Soppsches

Neuer Drivetesten-Befehl ",,/"
Die Library vergrößern
Mehrere Sysfiles gleichzeitig
Neuer DOS-Befehl OUT port#, xx
SYS-Files und wie man sie macht
Die Library-Befehle des G-DOS
Die Records handhaben
BASIC frei im RAM verlegen
PUT TO adresse - ein neuer BASIC-Befehl (s.o.)
Die Memory Size automatisch setzen
Kaffee kochen ab sofort gestattet (BEL bei Fehlern?)
BEL ohne Nachladen autom. bei BOOT (siehe oben?)
DSMBLR ein wenig komfortabler
Mehr über DDE
LPRINT CHR$ irgendwas
Und es geht doch: LPRINT CHR$(10)
ColourGenie liest NEWDOS und umgekehrt?
Wie ist GENIETEXT aufgebaut
Die Systemoptik ausgetrickst?
ITEM: PRINT @
Convert SCRIPSIT → ASCII

Neuer Dreitastenbefehl ",./"

Das mehrfach erwähnte Problem, innerhalb des residenten Teils von SYS0/SYS eine freie Stelle für weitergehende Features zu finden, ist bisher nur unbefriedigend gelöst: Für das heute vorgestellte Programm mußte vorerst der DOS-Eingabepuffer herhalten. In dessen oberem Teil gehen nun 11 Bytes verloren, so daß sich die maximale Befehlslänge von 80 Zeichen (inkl. NEW LINE, also 79 Schriftzeichen) auf gut eine Bildschirmzeile verringert. Das könnte alle halbe Jahre problematisch werden. Trotzdem, wer eine bessere Stelle in SYS0 weiß, bitte weitersagen!

Das Programm, das durch die Zaps in diesem Beitrag aufgerufen wird, soll bei mir mal wieder den Memory-Banking-Adapter EG 64 MBA unterstützen - ich weiß, Ihr habt's geahnt. Dieses Programm stelle ich aber nicht vor, sondern nur eine Methode, wie man es aufrufen kann. Folglich könnt Ihr auf diese Weise ein beliebiges Maschinensprache-File starten, meinetwegen Space Invaders.

Eine Memory-Banking-Utility oder ein entsprechendes Programm Eurer Wahl ist ihrer/seiner Natur nach ein Bestandteil des Betriebssystems. Es sollte deshalb möglich sein, es jederzeit, und sei es mitten in einem BASIC-Programm, aufzurufen, ähnlich DEBUG, um anschließend nahtlos fortfahren zu können. Das ist genau die Anforderung, die an ein SYS-File gestellt wird. So ist es nur logisch, hierfür eine SYS-Datei zu kreieren, die nach Möglichkeit auch im Bereich 4D00-51FFh arbeitet, so daß der Anwenderspeicher frei bleibt.

G-DOS/NEWDOS kann 32 Systemdateien ansprechen. Zwei davon heißen GDOS/SYS (BOOT/SYS) und INHALT/SYS (DIR/SYS). Die übrigen sind durchnumeriert von SYS0 bis SYS29. Sie alle existieren bereits auf der Systemdiskette. Also keine Chance mehr? Schaut man sie sich der Reihe nach mit DDE bzw. SUPERZAP/CMD an, stellt man fest, daß einige (je nach DOS verschiedene) zwar eingerichtet sind, aber nichts als Nullen enthalten (FFh bei TRSDOS, soviel ich weiß). Sie werden vom System auch nicht aufgerufen. Diese Files sind frei verfügbar und können für das hier besprochene Anliegen herhalten!

Ein SYS-File zu erstellen, ist überhaupt kein Problem. Man kann unter dem Namen z. B. SYS26/SYS ein eigenes Maschinenprogramm assemblieren. Es wird dann genau den Diskettenplatz belegen, der für das bisher funktionslose SYS-File reserviert war. Oder man kopiert ein fertiges Programm mit COPY,filename/typ:dr,SYS26/SYS:0. Das Resultat ist dasselbe. Dabei spielt es auch keine Rolle, welchen Ladebereich das Programm belegt. Bei meinen Tests stellte sich allerdings eine Bedingung heraus, deren Gründe ich noch nicht durchschaue: Wenn das File länger als der vom SYS-File belegte Platz auf der Diskette ist und deshalb Extensions kreiert werden, wird die Datei beim Aufruf nicht gefunden. Daher muß zuvor sichergestellt sein, daß die Anzahl der benötigten Sektoren höchstens gleich ist.

Zur Sache: Wenn von der Floppy ein INT ankommt, landet seine Bearbeitung schließlich bei 45F2h. Die Befehle, die dort nach dem Retten der Register (PUSH) stehen, werden zunächst abgearbeitet, bis auf den letzten vor dem Restaurieren der Register (POP): CALL 45BEh wird in CALL 435Dh umgezapt. Ab 435Dh bis zum Ende des Eingabepuffers steht nun eine kleine Routine, die die Speicherstelle 3820h abfragt. Dort stehen bestimmte Werte, wenn eine der Satzzeichentasten gedrückt wurde. Waren es Komma, Punkt und Querstrich gleichzeitig, heißt dieser Wert 0Dh bzw. 208d. Trifft diese Bedingung nicht zu, wird endlich 45BEh angesprungen.

Andernfalls wird es jetzt interessant: Ein Dreitastenbefehl, da er z. B. den Bildschirm sauber läßt, ist so wertvoll, daß damit auch längere Routinen aufrufbar sein sollten. Deshalb die Notwendigkeit eines eigenen

SYS-Files, denn in SYS0/SYS ist es dafür zu eng. Eine Systemdatei wird mit RST 28h aufgerufen. Dieser Befehl braucht einen Parameter im Akku (Register A). Ihn zu bestimmen, ist ein bißchen kompliziert:

In binärer Schreibweise muß er das Format

uuubbsss

haben. Dabei bedeutet sss+2 den relativen Sektor im Directory, in dem der Eintrag des SYS-Files steht. Das hieße beim Sektor 2 (gezählt ab 0) z. B. sss=0. Diese drei Bits können maximal die Zahl 7 darstellen. Inkl. 0 sind das 8 ansprechbare Sektoren. bb bedeutet die laufende Nr. des Fileeintrags in diesem Sektor, ebenfalls gezählt ab 0. Diese beiden Bits ermöglichen nach dem obigen Strickmuster die Adressierung der vier oberen Dateien dieses Sektors. So errechnet sich die Möglichkeit, insg. 32 Files mit RST 28h zu laden. Und uuu kann zur Übergabe eines Parameters genutzt werden. Lt. NEWDOS-Handbuch muß uuu auf jeden Fall > 0 sein. Es scheinen noch mehr Bedingungen daran zu hängen, so daß man sicherheitshalber immer das Bit 7 setzen sollte. Dieses Berechnungsschema für den Parameter im Akku steht im deutschen Handbuch auf S. 168.

Hier nun das konkrete Fallbeispiel, wieder für SYS26/SYS: Diese Datei steht im 6. relativen Sektor des Inhaltsverzeichnisses (inkl. 0. Sektor). 6-2=4, also muß sss 4 lauten bzw. 100 binär. Es ist, die 0 wieder eingeschlossen, der dritte Eintrag in diesen Sektor. bb heißt deshalb 11 binär (=3). Soll ohne weitere Parameter das Bit 7 gesetzt sein, ergibt sich ein uuu-Code von 100b. uuu, bb und sss addieren sich ihrem Stellenwert entsprechend zu 10011100b, also 9Ch bzw. 156d. Demnach brauchen wir, um SYS26/SYS aufzurufen, die Befehlssequenz LD A,9CH - RST 28H.

Uff! Nachdem Euch die Lektüre dieser Erläuterung zunächst abgeschreckt haben dürfte, macht Euch bitte die Mühe, es ein paarmal zu probieren. Spätestens nach der fünften Berechnung schafft Ihr es bereits im Kopf!

Werden nun die Tasten <,./> gleichzeitig gedrückt, wird SYS26/SYS geladen und angesprungen. Was dieses File leisten soll, ist Euch überlassen. Es könnte z. B. ein hübscher Bildschirm in einen anderen Speicherbereich gerettet oder auf die Floppy ausgelesen werden. Der Phantasie sind nur die Grenzen Eurer Kreativität gesetzt.

Auf jeden Fall muß das Programm mit RET abschließen. Und zwar deshalb: An 4600h (s. Assemblerlisting) steht ein CALL nach 435Dh. Bei erfolgreicher Tastaturabfrage wird der RST-Befehl angesprungen, der ebenfalls für die CPU ein CALL ist, also den Stack um zwei Bytes nach unten verlängert. Bei der Bearbeitung des Befehls RST 28h wird allerdings der Stackpointer zweimal inkrementiert, so daß wir uns wieder in der ersten CALL-Ebene befinden. Daher genügt das simple RET, mit dem das Programm schließlich am Ende der INT-Serviceroutine weitermacht, von wo wir ursprünglich gestartet sind.

Das Assemblerlisting zeigt den Programmablauf der beiden Zaps in SYS0/SYS. Die Dumps darunter sind die Hex-Codes, die in das File gezapt werden müssen. Es genügen wie immer die unterstrichenen Codes. Die ersten 4 Stellen der linken Spalte bezeichnen den relativen Sektor des Files, die beiden folgenden Hex-Ziffern vor dem Doppelpunkt verweisen auf das relative Byte dieses Sektors, mit dem die angezeigte Zeile beginnt.

Der fortgeschrittene Leser wird bemerken, daß das Byte 4 des Sektors 0Eh nicht zum Programm gehört und dennoch umgezapt wird. Es hat mit der Record-Organisation zu tun. Auch die ersten vier Bytes ab E7h im selben Sektor erscheinen nicht im Listing. Dieser Artikel wäre endgültig unverdauliche Kost, wenn ich jetzt auch noch dieses erklären wollte. Nehmt es hin und zapt!

Da ich gerade Eure Aufmerksamkeit habe (danke!) noch kurz zu einer anderen Sache: Seit einiger Zeit erstelle ich mit Hilfe von DSMBLR, DEBUG, SUPERZAP, ziemlich viel Bier, EDTASM, DDE usw. einen Assembler-Quellcode von SYS0/SYS. Im Gegensatz zu den ziemlich abstrakten Hex-Codes, die uns z. B. SUPERZAP an den Kopf wirft, ist diese Source direkt lesbar. Mit ihrer Hilfe hoffe ich, zumindest in groben Zügen die Alchimistenküche dieser Datei zu durchschauen. Die Source ist bereits recht reichlich kommentiert, Befehlscodes sind von reinen Datencodes (hoffentlich vollständig) unterschieden.

Wer Interesse an diesem Quelltext hat, kann mir zu diesem Zweck eine formatierte Diskette (G-DOS oder NEWDOS) schicken. Für das Porto und einen speziellen "Floppy Disk Shipper" (Versandtasche für Disketten) fügt bitte DM 2,- in Briefmarken bei (bitte gängige Werte). Um mir Herumprobiererei zu ersparen, legt auch bitte einen Zettel mit den FDRIVE-Parametern dazu. Zwischen 80/DS/DD und 40/SS/DD nimmt meine Disco alle Formate an.

Und hier das Listing:

```
SYS0/ZAP                        SOFTSOFT-Software     00:02 23 Jun 84 Seite 1

                00100           TITLE   SYS0/ZAP
                00110
4600            00120           ORG     4600H           ;Ende der INT-Routine
4600 CD5D43     00130           CALL    435DH           ;Ergänzung aufrufen
                00140
435D            00150           ORG     435DH           ;Ende Eingabepuffer
435D 3A2038     00160           LD      A,(3820H)       ;Tastatur Satzzeichen
4360 FED0       00170           CP      0D0H            ;<,./> gedrückt?
4362 C2BE45     00180           JP      NZ,45BEH        ;sonst normal weiter
4365 3E9C       00190           LD      A,9CH           ;Code für SYS26/SYS
4367 EF         00200           RST     28H             ;laden und bearbeiten
                00210
0000            00220           END
00000 mal gepennt
```

Bemerkung: Wer EDTASM plus hat und in der Kopfzeile seiner Listings immer noch Reklame für Microsoft macht, hat mit denen wohl einen Vertrag. Es ist ganz einfach, diesen Text den eigenen Bedürfnissen anzupassen (s. o.).

Die Sektordumps auf der nächsten Seite sind mit DDE erstellt. Mit SUPERZAP wird SYS0/SYS mit "DFS" aufgerufen, anschließend kann man die beiden relativen Sektoren, wie oben beschrieben, anzeigen und ändern.

Sekt. Ø2h, alt

```
000200: 79DA 7D04 C358 043A 6943 EE20 E668 3ECB y.ü..X.:iC. .h>.
000210: CCDD 49C8 21BE 4536 C9E5 2136 407E B73E ..I.!.E6..!6§ß.>
000220: 0020 0332 8045 AFF6 0018 09B6 2004 3D32 . .2.E..... .=2
000230: 7345 3600 2E36 0101 3816 FF0A 5FAE 73A3 sE6..6..8..._.s.
000240: 200D 7AC6 0857 2CCB 01F2 4A45 AF18 0F5F  .z..W,...JE..._
000250: 140F 30FC C5D5 222A 45CD 0B04 E1C1 B757 ..0..."*E.....W
000260: 2818 7CFE FF32 7345 3E00 2006 0AA5 28EE (.ö..2sE>. ...(.
000270: 3EFF 3237 453E 0232 8045 CDBF 45E1 3600 >.27E>.2.E..E.6.
000280: FE1F 281D C9E6 DFD6 41FE 1F79 3815 FE20 ..(.....A..y8..
000290: C021 7F38 7E23 A60F 79D0 21B4 457E EEC9 .!.8ß#..y.!.Eß..
0002A0: 77AF C9EE 20C9 FE61 D8FE 7FD0 D620 C900 w... ..a..... ..
0002B0: 2169 437E E66C 2026 3A01 38FE D018 053E !iCß.l &:.8....>
0002C0: E30E 04EF 3A10 38FE 0E18 0E3A BE45 D6C9 ....:.8....:.E..
0002D0: CA0D 44F1 C1D1 E118 223A 0238 FE1C 7AC9 ..D.....":.8..z.
0002E0: 3EA5 EFF5 E5D5 C53A E037 07CD 5301 00FC >.....:.7..S...
0002F0: 4540 DC10 46CD BE45 C1D1 E1F1 FBC9 F53E E§..F..E.......>
```

Sekt. Ø2h, neu

```
000200: 79DA 7D04 C358 043A 6943 EE20 E668 3ECB y.ü..X.:iC. .h>.
000210: CCDD 49C8 21BE 4536 C9E5 2136 407E B73E ..I.!.E6..!6§ß.>
000220: 0020 0332 8045 AFF6 0018 09B6 2004 3D32 . .2.E..... .=2
000230: 7345 3600 2E36 0101 3816 FF0A 5FAE 73A3 sE6..6..8..._.s.
000240: 200D 7AC6 0857 2CCB 01F2 4A45 AF18 0F5F  .z..W,...JE..._
000250: 140F 30FC C5D5 222A 45CD 0B04 E1C1 B757 ..0..."*E.....W
000260: 2818 7CFE FF32 7345 3E00 2006 0AA5 28EE (.ö..2sE>. ...(.
000270: 3EFF 3237 453E 0232 8045 CDBF 45E1 3600 >.27E>.2.E..E.6.
000280: FE1F 281D C9E6 DFD6 41FE 1F79 3815 FE20 ..(.....A..y8..
000290: C021 7F38 7E23 A60F 79D0 21B4 457E EEC9 .!.8ß#..y.!.Eß..
0002A0: 77AF C9EE 20C9 FE61 D8FE 7FD0 D620 C900 w... ..a..... ..
0002B0: 2169 437E E66C 2026 3A01 38FE D018 053E !iCß.l &:.8....>
0002C0: E30E 04EF 3A10 38FE 0E18 0E3A BE45 D6C9 ....:.8....:.E..
0002D0: CA0D 44F1 C1D1 E118 223A 0238 FE1C 7AC9 ..D.....":.8..z.
0002E0: 3EA5 EFF5 E5D5 C53A E037 07CD 5301 00FC >.....:.7..S...
0002F0: 4540 DC10 46CD 5D43 C1D1 E1F1 FBC9 F53E E§..F.üC.......>
```

Sekt. ØEh, alt

```
000E00: 0000 0001 F1E6 5000 0000 0000 0000 0000 ......P.........
000E10: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E20: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E30: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E40: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E50: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E60: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E70: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E80: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E90: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EA0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EB0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EC0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000ED0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EE0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EF0: 0000 0000 0000 0104 2840 4800 0202 004D ........(§H....M
```

Sekt. ØEh, neu

```
000E00: 0000 0001 E2E6 5000 0000 0000 0000 0000 ......P.........
000E10: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E20: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E30: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E40: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E50: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E60: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E70: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E80: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E90: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EA0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EB0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EC0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000ED0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EE0: 0000 0000 0000 0001 0D5D 433A 2038 FED0 .........üC: 8..
000EF0: C2BE 453E 9CEF 0104 2840 4800 0202 004D ..E>....(§H....M
```

Arnulf Sopp, Tel. Ø451-791926

Die Library vergrößern

Die "Bibliothek" (engl. library) des G-DOS, H-DOS und NEWDOS/80 steht in SYS1, verteilt auf zwei Sektoren. In G-DOS 2.1 für das Genie 3 und in H-DOS für die Modelle 1 und 2 ist sie wegen der zusätzlichen Befehle nahezu randvoll. Gleichwohl ist am Ende von SYS1 noch etwas Platz. Der Haken ist aber, daß die Library eben nicht dort, sondern einen Sektor früher zuende ist. Es folgt die Bearbeitungsroutine für den LIB-Befehl nahezu unmittelbar.

Um für weitere Befehle Platz zu schaffen, gibt es ein paar Möglichkeiten. Z. B. ist in G-/H-DOS der Befehl LIB überflüssig, weil das Befehlsverzeichnis auch mit ? ausgegeben werden kann. Dasselbe gilt für mehrere andere Befehle. Aber eine Erweiterung der Library mit einer Verkürzung erkaufen? Dabei ergäbe sich auch das Problem der Kompatibilität meines H-DOS mit G-DOS. Also änderte ich zunächst nur solche Befehlswörter ab, die es nur in H-DOS gibt: INIT heißt jetzt INI, BANK? wurde zu B?. Der neue Befehl ID, mit dem PDRIVE-Parameter von Disketten automatisch erkannt werden, hat nun Platz.

Gut und schön, es soll aber nicht der letzte Mögliche gewesen sein. Mit einer etwas komplizierteren Manipulation gab es noch einmal Platz für einen weiteren Befehl, der genau ein Zeichen lang sein darf. Nach diesem Zeichen müssen noch drei weitere Bytes vorhanden sein für einen Code, der im Register C an das aufgerufene SYS-File übergeben wird, den Requestcode, der in den Akku kommt und ein Schlußbyte, das anzeigt, ob dieser Befehl die Angabe von zusätzlichen Operanden/Parametern erlaubt. Diese insgesamt vier Bytes wurden auf folgende Weise frei:

Die oben erwähnte LIB-Bearbeitungsroutine beginnt mit dem Befehl LD HL,4F58. Damit wird HL als Zeiger mit dem Beginn der Library geladen. Dieser Befehl kann ohne weiteres an eine andere Stelle verschoben werden. Im Prinzip ginge das auch mit den folgenden Maschinencodes, aber aus Gründen, die zu erläutern hier zu weit führen würde, hätte das einen Rattenschwanz von zusätzlichen Änderungen zur Folge. Also nur drei Bytes Gewinn, aber das reicht immerhin für einen neuen Befehl (hat jemand eine Idee, was der bewirken sollte?).

Der LIB-Befehl wird nun auf das Ende von SYS1 umgeleitet. An der Stelle 51E2 wird zuerst der erwähnte Ladebefehl für den Zeiger nachgeholt und sodann an die nächste Stelle in der alten Routine gesprungen. Um dies aber zu ermöglichen, mußte der letzte Record von SYS1 bis ans Ende des letzten Sektors verlängert werden. Das Zählbyte im Record-Header wurde entsprechend geändert, der EOF-Code rutschte ganz nach hinten.

Das war für die Theoretiker, hier nun die Praxis: Auf der letzten Seite dieses Beitrags stehen drei Sektordumps, in denen die modifizierten Bytes unterstrichen sind. Im 0. Sektor wurde der Sprungvektor für den LIB-Befehl auf die neue Adresse gezapt. Im 3. Sektor sind die drei Bytes gekennzeichnet, wo zuvor der Ladebefehl stand. Sie sind nun frei. Im letzten Sektor lautet das Record-Zählbyte jetzt EA, weil noch EAh = 234d Bytes folgen. Am Ende des Records stehen zwei NOPs dort, wo zuvor das EOF war. Es folgen der HL-Ladebefehl und der Sprungbefehl in die alte Routine, schließlich das neue EOF.

