, d.h. mit 0 beschrieben, weil im Beispiel der CODE an die gleiche Adresse geladen werden soll, unter der er abgespeichert wurde. Bei einer geänderten Ladeadresse sollte an dieser Stelle ein 'LD A,1' stehen! Der Rest ist nichts Neues mehr. Hier die komplette Routine:

```
                0010 (FUNKTION 14 (= LOAD CODE)
                0020 (
EA60            0030          org 60000          ;Startadresse beliebig!
                0040 ;
EA60 CD213D     0050          CALL  $3D21        ;TRDOS-Puffer vorhanden? Nein? Dann diesen installieren!
                0060;
EA63 3E00       0070          LD    A,0          ;A-Reg. Bit 0 laden und
EA65 32F75C     0080          LD    ($5CF7),A    ;TRD0S-Var. S5CF7 (Bedeutung noch nicht klar!!!)
EA68 32105D     0090          LD    $5D10),A     ;und $5D10 löschen (auch noch unklar!!!)
EA6B 0E01       0100          LD    C,1          ;C-Reg. mit Nummer der Funktion ' Laufwerksnummer wählen' laden!
EA6D CD133D     0110          CALL  $3D13        ;Nun diese Funktion aufrufen und ausführen!
                0120 ;
EA70 0E18       0130          LD    C,24         ;Funktionsnummer 24 (=Change disk) ins C-Reg. laden
EA72 CD133D     0140          CALL  $3D13        ;und diese Funktion aufrufen und ausführen!
                0150 ;
EA75 3E09       0160          LD    A,9          ;A-Reg. mit Zählvariable für Namens- und Typ-Vergleich laden
EA77 32065D     0170          LD    ($5D06),A    ;und diesen Wert in entsprechende TRDOS-Variable retten!
                0180 ;
EA7A 2192EA     0190          LD    HL,name      ;Adresse des Namenanfangs in HL,
EA7D 11DD5C     0200          LD    DE,S5CDD     ;Adr. der TRDOS-Variablen für Programmnamen in DE
EA80 010900     0210          LD    BC,9         ;und BC mit 9 laden (Länge des Namens + Typ-Code =9 Zeichen)!
EA83 EDB0       0220          LDIR               ;Die 9 Zeichen jetzt nach 55CDD-S5CE5 in den Puffer kopieren!
                0230 ;
EA85 AF         0240          XOR   A            ;A-Reg. löschen, wenn Laden an Originalstartadr. erfolgen soll!
                0250 ;
EA86 210040     0260          LD    HL,16384     ;Anfangsadr. des CODE's in HL (hier Screenanfang) und
EA89 11001B     0270          LD    DE.6912      ;CODE-Länge in DE (hier Länge eines Screens) laden!
EA8C 0E0E       0280          LD    C,14         ;Nun Funktionsnr. 14 (=Bytes laden ) ins C-Reg. bringen
EA8E CD133D     0290          CALL  $3D13        ;und auch diese Funktion ausführen (=CODE laden)!
EA91 C9         0300          RET                ;Danach zurück ins BASIC!
                0310 ;
                0320 ;Nachfolgend sind 9 Zeichen für Prog.-Namen + Typ-Code reserviert!
                0330 ;Als Typ-Code kommt hier nur 'C' in Frage!
                0340 ;
EA92            0350 name   DM    Codename
EA9A            0360        DM    C
```

Die nächsten beiden Funktionen 5 (Sektoren von Disk lesen) und 6 (Sektoren auf Disk schreiben) sind von besonderer Bedeutung, denn ohne sie läuft kein Datentransfer vom Speicher auf Disk und umgekehrt!

Allerdings ist die Handhabung manchmal etwas schwierig, da man genau wissen muß, auf welchen Track und Sektor der Disk man zugreifen will und wie viele Sektoren geladen oder gespeichert werden sollen. Im folgenden Beispiel werden ab Track 1, Sektor 0 die nächsten 27 Sektoren (27 * 256 = 6912 Bytes) von Disk in den Speicher geholt nach Adresse 16384 bis 23296, also in den Bild- und Attributspeicher! Das macht natürlich nur Sinn, wenn ab Track 1, Sektor 0 auch tatsächlich ein Screen auf Disk gespeichert wurde, denn dann kann man auch sofort den Erfolg seiner Bemühungen bewundern!