Wer gerne an seinem DOS herumzapt, sollte sich unbedingt auf die unterstrichenen Bytes beschränken. Es gibt in H-DOS noch eine ganze Reihe weiterer abgeänderter Bytes, auch in den hier abgebildeten Sektoren. Bitte unbedingt ignorieren, denn diese Änderungen beziehen sich auf wieder andere Stellen im DOS, ohne die eine Änderung das System zuverlässig zum Absturz bringt. Eltern haften für ihre Kinder!

Wem erzähle ich das? Mir ist inzwischen bekannt, daß ich nicht der einzige bin, der DOS-Erweiterungen ausheckt (warum laßt ihr anderen im Info kaum von euch hören?). Wem's in der Library zu eng wird, der kann nun etwas dagegen tun. Das gilt besonders für Benutzer des Genie 3, die in dieser Beziehung ganz schön angemeiert sind.

Arnulf Sopp

```
DRV 00   0102 004D FE23 CA8A 4DFE 4328 73FE 63CA ...M.#..M.C(s.c.
0   10   304E FE83 CA55 51FE A3CA 2A4F FEC3 283F 0N...UQ...*O..(?
0H  20   0D28 3A0D CAE2 510D CA32 4D0D CAF3 500D .(:...Q..2M...P.
     30  2852 0DCA 1251 0D28 430D CA34 4E0D 280C (R...Q.(C..4N.(.
DRS 40   0D0D CA68 510D 283C 3E2A B7C9 21DD 51C3 ...hQ.(<>*..!.Q.
143050   6744 D901 01E3 11D3 49C5 D5D9 EFF1 C9CD gD......I.......
596H60   4351 0100 00EB 216A 43CB 76CB F628 04ED CQ....!jC.v..(..
     70  4B9D 43C5 ED73 9D43 EBC3 354E F1F1 180F K.C..s.C..5N....
     80  AF37 180B 216A 437E E62F 772B CBAE AFF3 .7..!jC~./w+....
     90  216B 4336 002B 462B 4E1E 0BF5 CB50 2023 !kC6.+F+N....P.#
     A0  F1F5 3802 280A FE38 2806 1E04 CB69 2013 ..8.(..8(....i..
     B0  CBB6 CB70 205E 3A6C 43CB 7728 0BCB 6920 ...p.^:lC.w(..i.
FRS  C0  071E 0C16 EB7A 4BEF CB78 2014 31E0 41CB .....zK..x..1.A.
0    D0  6F21 B045 2213 433E C328 023E C932 1243 o!.E".C>.(.>.2.C
0H   E0  21CD 51CB 7828 07ED 7B9B 4321 C851 FB3E !.Q.x(..{.C!.Q.>
     F0  1ECD 3300 CB69 CC67 4421 6A43 CBEE 0108 ..3..i.gD!jC....

DRV 00   0049 802A 0049 4E46 4F81 FF00 0102 0050 .I.*.INFO......P
0   10   4A4B 4C80 A510 4B49 4C4C 8045 904C 4388 JKL...KILL.E.LC.
0H  20   E500 4C46 81FE 004C 4942 82E3 004C 4953 ..LF...LIB...LIS
     30  5485 F088 4C4F 4144 80A4 504C 5754 81F9 T...LOAD..PLWT..
DRS 40   004E 81E4 B04E 4446 C028 0050 4155 5345 .N...NDF.(.PAUSE
143350   88EB 0050 4483 E900 504F 5254 82FF 0050 ...PD...PORT...P
599H60   5249 4E54 86F0 8850 524F 5486 E900 5055 RINT...PROT...PU
     70  5247 4589 E900 5280 2300 5381 E900 5354 RGE...R.#.S...ST
     80  4D54 89EB 0055 4852 82E5 0056 2B84 E500 MT...UHR...V+...
     90  5632 3480 FA00 5A86 FF00 5A45 4954 8AE9 V24...Z...ZEIT..
     A0  0026 83E5 0021 83EB 8A3B 86E3 002F 85E3 .&...!...;.../..
     B0  003F 82E3 003E C048 004D 3E82 EBB0 4444 .?...>.H.M>...DD
FRS  C0  45C0 F100 494E 4980 BA00 423F 80DF 002A E...INI...B?...*
3    D0  809C 004F 5554 87FF 0049 4480 F800 0000 ...OUT...ID.....
3H   E0  0000 0E40 0608 7ECB 7F23 2005 CDB7 5110 ...@..~..#....Q.
     F0  F523 237E B7CA B551 0DCC B551 28E4 CDAD .##~...Q...Q(...

DRV 00   5118 E1F3 CD43 5121 6A43 7EE6 C020 333A Q....CQ!jC~...3:
0   10   01EA 0051 2240 F5ED 739B 43CB FEFB 3E0D ...Q"@..s.C...>.
0H  20   CD33 00C3 8A4D 216A 43CB 7ECA 444D ED7B .3...M!jC.~.DM.{
     30  9B43 3E0E CD33 00F1 B747 3E0F CC33 0078 .C>..3...G>..3.x
DRS 40   3222 40F3 CBBE AF08 FDE1 DDE1 F1C1 D1E1 2"@.............
143450   D9C1 D1E1 08FB C9F1 E5D5 C508 D9E5 D5C5 ................
59AH60   F5DD E5FD E5D9 08F5 C9CD 6851 F57E D603 ..........hQ.~..
     70  2802 D60A 2801 23F1 C911 8044 D506 20CD (...(.#....D....
     80  7251 D106 00C9 7EFE 2A20 0412 1323 05E5 rQ....~.*....#..
     90  7ED6 30FE 0ACD A151 3016 7ED6 2EFE 0DCD ~.0....Q0.~.....
     A0  A151 3806 3E03 12F1 AFC9 7E12 1323 10EA .Q8.>.....~..#..
     B0  F601 E17E C9D8 7ED6 41FE 1FD8 D620 FE1F ...~..~.A.......
FRS  C0  C93E 20CD B751 10F9 C93E 0DD5 F5CD 3300 .>...Q...>....3.
4    D0  F1D1 C943 4D44 4A4F 4254 4F00 074D 202D ...CMDJOBTO..M.-
4H   E0  204B 6F6D 6D2C 2068 6175 2072 6569 6E3A .Komm,.hau.rein:
     F0  2003 0000 0000 2158 4FC3 D250 0202 004D ......!XO..P...M
```

Mehrere SYS-Files gleichzeitig

Einige Systemdateien von NEWDOS, G-DOS, H-DOS, TRSDOS usw. arbeiten mit anderen zusammen. So werden beispielsweise von der G-DOS/H-DOS-Funktion DDE in SYS15/SYS weitere SYS-Files aufgerufen, um Sektoren zu laden, zu schreiben, das Inhaltsverzeichnis zu sichten usw.. Da sich diese Dateien nicht überlagern dürfen, hilft sich DDE mit dem sicherlich einfachsten Ausweg: Der Ladebereich von SYS15/SYS beginnt bei 5200.

Es ist kaum anzunehmen, daß man mitten in einem gerade laufenden Programm mit DDE eine Datei modifizieren will. Deshalb ist es unproblematisch, daß SYS15/SYS den unteren Teil des Anwenderspeichers zuschaufelt. Soll jedoch eine Systemdatei im Hintergrund jederzeit, also auch mitten in einem Programm abrufbar sein, ohne anderen Systemdateien den Platz wegzunehmen, muß man sich etwas einfallen lassen.

Bei meinem H-DOS war diese Notwendigkeit gegeben. Darin spielt der EG 64 MBA eine entscheidende Rolle. Er wird von mehreren SYS-Files angesteuert. Da in einigen von ihnen der Platz sehr beschränkt war, sollte ein gemeinsames Unterprogramm für alle da sein. Ich entschied mich für SYS24/SYS, eine freie Systemdatei (die beim Genie 3 übrigens belegt ist). Für den Ladebereich bleibt nicht mehr viel Auswahl: Nur noch der DOS-Eingabepuffer 4318-4367 und der Sektorpuffer 4200-42FF stellen unterhalb 5200 noch eine nennenswerte Lücke dar.

Wenn man wegen der Mutterschaft der Vorsicht über die Porzellankiste alle 5 Sektoren von SYS24/SYS erhalten will, entsteht nun ein gravierendes Problem: In den Sektorpuffer wird alles geladen, auch Records, die mit dem Code 05 (s. Sektordumps) als das gekennzeichnet sind, was in BASIC REM heißt. Das bedeutet, daß der letztenendes aktive Record, der direkt im Sektorpuffer laufen soll, von allem Nachfolgenden verschüttet würde. Folglich darf nichts nachfolgen. Dieser Record muß im unwiderruflich letzten Sektor stehen. Alles Davorliegende muß entweder "REM" (05) heißen, also für den Programmablauf belanglos sein oder in einen anderen Speicherbereich laden, z. B. den Input Buffer.

Es gibt eine weitere Besonderheit, die zu beachten ist: Sollte von einer solchen SYS-Datei aus eine andere aufgerufen werden, so muß die Kontrolle endgültig an diese übergehen, denn ihre Sektoren werden selbstverständlich ebenfalls über 4200 ff. an ihren Bestimmungsort geladen, wobei sie unseren Spezialfile in die ewigen Jagdgründe schicken. Kurz: Während unser SYS24/SYS arbeitet, ist jegliche Disk-I/O ganz einfach verboten.

Um SYS24/SYS aufzurufen, gibt es zwei Möglichkeiten. Auf jeden Fall muß der Akku mit xA geladen sein, wobei x eine ungerade Zahl >= 3 ist. Warum, das will ich jetzt nicht erklären, weil es zu weit führen würde. Der Leser wolle sich bitte mit dem Aufruf von SYS-Dateien in der einschlägigen Literatur vertraut machen, z. B. im Clubinfo des Genie/TRS80 User Club Bremerhaven. Die eine Möglichkeit besteht darin, direkt mit dem Befehl RST 28H SYS24/SYS anzuspringen. Das hat dieselbe Wirkung wie ein GOTO in BASIC. Oder man callt eine der vielen Stellen im DOS-Kern 4000-4CFF, wo ein RST 28H steht. Das kommt einem GOSUB gleich, so daß nach Erledigung an der alten Stelle fortgefahren werden kann.

Nanu!? Angeblich ist ein RST doch ein Unterprogrammaufruf wie ein CALL! Der Schlüssel zu dieser Merkwürdigkeit liegt in der Bearbeitungsroutine von RST 28. An 4BC2 wird nämlich der Stackpointer zweimal inkrementiert, wodurch die RET-Adresse vom Stack "verschwindet" (richtiger: Ein RET würde die falsche, die darüberliegende Adresse finden).

Mein MBA-Problem, das auf diese Weise gelöst werden konnte, ist nur ein Beispiel. Es wäre interessant, in einem der kommenden Infos weitere

Anwendungen dieser Technik aus den Reihen des Clubs zu finden.

Noch etwas zu den Sektordumps: Da die Sektoren 0-3 völlig identisch sind, sind nur pars pro toto der Sektor 0 und als einziger Programmmsektor der Sektor 4 wiedergegeben. Das erste Byte jedes Sektors bedeutet die Gattung seines Inhalts. Sektor 0 ist Füllwerk und deshalb mit 05 gekennzeichnet. In Sektor 4 steht ein Maschinenprogramm, das mit dem Code 01 beginnt.

Arnulf Sopp

SYS24/SYS, Sekt. Ø

```
000000: 05FE 0000 0000 0000 0000 0000 0000 0000 ................
000010: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000020: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000030: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000040: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000050: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000060: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000070: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000080: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000090: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0000A0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0000B0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0000C0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0000D0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0000E0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
0000F0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

SYS24/SYS, Sekt. 4

```
000400: 01E2 0042 CD54 4428 0B7E E6DF FE4A 2877 ...B.TD(.ß...J(w
000410: FE4E 2804 3E2F 182B CD5D 42DB DF21 5348 .N(.>/.+.ÜB..!SH
000420: 2208 453E 3A32 D345 2110 3822 D445 AF18 ".E>:2.E!.8".E..
000430: 13E5 D5C5 F3FE 9A28 4EFE BA28 C7FE DA28 .......(N..(...(
000440: D73E 37B7 C1D1 E1FB C9FE FA20 E43E 0BD3 .>7........ .>..
000450: DF3D D3DF 21FF 3536 007E B73E 08C0 3EFD .=..!.56.ß.>..>.
000460: EF06 043E 0ED3 DF3D 10FB 3DD3 DF01 0037 ...>...=..=....7
000470: 6169 5159 EDB0 707E B728 05F1 3E08 18C3 aiQY..pß.(..>...
000480: 1C01 FF08 EDB0 C9CD 5D42 21B5 4211 7500 ........ÜB!.B.u.
000490: D501 2B00 EDB0 2185 0022 0845 AF32 FF2F ..+...!.."E.2./
0004A0: 32FF 35DB DF3E 08D3 DF3E 0FD3 DFE1 3ECD 2.5..>...>....>.
0004B0: 32D3 4522 D445 AF18 8B3A 2038 FED0 2806 2.E".E...: 8..(.
0004C0: 3A10 38FE E0C0 F61C EF79 FE07 C253 480E :.8......y...SH.
0004D0: B041 3E01 D3FF 10FE 413C D3FF 10FE 0D41 .A>.....A<.....A
0004E0: 10F0 F1C9 0202 4542 0000 0000 0000 0000 ......EB........
0004F0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
```

Neuer DOS-Befehl: OUT port#,xx,yy,...

Wenn man sich sein DOS näher ansieht, wird man je nach Version feststellen, daß möglicherweise nicht jede Funktion der Library auch wirklich lauffähig ist. Bei meinem G-DOS 2.1b (das ich inzwischen mal all' den Zaps frech 2.1c nenne) ist beispielsweise der Befehl V24 wirkungslos. Er soll eigentlich die V24-Schnittstelle, sofern eingebaut, initialisieren. Gibt man diesen Befehl ein, erscheint jedoch die Meldung, daß diese Funktion noch nicht implementiert sei. Der Speicherbereich, wo diese Meldung steht und die Routine, die sie anzeigt, stehen deshalb für Sinnvolleres zur Verfügung.

Eine der großen Stärken des Z80 ist seine Fähigkeit, 256 Ports anzusprechen. In BASIC ist das sehr einfach durch den OUT-Befehl. Vom DOS aus geht es aber nicht (Geduld, gleich geht es!). Wer immer in BASIC arbeitet, kann jetzt weiterblättern, falls er nicht sein DOS für den Wiederverkauf aufwerten will. In Maschinensprache ist die Ausgabe einer Größe auf einen Port zwar genauso simpel, es gibt dafür aber keinen DOS-Befehl. So war das hier vorgestellte Programm kein Problem, nur die Frage, wie man diesen Befehl implementieren könnte.

In SYS1/SYS sind alle DOS-Befehle gespeichert. Das System checkt bei einer Eingabe, ob einer dieser Befehle eingetippt wurde. Ist das der Fall, wird die entsprechende Routine geladen und abgearbeitet. Dabei wird bei dieser Befehlstabelle nicht überprüft, ob sie die Original-Apparat-Befehle oder die TCS-Verballhornungen enthält. Man kann daher problemlos z. B. "LOAD" in "LADE" umzappen. Mit etwas mehr Aufwand wäre sogar "SCHLÜRFE DIR 'REIN" möglich. So änderte ich kurzerhand "V24" in "OUT" um (oberer Sektordump).

Damit allein ist es allerdings nicht getan, denn der OUT-Befehl soll bitteschön auch befolgt werden, und dazu braucht es eine Bearbeitungsroutine. Nach dem Befehl V24 wird nach 519E in SYS29/SYS verzweigt, wo die Routine zur Anzeige der Denkste-Meldung steht. Die wiederum steht ab 515E. Also ist der ganze Bereich dazwischen verfügbar. Um Platz zu sparen, wird deshalb der Sprungbefehl am Anfang von SYS29/SYS auf 515E umgezapt (mittlerer Sektordump). Ab 515E folgt nun die Bearbeitungsroutine (unterer Dump). Und die geht so:

Gemäß der Befehlssyntax wird als erste Hexzahl die Portnummer erwartet (s. Überschrift). Sie wird eingelesen und dem Register C übergeben, mit dem man unabhängig von der Zahl einen Port indirekt adressieren kann. Sodann kommen die Werte, die auf diesen Port ausgegeben werden sollen. Es ist möglich, gleichzeitig mehrere Werte einzugeben. Die Obergrenze ist durch das DOS gegeben: Der Eingabepuffer faßt maximal 80 Zeichen inkl. NEW LINE am Ende.

Das Befehlswort OUT, die Portnummer und die einzelnen Werte zur Ausgabe werden wie im DOS üblich wahlweise durch Komma oder Blank getrennt. In Zeile 280 erfolgt deshalb ein CALL nach 4454, wo diese Trennzeichen erkannt werden. Falls in diesem Unterprogramm ein NEW LINE festgestellt wurde, falls also der Befehl zuende ist, steht das Zero-Flag auf 1. In diesem Falle wird die Bearbeitung beendet. Ansonsten wird aus den eingegebenen Hexzahlen der entsprechende binäre Code gebastelt und auf den Port ausgegeben. Sollte ein anderes Zeichen als eine Hexziffer gefunden werden, erfolgt die Fehlermeldung "falsche Parameter".

Sowohl die Rückkehr beim Ende der Befehlsbearbeitung als auch die Fehlerbehandlung haben einen kleinen Haken: Im Unterprogramm GETCHR befinden wir uns bereits in der zweiten Unterprogrammebene. Deshalb muß der Stack mit zwei POP-Befehlen auf den alten Stand gebracht werden, bevor das Programm in die DOS-Befehlseingabe oder die Fehleranzeige zurückspringt. Das ist alles.

* Bei NEWDOS 4C5D

```
                00100 ;* * * * * * * * * * * * * * * * * *
                00110 ;*                                   *
                00120 ;*      OUT port#,xx,yy,...          *
                00130 ;*     (C) '84 by St. Arnulf         *
                00140 ;*                                   *
                00150 ;* * * * * * * * * * * * * * * * * *
                00160
4D1D            00170          ORG     4D1DH   ;Operand des Sprungbefehls
4D1D 5E51       00180          DEFW    OUT     ;nach dort springen
                00190
515E            00200          ORG     515EH   ;hier OUT-Routine
515E CD6951     00210 OUT      CALL    GETVAL  ;Hexzahl einlesen
5161 4F         00220          LD      C,A     ;Portnummer in C laden
5162 CD6951     00230 LOOP     CALL    GETVAL  ;Hexzahl einlesen
5165 ED79       00240          OUT     (C),A   ;auf Port ausgeben
5167 18F9       00250          JR      LOOP    ;bis zum bitteren Ende
5169 CD7651     00260 GETVAL   CALL    GETCHR  ;eine Hexziffer einlesen
516C 17         00270          RLA             ;Stellenwert korrigieren,
516D 17         00280          RLA             ;d. h. in linkes Nibble
516E 17         00290          RLA             ;schieben
516F 17         00300          RLA             ;(4 Bits nach links)
5170 57         00310          LD      D,A     ;linke Hexziffer merken
5171 CD7651     00320          CALL    GETCHR  ;nächste Hexziffer einlesen
5174 B2         00330          OR      D       ;LSN mit MSN vereinigen
5175 C9         00340          RET             ;jetzt der korrekte Wert in Akku
5176 CD5444     00350 GETCHR   CALL    4454H*  ;Trennzeichen und CR erkennen
5179 2817       00360          JR      Z,EXIT  ;Ende bei NEW LINE
517B 7E         00370          LD      A,(HL)  ;Hexziffer laden
517C FE30       00380          CP      '0'     ;Dezimalziffer?
517E 3816       00390          JR      C,ERROR ;falls ASCII < Dezimalziffer
5180 FE3A       00400          CP      ':'     ;> ASCII "9" ?
5182 380A       00410          JR      C,RETURN;falls korrekte Dezimalziffer
5184 FE41       00420          CP      'A'     ;Hexziffer > ASCII "9" ?
5186 380E       00430          JR      C,ERROR ;falls < ASCII "A"
5188 FE47       00440          CP      'G'     ;> ASCII "F" ?
518A 300A       00450          JR      NC,ERROR;falls falsche Eingabe
518C D607       00460          SUB     7       ;falls Alpha-Hexziffer
518E E60F       00470 RETURN   AND     0FH     ;ASCII nach binär umwandeln
5190 23         00480          INC     HL      ;nächste Bildschirmzelle
5191 C9         00490          RET             ;erledigt
5192 F1         00500 EXIT     POP     AF      ;Stack korrigieren
5193 F1         00510          POP     AF      ;(2. CALL-Ebene)
5194 AF         00520          XOR     A       ;Z-Flag für "kein Fehler"
5195 C9         00530          RET             ;ins Betriebssystem
5196 F1         00540 ERROR    POP     AF      ;Stack korr. (s. o.)
5197 F1         00550          POP     AF
5198 3E2F       00560          LD      A,2FH   ;Fehlercode "falsche Parameter"
519A B7         00570          OR      A       ;Z-Flag rücksetzen
519B C9         00580          RET             ;ins Betriebssystem
002F            00590          END             ;wohlverdient
00000 mal gepennt
33470  Zeichen verfügbar


ERROR  5196 00540   00390 00430 00450
EXIT   5192 00500   00360
GETCHR 5176 00350   00260 00320
GETVAL 5169 00260   00210 00230
LOOP   5162 00230   00250
OUT    515E 00210   00180
RETURN 518E 00470   00410
```

Arnulf Sopp
Tel. 0451-791926

(10)

SYS-Files, und wie man sie macht

In der folgenden Tabelle sind alle SYS-Files von G-DOS 2.1b verzeichnet mit ihren Aufgaben und ihrer Länge. Die Länge deshalb, weil man sie beim Benutzen der freien SYS-Dateien kennen sollte. In den beiden letzten Spalten steht in binär und sedezimal, mit welchem Wert der Akku geladen sein muß, wenn man sie mit RST 28h aufrufen will. Manche Files erwarten außer dem Code in A weitere Parameter; Vorsicht ist also geboten!

File	Aufgabe	Sekt.	Akku bin.	hex.[1]
GDOS	DOS booten	5	xxx00000	g0
INHALT	Inhaltsverzeichnis	30	xxx00001	g1
SYS0	DOS-Kern, bis 4CFFh resident	15	xxx00010	g2
SYS1	DOS-Befehle interpretieren	5	xxx00011	g3
SYS2	File-Handling	5	xxx00100	g4
SYS3	dto., JKL, DEBUG usw.	5	xxx00101	g5
SYS4	DOS-Fehlermeldungen	5	xxx00110	g6
SYS5	DEBUG	5	xxx00111	g7
SYS6	NDF, COPY, APPEND, PD, S	35	xxx01000	g8
SYS7	UHR, DATUM, AUTO, ATTRIB usw.	5	xxx01001	g9
SYS8	I (DIR), FREE	5	xxx01010	gA
SYS9	B2, BOOT, Chaining-Kommandos	5	xxx01011	gB
SYS10	BASIC-Befehle GET und PUT	5	xxx01100	gC
SYS11	BASIC-Befehl RENUM (Teil)	5	xxx01101	gD
SYS12	BASIC-Befehl REF	5	xxx01110	gE
SYS13	BASIC-Fehlermeldungen, RENUM	5	xxx01111	gF
SYS14	CLEAR, CREATE, E, LIST, @, DR	5	xxx10000	u0
SYS15	FORM, V24	4	xxx10001	u1
SYS16	Hauptteil von PD	5	xxx10010	u2
SYS17	Hauptteil von S, AIK	5	xxx10011	u3
SYS18	direkte BASIC-Kommandos	5	xxx10100	u4
SYS19	versch. BASIC-Befehle	5	xxx10101	u5
SYS20	dto.	5	xxx10110	u6
SYS21	BASIC-Befehl CMD"D"	5	xxx10111	u7
SYS22	frei	5	xxx11000	u8
SYS23	LWT[2]	5	xxx11001	u9
SYS24	frei	5	xxx11010	uA
SYS25	frei	10	xxx11011	uB
SYS26	frei	5	xxx11100	uC
SYS27	frei	5	xxx11101	uD
SYS28	FORM (Druckercodes)	5	xxx11110	uE
SYS29	INFO	5	xxx11111	uF

[1]) x = 1 oder 0, g = gerade Hex-Ziffer, u = ungerade Hex-Ziffer

[2]) Diese Tabelle gilt für G-DOS 2.1b. Dort ist der Befehl LWT nicht mehr implementiert. Der Laufwerkstest funktioniert aber mit mindestens 3 Blanks und NEW LINE.

Die Aufgaben der SYS-Files sind teilweise der deutschen Anleitung zu NEWDOS-80 entnommen, soweit sie dort verzeichnet sind. Den Rest habe ich selber herausgetüftelt. Da G-DOS und NEWDOS nicht völlig identisch sind, und da ich entgegen anderslautenden Gerüchten nicht unfehlbar bin, sind Irrtümer in der Spalte "Aufgabe" möglich.

Wie SYS-Files aufgerufen werden, ist im Prinzip in dem Artikel "Neuer Dreitastenbefehl <,./>" erklärt. Die obige Tabelle erleichtert hoffentlich die Berechnung des Parameters im Akku. Der geneigte Leser muß sich nun nicht mehr mit einzelnen Bits herumschlagen, es genügt, zu wissen, worin sich eine gerade von einer ungeraden einstelligen Zahl unterscheidet. Darüber hinaus möchte ich diesmal erklären, wie ein eigenes SYS-File geschrieben werden kann. Dazu sind ein paar Vorkenntnisse nötig, deshalb wieder als hors d'oeuvre ein wenig Theorie:

Was später im Hauptspeicher stehen soll, liegt Byte an Byte auf der Diskette wie die einzelnen Töne einer Melodie auf einer Schallplatte. Es ist jedoch in einzelne Portionen, sog. Records unterteilt. Am Anfang eines jeden Records steht ein Byte, das die Art der folgenden Codes kennzeichnet. Bei einem Maschinenprogramm hat es den Wert 01. Das zweite Byte ist die Anzahl der Bytes, die dieser Record enthält. Dabei steht 00 für 256 dez.. Die relativen Bytes 3 und 4 des Records enthalten die Ladeadresse des ersten Codes (Byte 5), und zwar in der gewohnten Reihenfolge LSB-MSB. Die beiden Bytes der Adresse werden im Byte 2 übrigens mitgezählt. Die Abb. 1 zeigt ein beliebiges Beispiel. Die Bytes der Record-Organisation sind unterstrichen.

In einem nicht belegten SYS-File (22, 24, 25, 26 und 27) enthalten alle Records 256 Bytes, also ist der Zähler (rel. Byte 2) immer 00. Bei eigenen SYS-Files (wofür man diese freien Dateien ausnutzen muß) sollte man dieses Schema beibehalten.

Am Ende einer Datei schließlich steht die Einsprungsadresse des Programms. Sie wird mit dem Kenncode 02 eingeleitet. Es folgt auch hier die Anzahl der folgenden Bytes. Da es sich nur um diese Adresse handelt, lautet der Wert dieses Bytes ebenfalls 02 (2 Bytes für LSB und MSB der Adresse, s. Abb. 2a).

Die freien SYS-Files belegen 5 bzw. 10 Sektoren. Eine eigene SYS-Routine mag zwar kürzer sein, aber im Hinblick auf spätere Erweiterungen wäre es verschwenderisch, auf den Rest zu verzichten. Und das macht die Geschichte leider etwas kompliziert, wie wir später sehen werden. Grundsätzlich ist es aber ohne weiteres möglich, unter dem Namen z. B. SYS26/SYS mit EDTASM ein Maschinenprogramm zu assemblieren, das sich (außer der Länge) in nichts von einer Apparat-Systemdatei unterscheidet. Man kann sogar mit POKE-Befehlen ein Maschinenprogramm von BASIC aus zusammenschustern und mit CMD"DUMP,..." sein SYS-File kreieren.

Besser als diese Partisanen-Heimarbeit ist die Methode der sektorweisen Kopie. Dabei bleiben die Lage und Länge sowie alle anderen Eigenschaften der freien SYS-Datei erhalten. EDTASM erzeugt beispielsweise das File SYS26/CMD. Mit SUPERZAP und seinem Befehl DFS kriegt man raus, welche Sektoren dieses Programm belegt. Es sind später für den Befehl CDS die Quellsektoren. Auf die gleiche Weise findet man die Zielsektoren von SYS26/SYS.

Wenn SYS26/CMD auf einer ziemlich vollen Diskette nicht mehr zusammenhängend aufgezeichnet wurde (vorher mit SUPERZAP feststellen!), kann man leider bei der Frage SECTOR COUNT nicht mehr alles auf einen Rutsch erledigen. Wer Sorge hat, bei der Kopie auf Raten Fehler zu machen, kann für SYS26/CMD eine frische Diskette nehmen.

Zuletzt ist von Fall zu Fall noch etwas Kosmetik notwendig. Die Record-Grenzen liegen bei SYS26/SYS nicht notwendigerweise an denselben Stellen wie bei SYS26/CMD. Sie lassen sich einfach umzappen, wie in Abb. 2a (SYS26/CMD) und 2b (SYS26/SYS) gezeigt. Die zu modifizierenden Codes sind unterstrichen. Außerdem enthalten die CMD-Sektoren vielleicht noch Müll des Assemblers (Abb. 2a), den man aus optischen Gründen auf 00 zappen kann (Abb. 2b). Die Einsprungsadresse wird ebenfalls mit Nullen überschrieben, denn am Ende einer jeden SYS-Datei steht sie bereits: 4D00h. Das ist das Bytemuster 02 02 00 4D im letzten Sektor (der hier

nicht abgebildet ist, um nicht eine neue Infoseite zu beginnen). In Abb. 2a stehen diese Codes mitten im Sektor.

Auch bei diesem Beitrag habe ich das Gefühl, Euch eher abgeschreckt als ermutigt zu haben. Wenn Ihr Euer DOS-Original im Panzerschrank laßt und nur mit Kopien arbeitet, darf aber gerne alles schiefgehen. Und es wird einiges schiefgehen, das walte Murphy. Nach zwei Stunden Training, zwei Litern Schweiß und ebensoviel Bier habt Ihr es aber im Griff. Merke: Die teure Disco geht nur mit Hardware von der Art eines Vorschlaghammers kaputt.

```
                 ↓  1  2  3  4  5  6  7  8  9  A  B
DRV  00   010B 0C40 C3C2 4BC3 0946 C3F2 4501 0B2D ...§..K..F..E..-
0    10   40C3 0044 3E43 EFC3 DB4A 0121 3E40 0000 §..D>C...J.!>§..
0H   20   0032 1E08 530A 0500 52FF FF00 0000 0000 .2..S...R.......
     30   0000 00F5 07DC E847 F1C9 0000 0001 1F63 .......G........
DRS  40   407A CD68 407B F50F 0F0F 0FCD 7140 F1E6 §...§.........§..
5    50   0FC6 9027 CE40 2777 23C9 3B3B 1700 0112 ...'.§'.#.;;.....
5H   60   0843 0000 1123 0323 0A02 0000 C3B0 4501 .C...#.#......E.
     70   0000 0143 6843 A540 0000 0000 0000 5A11 ...C.C.§.......Z.
     80   2303 230A 0200 0011 02FF 0100 0100 0000 #.#.............
     90   00FF 00FF 0100 0100 0000 00FF 00FF 0100 ................
     A0   0100 0000 00FF 0071 4300 0000 0004 0100 ........C.......
     B0   0100 0100 000D 0D01 30B2 4300 0000 FFFF ........0.C.....
FRS  C0   FC4C 0000 00FF FF00 0000 0000 FFFF 0000 .L..............
0    D0   0000 00FF FF00 0080 2800 0042 0000 FF00 ........(..B....
0H   E0   0000 0040 0000 00FF FF01 0000 443E 23EF ...§........D>#.
     F0   0000 3E63 EFC8 F53E 26EF C309 463E 65EF ..>....>&...F>..
```

Abb. 1

```
DRV  00   0162 FC4D 3063 2C30 642C 3065 2C30 662C ...MO..O.,O.,O.,
0    10   3067 2C30 6820 2D20 456E 6465 206D 6974 O.,O..-.E.......
0H   20   203C 4E45 5720 4C49 4E45 3E03 527D 636B .<NEW.LINE>.R...
     30   6B65 6872 6164 7265 7373 653F 2028 4865 ...........?.(H.
DRS  40   783B 2066 7D72 2052 4554 206E 7572 203C .;.....RET.....<
161  50   4E45 5720 4C49 4E45 3E20 6569 6E67 6562 NEW.LINE>.......
A1H  60   656E 2903 0202 004D 462C 3061 2C30 622C ..)....MF,O.,O.,
     70   3063 2C30 642C 3065 2C30 662C 3067 2C30 O.,O.,O.,O.,O.,O
     80   6820 2D20 456E 6465 206D 6974 203C 4E45 ..-.E.........<NE
     90   5720 4C49 4E45 3E27 0DB0 B1B2 B3B0 2009 W.LINE>'.........
     A0   4445 4642 0930 3348 0DB0 B1B2 B4B0 2054 DEFB.03H.......T
     B0   4558 5433 0944 4546 4D09 2752 7D63 6B6B EXT3.DEFM.'R....
FRS  C0   6568 7261 6472 6573 7365 3F20 2848 6578 .........?.(H..
1    D0   3B20 667D 7220 5245 5420 6E75 7220 3C4E ;.....RET.....<N
1H   E0   4557 204C 494E 453E 2065 696E 6765 6265 EW.LINE>.......
     F0   6E29 270D B0B1 B2B5 B020 0944 4546 4209 .)'........DEFB.
```

Abb. 2a

```
DRV  00   3063 0100 FE4D 2C30 642C 3065 2C30 662C O....M,O.,O.,O.,
0    10   3067 2C30 6820 2D20 456E 6465 206D 6974 O.,O..-.E.......
0H   20   203C 4E45 5720 4C49 4E45 3E03 527D 636B .<NEW.LINE>.R...
     30   6B65 6872 6164 7265 7373 653F 2028 4865 ...........?.(H.
DRS  40   783B 2066 7D72 2052 4554 206E 7572 203C .;.....RET.....<
135150    4E45 5720 4C49 4E45 3E20 6569 6E67 6562 NEW.LINE>.......
547H60    656E 2903 0000 0000 0000 0000 0000 0000 ..)............
     70   0000 0000 0000 0000 0000 0000 0000 0000 ................
     80   0000 0000 0000 0000 0000 0000 0000 0000 ................
     90   0000 0000 0000 0000 0000 0000 0000 0000 ................
     A0   0000 0000 0000 0000 0000 0000 0000 0000 ................
     B0   0000 0000 0000 0000 0000 0000 0000 0000 ................
FRS  C0   0000 0000 0000 0000 0000 0000 0000 0000 ................
1    D0   0000 0000 0000 0000 0000 0000 0000 0000 ................
1H   E0   0000 0000 0000 0000 0000 0000 0000 0000 ................
     F0   0000 0000 0000 0000 0000 0000 0000 0000 ................
```

Abb. 2b

Arnulf Sopp, Tel. 0451-791926

Die Library-Befehle des G-DOS 2.x

In meinem Artikel "SYS-Files und wie man sie macht" (Info 9/84) äußere ich noch skeptisch, daß in der Rubrik "Aufgaben" meiner Tabelle evtl. Fehler sein könnten. Das liegt daran, daß ich verschiedene Systemdateien disassembliert und mir auf die Programme vorsichtig einen Reim gemacht hatte. Die Tabelle war zwar korrekt, aber jetzt kann ich eine lückenlose abliefern. Sie ist diesmal (nicht ganz konsequent) alphabetisch in der Reihenfolge der LIB-Bildschirmausgabe geordnet. Die SYS-Dateien sind dezimal angegeben, die Registerinhalte von A und C (s. u.) sedezimal. Es ist meine persönliche Library nach ein paar Zaps, die nicht mehr überall mit dem Original-DOS übereinstimmt:

LIB-CMD	SYS	A	C	LIB-CMD	SYS	A	C	LIB-CMD	SYS	A	C
0	14	F0	04	§	14	F0	01	AIK	17	53	00
APPEND	6	68	40	ATTRIB	7	E9	05	AUTO	7	E9	04
B2	9	EB	06	BL	3	E5	01	BOOT	9	EB	0A
BREAK	3	E5	05	CLS	1	E3	09	CONT	9	EB	45
COPY	6	48	40	CREATE	14	F0	02	DATUM	7	E9	0B
DIR	8	2A	00	DISK	29	FF	03	DO	9	EB	43
DR	28	FE	02	DUMP	7	E9	07	E	14	F0	07
FORM	28	FE	08	FREE	8	4A	00	HIMEM	7	E9	02
I	8	2A	00	INFO	29	FF	01	JKL	3	A5	00
KILL	3	45	00	LC	3	E5	08	LF	28	FE	01
LIB	1	E3	02	LIST	14	F0	05	LOAD	2	50	24
LWT	23	F9	01	N	2	E4	01	NDF	6	28	40
PAUSE	9	EB	08	PD	7	E9	03	PORT	29	FF	02
PRINT	14	F0	06	PROT	7	E9	06	PURGE	7	E9	09
R	1	23	00	S	7	E9	01	STMT	9	EB	09
UHR	3	E5	02	V+	3	E5	04	OUT	29	FF	07
Z	29	FF	06	ZEIT	7	E9	0A	&	3	E5	03
!	9	EB	03	;	1	E3	06	/	1	E3	05
?	1	E3	02	>	6	48	40	M>	9	EB	02
DDE	15	F1	40	123	5	27	00	567	26	FC	00
,./	26	DC	00								

Die angegebenen SYS-Files sind diejenigen, bei denen eingesprungen wird. Es gibt durchaus Befehle, die mehrere Systemfiles durchlaufen. Man kann das daran erkennen, daß in der betreffenden Routine ein RST 28h vorkommt. Der jeweilige Akkuinhalt verrät dann, wo es hingeht. Die obige Tabelle und die erstgenannte helfen dabei, das Ziel herauszufinden.

Diese neue Tabelle ist das Resultat einer Spielerei: Zapt einmal irgendeinen Befehl um; nennt meinetwegen BOOT einfach KAHN. Wenn Ihr nachher in der Befehlsebene KAHN eintippt, wird das System neu gebootet. Auf dieser Basis konnte ich auch meinen neuen Befehl OUT (auch in diesem Info?) implementieren, der nun den sinnlosen Befehl V24 ersetzt. Bei diesem Zufallsfund blieb es natürlich nicht, sondern die Untersuchungen in SYS1/SYS, wo die DOS-Befehle erkannt werden, zeitigten weitere Ergebnisse:

Hinter jedem Befehlswort stehen drei weitere Bytes. Bei dem ersten ist immer das Bit 7 gesetzt. Danach folgt der DOS-Request-Code, der in den Akku geladen wird, um mit RST 28h in das zuständige SYS-File springen zu können. Das dritte Byte ist zumeist 00. Ein paarmal ist dies nicht der Fall. Bisher bin ich noch nicht dahintergekommen, welche Bedeutung das hat. Die beiden abgedruckten Sektordumps aus SYS1/SYS zeigen die Befehlswörter und diese drei Bytes für jeden Befehl.

Befehlswort, Akkuinhalt und (meistens) 00 als Ende-Markierung (?) waren leicht zu identifizieren. Das erste Schlußbyte aber hat eine wichtige Bedeutung: In vielen Systemdateien steht am Anfang eine Art Hühnerleiter, wo das Register C wiederholt dekrementiert wird. Ist es bei 00 angekommen, wird in das zuständige Segment des Programms gesprungen.

Dieses Byte gelangt aber nicht unverändert durch die Routine, die den Befehl erkennt, sondern das Bit 7 wird zurückgesetzt. So wird aus 82 beispielsweise 02. Nach zweimaligem Dekrementieren ist die Zero-Bedingung erfüllt und der Befehl JP Z,xxxx wird ausgeführt. In einigen SYS-Dateien genügt als Zeiger auf die betreffende Routine allerdings auch der Akku. Dies scheint z. B. in SYS6/SYS der Fall zu sein, wo ich diese Hühnerleiter nicht fand.

Was kann man nun mit diesen Informationen anfangen? Mein DEBUG kommt nicht nur, wenn ich gleichzeitig <123> drücke, sondern auch, wenn ich die Ziffern nacheinander eingebe. Dazu war es nur nötig, das zweite Byte, das in den Akku kommt, von 00 auf 87 zu zappen. Eigene DOS-Befehle, die nicht, wie OUT, einen alten Befehl ersetzen, lassen sich ebenfalls leicht einschummeln. Man muß nur in dem Bereich hinter dem letzten Befehl zuerst das Wort, dann irgendetwas >= 80, dann den korrekten Akku-Wert für das zuständige SYS-File und schließlich 00 in die Library einzappen.

Freie SYS-Dateien gibt's genug. Sie sollen nicht bloß Platz auf der Diskette kosten sondern etwas leisten. Es ist kein Problem, wie man sieht. In einer "NEWDOS8052" genannten DOS-Verschönerung, die angeblich copyrightswidrig in den Kreisen des Clubs kursieren soll, ist PRINT in LLIST umbenannt, um eine Analogie zu BASIC zu schaffen. Wer dergleichen sinnvoll findet, kann das nach der beschriebenen Methode ebenfalls machen und noch einiges mehr. Viel Spaß dabei!

Arnulf Sopp

Beispiel: BOOT: A=EB für SYS9/SYS, C=0A (8A-80=0A)