Aber natürlich lassen sich auch andere Daten in geschützte Speicherbereiche transferieren und dort weiterverarbeiten und auch wieder zurückschreiben auf Disk. Genau das macht das TRDOS ja auch mit Hilfe dieser Funktionen, wenn der Katalog oder Organisationssektor bearbeitet wird, also z.B. ein Programmname gelöscht, geändert, gesucht oder zugefügt wird! Hier liegt auch die Stärke dieser Funktionen.

Sie können im Gegensatz zu den Funktionen 11 und 14 (Bytes laden/saven) ausgelesene und eventuell veränderte Sektoren auch an die Original-Position auf der Disk zurückschreiben. Diese wichtige Eigenschaft wird auch vom 'doctor'-Programm auf der Utility-Disk genutzt! Damit lassen sich dann u.a. 'defekte' Sektoren korrigieren oder gelöschte Files wieder lesbar machen, aber dazu ein anderes Mal mehr.

Man sollte sich aber immer der Tatsache bewußt sein, daß man speziell beim überschreiben von Sektoren (besonders im Katalog und Org.-Sektor) sehr umsichtig zu Werke gehen muß. Versuche in dieser Richtung deshalb nie mit Original-Disketten vornehmen! Nun die Routinen, die bis auf die Funktionsnummern identisch sind, deshalb nur ein Listing!

```
                0010 ;FUNKTION 5 (= LOAD SEKTOR) / FUNKTION 6 (= SAVE SEKTOR)
                0020 ;
EA60            0030     ORG 60000      ;Startadresse beliebig!
                0040 ;
EA60 CD213D     0050 ;    CALL $3D21     ;TRDOS-Puffer vorhanden? Nein? Dann diesen installieren!
                0060 ;
EA63 3E00       0070     LD   A,0       ;Laufwerksnummer ins A-Register (A=0, B=1, C=2, 0=3) und
EA65 0EO1       0080     LD   C, 1      ;(C-Reg. mit Nummer der Funktion 'Laufwerksnummer wählen' laden!
EA67 CD133D     0090     CALL $3D13     ;Nun diese Funktion aufrufen und ausführen!
                0100 ;
EA6A 210040     0110     LD   HL,16384  ;Adr., ab welcher geladen bzw. gesavet wird (hier Screen)!
EA6D 061B       0120     LD   B,27      ;Zahl zu ladender/speichernder Sektoren (hier kompl. Screen)!
EA6F 1601       0130     LD   D,1       ;Tracknr. des 1. zu ladenden bzw. zu speichernden Sektors in D!
EA71 1E00       0140     LD   E,0       ;Sektornr. des 1. zu ladenden bzw. zu speichernden Sektors in E!
EA73 0E05       0150     LD   C,5       ;Nun Funktionsnr. 5 (= lesen! bzw. 6 (=schreiben! in C laden
EA75 CD133D     0160     CALL $3D13     ;und auch diese Funktion ausführen!
EA78 C9         0170    RET             ;Danach zurück ins BASIC!
```

Daß bei manchen Funktionen (wie z.B. bei 'Bytes saven'),so umfangreiche Vorbereitungen zu treffen sind, liegt in der Regel daran, daß hier andere Einstiegsadressen benutzt werden als beim normalen entsprechenden Befehl, wo diese gewöhnlich schon vorweg automatisch getroffen werden!

Will man mehrere der beschriebenen Funktionen gleichzeitig in einem Programm nutzen, brauchen natürlich die immer wieder verwendeten Teile nur einmal eingebaut werden mit einem 'RET' als Abschluß! Man kann sie dann mit einem 'CALL xxxx' von überall erreichen und spart wertvollen Speicherplatz!

Dazu dann nächstes Mal ein Assemblerlisting der MC-Routinen des Disk-Doktor-Programms der BETA-Version 5.03! Da wird dann gezeigt, wie man am Besten mehrere Funktionen geschickt miteinander kombiniert und sie nutzt.