```
000200: D54C 20A9 CB59 2802 0102 004F E3E5 79E6 .L ..Y(....O..y.
000210: 0728 0EE5 21BC 5123 2323 3D20 FACD 2A4F .(..!.Q###= ..*O
000220: E179 01D3 49C5 CB7F C806 0021 0042 CB77 .y..I......!.B.w
000230: CA24 44C3 2044 D5C5 011C 091A FE3A 280A .$D. D.......:(.
000240: FE2F 3806 281B 0D13 10F1 2323 E5EB 0600 ./8.(.....##....
000250: 0954 5D2B 1313 13ED B8E1 0E03 EDB8 3E2F .TÜ+..........>/
000260: 12C1 D1C9 3084 F000 4081 F000 4149 4B80 ....0...§..,AIK.
000270: 5300 4150 5045 4E44 C068 0041 5454 5249 S.APPEND.h.ATTRI
000280: 4285 E988 4155 544F 84E9 0042 3286 EB00 B...AUTO...B2...
000290: 424C 81E5 0042 4F4F 548A EB10 4252 4541 BL...BOOT...BREA
0002A0: 4B85 E500 434C 5389 E310 434F 4E54 C5EB K...CLS...CONT..
0002B0: 0043 4F50 59C0 4800 4352 4541 5445 82F0 .COPY.H.CREATE..
0002C0: 4044 4154 554D 8BE9 0044 4952 802A 0044 §DATUM...DIR.*.D
0002D0: 4953 4B83 FF00 444F C3EB 8A44 5282 FE00 ISK...DO...DR...
0002E0: 4455 4D50 87E9 C845 87F0 0046 4F52 4D88 DUMP...E...FORM.
0002F0: FE00 4652 4545 804A 0048 494D 454D 82E9 ..FREE.J.HIMEM..
```

SYS1/SYS, rel. Skt. 2

```
000300: 0049 802A 0049 4E46 4F81 FF00 0102 0050 .I.*.INFO......P
000310: 4A4B 4C80 A510 4B49 4C4C 8045 904C 4388 JKL...KILL.E.LC.
000320: E500 4C46 81FE 004C 4942 82E3 004C 4953 ..LF...LIB...LIS
000330: 5485 F088 4C4F 4144 80A4 504C 5754 81F9 T...LOAD..PLWT..
000340: 004E 81E4 B04E 4446 C028 0050 4155 5345 .N...NDF.(.PAUSE
000350: 88EB 0050 4483 E900 504F 5254 82FF 0050 ...PD...PORT...P
000360: 5249 4E54 86F0 8850 524F 5486 E900 5055 RINT...PROT...PU
000370: 5247 4589 E900 5280 2300 5381 E900 5354 RGE...R.#.S...ST
000380: 4D54 89EB 0055 4852 82E5 0056 2B84 E500 MT...UHR...V+...
000390: 4F55 5487 FF00 5A86 FF00 5A45 4954 8AE9 OUT...Z...ZEIT..
0003A0: 0026 83E5 0021 83EB 8A3B 86E3 002F 85E3 .&...!...;.../..
0003B0: 003F 82E3 003E C048 004D 3E82 EBB0 4444 .?...>.H.M>...DD
0003C0: 45C0 F100 3132 3380 8700 3536 3780 FC00 E...123...567...
0003D0: 2C2E 2F80 DC00 0000 0000 0000 0000 0021 ,./............!
0003E0: 584F 0E40 0608 7ECB 7F23 2005 CDB7 5110 XO.§..ß..# ...Q.
0003F0: F523 237E B7CA B551 0DCC B551 28E4 CDAD .##ß...Q...Q(...
```

SYS1/SYS, rel. Skt. 3

Die Records handhaben

In den beiden Beiträgen "Neuer Dreitastenbefehl ..." und "SYS-Files ..." war bereits von der Record-Organisation die Rede. Mich läßt das dumpfe Gefühl nicht los, daß ich mich dort etwas zu global, nur für den Experten verständlich ausdrückte. Da meine literarischen Absonderungen letztenendes für die Praxis auch des weniger geübten Infolesers etwas bringen sollen, möchte ich einen weiteren Beitrag zu diesem Thema nachschieben. Wer bei seiner Zapperei möglichst wenige Fehler machen will, muß mit Records einfach umgehen können.

Es mag auf Anhieb paradox klingen, daß ich als Beispiel eine Datei der Systemdiskette nehme, die als einzige eben nicht in Records gegliedert ist: Habt Ihr schon einmal versucht, GDOS/SYS (BOOT/SYS) zu disassemblieren? Disassembler erwarten als allererstes Byte des Files den Code 01. Er signalisiert ein Maschinenprogramm bzw. ein Datenfeld, das ähnlich einem Programm einem ganz bestimmten Speicherbereich zugeordnet ist. In GDOS/SYS findet sich aber nur der Maschinencode dieses Urladers, nichts von Adressen, kein Kenncode 01. Das liegt daran, daß es DOS erst laden muß und selbst vom Microsoft-ROM geladen wird.

Für unser Problem ist zunächst nur der erste Sektor von GDOS, der eigentliche Urlader interessant und hiervon auch nur die ersten 238 Bytes. Der Rest des Sektors enthält einen Copyright-Vermerk und der Rest des Files Daten zur späteren Verwendung.

Zur Wiederholung: Das erste Byte eines Maschinenprogramms lautet immer 01. Das zweite hält die Anzahl der zu diesem Record, dieser Portion des Programms gehörigen Bytes. Dabei steht 00 für 100h (256d). Das dritte und vierte Byte (die bei der Anzahl bereits mitgezählt werden) stellen in der Folge LSB-MSB die Ladeadresse des ersten zum Maschinencode gehörenden Bytes (des fünften) dar. Erst jetzt folgt das eigentliche Programm. Am Ende des Files schließlich finden wir mit dem Kenncode 02 die Einsprungsadresse des Programms. Nach 02 folgt wieder ein Bytezähler. Er lautet ebenfalls 02, weil nur noch die Adresse (2 Bytes) folgt.

Um z. B. für DSMBLR oder DISASSEM aus GDOS ein lesbares File zu machen, müssen wir für Record-Codes sorgen. Es wäre schade, die ersten vier Bytes dafür zu überschreiben. Davor ist dem File aber kein Sektor mehr zugeordnet, den man dafür verwenden könnte. Die Lösung ist simpel: Wir "borgen" uns ein unbenutztes File, das mindestens zwei Sektoren Platz bietet. Die bereits früher erwähnten freien SYS-Dateien eignen sich hervorragend. Hier kommt willkürlich SYS22/SYS zur Anwendung.

In dessen zweiten Sektor kopieren wir den ersten Sektor von GDOS. Das geht gut mit DEBUG oder SUPERZAP. Die letzten vier Bytes des ersten SYS22-Sektors halten jetzt für die Record-Organisation her. Es beginnt mit 01. Weshalb es ausgerechnet mit FE weitergeht, erkläre ich später. Danach folgt die Ladeadresse 4200h, mit dem LSB beginnend.

Die 252 Nullen davor würde der Disassembler aber ebenfalls nicht verzeihen, deshalb werden sie mit entsprechenden Record-Codes zu dem erklärt, was in BASIC REM heißt: Der Code 05 bezeichnet Bereiche auf der Diskette, die Kommentare enthalten und ansonsten ignoriert werden sollen. Es muß auch hier ein Bytezähler folgen, damit DOS "weiß", ab wo es wieder interessant wird. In unserem Beispiel lautet er FAh (250d Bytes bis zum nächsten Record-Header).

Erinnern wir uns: Die letzte Dump-Zeile von GDOS, das ist jetzt der zweite Sektor von SYS22, enthält nur noch den Copyright-Vermerk. Seine

letzten vier Buchstaben (s. Abb.) sind ohnehin hochgestapelt, also können sie für die Einsprungsadresse dienen. Sie wird, wie gesagt, mit der Bytefolge 02 02 1b mit dargestellt, wobei in diesem Fall konkret 00 42 für 4200h einzusetzen ist. Zwischen dem Kennzeichen 02 und dem letzten Zählbyte im ersten Sektor liegen 254 Bytes, und so erklärt sich der Zähler FE dort.

Alles klar? Alles klar.

GDOS/SYS, Sektor Ø

```
000000: 00FE 30F3 21EC 3736 FF36 D023 3600 2336 ..0.!.76.6.#6.#6
000010: 0011 0501 D931 E041 21FF 51CD 5242 FE20 .....1.A!.Q.RB.
000020: 4730 2957 CD52 424F CD52 425F 1012 CD52 G0)W.RBO.RB_...R
000030: 4257 0D0D 2CCC 5542 7E12 130D 20F6 18DB BW....UB~... ...
000040: 10F9 CD52 4257 1AFE A513 D5C8 21E5 42C3 ...RBW......!.B.
000050: C342 2C7E C0D9 060A 21E1 3736 01D5 C57B .B,~....!.76...ä
000060: D612 3803 5F36 0921 EC37 CDCE 42ED 53EE ..8._6.!.7..B.S.
000070: 3736 1BCD CE42 3688 11EF 3701 0051 CDD7 76...B6...7..Q..
000080: 427E E683 E281 421A 0203 CB4E C287 42CB B~....B....N..B.
000090: 4EC2 8742 CB4E 20EF CB46 2808 CB4E 20E7 N..B.N ..F(..N .
0000A0: CB7E 28E6 7E36 D0C1 D1E6 FC20 0C1C 7BD6 .~(.~6..... ..ä.
0000B0: 2420 0314 1E00 D97E C9CD D742 360B 1098 $ .....~...B6...
0000C0: 21DD 427E FE03 28FB 23CD 3300 1BF5 CDD7 !.B~..(.#.3.....
0000D0: 42CB 4620 FC7E C93E 063D 20FD C91C 1F52 B.F .~.>.= ....R
0000E0: 4553 4554 031C 1F47 2D44 4F53 3F03 0000 ESET...G-DOS?...
0000F0: 4027 3832 2F38 3420 5443 532F 536F 7070 @'82/84 TCS/Sopp
```

SYS22/SYS, Sektor Ø

```
000000: 05FA 2020 2020 2020 2020 2020 2020 2020 ..
000010: 2020 2020 2020 2020 2020 2020 2020 2020
000020: 4465 7220 436F 6465 2030 3520 6265 2D20 Der Code 05 be-
000030: 7A65 6963 686E 6574 2065 696E 2020 2020 zeichnet ein
000040: 4665 6C64 2C20 6461 7320 766F 6D20 2020 Feld, das vom
000050: 444F 5320 6967 6E6F 7269 6572 7420 2020 DOS ignoriert
000060: 7765 7264 656E 2073 6F6C 6C2E 2046 4120 werden soll. FA
000070: 6461 6869 6E74 6572 2069 7374 2064 6173 dahinter ist das
000080: 5A7B 686C 6279 7465 2C20 6461 6D69 7420 Zählbyte, damit
000090: 6465 7220 666F 6C67 656E 6465 2020 2020 der folgende
0000A0: 4B65 6E6E 636F 6465 2030 3120 2873 2E20 Kenncode 01 (s.
0000B0: 752E 2920 6765 6675 6E64 656E 2020 2020 u.) gefunden
0000C0: 7769 7264 2E20 2020 2020 2020 2020 2020 wird.
0000D0: 2020 2020 2020 2020 2020 2020 2020 2020
0000E0: 2020 2020 2020 2020 2020 2020 2020 2020
0000F0: 2020 2020 2020 2020 2020 2020 01FE 0042                ...B
```

SYS22/SYS, Sektor 1

```
000100: 00FE 30F3 21EC 3736 FF36 D023 3600 2336 ..0.!.76.6.#6.#6
000110: 0011 0501 D931 E041 21FF 51CD 5242 FE20 .....1.A!.Q.RB.
000120: 4730 2957 CD52 424F CD52 425F 1012 CD52 G0)W.RBO.RB_...R
000130: 4257 0D0D 2CCC 5542 7E12 130D 20F6 18DB BW..,.UB~... ...
000140: 10F9 CD52 4257 1AFE A513 D5C8 21E5 42C3 ...RBW......!.B.
000150: C342 2C7E C0D9 060A 21E1 3736 01D5 C57B .B,~....!.76...ä
000160: D612 3803 5F36 0921 EC37 CDCE 42ED 53EE ..8._6.!.7..B.S.
000170: 3736 1BCD CE42 3688 11EF 3701 0051 CDD7 76...B6...7..Q..
000180: 427E E683 E281 421A 0203 CB4E C287 42CB B~....B....N..B.
000190: 4EC2 8742 CB4E 20EF CB46 2808 CB4E 20E7 N..B.N ..F(..N .
0001A0: CB7E 28E6 7E36 D0C1 D1E6 FC20 0C1C 7BD6 .~(.~6..... ..ä.
0001B0: 2420 0314 1E00 D97E C9CD D742 360B 1098 $ .....~...B6...
0001C0: 21DD 427E FE03 28FB 23CD 3300 1BF5 CDD7 !.B~..(.#.3.....
0001D0: 42CB 4620 FC7E C93E 063D 20FD C91C 1F52 B.F .~.>.= ....R
0001E0: 4553 4554 031C 1F47 2D44 4F53 3F03 0000 ESET...G-DOS?...
0001F0: 4027 3832 2F38 3420 5443 532F 0202 0042 @'82/84 TCS/...B
```

Arnulf Sopp, Tel. 0451-791926

BASIC FREI IM RAM VERLAGERN

Üblicherweise werden Maschinenprogramme geschützt, indem man nach dem Einschalten die MEMSIZE-Frage mit der entsprechenden Adresse beantwortet. Wozu der Umstand? Das kann ein Programm auch selbst erledigen. Doch davon mehr im nächsten Artikel.

Hier soll das genaue Gegenteil besprochen werden: BASIC kann einem Programm nichts anhaben, das unterhalb residiert, denn BASIC wächst immer nur nach oben. Der Programmtext (PST, program statement table) und die Variablenliste (VLT, variable list table) bekommen vom Betriebssystem einen Platz zugewiesen, der die Untergrenze bestimmt, von wo sie sich nach den Wünschen des Benutzers nach oben ausdehnen.

Und damit ist für den aufmerksamen Leser auch schon alles gesagt: Man braucht nur die Zeiger auf die jeweils zuzuweisenden Plätze zu "verbiegen", um sich unterhalb von BASIC beliebig viel Platz zu schaffen, wo man ein Maschinenprogramm einschummeln kann. Doch zum Verständnis sind einige Informationen nötig:

Je nach dem, ob man Level 2 oder ein DOS fährt, liegen gewisse wichtige Adressen an unterschiedlicher Stelle. Die Beschaffenheit der Peripherie kann unser Tandy oder Genie nicht ahnen, deshalb gibt es einen reservierten Speicherbereich im RAM, der z. B. von einer angeschlossenen Floppy umgeschrieben wird. Da dieser Bereich u. a. der Kommunikation mit der Peripherie dient, heißt er "communications region". Dort finden sich an festgelegter Stelle auch unsere Zeiger auf BASIC.

Für den unteren BASIC-Bereich, den wir verlagern wollen, sind nur fünf Zeiger von Belang, und zwar der auf den Anfang der PST, den der Liste einfacher Variabler, den Anfang der Liste dimensionierter Variabler, ihr Ende und den Pointer auf das zuletzt gelesene DATA-Statement (der anfangs auf eine Stelle vor der PST zeigt). Die String-Variablen sind ganz oben gespeichert, so daß wir uns hier nicht um sie kümmern müssen. Wo diese Pointers stehen, geht aus dem anhängenden Assembler-Listing hervor.

Dieses Assembler-Programm ist für sich allein völlig witzlos. Wer möchte schon gerne ein Programm laden und fahren, bis endlich sein Zielprogramm drankommt? Sinn der Sache ist, es zum Bestandteil einer eigenen Routine zu machen. Dabei sollte man beachten, daß das kleine Programm nach dem Start überflüssig geworden ist und nicht mehr geschützt zu werden braucht. Daher sollte zuerst das eigene Programm kommen, dann die drei Nullen ab "ENDE", dann das Segment ab "ANFANG". Der Befehl "JP BASIC" ist dann natürlich durch einen Sprung ins eigene Zielprogramm zu ersetzen (das seinerseits nach BASIC springen kann).

Es muß unbedingt beachtet werden, daß das Programm nur von BASIC aus (mit SYSTEM bzw. mit CMD"FILENAME") geladen werden darf. Unter DOS stehen an der Stelle der Zeiger völlig andere Werte, die alles aus dem Lot bringen würden.

Arnulf Sopp. Tel. 0451-791926

```
                00100 ;*************************************************
                00110 ;*                                               *
                00120 ;*        BASIC RELOZIEREN                       *
                00130 ;*                                               *
                00140 ;*************************************************
                00150 ;(C) 1984 by A. Sopp, Wakenitzstr. 8, 2400 Lübeck 1
                00160
                00170
                00180 ;SYMBOLVEREINBARUNGEN:
40A4            00190 BASPTR  EQU     40A4H   ;START BASIC-PROGRAMMTEXT
40F9            00200 VARPTR  EQU     40F9H   ;START EINFACHE VARIABLE
40FB            00210 ARRANF  EQU     40FBH   ;START DIMENS. VARIABLE
40FD            00220 ARREND  EQU     40FDH   ;DTO. ENDE
40FF            00230 DATPTR  EQU     40FFH   ;ZEIGER AUF LETZTES DATA-STMT.
1A19            00240 BASIC   EQU     1A19H   ;BASIC-WARMSTARTADRESSE
                00250
                00260 ;BASIC VERSCHIEBEN:
                00270
6A45            00280         ORG     6A45H           ;FUER LEVEL 2: 42E8H
                00290
6A45 215D6A     00300 ANFANG  LD      HL,ENDE         ;NEUER DATA-ZEIGER
6A48 22FF40     00310         LD      (DATPTR),HL     ;VERBIEGEN
6A4B 23         00320         INC     HL              ;EINE STELLE HOEHER
6A4C 22A440     00330         LD      (BASPTR),HL     ;NEUER BASIC-ANFANG
6A4F 23         00340         INC     HL              ;ZWEI STELLEN HOEHER
6A50 23         00350         INC     HL
6A51 22F940     00360         LD      (VARPTR),HL     ;NEUER ZEIGER EINF. VAR.
6A54 22FB40     00370         LD      (ARRANF),HL     ;DTO. ANFANG DIMENS. VAR.
6A57 22FD40     00380         LD      (ARREND),HL     ;DTO. ENDE DTO.
6A5A C3191A     00390         JP      BASIC           ;WARMSTART
6A5D 00         00400 ENDE    DEFB    0               ;BASIC BEGINNT MIT 3 X 0
6A5E 00         00410         DEFB    0               ;2. 0
6A5F 00         00420         DEFB    0               ;3. 0
                00430
6A45            00440         END     ANFANG
00000 TOTAL ERRORS
34013  TEXT AREA BYTES LEFT


ANFANG 6A45 00300   00440
ARRANF 40FB 00210   00370
ARREND 40FD 00220   00380
BASIC  1A19 00240   00390
BASPTR 40A4 00190   00330
DATPTR 40FF 00230   00310
ENDE   6A5D 00400   00300
VARPTR 40F9 00200   00360
```

P U T T O a d r e s s e - ein neuer BASIC-Befehl

Nein, es ist durchaus nicht mein Lieblingsthema, BASIC im RAM auf- und abzuschubsen. Hat man aber einmal über ein derartiges Thema nachgedacht, um ein paar Seiten für das Clubinfo daraus zu machen, kommen in der Folge ständig neue Ideen zumgleichen Problemkreis.

Heute soll der BASIC-Programmierer etwas davon haben: Der neue Befehl PUT TO geht noch ein Stück weiter als die bisherigen Vorschläge. Er verschiebt ebenfalls BASIC, aber im Speicher befindliche Programme bleiben erhalten. Sie rutschen kurzerhand an die neue Stelle mit und tun dort so, als sei nichts gewesen. In BASIC merkt der User nur dann etwas, wenn er mit PRINT MEM fragt, wieviel Platz er noch im RAM hat.

Die Befehlssyntax lautet wie in der Überschrift. Dabei können die Blanks entfallen, die Adresse muß dezimal eingeben werden. Mit dieser Adresse ist die BASIC-Untergrenze gemeint. Es ist die Stelle, auf die der DATA-Zeiger deutet, bevor RUN eingegeben wurde. In diesem Vor-RUN-Zustand befindet sich übrigens auch das Programm nach der Verschiebung mit PUT TO: Variable sind gelöscht, der DATA-Zeiger weist unmittelbar vor den Programmtext.

Die einzugebende Adresse muß ohne den zuvor besprochenen EG 64 MBA mindestens 17128 betragen, nach oben sind nur die physikalischen Grenzen des RAM gesetzt. Es obliegt dem Anwender, darauf zu achten, daß von dieser Adresse an nach oben für das Programm und später hinzukommende Variable genügend Platz ist. Diese Eigenverantwortung des Users macht deutlich, daß gewisse Grundkenntnisse über die Speicherverwaltung von BASIC nötig sind wie etwa auch bei den Befehlen VARPTR und USR.

In diesem Zusammenhang ist auch der Grund zu sehen, weshalb ich einerseits eine BASIC-Utility vorstelle, andererseits aber nur ein Assembler-Listing anbiete. Der BASIC-Programmierer ohne Maschinenspracheкenntnisse wird den Befehl PUT TO wohl nie brauchen, der Z80-Freak hat aber einen Assembler, mit dem er PUT TO seinen eigenen Bedürfnissen anpassen kann (z. B. ORG verlegen).

Das Programm läuft nicht unter Disk-BASIC, weil der PUT-Befehl zweckentfremdet wird. Mit einer kleinen Änderung ist aber auch das möglich: Die Befehle

```
RST     08H
DEFB    TO
```

werden ersetzt durch

```
DEC     HL          ;als Zeiger für RST 10H
RST     10H         ;nächstes Zeichen laden
CP      TO          ;TO-Token?
JP      NZ,627BH    ;PUT-Routine Disk-BASIC
INC     HL          ;war in RST 8 enthalten
```

(Die Adresse 627BH kann von DOS zu DOS variieren. Schaut euch in 4183/4184H den genauen Vektor an und ermittelt seinen dezimalen Wert mit PRINT PEEK(&h4183)+PEEK(&h4184)*256.)

Und was das Ganze soll? Das Himem wird für tausend kleine Popelprogramme gebraucht; man setzt pausenlos die Memsize für alles Mögliche. Mit PUT TO wird stattdessen die BASIC-Untergrenze bestimmt, so daß

man z. B. ganz oben einen Monitor fahren kann und unten die Tasten entprellt oder (nein, es ist wirklich nicht mein Lieblingsthema) für EDTASM Platz hat. Irgendwo lümmelt sich dann BASIC beliebig dazwischen.

Auch dies ist ein solches Popelprogramm, denn man wird es recht selten brauchen. Immerhin, es kann im Gegensatz zum Meisten dieser Art mitten in der BASIC-Arbeit nachgeladen werden. So war für mich an der ganzen Geschichte hauptsächlich die Frage interessant, wie man überhaupt einen neuen BASIC-Befehl implementiert. Es gibt einige Möglichkeiten:

Zum gleichen Zweck hätte auch das Befehlswort MOVE BASIC dienen können. Dabei würde zunächst ein Syntax Error erkannt werden. Im Verlauf der Fehlerbehandlungsroutine wird auch ins RAM verzweigt, das wir für diesen Zweck umprogrammieren können. Da steht jetzt ein Jump zu unserer neuen Verb Action-Routine. Dort wird der Stack nach der Herkunft des Fehlers durchpflügt. Kommt er aus der Variable Assignment-Routine, checken wir, ob das fehlerhafte Statement genau MOVE BASIC oder MOVEBASIC und nicht anders lautete. Ist das der Fall, ist es kein Fehler und der Spaß geht los.

Zu umständlich? Einverstanden. Da der BASIC-Verschiebebefehl wohl hauptsächlich nicht programmiert, sondern als direktes Statement eingetippt wird, vereinfacht sich die Sache: Der hierfür zuständige DOS-Exit ist 41B2H (JP 6033H, je nach DOS). Wir verbiegen ihn auf unsere Routine und können den Stack Stack sein lassen. Jetzt wird wieder untersucht, ob die folgenden Zeichen genau M-O-V-E-(-)B-A-S-I-C hießen.

Immer noch zu umständlich? Wieder einverstanden. Die einfachste Lösung ist immer, einen Befehl zu "mißbrauchen", den es bereits gibt, der ein Token hat. Wird ein Token angetroffen, springt der Interpreter in die zuständige Verb Action-Routine. So ziemlich jede dieser Routinen verzweigt, z. T. mehrmals, in das freie RAM, wo wir Vektoren auf unsere Ergänzung richten können. Nur ist leider bis zum Sprung ins RAM häufig schon allerhand passiert, das wir wieder einnorden müssen. Deshalb ist eigentlich nur eine einzige Methode wirklich empfehlenswert:

Die reinen Disk-BASIC-Befehle, die unter Level 2 zu einem ?L3 Error oder ?SN Error führen, springen das RAM sofort an. Noch ist alles im Lot, und unser Programm kann kurz und bündig bleiben. Deshalb die zahllosen BASIC-Erweiterungen, die mit NAME aufgerufen werden. Wenn dem Befehl (hier PUT) noch ein weiteres Zeichen oder Token (hier TO) folgt, kann gleich am Anfang festgestellt werden, ob alles seine Richtigkeit hat. Spätere Syntaxchecks übernimmt wieder der Interpreter. Wenn z. B. PUT TO 20000? eingegeben wurde, wird der Befehl zunächst korrekt ausgeführt. Erst dann erkennt Mr. Microsoft das Fragezeichen und versteht Bahnhof: ?SN Error. Wir haben ihn ja nur ergänzt, nicht verändert.

Mit EDTASM ist die Eingabe des Programms am bequemsten. Wer diesen Assembler nicht hat, muß ihn entweder für sehr viel Geld kaufen oder gewisse Mitglieder unseres Clubs ansprechen, die angeblich jedes Copyright mißachten und Programme weitergeben. Einem böswilligen Gerücht zufolge soll es kein Mitglied geben, das es nicht tut. Muß ich deutlicher werden?

Arnulf Sopp, Wakenitzstr. 8, 2400 Lübeck 1, Tel. 0451-791926

```
                00100 ;***                    PUT TO adresse                ***
                00110 ;***                                                  ***
                00120 ;*** Ein neuer BASIC-Befehl zum freien Verschieben    ***
                00130 ;*** von BASIC inkl. Programmtext im RAM. Variable    ***
                00140 ;*** werden dabei gelöscht.                           ***
                00150
                00160 ;(C) 1984 by Arnulf Sopp, Wakenitzstr. 8, D-2400 Lübeck 1
                00170
                00180
                00190 ;Symbolvereinbarungen:
4182            00200 PUT     EQU     4182H   ;Adresse PUT-Verb Action-Routine
06CC            00210 BASIC   EQU     06CCH   ;BASIC-Warmstartadresse
1E5A            00220 ASCINT  EQU     1E5AH   ;ASCII-String -> DE als Integer
40FF            00230 DATPTR  EQU     40FFH   ;DATA-Pointer
40A4            00240 PSTPTR  EQU     40A4H   ;Pointer auf BASIC-Textanfang
40F9            00250 VLTPTR  EQU     40F9H   ;dto. auf Anfang Variablenliste
40FB            00260 ARRANF  EQU     40FBH   ;dto. Anf. dimens. Var.
40FD            00270 ARREND  EQU     40FDH   ;dto. Ende (Anfang freies RAM)
00BD            00280 TO      EQU     0BDH    ;TO-Token
                00290
                00300 ;PUT-Vektor verbiegen:
4182            00310         ORG     PUT
4182 C30080     00320         JP      PUTTO           ;neue Verb Action-Routine
                00330
                00340 ;neue Verb Action-Routine:
8000            00350         ORG     8000H           ;oder je nach RAM-Größe
8000 CF         00360 PUTTO   RST     08H             ;Syntax Error, falls
8001 BD         00370         DEFB    TO              ;nicht TO angetroffen
8002 CD5A1E     00380         CALL    ASCINT          ;Adresse in DE einlesen
8005 2AFF40     00390         LD      HL,(DATPTR)     ;für die Ermittlung
8008 ED4BF940   00400         LD      BC,(VLTPTR)     ;der Programmlänge
800C 0B         00410         DEC     BC              ;auf die 0 davor
800D DF         00420         RST     18H             ;rauf oder runter?
800E C8         00430         RET     Z               ;falls dieselbe Adresse
800F E5         00440         PUSH    HL              ;HL und BC vertauschen
8010 C5         00450         PUSH    BC
8011 E1         00460         POP     HL
8012 C1         00470         POP     BC
8013 D5         00480         PUSH    DE              ;neue Adresse für später
8014 380B       00490         JR      C,UPLOAD        ;Carry, falls rauf
8016 C5         00500         PUSH    BC              ;als Quelle für LDIR
8017 ED42       00510         SBC     HL,BC           ;Programmlänge
8019 E5         00520         PUSH    HL              ;als Zähler für LDIR
801A C1         00530         POP     BC              ;nach BC laden
801B 03         00540         INC     BC              ;Bytezähler korrig.
801C E1         00550         POP     HL
801D EDB0       00560         LDIR                    ;BASIC umschaufeln
801F 180C       00570         JR      CNGPTR          ;dort weiter
8021 E5         00580 UPLOAD  PUSH    HL              ;spätere Quelle für LDDR
8022 B7         00590         OR      A               ;Carry löschen
8023 ED42       00600         SBC     HL,BC           ;wie oben
8025 E5         00610         PUSH    HL
8026 C1         00620         POP     BC
8027 03         00630         INC     BC              ;Bytezähler korrig.
8028 19         00640         ADD     HL,DE           ;ans Ende des Programms
802A EB         00650         EX      DE,HL           ;Ziel DE laden
802A E1         00660         POP     HL              ;Quelle
802B EDB8       00670         LDDR                    ;wie oben, von hinten
                00680
                00690 ;BASIC-Zeiger nachstellen:
802D E1         00700 CNGPTR  POP     HL              ;neue Adresse
802E 22FF40     00710         LD      (DATPTR),HL     ;DATA-Pointer laden
8031 23         00720         INC     HL              ;eins höher
8032 22A440     00730         LD      (PSTPTR),HL     ;PST-Pointer laden
                00740
```

```
                00750 ;Zeilenzaiger im Programmtext nachstellen:
8035 E5         00760         PUSH    HL              ;erster Zeilenpointer
8036 23         00770         INC     HL              ;insges. 4X erhöhen
8037 D1         00780 LOOP    POP     DE
8038 23         00790         INC     HL              ;auf 1. Programmstatement
8039 23         00800         INC     HL              ;erhöhen (die nächste 0
803A 23         00810         INC     HL              ;muß EOL sein.)
803B AF         00820         XOR     A               ;A=0 für CPIR
803C 47         00830         LD      B,A             ;dto. BC (=256!)
803D 4F         00840         LD      C,A
803E EDB1       00850         CPIR                    ;0 als EOL aufsuchen
8040 EB         00860         EX      DE,HL           ;Register umordnen
8041 73         00870         LD      (HL),E          ;Zeilenpointer korri-
8042 23         00880         INC     HL              ;gieren
8043 72         00890         LD      (HL),D
8044 EB         00900         EX      DE,HL           ;alte Registerordnung
8045 E5         00910         PUSH    HL              ;neuer Zeilenpointer
8046 7E         00920         LD      A,(HL)          ;auf EOL prüfen
8047 23         00930         INC     HL              ;eine zweite 0?
8048 B6         00940         OR      (HL)
8049 20EC       00950         JR      NZ,LOOP         ;nein, weiter
                00960
                00970 ;Variablenpointers laden und Ende:
804B 23         00980         INC     HL              ;auf neuen VLTPTR erhöhen
804C 22F940     00990         LD      (VLTPTR),HL
804F 22FB40     01000         LD      (ARRANF),HL
8052 22FD40     01010         LD      (ARREND),HL
8055 E1         01020         POP     HL              ;Stack korrigieren
8056 C3CC06     01030         JP      BASIC
                01040
06CC            01050         END     BASIC
00000 TOTAL ERRORS
32283  TEXT AREA BYTES LEFT


ARRANF 40FB 00260   01000
ARREND 40FD 00270   01010
ASCINT 1E5A 00220   00380
BASIC  06CC 00210   01030 01050
CNGPTR 802D 00700   00570
DATPTR 40F5 00230   00390 00710
LOOP   8037 00780   00950
PSTPTR 40A4 00240   00730
PUT    4182 00200   00310
PUTTO  8000 00360   00320
TO     00BD 00280   00370
UPLOAD 8021 00530   00490
VLTPTR 40F9 00250   00400 00990
```

ML

Die MEMORY SIZE automatisch

In einem meiner früheren Artikel war beiläufig davon die Rede, daß ein Maschinenprogramm sich selbsttätig vor BASIC schützen kann, indem es selber die obere Speichergrenze für BASIC setzt. Dies ist sogar sehr einfach, wie wir sehen werden, denn auch der Microsoft-Interpreter kocht nur mit Wasser. Zum Verständnis der Materie möchte ich jedoch zuvor erklären, woher BASIC "weiß", bis wohin es sich ausbreiten darf.

Das RAM unseres Computers ist an der Speicherstelle FFFFh (65535d) zuende. Dies gilt jedenfalls für 64KB-Systeme. Wenn die Memory Size nach dem Einschalten nicht definiert wurde (nur <ENTER> gedrückt), können alle BASIC-Bestandteile (Programmtext, numerische Variable, Stringvariable) bis dorthin anwachsen.

Anderfalls wird die Obergrenze an der Stelle 40B1/40B2h (16561/16562d) in der Communications Region abgelegt. Bei jedem Befehl, der BASIC anwachsen läßt (Einfügen einer Programmzeile, Zuweisung einer Variablen), wird zunächst geprüft, ob dafür bis zur Obergrenze noch genügend Platz ist. Falls nicht, wird ein "?OM Error" ausgegeben. Dies geschieht übrigens auch, wenn der Stack nach unten bis in die VLT (s. früheren Artikel) anwächst, wenn beispielsweise ein GOSUB oder eine FOR-NEXT-Schleife aktiv ist.

Die beiden Bytes, die die Memory Size halten, lassen sich ohne weiteres verändern, sogar mit POKE von BASIC aus. Man braucht daher nicht den Comcuter aus- und wieder einzuschalten (oder mit SYSTEM /0 die Einschaltroutine anzuspringen), wenn man während der Arbeit ein zu schützendes Programm nachladen will. Hier möchte ich als bekannt voraussetzen, wie man eine Integerzahl wie die Speichergröße in zwei Bytes aufteilt, um sie nach 40B1/40B2h zu laden.

Damit ist allerdings noch nicht alles getan. Hierzu ein Blick auf den oberen Speicherbereich: Ganz oben liegt gegf. der geschützte Bereich. Direkt darunter schließt sich die String Area an, der Bereich, wo Zeichenketten gespeichert werden, die nicht in der Form V$="..." im Programmtext stehen. Unterhalb beginnt der Stack, der sich nach unten vergrößert. Der Bereich zwischen der Stack-Untergrenze und der VLT ist frei.

Der String Space umfaßt nach dem Einschalten zunächst 50 Bytes. Mit CLEAR kann er nach Bedarf vergrößert werden. Seine Untergrenze (Memory Size minus 50 bzw. CLEAR-Argument) wird ebenfalls in der Communications Region abgelegt, und zwar an der Stelle 40A0/40A1h (16544/16545d). Wird diese Grenze durch Zuweisung von Zeichenketten unterschritten, wird ein "?OS Error" ausgegeben.

Beim Verändern der Memory Size muß nun zuletzt noch der Stack den neuen Werten angepaßt werden, denn er würde sonst unser zu schützendes Maschinenprogramm oder die Zeichenketten zuschaufeln. Hierzu wird der Stack Pointer (Stapelzeiger) auf den Wert der String-Untergrenze gesetzt. Daraus geht logischerweise hervor, daß der Stack im Augenblick der Veränderung nichts enthalten darf (z. B. GOSUB-Rücksprungadresse), denn der Zeiger deutet danach gewissermaßen in die falsche Richtung.

Leider kann man den Stack Pointer nicht mit einem BASIC-Befehl verbiegen. Ein kleines Maschinenprogramm, dessen entscheidender Bestandteil die Befehlsfolge

```
LD HL,(40A0H)
LD SP,HL
```

ist, kann aber durchaus über DATA eingelesen und angestrungen werden. Näher möchte ich darauf hier nicht eingehen, denn dieser Artikel ist ohnehin eher für den Assembler-Programmierer gedacht.

Das ist bereits alles. Abschließend wäre noch zu bemerken, daß das Programmsegment, das die Memory Size setzt, an den unteren Adressen des

Programme stehen sollte. Es ist nach der Initialisierung nämlich überflüssig geworden und braucht somit nicht vor BASIC geschützt zu werden. Das untenstehende Assembler-Programm gibt ein Beispiel. Das Hauptprogramm selbst löscht nur den Bildschirm (CALL 01C9H hätte genügt), denn es geht hier nur darum, die selbsttätige Manipulation der Memory Size zu demonstrieren.

```
                00100 ;PROGRAMM SETZT SEINE MEMORY SIZE SELBSTTAETIG
                00110
                00120 ;AUFGEWAERMT BY ARNULF SOPP. TEL. 0451-791926
                00130
                00140 ;SYMBOLVEREINBARUNGEN:
0033            00150 OUTACC  EQU     0033H   ;AKKUINHALT -> BILDSCHIRM
40B1            00160 MEMSIZ  EQU     40B1H   ;HIER IST DIE MEM SIZE ABGELEGT
40A0            00170 STRING  EQU     40A0H   ;DTO. STRING SPACE-UNTERGRENZE
001C            00180 HOME    EQU     1CH     ;VIDEO-STEUERCODE "HOME CURSOR"
001F            00190 CLEAR   EQU     1FH     ;DTO. "CLEAR TO END OF FRAME"
                00200
                00210
7000            00220         ORG     7000H           ;ODER WO AUCH IMMER
                00230
                00240 ;MEM SIZE, STRING SPACE, STACK EINNORDEN:
7000 210E70     00250 INIT    LD      HL,ANFANG-2     ;AB HIER GESCHUETZT
7003 22B140     00260         LD      (MEMSIZ),HL     ;-> COMMUN. REGION
7006 113200     00270         LD      DE,50           ;FUER "CLEAR 50"-AEQUIV.
7009 97         00280         OR      A               ;CARRY = 0 WEGEN SBC
700A ED52       00290         SBC     HL,DE           ;HL = MEM SIZE - 50
700C 22A040     00300         LD      (STRING),HL     ;STRING SPACE BEGRENZEN
700F F9         00310         LD      SP,HL           ;NEUER STACK-ANFANG
                00320
                00330 ;HAUPTPROGRAMM (WAHRLICH EINE EDV-OFFENBARUNG!):
7010 3E1C       00340 ANFANG  LD      A,HOME          ;CURSOR NACH LINKS OBEN
7012 CD3300     00350         CALL    OUTACC          ;AKKU -> BILDSCHIRM
7015 3E1F       00360         LD      A,CLEAR         ;ALLES DAHINTER LOESCHEN
7017 CD3300     00370         CALL    OUTACC          ;WIRD'S BALD?
701A F7         00380         RST     30H             ;UNTER DOS "LOAD DEBUG",
                00390                                 ;UNTER L2 BASIC-WARMSTART
                00400                                 ;VON HINTEN DURCH DAS
                00410                                 ;KNIE INS AUGE
                00420
7000            00430         END     INIT            ;START AB "INIT" MIT "/"
00000 TOTAL ERRORS
3F940  TEXT AREA BYTES LEFT