Damit soll's für heute wieder genug sein. Nächstes mal dann vielleicht ein paar Worte zu den übrigen Funktionen und andere interessante Routinen. Bis dann!

MfG
Wilhelm

Folge 9

7.1.90

Hallo Spectrum-User !

Nun will ich versuchen, meinen ersten Beitrag im neuen Jahr für’s Club-Info zu schreiben. Hoffentlich bleibt die Mehrzahl der Mitglieder auch 1990 dem SPECTRUM treu und damit hoffentlich auch dem Club! Ich werde jedenfalls noch lange den SPECTRUM nutzen, denn bisher habe ich seine Leistungsfähigkeit noch lange nicht voll ausgeschöpft!

Da mich auch besonders das Programmieren und Analysieren von Maschinensprache-Programmen interessiert, bieten sich noch reichlich Möglichkeiten. Schließlich kann der SPECTRUM mit intelligent gemachten Programmen und Peripherie fast alles, wenn auch langsamer und vielleicht nicht immer so komfortabel, was wesentlich teurere Rechner leisten! Daher kann ich den Beiträgen im Info 11+12/89 nur zustimmen, denn ich will schließlich nicht nur ’Anwender’ sein, sondern auch noch verstehen, was da abläuft.

Jetzt soll’s aber endlich weitergehen mit meiner Serie zur BETA-Disk-Version 5.03! Letztes Mal habe ich versucht, die wichtigsten Funktionen (zur Erinnerung: sie dienen dem Aufruf verschiedener Befehle aus MC-Programmen) zu erläutern.

Inzwischen ist also geklärt, was die Funktionen 1, 5, 6, 7, 11, 12, 14 und 18 bewirken und wie sie sich ansprechen lassen. Wie sieht’s aber bei den übrigen aus? Also, Funktion 13 und 15 - 17 können wir sofort abhaken, den sie bewirken nur eine Rückkehr ins BASIC. Hier kann man bei einem geänderten EPROM eventuell eigene Aufrufadressen einbauen und damit eigene Funktionen aufrufen.

Von großer Bedeutung ist Funktion 24! Sie bewirkt unter anderem, daß gegebenenfalls der Puffer erweitert und der Organisationssektor geladen wird. Danach findet eine Überprüfung des Disk-Formates mit Anpassung der entsprechenden Systemvariablen. Diese Funktion muß immer vorgeschaltet werden, wenn sichergestel1t werden muß, das unterschied1iehe Disk-Formate erkannt werden.

Das ist eigentlich bei allen Schreib- und Lese—Operationen auf und von Disk erforder1ich. Bei einigen Funktionen wird dies schon automatisch berücksichtigt (z.B. bei den Funktionen 7, 11, 12 und 18). Bei Nummer 14 (Bytes laden) ist dies jedoch nicht der Fall, deshalb wurde es im entsprechenden Programm (Folge 7) in Zeile 130-140 nachgeholt! Näheres zu Funktion 24 ist dem Assemblerlisting in Folge 6 dieser Serie zu entnehmen (Einstiegsadresse $0405)!

Funktion 0 dient hauptsächlich der BREAK-Abfrage. Da sie fester Bestandteil der Schreib- und Lese-Operationen ist, kann man gewöhnlich auf einen gesonderten Aufruf verzichten.

Die auch noch nicht behandelte Funktion 2 bewirkt eine Einstellung des angesprochenen Laufwerks auf eine im A-Register übergebene Tracknummer! Auch diese wird praktisch bei allen Schreib-/Lese-Zugriffen genutzt. Eigene Verwendung habe ich bisher noch nicht dafür gehabt. Wer will, kann ja mal folgende kleine Routine per Assembler eingeben und testen:

```
CALL $3D21 ;TRDOS-Puffer testen bzw. installieren
LD A,33    ;Tracknummer ins A-Reg. schreiben
LD C,2     ;Funktionsnummer übergeben
CALL $3D13 ;Funktion aufrufen
RET        ;Zurück ins BASIC
```

Bei Aufruf wird nun der Schreib-/Lese-Kopf auf Track 33 eingestellt! Andere Tracknummern können natürlich im Rahmen des technisch Möglichen eingegeben werden (also bei 40 Track- Laufwerk 0-39, bei 80 Tracks 0-79 für einseitige und 0-79 bzw. 0-159 bei doppelseitigen).