ANFANG 7010 00340   00250
CLEAR  001F 00190   00360
HOME   001C 00180   00340
INIT   7000 00250   00430
MEMSIZ 40B1 00160   00260
OUTACC 0033 00150   00350 00370
STRING 40A0 00170   00300
```

Arnulf Sopp, Wakenitzstr. 3, D-2400 Lübeck 1, Tel.: 0451-791926

Kaffeekochen ab sofort gestattet.

Dies sind eigentlich zwei Artikel. Ich will mich aber kurz- und beide zusammenfassen. Der eine handelt von einem Problem und seiner Lösung, der andere von dem wesentlich kniffligeren Problem, das nämliche zu lösen und wiederum dessen Lösung. Oder so ähnlich.

Jedenfalls haben Wolfgang Frey und ich vor kurzem ein bißchen Assembler trainiert. Wir schusterten eine kleine Routine zusammen, die bei jedem Tastendruck einen kurzen Rülpser auf den Lautsprecher ausgab. Derlei geht mit der Zeit freilich auf die Nerven. Um aber nicht die Zeit mit Spielkram vergeudet zu haben, zapte ich diese Routine jeweils ein wenig abgewandelt in SYS4/SYS und SYS13/SYS ein. Nun piept es nur noch, wenn ein DOS- oder BASIC-Fehler auftritt. Aber wirklich sinnvoll ist es in SYS6/SYS. Dort wird kopiert und formatiert.

Es dauert schon einige Minuten, bis z. B. die Systemdiskette inkl. Formatierung kopiert ist. Während dieser Zeit geht der Hacker normalerweise zum Briefkasten oder Kaffee kochen. Tritt nun aber ein Fehler auf, der mit der Eingabe "<A>bbruch, <W>iederholung, <F>ortfahren" beantwortet werden muß (G-DOS; in NEWDOS erscheint dergl. in Englisch), dann ist der Hacker woanders. Ein Rufsignal wie oben beschrieben wäre hochwillkommen.

Leider ist SYS6/SYS nahezu bis auf das letzte Bit voll. Eine Lärmroutine dieser Art hat ohne Änderungen keinen Platz mehr. Deshalb habe ich eine besonders lange Fehlermeldung abgekürzt: Aus "Schlechte Parameter oder Konflikt mit Pdrivedaten" wurde "PD/Par.!". Der so freigewordene Platz faßt die Krawallroutine leicht und läßt sogar noch ein paar Bytes Platz für schlechte Zeiten.

Dieses Pfeifprogramm ist auf der übernächsten Seite gelistet. Es geht von folgender Überlegung aus: Die Meldung "<A>bbruch ..." wird in SYS6/SYS an der Speicherstelle 58EBh mit dem Befehl CALL 4467h angezeigt. Stattdessen steht nun hier ein CALL nach 6E5Eh. Das ist im Bereich der genannten nunmehr abgekürzten Fehlermeldung. Zunächst wird die Anzeige nachgeholt, die im ersten Zap (oberer Sektordump, geänderte Adresse unterstrichen) unterdrückt worden ist. Sodann wird ein Tatütata aus zwei Tönen generiert. Dabei werden abwechselnd ein positives und ein negatives Signal auf den Cassettenport FFh gelegt. Die frequenzbestimmenden Verzögerungen zwischen zwei Signalen verhalten sich zur Tonlänge je nach Ton umgekehrt (oder wie drückt man das mathematisch korrekt aus?).

Nach jedem Tatü wird die Tastatur abgefragt. Wurde keine Taste gedrückt, ist der User wohl noch nicht zugegen. Dann lärmt es eben weiter. Andernfalls passiert dasselbe wie gewohnt: Nach irgendeinem Tastendruck und anschließend der Eingabe von A, W oder F tut der Computer, was er nicht lassen kann. Es mutet vielleicht befremdlich an, daß das Unterprogramm KRLOOP nicht mit RET abgeschlossen ist. Es ist! Am Ende der Routine ab 002Bh, die die Tastatur befragt, steht ein RETurnbefehl.

Die Hexcodes, die dieses Programm darstellen, sind in der zweiten Spalte des Assemblerlistings zu sehen. Sie finden sich wieder im unteren Sektordump. Dabei ist zu beachten, daß die ersten 9 unterstrichenen Codes die gekürzte Fehlermeldung sind. In der nächsten Zeile beginnt das eigentliche Programm.

Die zweite Geschichte handelt von der aufregenden Suche nach einem Caller, die schließlich in einem wahren Showdown endete. Die mehrfach zitierte Fehlermeldung, bei der es piepen sollte, wird über 4467h angezeigt. Dort steht der Befehl JP 4BA6h. Das Registerpaar HL muß zu diesem Zweck als Zeiger mit der Adresse des Textes geladen werden. Das erste Byte dieser Meldung steht in 5A88h. So lag es nahe, mit SUPERZAP nach der Bytefolge 21-88-5A (LD HL,5A88H) zu suchen. Es gibt sie aber nicht. Da

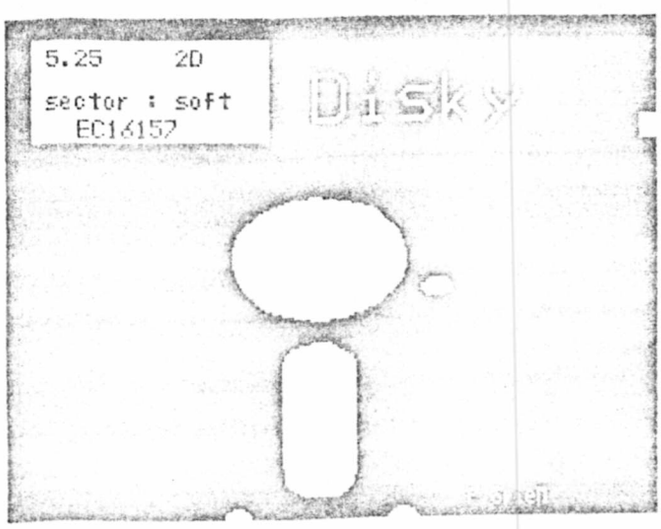
sind zu viele Möglichkeiten, HL mit einer davor oder dahinter liegenden Adresse zu laden, deshalb schrieb ich anstelle einer endlosen Suche eine kleine Routine, die mir die Arbeit abnehmen sollte:

Zunächst werden die benutzten Register in einen eigenen Puffer geschrieben, denn ein PUSH hätte die Ermittlung des Callers erschwert. Anschließend wird die RET-Adresse (Caller +3) vom Stack gepopt (und sofort wieder draufgepusht) und über die DOS-Routine 4063h angezeigt. Ein Blank dahinter sorgt für Übersichtlichkeit. Wegen der Scrollerei während der DOS-Arbeit setzte ich vorsichtshalber die erste Anzeigestelle auf 3F00h. So können Zeilenvorschübe die angezeigte Adresse nicht zum Verschwinden bringen.

Das Resultat zeigt die Bildschirm-Hardcopy unter dem Assembler-Listing. Mitten im Bildschirm, durch Zeilenvorschübe des DOS auseinandergezogen, finden sich mehrere Hexzahlen. Die letzte vor der "erzwungenen Beendigung der Funktion" mit dem Kommando A lautet 58EEh. Demzufolge wurde die mehrfach erwähnte Fehlermeldung von 58EBh aus aufgerufen. Dort liegt jetzt als Kuckucksei der erste Zap, der den zweiten, das Krachprogramm, in Gang bringt. So einfach ist das. Oder so schwierig, je nach dem.

Den zweiten Teil dieses Beitrags bringe ich nicht, um die spannende Geschichte einer Verfolgungsjagd nach einer Adresse zu schildern. Vielmehr stellen sich derartige Probleme ständig bei der Zapperei. Dies ist ein Weg, sie zu lösen.

Arnulf Sopp, Tel. 0451-791926



```
000B00: 2195 59CB 7EC2 1A52 F6C0 CD09 44C3 C858 !.Y.ß..R....D..X
000B10: CDDA 57C0 DD34 0A20 03DD 340B AFC9 AFF6 ..W..4. ..4.....
000B20: 00C8 21E0 59CD 6744 CDDD 56CD A058 CD81 ..!.Y.gD..V..X..
000B30: 58F6 FFC9 CD67 443E 0DC3 BD58 7EFE 2023 X....gD>...Xß. #
000B40: 3804 FE80 3802 3E20 CDBD 5810 EFC9 3E20 8...8.> ..X...>
000B50: CDBD 5810 F9C9 7DC6 056F 0608 CDB3 587E ..X...ü..o....Xß
000B60: FE20 0603 3E2F C4BD 587E FE20 23C4 BD58 . ..>/..Xß. #..X
000B70: 10F7 C9D5 F5CD 3300 F1D1 C9CD 6744 3A95 ......3.....gD:.
000B80: 59CB 7FC2 4952 2184 5ACD E458 FE01 D03E Y...IR!.Z..X...>
000B90: 39C3 1A52 CD67 4421 B35A C5E5 7EB7 2320 9..R.gD!.Z..ß.#
000BA0: FBCD 5E6E CD2B 57E1 0EFF E534 0C35 28F4 ..^n.+W....4.5(.
000BB0: BE23 20F7 CDBD 0102 0059 58CD 8158 79B7 .# ......YX..Xy.
000BC0: E1C1 C901 0004 2133 59C5 4E23 4623 EB3E ......!3Y.N#F#.>
000BD0: 2F3C 0938 FCED 42C1 EBFE 3020 040C 0D28 /<.8,.B...0 ...(
000BE0: 040C CDBD 5810 E27B C630 C3BD 58F0 D818 ....X..ä.0..X...
000BF0: FC9C FFF6 FF00 FF6F 4259 0007 0001 0000 .......oBY......

002100: 736B 6574 7465 2077 6972 6420 6B6F 7069 skette wird kopi
002110: 6572 740D 5369 6E64 2053 7973 7465 6D20 ert.Sind System
002120: 756E 6420 0320 6964 656E 7469 7363 683F und . identisch?
002130: 0344 6973 6B65 7474 6520 666F 726D 6174 .Diskette format
002140: 6965 7265 6E3F 0350 442F 5061 722E 210D ieren?.PD/Par.!.
002150: CD67 44C5 01C0 60C5 CD71 6EC1 7948 4728 .gD...`..qn.yHG(
002160: F6C1 C9C5 413E 01D3 FF10 FE41 3CD3 FF10 ....A>.....A<...
002170: FEC1 10EF C32B 0000 0000 0046 6F72 6D61 .....+.....Forma
002180: 7466 6568 6C65 7220 4672 6F6E 7473 6569 tfehler Frontsei
002190: 7465 2076 6F01 02A3 6E6E 2053 7075 7220 te vo...nn Spur
0021A0: 0346 726F 6E74 2052 7D63 6BCD C06E C8CD .Front Rück..n..
0021B0: D94C D8C8 2BC9 7EFE 0DC8 CDD9 4CD0 C31A .L..+.ß.....L...
0021C0: 527E FE3A 2001 237E D630 FE0A D0CD E76E Rß.: .#ß.0.....n
0021D0: CD76 4720 E97B 37C9 CDF1 6E18 03CD 0F6F .vG .ä7...n....o
0021E0: 7AB7 7BC8 C318 52E5 CD14 6F7E D641 FE08 z.ä...R...oß.A..
0021F0: 300D E106 01E5 CD16 6F7E FE48 2320 E5CB 0.......oß.H# ..
```

*6E55

```
6E5E            00100           ORG     6E5EH
6E5E CD6744     00110           CALL    4467H           ;Meldung anzeigen
6E61 C5         00120           PUSH    BC
6E62 01C060     00130           LD      BC,60C0H        ;2 Zähler: 60 und C0
6E65 C5         00140 KRACH     PUSH    BC              ;retten
6E66 CD716E     00150           CALL    KRLOOP          ;1 Schwingung erzeugen
6E69 C1         00160           POP     BC              ;zurückholen
6E6A 79         00170           LD      A,C             ;B und C vertauschen
6E6B 48         00180           LD      C,B
6E6C 47         00190           LD      B,A
6E6D 28F6       00200           JR      Z,KRACH         ;falls keine Taste gedr.
6E6F C1         00210           POP     BC              ;T. gedr.: Register rest.
6E70 C9         00220           RET                     ;weiter mit Fehlerbeh.
6E71 C5         00230 KRLOOP    PUSH    BC              ;1. Zähler retten
6E72 41         00240           LD      B,C             ;2. Zähler laden
6E73 3E01       00250           LD      A,1             ;posit. Signal
6E75 D3FF       00260           OUT     (0FFH),A        ;auf Krawallport
6E77 10FE       00270           DJNZ    $               ;Warteschleife
6E79 41         00280           LD      B,C             ;2. Zähler erneuern
6E7A 3C         00290           INC     A               ;Akku = 2, neg. Signal
6E7B D3FF       00300           OUT     (0FFH),A        ;auf Lärmport
6E7D 10FE       00310           DJNZ    $               ;Warteschleife
6E7F C1         00320           POP     BC              ;1. Zähler restaur.
6E80 10EF       00330           DJNZ    KRLOOP          ;bis 1 Schwingg. zuende
6E82 C32B00     00340           JP      002BH           ;Tast. abfr. (dort RET)
0000            00350           END
00000 mal gepennt
34328  Zeichen verfügbar
```

```
8000            00100           ORG     8000H           ;beliebige Adresse
8000 210980     00110 START     LD      HL,DEVIAT       ;Adr. d. Umleitung laden
8003 226844     00120           LD      (4468H),HL      ;Sprungbefehl verbiegen
8006 C32D40     00130           JP      402DH           ;retour ins DOS
8009 223880     00140 DEVIAT    LD      (HLBUFF),HL     ;Register retten
800C ED533A80   00150           LD      (DEBUFF),DE
8010 323C80     00160           LD      (ABUFF),A
8013 2A3680     00170           LD      HL,(VDBUFF)     ;Bildschirmstelle laden
8016 D1         00180           POP     DE              ;Caller ermitteln
8017 D5         00190           PUSH    DE              ;RET-Adresse restaurieren
8018 CD6340     00200           CALL    4063H           ;DE in Hex anzeigen
801B 3E20       00210           LD      A,' '           ;Blank zwischen den
801D 77         00220           LD      (HL),A          ;  Adressen anzeigen
801E 23         00230           INC     HL              ;nächste Bildschirmstelle
801F 7C         00240           LD      A,H             ;Screen zuende, d. h.
8020 FE40       00250           CP      40H             ;  HL >= 4000h?
8022 2002       00260           JR      NZ,GOON         ;keine Panik, falls nein
8024 263F       00270           LD      H,3FH           ;sonst wieder 3F00h
8026 223680     00280 GOON      LD      (VDBUFF),HL     ;neue Stelle merken
8029 2A3880     00290           LD      HL,(HLBUFF)     ;Register restaurieren
802C ED5B3A80   00300           LD      DE,(DEBUFF)
8030 3A3C80     00310           LD      A,(ABUFF)
8033 C3A64B     00320           JP      4BA6H           ;jetzt Meldung anzeigen
8036 003F       00330 VDBUFF    DEFW    3F00H           ;hält Bildschirmadresse
0002            00340 HLBUFF    DEFS    2               ;Puffer für Register
0002            00350 DEBUFF    DEFS    2
0001            00360 ABUFF     DEFS    1
8000            00370           END     START           ;Einsprungadresse
00000 mal gepennt
34106  Zeichen verfügbar
```

2.1b - mod.
1984 durch
Arnulf Sopp

Komm, hau 55C7 55CC 55D8

Komm, hau rein:ndf 1

Diskette wird formatiert 6810

" NEW LINE ", wenn Zieldiskett693E 6944 erk Nr. 1

58EE

Diskette hat Daten

Zieldisketten name, -datum: G-DOS FC SOFTSOPP

<A>bbruch, <W>iederholung, <F>ortfahren

A 4DF5

Erzwungene Beendigung der Funktion

Komm, hau rein:

(C) '84 by A. Sopp

BEL ohne Nachladen gleich bei BOOT

Im letzten Artikel zu diesem Thema stellte ich ein Programm vor, das den ASCII-Code 07 nutzbar macht, den unsere Maschinen normalerweise ignorieren. Er wird BEL genannt und führt bei der Ausgabe "auf den Bildschirm" (mit PRINT oder einem entsprechenden Maschinenbefehl) zu einem akustischen Signal.

Wie bereits angedroht, folgt hier eine Version, die Bestandteil von SYS0/SYS ist und daher gleich beim Booten aktiviert wird. Sie residiert im Adreßbereich des L2-ROMs an einer Stelle, die nur während der IPL-Sequenz unmittelbar nach dem Einschalten benötigt wird. Dieser Speicherbereich ist daher Sekundenbruchteile nach dem Druck auf den Knopf frei. Da man im ROM aber bekanntlich nicht schreiben kann, muß der EG 64 MBA angeschlossen sein, der das dort liegende RAM zugänglich macht.

Zum Programm selbst muß hier nicht mehr viel gesagt werden, das war im vorigen Beitrag hoffentlich ausführlich genug. Auch hier wird an der Stelle CPBEL zunächst geprüft, ob ASCII 07 anliegt und gepfiffen, falls ja. Der Unterschied besteht in der Einsprungstelle: Im Videotreiber wird an der Stelle 0506 DE mit 0480 geladen und dieser Wert als RET-Adresse auf den Stack gepusht. Hier steht nun aber der CALL nach CPBEL. Daher muß in Zeile 670 der Befehl LD DE,0480H ausgeführt werden, falls im Akku nicht 07 stand. Diesen notwendigen Befehl habe ich ja mit meinem CALL einfach übertüncht.

Dieses Programm ist demnach ein Eingriff mitten im Interpreter, der ohne Memory Banking nicht denkbar wäre. Der Vorteil liegt darin, daß der Anwenderspeicher oberhalb des ROM-Adreßbereichs nicht tangiert wird. Die einfachere Routine vom letzten Mal besetzt nun einmal leider das Himem.

Zum Verständnis des Zaps ist er im Listing komplett wiedergegeben, nicht nur der BEL-Bestandteil. Im ersten Teil wird zerst geprüft, ob der Linkspfeil gedrückt wurde. Mit ihm kann man nämlich verhindern, daß die ganzen Änderungen aktiv werden. Also ein abschaltbarer Zap, sozusagen. Ohne Linkspfeil wird zunächst das ROM auf das RAM kopiert, das RAM im I/O-Adreßbereich wird auf 00 gesetzt. Ab Zeile 340 wird nun das Kukkucksei namens NEWCOD in den Interpreter gelegt. Die Routinen, die es anspringen sollen (Videotreiber und INT-Service-Routine) werden entsprechend verbogen. Übrigens ist die INT-Ergänzung für das Erkennen der Dreitastenbefehle für das Banking und den Spooler (s. frühere Infos) gut. Zum genaueren Verständnis bitte ich den Leser, diese Routinen an 0506 (mitten im Videotreiber) und 45D3 (mitten in der INT-Bearbeitung) selbst zu disassemblieren. Das würde hier zu weit führen.

Die beiden Sektordumps zeigen den Zap in den letzten beiden Sektoren von SYS0/SYS. Wie üblich sind die unterstrichenen Codes zu ändern. Dieser neue Programmteil wird als Unterprogramm aufgerufen aus 4F2A. Deshalb muß dort der Befehl LD A,(3840H) ersetzt werden durch CALL 50A8H. Dieser zusätzliche Zap steht im Sektor 0C von SYS0/SYS an den Bytes 43/44/45H. Hier sind die Codes CD-A8-50 einzuzappen. Dieser Zap ist nicht als Sektordump abgebildet, um für 3 Bytes das Info nicht unnötig dick zu machen.

Mir ist klar, daß kaum jemand von Euch den MBA hat. Mit meinen Beiträgen möchte ich Euch deshalb für das Ding interessieren, denn man kann sein Geld kaum besser anlegen.

Arnulf Sopp, Tel. 0451-791926