Die Funktionen 3 und 4 sind meiner Meinung nach überflüssig, da ihr Aufruf platzsparender durch direkte Ladebefehle ersetzt werden kann!

Bei Nummer 3 wird nämlich nur eine Sektornummer aus dem A-Reg. in die Systemvariab1e $5CFF übergeben, während Funktion 4 eine Adresse aus dem HL-Register nach S5D00 schreibt!

Funktion 10 vergleicht einen im TRDOS-Systemvariablenbereich ab $5CDD abgelegten Programmnamen der Reihe nach mit denen in der zuvor im erweiterten Puffer geladenen Katalog-Kopie. Wenn er gefunden wird, steht im C-Reg. die Nummer des gefunden Namens im Katalog.

Wird diese nun ins A-Reg. übergeben, kann durch Aufruf von Funktion 8 oder 9 daraus der richtige Sektor errechnet und dann der komplette Datensatz des entsprechenden Programms bei 8 aus der CAT-Kopie in den TRDOS-Puffer und bei 9 in umgekehrte Richtung kopiert werden. Praktisch angewendet habe ich dies bisher noch nicht in eigenen Routinen, daher erfolgen diese Angaben nur unter Vorbehalt!

Letzteres gilt auch für Funktion 19 und 20! Soweit zu übersehen, läßt sich auch damit der zu jedem Programm im Katalog gehörende Datensatz (bestehend aus Namen, Typ, Startadresse, Blocklänge, Fileanzahl, Startsektor und Starttrack, (vergleiche dazu Folge 4) mit Funktion 19 von einer im HL-Reg. übergebenen Adresse in den TRDOS-Systemvariablenbereich nach $5CDD-$5CEC kopieren.

Funktion 20 erlaubt den Transfer der Programmparameter in die umgekehrte Richtung. Damit kann man eventuell eine Routine stricken, die es erlaubt, den Kataloginhalt in jeden gewünschten Speicherbereich zur weiteren Bearbeitung zu laden. Das läßt sich vielleicht nutzen, um die Parameter z.B. für die Verwendung in einem Textverarbeitungsprogramm aufzubereiten, wo sich dann auch weiterführende Kommentare einbauen ließen für detailierte Programmlisten usw.! Der Fantasie sind also keine Grenzen gesetzt, nur ist die Realisierung der Ideen meist sehr zeitraubend!

Bleiben noch die Funktionen 21-23! Nummer 21 dient zur Überprüfung der einzelnen Tracks auf Fehlerfreiheit, Funktion 22 wählt die Seite 0 einer Disk an und 23 entsprechend Seite 1. Das ist ja erforderlich, sobald bei einer doppelseitigen Diskette eine Tracknummer >39 (bei 40 Trk DS) oder >79 (bei 80 Trk DS) gelesen oder beschrieben werden soll. Gewöhnlich geschieht das Umschalten zwischen beiden Seiten unbemerkt, da automatisch.

Wie die letzten 3 Funktionen und einige schon genauer beschriebene genutzt werden können, geht aus dem kommentierten Assemblerlisting des Programms "doctor” von der Utility-Disk der BETA-.03-Version hervor. Dies bietet ein anschauliches Beispiel, wie sich mehrere Funktionen übersichtlich und platzsparend aufrufen lassen!

Wer allerdings den BASIC-Teil von ’doctor’ und den MC-Teil vergleicht, wird sofort auf Unstimmigkeiten stoßen. In den REM- Zeilen 40 bis 60 im BASIC-Programm werden nämlich falsche Adressen angegeben! Korrekt muß es dort heißen:

```
40 REM 60015 sector    (statt 60014)
50 REM 60016 track     (statt 60015)
60 REM 60017 drive no  (statt 60016)
```

Praktische Auswirkungen hat der Fehler aber nicht, da im Programm selbst die richtigen Adressen benutzt werden. Aber es kann irritierend wirken!

Übrigens möchte ich nochmals darauf hinweisen, daß es ratsam ist, zuerst nach Einlegen einer zu untersuchenden Diskette Menu-Punkt 8 (Change disk) aufzurufen. Meine Utility-Disk ist nämlich mit 80 Trk SS formatiert und darauf werden auch die TRDOS-Systemvariablen eingestellt.

Wenn die neu eingelegte Disk jedoch in einem anderen Format formatiert ist, werden die übrigen Menu-Punkte nicht korrekt behandelt. Das richtige Format wird erst durch ’Change disk’, was exakt einen Aufruf der Funktion 24 bedeutet, eingestellt!

Wie im Listing zu sehen, werden alle Funktionen über JUMP-Befehle (JP xxxx) erreicht, die wiederum aus dem BASIC jeweils mit ’RANDOMIZE USR xxxxx’ angesprungen werden. Weil alle benutzten Routinen und Einsprünge in der sogenannten Sprungleiste stehen, ist das Programm sehr ’pflege1eicht’, das heißt, leicht zu erweitern und zu ändern. Für eigene MC-Programme sollte man sich diese Vorgehensweise merken, denn nebenbei wird’s auch noch übersichtlicher!

Nun aber erst mal das Assemblerlisting:

```
EA60            0000            0RG 60000           ;Startadresse von CODE 'doctor'
                0010 ;
                0020             JP     readsek      ;Aufrufadr. 60000  für Funktion 5 (= Sektor lesen)
EA63 C393EA     0030             JP     writsek      ;Aufrufadr. 60003 für Funktion 6  (=Sektor schreiben)
EA66 C399EA     0040             JP     seldriv      ;Aufrufadr. 60006  für Funktion 1 (=Laufwerk wählen)
EA69 C3A2EA     0050             JP     chadisk      ;Aufrufadr. 60009  für Funktion 24 (= Change Disk)
EA6C C3A8EA     0060             JP     veritrk      ;Aufrufadr. 60012 für Funktion 21 (=Verify komplette Disk)
                0065 ;
EA6F            0070 seknr       DB     $00          ;Platz für Sektornummer (Wert 0-15)
EA70            0080 trknr       DB     $00          ;Platz für Tracknr. (Wert 0-39/79/159 je nach Diskformat)
EA71            0090 drivenr     DB     $00          ;Platz für Drivenummer (Wert 0-3)
                0095 ;
EA72 C3AEEA     0100             JP     side0        ;Aufrufadr. 60018 für Funktion 22 (= Disk-Seite 0 wählen)
EA75 C3B4EA     0110             JP     side1        ;Aufrufadr. 60021 für Funktion 23 (= Disk-Seite 1 wählen)
                0115 ;
EA78 CD133D     0120 runfunk     CALL   $3D13        ;Gewählte Funktion ausführen!
EA7B C9         0130             RET                 ;Zurück ins BASIC!
                0140 ;
EA7C 0601       0150 sekload     LD     B,1          ;Zu ladende bzw. speichernde Sektorenzahl ins 6-Reg. und
EA7E 218CEB     0160             LD     HL,$EB8C     ;Zieladr. bzw Startadr. (hier 60300) dafür in HL laden!
EA81 3A70EA     0170             LD     A,(trknr)    ;Eingegebene Tracknummer ins A-Reg. einlesen
EA84 57         0180             LD     D,A          ;und ins D-Reg. übertragen!
EA85 3A6FEA     0190             LD     A,(seknr)    ;Eingegebene Sektornummer ins A-Reg. einlesen
EA88 5F         0200             LD     E,A          ;und ins E-Reg. bringen!
EA89 CD133D     0210             CALL   $3D13        ;Gewählte Funktion ausführen!
EA8C C9         0220             RET                 ;Zurück ins BASIC!
                0230 ;
EA8D 0E05       0240 readsek     LD     C,5          ;Funktionsnr. 5 laden (= Sektor lesen)
EA8F CD7CEA     0250             CALL   sekload      ;Parameter laden und Funktion ausführen.
EA92 C9         0260             RET                 ;Zurück ins BASIC?
                0270;
EA93 0E06       0280 writsek     LD     C,6          ;Funktionsnr. 6 laden (=Sektor auf Disk schreiben)!
EA95 CD7CEA     0290             CALL   sekload      ;Parameter laden und Funktion ausführen.
EA98 C9         0300             RET                 ;Zurück ins BASIC!
                0310 ;
EA99 3A71EA     0320 seldriv LD A,(drivenr)          ;Eingegebene Laufwerksnummer ins A-Reg. holen
EA9C 0E01       0330             LD     C,1          ;und Funktionsnr. 1 laden (= Laufwerksnr. wählen)
EA9E CD78EA     0340             CALL   runfunk      ;und 'Funktion ausführen' aufrufen!
EAA1 C9         0350             RET                 ;Zurück ins BASIC!
                0360 ;
EAA2 0E18       0370 chadisk LD      C,24            ;Funktionsnr. 24 laden (= Diskformat überprüfen)
EAA4 CD78EA     0380             CALL   runfunk      ;und 'Funktion ausführen' aufrufen!
EÄA7 C9         0390             RET                 ;Zurück ins BASIC!
                0400 ;
EAA8 0E15       0410 veritrk     LD C,21             ;Funktionsnr. 21 laden Verify track)
EAAA C07CEA     0420             CALL sekload        ;Parameter laden und Funktion ausführen.
EAAD C9         0430             RET                 ;Zurück ins BASIC!
                0440 ;
EAAE 0E16       0450 side0       LD C,22             ;Funktionsnr. 22 laden (= Seite 0 der  Disk wählen)
EABO CD78EA     0460             CALL runfunk        ;und 'Funktion ausführen' aufrufen!
EAB3 C9         0470             RET                 ;Zurück ins BASIC!
                0480 ;
EAB4 0E17       0490 side1       LD C,23             ;Funktionsnr. 23 laden (= Seite 1 der Disk wählen)
EAB6 CD78EA     0500             CALL runfunk        ;und 'Funktion ausführen' aufrufen!
EA69 C9         0510             RET                 ;Zurück ins BASIC!

#               61A8    seknr    EA6F    trknr    EA70    drivenr  EA71    runfunk  EA78    sekload  EA7C    readsek  EA8D
                writsek  EA93    seldriv  EA99    chadisk  EAA2    veritrk  EAA8    side0    EAAE    sidel    EAB4
```