```
0072            00010 DESTIN  EQU   0072H          ;dort NEWCOD ablegen
50A8            00100         ORG   50A8H          ;am Ende von SYS07SYS
50A8 3A4038     00110         LD    A,(3840H)      ;Tastatur abfragen
50AB CB6F       00120         BIT   5,A            ;Linkspfeil gedrückt?
50AD C0         00130         RET   NZ             ;nichts veränd., falls ja
50AE E5         00140         PUSH  HL             ;benutzte Register retten
50AF F5         00150         PUSH  AF
50B0 F3         00160         DI                   ;bloß keine Störungen!
50B1 0604       00170         LD    B,04H          ;4 Codes auf MBA ausgeben
50B3 3E0E       00180         LD    A,0EH          ;1. Code
50B5 D3DF       00190 BANK    OUT   (0DFH),A       ;auf Banking-Port
50B7 3D         00200         DEC   A              ;nächster Code
50B8 10FB       00210         DJNZ  BANK           ;usw.
50BA 3D         00220         DEC   A              ;0A aussparen
50BB D3DF       00230         OUT   (0DFH),A       ;09 ausgeben
50BD 010036     00240         LD    BC,3600H       ;Zähler für L2/4-ROM
50C0 61         00250         LD    H,C            ;H=00
50C1 69         00260         LD    L,C            ;HL=0000
50C2 51         00270         LD    D,C            ;D=00
50C3 59         00280         LD    E,C            ;DE=0000
50C4 EDB0       00290         LDIR                 ;ROM auf RAM kopieren
50C6 70         00300         LD    (HL),B         ;(3600)=00
50C7 1C         00310         INC   E              ;Ziel DE=3601
50C8 01FF09     00320         LD    BC,09FFH       ;Zähler für I/O-Bereich
50CB EDB0       00330         LDIR                 ;dort Nullen einschreiben
50CD 21FB50     00340         LD    HL,NEWCOD      ;Anfang Zap für "ROM"
50D0 117200     00350         LD    DE,DESTIN      ;Ziel des Zaps
50D3 D5         00360         PUSH  DE             ;brauchen wir noch
50D4 012D00     00370         LD    BC,FINITO-NEWCOD       ;Länge des Zaps
50D7 EDB0       00380         LDIR                 ;Zap übertragen
50D9 3EC3       00390         LD    A,0C3H         ;JP-Opcode
50DB 320605     00400         LD    (0506H),A      ;Videotreiber verwanzen
50DE 218200     00410         LD    HL,CPBEL-OFFSET ;0506: JP 0082
50E1 220705     00420         LD    (0507H),HL     ;Sprungadresse ablegen
50E4 DBDF       00430         IN    A,(0DFH)       ;reset MBA
50E6 3E08       00440         LD    A,08H          ;read RAM 0000-2FFF
50E8 D3DF       00450         OUT   (0DFH),A       ;auf Banking-Port
50EA 3E0F       00460         LD    A,0FH          ;reset nicht mit R-Taste
50EC D3DF       00470         OUT   (0DFH),A       ;auch ausgeben
50EE E1         00480         POP   HL             ;HL=0072
50EF 3ECD       00490         LD    A,0CDH         ;CALL-Opcode
50F1 32D345     00500         LD    (45D3H),A      ;in INT-Service-Routine
50F4 22D445     00510         LD    (45D4H),HL     ;45D3: CALL 0072
50F7 F1         00520         POP   AF             ;Register restaurieren
50F8 E1         00530         POP   HL
50F9 FB         00540         EI                   ;INT wieder zulassen
50FA C9         00550         RET                  ;und zuende booten
50FB            00560 OFFSET  EQU   $-DESTIN       ;Subtrahend f. Relokation
50FB 3A2038     00570 NEWCOD  LD    A,(3820H)      ;Tastaturabfrage
50FE FED0       00580         CP    0D0H           ;<,./> gedrückt?
5100 2806       00590         JR    Z,RST28        ;verarbeiten, falls ja
5102 3A1038     00600         LD    A,(3810H)      ;Tastaturabfrage
5105 FEE0       00610         CP    0E0H           ;<567> gedrückt?
5107 C0         00620         RET   NZ             ;norm. weiter, falls nein
5108 F61C       00630 RST28   OR    1CH            ;Akku für RST 28 vorber.
510A EF         00640         RST   28H            ;und SYS26/SYS laden
510B FE07       00650 CPBEL   CP    07H            ;BEL-Code?
510D 2806       00660         JR    Z,BEL          ;falls ja
510F 118004     00670         LD    DE,0480H       ;überschriebener Befehl
5112 C30905     00680         JP    0509H          ;dahinter weiter
5115 0EB0       00690 BEL     LD    C,0B0H         ;Anfangswert f. Schleifen
5117 3E01       00700 BEEP    LD    A,01H          ;positiver Impuls
5119 D3FF       00710         OUT   (0FFH),A       ;auf Port ausgeben
511B 10FE       00720         DJNZ  $              ;etwas warten
511D 41         00730         LD    B,C            ;Schleifenzähler erneuern
511E 3C         00740         INC   A              ;A=2, negativer Impuls
511F D3FF       00750         OUT   (0FFH),A       ;ausgeben
5121 10FE       00760         DJNZ  $              ;ein wenig trödeln
5123 0D         00770         DEC   C              ;Zähler erniedrigen
5124 41         00780         LD    B,C            ;und laden
5125 10F0       00790         DJNZ  BEEP           ;bis Ton zuende
5127 C9         00800         RET                  ;ins Betriebssystem
5128            00810 FINITO  EQU   $
0000            00820         END
00000 mal gepennt
32739 Zeichen verfügbar

                                        C. '84 by Sopp
```

DIE GEANDERTEN SEKTOREN:

C. '84 by Sopp

```
000D00: BF01 00E0 4F80 BCBC 2054 4353 2D20 BFC2 ....O    TCS- .
000D10: BFC2 BFC2 BFC2 8B8C BCB4 C400 0000 0000 ..... .....
000D20: 0000 A09E 81C1 8020 8080 BF20 C020 2020 ..... .  .
000D30: BE83 838D C431 3938 34C2 6475 7263 680A .....1984.durch.
000D40: AFBC BC9F C6BF BCBC 9F20 20AF BCBC 9FC2 . . .  .  .
000D50: AFBC BC9F C3B8 BFBC BC20 A894 C208 BCBF . . .  .. .
000D60: BC20 C120 AFB0 B09C 2020 C241 726E 756C .  .  .Arnul
000D70: 6620 536F 7070 CE0D 0000 0000 0000 0000 f Sopp........
000D80: 4441 5455 4D3F 2028 5454 2E4D 4D2E 4A4A DATUM? (TT.MM.JJ
000D90: 2920 035A 4549 543F 2020 2848 483A 4D4D ) .ZEIT?  (HH:MM
000DA0: 3A53 5329 2003 5454 2E4D 4D2E 4A4A 2020 :SS) .TT.MM.JJ
000DB0: 4848 3A4D 4D3A 5353 0D01 1F01 0C53 6400 HH:MM:SS.....Sd.
000DC0: 1800 3C00 3C3A 4038 CB6F C0E5 F5F3 0604 ..<.<:@8.o......
000DD0: 3E0E D3DF 3D10 FB3D D3DF 0100 3661 6951 >...=..=....6aiQ
000DE0: 59ED B070 1C01 FF09 EDB0 21FB 5011 7200 Y. p..... !.P.r.
000DF0: D501 2D00 EDB0 3EC3 3206 0521 8200 2207 ..-.. >.2..! .".
```

```
000E00: 05DB DF01 F7E6 503E 08D3 DF3E 0FD3 DFE1 ......P>...>....
000E10: 3ECD 32D3 4522 D445 F1E1 FBC9 3A20 38FE >.2.E".E....: 8.
000E20: D028 063A 1038 FEE0 C0F6 1CEF FE07 2806 .(.:.8........(.
000E30: 1180 04C3 0905 0EB0 3E01 D3FF 10FE 413C . ..... >.....A<
000E40: D3FF 10FE 0D41 10F0 C900 0000 0000 0000 .....A..........
000E50: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E60: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E70: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E80: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000E90: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EA0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EB0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EC0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000ED0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EE0: 0000 0000 0000 0000 0000 0000 0000 0000 ................
000EF0: 0000 0000 0000 0000 0000 0000 0202 004D ...............M
```

DSMBLR/CMD — ein wenig komfortabler

Es ist nicht besonders sinnvoll, Utilities hier im Info vorzustellen, denn bei unseren Tauschaktivitäten (legal oder nicht) hat eh' jeder früher oder später alles. Aber interessante Änderungen in Dienstprogrammen sind das Papier schon wert.

DSMBLR/CMD ist ein Disassembler, der (u. a.) EDTASM-kompatiblen Sourcecode erstellt. Damit ist es möglich, fremde Maschinenprogramme praktisch beliebig zu verändern (sofern man sich in die Programme eingedacht hat und ihren Ablauf versteht, damit man keine wichtigen Funktionen ruiniert).

Wenn die DSMBLR-Source von EDTASM "verstanden" werden soll, muß sie einen Header von 6 Zeichen und Zeilennummern haben. Zu diesem Zweck wird DSMBLR mit der Syntax DSMBLR,(HEADER,NUMBER) aufgerufen. Das ist ein bißchen umständlich. Ein Job, der mit DO,DIS aufgerufen wird und dieses Bandwurmkommando enthält, ist bequemer. Noch simpler ist die Lösung, die in den beiden Listings auf der nächsten Seite vorgestellt ist: Ein kleines Programm wird mitten in den DOS-Befehlspuffer geladen. Es enthält zunächst das Aufrufkommando und anschließend zwei Befehle, die den Aufruf bewerkstelligen (Zz. 5 und 6). Jetzt wird nur noch DIS eingegeben.

Der String in Zz. 2 und 3 kürzt die Optionen HEADER und NUMBER ab, was erlaubt ist. Außerdem enthält er weitere Parameter, auf die ich aber nicht weiter eingehen will. Sie sind hier nicht wichtig. Da solche Strings mit einer Länge von 18 Zeichen dargestellt werden, hat DSMBLR aus dieser Zeichenkette zwei Zeilen gemacht.

Die beiden Listings zeigen, wie DSMBLR arbeitet. Nach zwei Kopfzeilen, die auf jeder Seite einer Disassembly wiederholt werden, folgt der Code. Auf einfache Weise kann Daten- und Befehlscode unterscheidbar gemacht werden. Leider müssen aber die Pseudo-Ops DW (kommt hier nicht vor) und DB mit dem Editor/Assembler je nach dem in DEFB, DEFW oder DEFM verwandelt werden, damit die Syntax für EDTASM paßt. Um nahezu beliebig wursteln zu können, werden alle Adressen in Labelform wiedergegeben.

An den Kopfzeilen ist zwar nicht viel auszusetzen, aber auf deutsch lesen sie sich leichter. Das zweite Linefeed vor der ersten Programmzeile stört die optische Verbindung der Spaltenüberschriften und der Spalten selbst. Zusätzlich kann man, wenn man mag, eine persönliche Kopfzeile wie im unteren Listing entwerfen. Kleiner Schönheitsfehler: So wie diese Änderung sieht dann auch das Hello des Disassemblers aus, denn es wird aus eben diesem String erzeugt. Na und? Die kleine Routine ist übrigens kein "Fremdprogramm". Ich habe dieses Wort in die Zeile aufgenommen, weil ich von eigenen Programmen die Source habe und in diesen Fällen nicht auf DSMBLR angewiesen bin. Dies ist ja nur eine Demonstration.

Die notwendigen Änderungen, die im relativen Sektor 14h (20d) durchgeführt werden, gehen aus den beiden Hexdumps hervor (alt oben, neu unten). Das letzte Byte ist das P von "PAGE" bzw. das S von "Seite". Die ersten Bytes des nächsten Sektors müssen entsprechend geändert werden ("AGE" mit "eite" überschreiben). Eine zusätzliche Änderung ist sinnvoll, wenn der Computer/Drucker kleine Umlaute und ß hat: Im relativen Sektor 0Eh (14d) des Files wird das relative Byte 17h von 7B auf 7F gezapt. Jetzt werden diese Codes ausgedruckt und nicht durch Punkte ersetzt.

Arnulf Sopp, Tel. 0451-791926

```
MISOSYS Disassembler - Disk Version 3.0  DIS/CMD      PAGE  00001

ADDR CONTENTS LINE# LABEL  INSTRUCTION      ASCII

                00001        ORG  4318H
4318            00002 M4318  DB   'DSMBLR (C,H,L=62,N'
432A            00003        DB   ',S=32)'
4330            00004        DB   0DH
4331 211843     00005 M4331  LD   HL,M4318      !.C
4334 C30544     00006        JP   M4405          C.D
                00007 M4405  EQU  4405H
                00008 MFFFF  EQU  0FFFFH
                00009        END  M4331
```

C. 1984 A. Sopp

```
SOFTSOFT Software - Fremdprogramm:     DIS/CMD      Seite 00001

Adr. Inhalt   Zeile Symbol Befehl          ASCII

                00001        ORG  4318H
4318            00002 M4318  DB   'DSMBLR (C,H,L=62,N'
432A            00003        DB   ',S=32)'
4330            00004        DB   0DH
4331 211843     00005 M4331  LD   HL,M4318      !.C
4334 C30544     00006        JP   M4405          C.D
                00007 M4405  EQU  4405H
                00008 MFFFF  EQU  0FFFFH
                00009        END  M4331
```

Tel. 0451-791926

wichtig, ... optional

```
001400: 2061 6E64 2045 6E74 6572 203C 482C 4C3E  and Enter <H,L>
001410: 0320 2020 2020 203A 2053 7461 7274 3D78 .      : Start=x
001420: 7878 782C 2045 6E64 0100 D867 3D78 7878 xxx, End...g=xxx
001430: 782C 2054 7261 6E73 6665 723D 7878 7878 x, Transfer=xxxx
001440: 0D50 726F 6772 616D 2077 696C 6C20 6F76 .Program will ov
001450: 6572 7772 6974 6520 4469 7361 7373 656D erwrite Disassem
001460: 626C 6572 0D52 6561 6479 2070 7269 6E74 bler.Ready print
001470: 6572 2061 6E64 2065 6E74 6572 2074 6974 er and enter tit
001480: 6C65 0D41 4444 5220 434F 4E54 454E 5453 le.ADDR CONTENTS
001490: 204C 494E 4523 204C 4142 454C 2020 494E  LINE# LABEL  IN
0014A0: 5354 5255 4354 494F 4E20 2020 2020 2041 STRUCTION      A
0014B0: 5343 4949 0D50 726F 6365 7373 696E 6720 SCII.Processing
0014C0: 7061 7373 2031 0D0A 4D49 534F 5359 5320 pass 1..MISOSYS
0014D0: 4469 7361 7373 656D 626C 6572 202D 2044 Disassembler - D
0014E0: 6973 6B20 5665 7273 696F 6E20 332E 3003 isk Version 3.0.
0014F0: 2020 2020 2020 2020 2020 2020 2020 2050                P
```

```
001400: 2061 6E64 2045 6E74 6572 203C 482C 4C3E  and Enter <H,L>
001410: 0320 2020 2020 203A 2053 7461 7274 3D78 .      : Start=x
001420: 7878 782C 2045 6E64 0100 D867 3D78 7878 xxx, End...g=xxx
001430: 782C 2054 7261 6E73 6665 723D 7878 7878 x, Transfer=xxxx
001440: 0D50 726F 6772 616D 2077 696C 6C20 6F76 .Program will ov
001450: 6572 7772 6974 6520 4469 7361 7373 656D erwrite Disassem
001460: 626C 6572 0D52 6561 6479 2070 7269 6E74 bler.Ready print
001470: 6572 2061 6E64 2065 6E74 6572 2074 6974 er and enter tit
001480: 6C65 0D41 6472 2E20 496E 6861 6C74 2020 le.Adr. Inhalt
001490: 205A 6569 6C65 2053 796D 626F 6C20 4265  Zeile Symbol Be
0014A0: 6665 686C 2020 2020 2020 2020 2020 2041 fehl          A
0014B0: 5343 4949 0D50 726F 6365 7373 696E 6720 SCII.Processing
0014C0: 7061 7373 2031 0D0A 534F 4654 534F 4654 pass 1..SOFTSOFT
0014D0: 2053 6F66 7477 6172 6520 2020 4672 Software    Fr
0014E0: 656D 6470 726F 6772 616D 3A 2020 2003 emdprogramm:
0014F0: 2020 2020 2020 2020 2020 2020 2020 2020
```

Mehr über DDE

Die G-DOS-Anleitung ist notorischerweise ein Groschenheft. Wer nicht total computerkrank ist und nach und nach sein DOS selber auseinanderpflückt, ahnt nichts von dem, was noch drinsteckt.

DDE steht in SYS15/SYS und ist eine Utility, die die Inspektion einer Diskette erlaubt, ähnlich SUPERZAP von NEWDOS-80. Dabei können lt. Anleitung nur Dateien aufgerufen werden, keine Sektoren "an sich", was jedoch SUPERZAP erlaubt. In der Anzeige eines Sektors steht links auch nur die relative Sektornummer des gerade geladenen Files, nicht die absolute Sektornummer der Diskette. Sagt die Anleitung.

Da in SYS15/SYS noch reichlich Platz ist, wollte ich DDE ein wenig aufmotzen (Druckerausgabe, Anzeige der absoluten Sektornummer usw.). Der erste Schritt dazu ist immer eine Disassembly, um zu sehen, wo was geschieht. Beim Lesen des Quelltextes staunte ich nicht schlecht:

DDE residiert von 5200-54FF. Zusätzlich reserviert es 256 Bytes ab 5500 für einen systeminternen I/O-Puffer. Weshalb nicht einfach der DOS-Puffer an 4200 hergenommen wird, bleibt dunkel. Das bedeutet, daß Programme im unteren Anwenderbereich gnadenlos zugeschaufelt werden. Unser G-DOS-Manual (welch Wort für das!) weiß nichts davon.

Es gibt zusätzliche Funktionen, die dort ebenfalls nicht erläutert sind. Bei der Frage nach dem Dateinamen wird mit # die zuletzt angezeigte Datei "vergessen", aber ihr relativer Sektor nicht. Jetzt steht dieser relative Sektor, aber von der ganzen Diskette, auf dem Display. Wurde nach # eine Laufwerksnummer angegeben, wird dieses Laufwerk angewählt. Nur mit NEW LINE wird Lw. 1 geschaltet, weil 0D (NEW LINE) für die Nummer gehalten und mit 03 UND-verknüpft wird.

Beispiel: Wurde zuletzt der relative Sektor 04 des Files PROG/BAS untersucht, so steht da nach #2 der relative Diskettensektor 04 von Laufwerk 2 auf dem Bildschirm. Das ist Sektor 04 von GDOS/SYS.

In diesem Modus können alle bekannten Funktionen abgerufen werden. Mit Shift ;/+ kommt man normalerweise ans Ende einer Datei. In diesem Modus wird ebenfalls ein sehr hoher Sektor angezeigt, nämlich immer 018F, der aber natürlich nichts mehr mit GDOS/SYS zu tun hat. Das Ende der Diskette ist es auch nicht, denn mit 80/DS/DD habe ich 2880 (0B40) Sektoren. Diese Sektornummer ist an der Adresse 54EA in zwei Bytes niedergelegt, das sind die relativen Bytes 02 und 03 des rel. Sektors 03 von SYS15/SYS. Klar, daß ich sie sofort in 40 und 0B umzappte. Bisher läuft damit alles normal. Nichts spricht dagegen, auch bei anderen Spurenzahlen und Dichten diese Möglichkeit auszunutzen. Man denke aber bitte an die Reihenfolge LSB-MSB.

Eine weitere Funktion ist der Anleitung nicht bekannt: Mit * anstelle eines Dateinamens wird der zuletzt bearbeitete Sektor angezeigt. Dabei darf gerne zwischen zwei DDE-Aufrufen allerhand andere EDV gelaufen sein. Bedingung ist allerdings, daß der untere Bereich des Anwender-RAMs nicht verändert wurde. DDE legt dort nämlich die Sektoradressen und noch einiges auf Eis. Übrigens macht das BASIC genauso; der Befehl BASIC * ruft BASIC/CMD aus dem DOS auf und beläßt alle alten Zeiger und den Programmtext.

Eine dritte Besonderheit ist nicht gerade aufregend und bietet keine nennenswerte Bereicherung der DDE-Routine. Gleichwohl gehört das in die Anleitung: Beim S-Kommando, mit dem man einen bestimmten Sektor des Files aufrufen kann, wird jede Eingabe < ASCII 31h ("1") als Anwahl des Sektors 0 verstanden. Zusätzlich wird NEW LINE überflüssig, wenn die Eingabe < ASCII 21h ist. Um also in den ersten Sektor einer Datei zu

kommen, kann man statt Shift -/= auch z. B. S und Blank eingeben.

Ein neues Manual ist von der Fa. TCS angekündigt. Es sei vollständiger, heißt es. Aber der Käufer des alten Heftchens bleibt angeschmiert, denn der Kaufvertrag des G-DOS scheint bisher beinhaltet zu haben, daß der Käufer auf eine vernünftige Anleitung verzichtet. Auf Deutsch: Jeder kann das neue Manual bekommen - gegen Cash.

Bekanntlich verbraucht jede Datei auf einer Diskette mindestens eine Einheit, also fünf Sektoren. Das bedeutet, daß alle Sektoren, die beim Schreiben eines Files zum Vielfachen von fünf Sektoren noch fehlen, für alle Zeiten verloren sind (jedenfalls ohne drastische Änderung des Systems). So ist es unverständlich, daß ausgerechnet eine SYS-Datei, nämlich SYS15/SYS, nur vier Sektoren belegt.

Man könnte sich damit abfinden, denn sogar bei 40/SS/SD hat eine Platte noch immer 720 Sektoren. Systemdateien haben jedoch einen unschätzbaren Wert, denn sie können mit dem Befehl RST 28h in eigenen Programmen ohne Tastatureingabe aufgerufen werden, ohne sich der diffizilen Handhabung der Disk-I/O bedienen zu müssen. So ist es um den fünften Sektor in SYS15/SYS besonders schade.

Aber dagegen ist ein Kräutlein gewachsen. Leider ist dabei diesmal die Zapperei nicht nur auf die Zieldatei SYS15/SYS beschränkt. Gleichzeitig müssen zwei Bytes in INHALT/SYS geändert werden. Es handelt sich um die Bytes 42 und 54 (beide hex) im relativen Sektor 03 von INHALT/SYS. Das Letztere gibt schlicht die Anzahl der vom File SYS15/SYS belegten Sektoren wieder. So wird aus 04 eben 05. Das Erstere ist das EOF-Byte. Es besagt, im wievielten Byte des letzten Sektors der Datei das File zuende ist. Es zeigt also auf das erste nicht mehr zur Datei gehörige Byte. Da in meiner Modifikation der fünfte Sektor bis zum letzten Bit ausgenutzt wird, muß das EOF-Byte 00 lauten, das entspricht 256.

SYS15/SYS enthält nur die Routine zur Bearbeitung des DOS-Befehls DDE. Es liegt daher nahe, die Änderungen nicht ausgerechnet mit DDE durchzuführen. SUPERZAP ist ohnehin besser (wenngleich wesentlich unbequemer). Mit DDE geht es dann, wenn man die Zaps im Laufwerk 1 oder höher einbringt. Es empfiehlt sich, zunächst den Dateieintrag im Inhaltsverzeichnis zu ändern (oberer Sektordump). Dabei bleibt DDE voll funktionsfähig und kann notfalls sich selbst bearbeiten (nicht übel, wenn man nur ein Laufwerk und kein SUPERZAP hat).

Die Änderungen im Sektor 03 von SYS15/SYS sind kein Problem; dergleichen wurde schon in diversen Infos vorgestellt. Wer schon gelegentlich zapte, wird sich wundern, daß es selbst mit DDE auch im (überhaupt nicht belegten) Sektor 04 keine Kunst ist, obwohl das EOF im Sektor 03 bereits definiert ist. Das liegt eben daran, daß das soeben bereits verwanzte Inhaltsverzeichnis DDE glauben macht, es habe 5 Sektoren. Man kommt deshalb mit dem gewohnten Druck auf die ";"-Taste in den nachfolgenden Sektor. Der Rest ist Handwerk.

Im ersten Teil dieses Beitrags steht zu lesen, daß SYS15/SYS den Platz von 5200-55FF beansprucht (inkl. Sektorpuffer). Um nicht ohne Not bei jedem Laden des Files noch mehr Speicherplatz zu besetzen, sollte man die durch den zusätzlichen Sektor gewonnenen Records als das definieren, was in BASIC REM heißt. Der Record-Header 05 (im mittleren und unteren Sektordump unterstrichen) sorgt dafür. So wird weiterer Platz erst beansprucht, wenn später entsprechende Zaps hinzukommen. Da noch der DOS-Eingabepuffer mit seinen 80 Zeichen frei ist, sollte man eigene Routinen in SYS15/SYS zunächst dorthin legen. Der Sektorpuffer ab 4200 wird von DDE ebenfalls nicht gebraucht, so daß auch dort Platz ist. Und schließlich ist der ganze Overlay-Bereich 4D00-51FF frei, wenn die neue Routine keine weiteren DOS-Moduln nachladen soll.

Auf jeden Fall ist jetzt in SYS15/SYS Platz, und um den wäre es schade gewesen. Versteht dies bitte nicht als Preisausschreiben, aber wer hat eine Idee, was man in diesen Raum legen könnte? Ihr müßt hierzu nicht gleich die fertigen Maschinenprogramme abliefern, so daß auch die BASIC-Spezialisten unter Euch über Bedarfslücken im DOS nachdenken können.

Arnulf Sopp

INHALT/SYS, rel. Skt. 03