Ich vermute, daß es auch für ältere BETA-Versionen einen Diskettenmonitor in ähnlicher Form gibt, der wahrscheinlich sogar nicht viel anders aufgebaut ist als dieser hier. Es lohnt sich bestimmt, einmal nachzusehen, bietet sich doch damit die Möglichkeit, etwas über die Funktionen und deren Aufrufadresse zu erfahren.

Wer übrigens mehrere Sektoren gleichzeitig laden will um sie sich z.B. mit Menu-Punkt 6 (Display sektor) anzusehen, kann dies leicht haben! Dazu braucht nur in Adresse $EA7D ein neuer Wert mit 'POKE 60029,x' eingegeben werden. Der Wert x sollte nicht größer als 16 sein, womit ein ganzer Track in einem Rutsch geladen (oder auf Disk zurückgeschrieben) wird! Eventuell kann man sich eine entsprechende Abfrage auch in den BASIC-Teil einbauen, etwa so:

```
9205 GOSUB 9400          bzw 9305 GOSUB 9400
......
9400 INPUT "Sektoranzahl? (1-16) ";sz : IF sz < 1 OR sz > 16
THEN GOTO 9400
9410 POKE 60029,sz :RETURN
```

Es bedarf einer weiteren Änderung und zwar in Zeile 8420, damit die richtige Sektorenanzahl auch gelistet werden kann:

```
8420 FOR i=a TO 255 * sz STEP 4
```

Dann läßt sich jedes mal die Anzahl neu festlegen. Das gleichzeitige Laden mehrerer Sektoren spart bei Sucherei auf der Disk viel Zeit! Vorsicht ist allerdings besonders beim Zurückschreiben von Sektoren auf Disk geboten! Dann sollte die eingegebene Sektorenzahl keinesfalls größer als die zuvor beim Lesen verwendete sein. Andernfalls werden eventuell wichtige Daten mit 'Müll' überschrieben und das kann unter Umständen die ganze Disk oder zumindest Teile davon unbrauchbar machen!

Natürlich gibt's noch viele Verbesserungmöglichkeiten, so könnte man sich, während der Sektorinhalt gelistet wird, ständig die zugehörige Track- und Sektornummer mit anzeigen lassen usw.! Das erspart einem dann hinterher eventuell die Rechnerei, herauszufinden, in welchem Track und Sektor man gerade liest. Dazu vielleicht ein anders Mal mehr!

Bei Menu-Punkt 6 ist Ausgabe auf einen Drucker manchmal wünschenswert! Das ist eigentlich kein großes Problem, da man im Grunde ja nur von Kanal 2 auf Kanal 3.umschalten muß. Folgende Änderungen sind dazu notwendig:

```
8402 INPUT "2) Bildschirm 3)Drucker? ";ch: IF ch<2 0R ch>3 THEN GOTO 8402
```

Des weiteren muß, falls auch mit Menu-Punkt 7 (Edit sector) mehrere Sektoren hintereinander editierbar sein sollen, Zeile 6090 ergänzt werden (sz = Sektoranzah1):

```
6090 LET a=a+l: IF a<256*sz THEN GOTO 6030
```

Dann empfehle ich noch, den STOP-Befehl in Zeile 910 durch ein GOTO 200 zu ersetzen, damit man aus dem TRDOS sofort wieder ins "doctor"-Programm zurück kommt ohne den Umweg über's BASIC!

Nun muß noch in Zeile 8040, 8100, 8120, 8336, 8360, 8365, 8405, 8430 und 8445 der PRINT-Befehl ergänzt werden zu 'PRINT #ch;' und Zeile 8460 wird folgendermaßen erweitert:

```
8460 LET ch=2: PRINT #ch; "press key for menu";: PAUSE 0: PAUSE 0: RETURN
```

Die Initialisierung des Druckers sollte etwa in einer Zeile 202 geschehen. Bei meinem Kempston E-Interface brauche ich dazu nur

```
202 COPY: REM /1
```

eingeben (= Line feed und Cariage return on)! Hier können auch noch andere Befehle an den Drucker, z.B. Zeilenabstand oder Schrift ändern, eingebaut werden. Natürlich läßt sich das auch erst ab Zeile 8400 einbauen, wenn man will. Die Beispiele sollen schließlich nur als Anregung und kleine Starthilfe dienen!

Da das Handbuch leider kaum Angaben zur Handhabung des Disk-Monitors macht, noch ein paar Tips zur Adresseneingabe:

Wenn bei Menu-Punkt 6 bzw. 7 die Startadresse abgefragt wird, hat man mehrere Eingabemöglichkeiten und zwar dezimal oder hexadazimal. Letzteres wird durch ein nachgestelltes 'H' oder 'h' gekennzeichnet! Dazu Beispiele:

```
Dezimal    :  127
Hexadezimal:  7fh, 7FH, 7fH, 7Fh, 07fh, 007Fh usw.
```

Es können also Hex-Zahlen mit 1-4 Stellen angegeben werden in beliebiger Mischung von Groß- und K1einschrift! Zu beachten ist lediglich, daß Startadresse nicht größer sein sollte als 'Sektorzahl * 256'. Beim Ändern von Sektorinhalten mit Menu-Punkt 7 werden nur Hex-Zahlen akzeptiert, zweistellig und ausnahmsweise ohne nachgestelltes 'h'!

Um die geänderte Version des 'doctor' abzuspeichern, kann man noch folgende Zeilen anhängen:

```
9990  RANDOMIZE USR 15619: REM: SAVE "doctor" LINE 5
9991  RANDOMIZE USR 15619: REM'. SAVE "secread" CODE 60000,90
```

Mit RUN 9990 aufgerufen, wird die neue Version am Besten auf eine neue Disk gespeichert. Es empfiehlt sich nicht, die Originaldisk dafür zu benutzen, denn die sollte immer schreibgeschützt in Reserve bleiben.
Nun Schluß für diesen Monat! Wenn die Zeit reicht, habe ich vor, daß nächste Mal u.a. zu erläutern, wie man gelöschte Files wieder retten kann und anderes mehr. Bis dann!

MfG
Wilhelm