```
000300: 5D20 0000 0049 4E4B 414C 5420 2053 5953 Ü ...INHALT  SYS
000310: A71D F9E5 1E00 3005 FFFF FFFF FFFF FFFF .....0.........
000320: 5F20 0000 0053 5953 3720 2020 2053 5953 _ ...SYS7    SYS
000330: 5678 1234 0500 2E80 FFFF FFFF FFFF FFFF Vx.4.....
000340: 5F20 0000 0053 5953 3135 2020 2053 5953 _ ...SYS15   SYS
000350: 5678 1234 0500 2E40 FFFF FFFF FFFF FFFF Vx.4...§.....
000360: 5F20 0000 0053 5953 3233 2020 2053 5953 _ ...SYS23   SYS
000370: 5678 1234 0500 2D80 FFFF FFFF FFFF FFFF Vx.4..-.....
000380: 1020 00F4 0047 4553 4348 4252 494B 5046 . ...GESCHBRIKPF
000390: 9642 9642 0500 0160 FFFF FFFF FFFF FFFF .B.B...`.....
0003A0: 1020 0000 0052 4F55 5445 2020 2041 534D . ...ROUTE   ASM
0003B0: 9642 9642 1100 0183 FFFF FFFF FFFF FFFF .B.B.....
0003C0: 1020 00FC 0041 5254 494B 454C 3220 2020 . ...ARTIKEL2
0003D0: 9642 9642 1100 3883 FFFF FFFF FFFF FFFF .B.B..8.....
0003E0: 0020 0017 004C 5052 494E 5420 2043 4D44 . ...LPRINT  CMD
0003F0: 9642 9642 0100 3F20 FFFF FFFF FFFF FFFF .B.B..? .....
```

SYS15/SYS, rel. Skt. 03

```
000300: 0000 400B FFFF FFFF FFFF FFFF FFFF FFFF ..§.....
000310: FFFF FFFF FFFF 0500 0000 0000 0000 0000 .....
000320: 496E 2064 6965 7365 6D20 4265 2D20 2020 In diesem Be-
000330: 7265 6963 6820 6465 7320 4669 6C65 7320 reich des Files
000340: 5359 5331 352F 5359 5320 7374 6568 7420 SYS15/SYS steht
000350: 6B65 696E 207A 7520 6C61 6465 6E64 6572 kein zu ladender
000360: 4D61 7363 6869 6E65 6E63 6F64 652E 2020 Maschinencode.
000370: 4B65 6E6E 746C 6963 6820 6475 7263 6820 Kenntlich durch
000380: 6465 6E20 5265 636F 7264 2D48 6561 2D20 den Record-Hea-
000390: 6465 7220 3035 2069 6D20 4279 7465 2020 der 05 im Byte
0003A0: 3136 682C 2068 616E 6465 6C74 2065 7320 16h, handelt es
0003B0: 7369 6368 206C 6564 6967 6C69 6368 2020 sich lediglich
0003C0: 756D 2065 696E 656E 204C 7D63 6B65 6E2D um einen Lücken-
0003D0: 667D 6C6C 6572 2C20 756D 2066 7D6E 6620 füller, um fünf
0003E0: 5365 6B74 6F72 656E 2069 6D20 4869 6E2D Sektoren im Hin-
0003F0: 626C 6963 6B20 6175 6620 7370 7B2D 2020 blick auf spä-
```

SYS15/SYS, rel. Skt. 04 (neu)

```
000400: 2846 6F72 7473 2E20 6175 7320 6465 6D20 (Forts. aus dem
000410: 766F 7269 6765 6E20 05E2 5365 6B74 2E29 vorigen ..Sekt.)
000420: 2020 2020 2020 2020 2020 2020 2020 2020
000430: 7465 7265 2056 6572 7765 6E64 756E 6720 tere Verwendung
000440: 7A75 2062 6573 6574 7A65 6E2E 2020 2020 zu besetzen.
000450: 4869 6572 2068 6162 656E 206E 6F63 6820 Hier haben noch
000460: 7265 6368 7420 756D 6661 6E67 7265 692D recht umfangrei-
000470: 6368 6520 5072 6F67 7261 6D6D 6520 2020 che Programme
000480: 506C 6174 7A2E 2020 2020 2020 2020 2020 Platz.
000490: 2020 2020 2020 2020 2020 2020 2020 2020
0004A0: 2020 2020 2020 2020 2020 2020 2020 2020
0004B0: 2020 2020 2020 2020 2020 2020 2020 2020
0004C0: 2020 2020 2020 2020 2020 2020 2020 2020
0004D0: 2020 2020 2020 2020 2020 2020 2020 2020
0004E0: 2020 2020 2020 2020 2020 2020 2020 2020
0004F0: 2020 2020 2020 2020 2020 2020 0202 0052                ...R
```

LPRINT CHR$(irgendwas)

Der Druckertreiber im Microsoft-ROM ist auf der nächsten Seite aufgelistet und kommentiert. Wie man sieht, bezieht sich der größte Aufwand darauf, im Drucker-DCB den Zeilenzähler auf dem laufenden zu halten und bei senkrechter Tabulation lauter Zeilenvorschübe auszugeben. Das sind Features, die modernere Drucker beherrschen. Doppelt genäht hält besser, also sei's drum. Aber da ist eine andere Eigenschaft, die sich durchaus störend auswirkt. Welchen Nutzen sie hat, ist mir nicht klar:

Bekanntlich kann ein NUL-Code (ASCII 0) nicht mit LPRINT ausgedruckt werden. Man muß sich mit OUT 253,0 behelfen (beim TRS-80: POKE 14312,0). Bei gewöhnlichen Texten, Listings usw. spielt das keine Rolle, denn NUL kommt da nicht vor. Will man an seinen intelligenten Drucker jedoch die Null ausgeben, weil sie etwa Bestandteil eines Steuercodes ist, ist es immer wieder ärgerlich, den zusammenhängenden LPRINT-Befehl durch ein OUT-Statement unterbrechen zu müssen.

Eine entsprechende Änderung des Druckertreibers ist sehr einfach. Die Treiberadresse steht im Drucker-DCB an der Stelle 4026/4027h (16421/16422d). Bei einem Druckerbefehl wird dort die Adresse ausgelesen und die Routine angesprungen, die an dieser Adresse steht. Das ist normalerweise 05D8h im ROM. Wir können dort aber eine beliebige andere Adresse einschreiben. An dieser neuen Adresse (im abgedruckten Beispiel 7000h) wird zunächst der Akku wieder mit dem Zeichen geladen, das im ROM zuvor in das Register C übernommen wurde. Der nächste Schritt besteht lediglich darin, beliebige Teile des Originaltreibers einfach zu überspringen.

Wenn wir nur den Ausdruck von NUL ermöglichen wollen, springen wir an die Stelle NOZERO (Listing des Originaltreibers) zurück. Der Zeilenzähler bleibt dann in Betrieb. Wenn die Prüfung des Zeichens auf vertical tab nicht erwünscht ist (weil es nur Zeit kostet, denn der Drucker kann das auch), kann man nach NOVT springen. Soll auch nicht auf form feed getestet werden (dto.), geht es bei NOFF weiter. Diese Version zeigt das Listing unten.

Beim Austesten hat sich herausgestellt, daß LPRINT und LLIST durch diese Manipulation in keiner Weise beeinträchtigt werden. Die JKL-Option (DOS) bleibt ebenfalls unverändert intakt. Wozu also überhaupt dieser Aufwand im ROM? Ältere Drucker mögen einen Zeilenzähler im Speicher des Computers nötig gehabt haben. Es ist auch durchaus denkbar, daß bestimmte Textverarbeitungsprogramme ihn benutzen. Das NUL-Verbot mag auch einmal sinnvoll gewesen sein. Wer aber im Normalbetrieb einfach nur LLISTen und LPRINTen möchte, ist mit der vorgestellten Treiber-Umleitung gut bedient:

```
7000            00100           ORG     7000H           ;oder wo auch immer
7000 210770     00110 START     LD      HL,DRIVER       ;neuer Anfang d. Treibers
7003 222640     00120           LD      (4026H),HL      ;in den DCB laden
7006 C9         00130           RET                     ;-> DOS oder BASIC
                00140
7007 79         00150 DRIVER    LD      A,C             ;Zeichen in Akku laden
7008 C3B405     00160           JP      05B4H           ;ausdrucken "as is"
                00170
7000            00180           END     START           ;Einsprung dort
```

Arnulf Sopp, Tel. 0451-791926

```
                00100 ;***   Der Druckertreiber im ROM   ***
                00110
                00120 ;CALL 003BH druckt immer:
003B            00130         ORG     003BH
003B 112540     00140         LD      DE,4025H        ;Adr. des Printer-DCB
003E 18DB       00150         JR      M001B
                00160
                00170 ;CALL 001BH druckt, wenn DE mit DCB geladen:
001B            00180         ORG     001BH
001B C5         00190 M001B:  PUSH    BC              ;retten, wird verändert
001C 0602       00200         LD      B,02H           ;Bit 1 des Device-Typs
001E 1826       00210         JR      M0046
                00220
                00230 ;CALL 0046H druckt, wenn DE und B bereits geladen:
0046            00240         ORG     0046H
0046 C3C203     00250 M0046:  JP      M03C2           ;nur Hühnerleiterfunktion
                00260
                00270 ;dasselbe, denn 0049H enthält nur den Sprungbefehl:
03C2            00280         ORG     03C2H
03C2 E5         00290 M03C2:  PUSH    HL              ;Register retten
03C3 DDE5       00300         PUSH    IX
03C5 D5         00310         PUSH    DE              ;DCB-Adresse
03C6 DDE1       00320         POP     IX              ;nach IX laden
03C8 D5         00330         PUSH    DE              ;Adr. retten (wozu?)
03C9 21DD03     00340         LD      HL,M03DD        ;RET-Adresse
03CC E5         00350         PUSH    HL              ;auf den Stack
03CD 4F         00360         LD      C,A             ;zu druckendes Zeichen
03CE 1A         00370         LD      A,(DE)          ;Devic-Typ
03CF A0         00380         AND     B               ;Bit 1 maskieren
03D0 B8         00390         CP      B               ;Bit vorhanden?
03D1 C23340     00400         JP      NZ,4033H        ;DOS-Exit, falls nein
03D4 FE02       00410         CP      02H             ;Flags setzen (wozu?)
03D6 DD6E01     00420         LD      L,(IX+01H)      ;HL mit der Treiber-
03D9 DD6602     00430         LD      H,(IX+02H)      ;adresse laden
03DC E9         00440         JP      (HL)            ;den Treiber anspringen
03DD D1         00450 M03DD:  POP     DE              ;nach RET: Reg. restaur.
03DE DDE1       00460         POP     IX
03E0 E1         00470         POP     HL
03E1 C1         00480         POP     BC
03E2 C9         00490         RET                     ;Druckvorgang beendet
                00500
                00510 ;Hier wird gedruckt und der Zeilenzähler verwaltet:
058D            00520         ORG     058DH
058D 79         00530         LD      A,C             ;zu druckendes Zeichen
058E B7         00540         OR      A               ;= NUL (0) ?
058F 2840       00550         JR      Z,M05D1         ;nichts tun, falls ja
0591 FE0B       00560 NOZERO  CP      0BH             ;vertical tab?
0593 280A       00570         JR      Z,M059F         ;falls ja
0595 FE0C       00580 NOVT    CP      0CH             ;form feed?
0597 201B       00590         JR      NZ,NOFF         ;falls nein
0599 AF         00600         XOR     A               ;form feed: A <- 0
059A DDB603     00610         OR      (IX+03H)        ;A <- Zeilen/Seite
059D 2815       00620         JR      Z,NOFF          ;falls 0 Z/S
059F DD7E03     00630 M059F:  LD      A,(IX+03H)      ;Zeilenzahl wieder laden
05A2 DD9604     00640         SUB     (IX+04H)        ;abzgl. bisher gedr. Z.
05A5 47         00650         LD      B,A             ;B <- Zähler f. restl. Z.
05A6 CDD105     00660 M05A6:  CALL    M05D1           ;Drucker bereit?
05A9 20FB       00670         JR      NZ,M05A6        ;abwarten, bis bereit
05AB 3E0A       00680         LD      A,0AH           ;line feed
05AD 00         00690         NOP                     ;TRS-80: LD (37E8H),A
05AE D3FD       00700         OUT     (0FDH),A        ;line feed drucken
05B0 10F4       00710         DJNZ    M05A6           ;bis Seite voll
05B2 1818       00720         JR      M05CC           ;dort weiter
05B4 F5         00730 NOFF    PUSH    AF              ;retten
05B5 CDD105     00740 M05B5:  CALL    M05D1           ;Drucker bereit?
05B8 20FB       00750         JR      NZ,M05B5        ;falls nein
05BA F1         00760         POP     AF              ;zu druckendes Zeichen
05BB 00         00770         NOP                     ;TRS-80: s. o.
05BC D3FD       00780         OUT     (0FDH),A        ;Zeichen drucken
05BE FE0D       00790         CP      0DH             ;war es carriage return?
05C0 C0         00800         RET     NZ              ;zurück, falls nein
05C1 DD3404     00810         INC     (IX+04H)        ;ja, Zeilenzähler erhöhen
05C4 DD7E04     00820         LD      A,(IX+04H)      ;Wieviele Zeilen bisher?
05C7 DDBE03     00830         CP      (IX+03H)        ;Seite schon voll?
05CA 79         00840         LD      A,C             ;Zeichen zurückholen
05CB C0         00850         RET     NZ              ;falls S. noch nicht voll
05CC DD360400   00860 M05CC:  LD      (IX+04H),00H    ;voll: Zähl. auf 0 setzen
05D0 C9         00870         RET                     ;und zurück
                00880
                00890 ;Druckerbereitschaft feststellen:
05D1 00         00900 M05D1:  NOP                     ;TRS-80: LD A,(37E8H)
05D2 DBFD       00910         IN      A,(0FDH)        ;Druckerstatus laden
05D4 E6F0       00920         AND     0F0H            ;linkes Nibble maskieren
05D6 FE30       00930         CP      30H             ;bereit, wenn Bits 4&5=1
05D8 C9         00940         RET                     ;mit Statusflags zurück
0000            00950         END
00000 mal gepennt
32487 Zeichen verfügbar
```

```
0010          TITL    '*** U m l a u t ***
0020          SBTL    'Umlautreiber für MULTIDOS, (c) 1984 by Ralf Folkerts
0030          COMM    '****************** Umlaut  **** (c) 1984 by: *
0040          COMM    '* Ralf Folkerts*****************
0050          ;Umlauttreiber fuer MULTIDOS und NEWDOS
0060          ;Copyright (c) 06/84 by
0070          ;Ralf Folkerts
0080          ;Nutzhorner Strasse 9
0090          ;2875  Bookholzberg
0100          ;********
0110          ;DIE TITL UND SBTL BEI ASSMEBLERN <>ZEUS
0120          ;BITTE WEGLASSEN. EBENSO COMM.
0130          ;ALLE LABEL UND HEXZAHLEN GROSS.
0140          ;FALLS DER TREIBER NICHT UNTER DOS LAUFEN
0150          ;SOLL, SIND DIE ZEILEN
0160          ;330 UND 240 WEGZULASSEN.
0170          ;FALLS UNTER NEWDOS LAUFT, KOENNEN DIE
0180          ;ZEILEN 310 UND 320 WEGFALLEN.
0190          ;**********
0200          ;
0210          ;NACH DER AKTIVIERUNG VERSCHIEBT DER TREIBER
0220          ;SICH SELBSTSTAENDIG AN TOPMEM. DANN GIBT ER
0230          ;EINE FERTIG - MELDUNG AUS. NUN KANN DIE UM-
0240          ;LAUTBILDUNG DURCH DRUECKEN DER <CLEAR> TASTE
0250          ;EINGELEITET WERDEN. WENN MAN NACH DER <CLEAR>
0260          ;TASTE EINE ANDERE TASTE DRUECKT, ERGEBEN SICH
0270          ;FOLGENDE FUNKTIONEN:
0280          ;<A> : AE;  <O> : OE;  <U> : UE;  <S> : SZ
0290          ;MIT <SHIFT> ENTSPRECHEND.
0300          ;<CLEAR> : EINGETLICHE <CLEAR> FUNKTION
0310          ;JEDE ANDERE TASTE: WIE NORMAL
0320          ;
0330          ;**********.
0340          ORG     5200h           ;Overlay
0350 start    LD      HL,end          ;Letzte Zeile
0360          LD      BC,byte-begin   ;Anzahl der Bytes
0370          LD      DE,(topmem)     ;Letzte Adresse
0380          LDDR                    ;Verschibe Prgm.
0390          EX      DE,HL           ;Hole neue Adresse
0400          LD      (topmem),HL     ;Schreibe neuen Topmem
0410          LD      (basic),HL      ;Letzte Adr. f. BASIC
0420          INC     HL              ;Counter + 1
0430          PUSH    HL              ;Sichere Adresse
0440          LD      HL,(kidcb)      ;Hole alte Treiber Adr.
0450          POP     IX              ;Hole HL in IX
0460          LD      (IX+2),H        ;Speichere LSB
0470          LD      (IX+1),L        ;Speichere MSB
0480          LD      (kidcb),IX      ;Neuen DCB Start
0490          LD      (IX+10),H
0500          LD      (IX+9),L
0510          LD      A,20h           ;Code fuer LC ein
0520          LD      (kidcb+2),A     ;Schalte LC ein
0530          LD      HL,bereit       ;Hole Text, bereit
0540          CALL    vod             ;Ausgabe auf Schirm
0550          JP      dos             ;Zurueck zum DOS
0560 ;
0570 ;        Bis hier Relocater und Init. Ab jetzt
0580 ;        folgt eigentlicher Treiber
0590 ;
0600 begin    CALL    0               ;Hier Dummy f. DCB Adr.
0610          OR      A               ;FLAG's setzen
0620          RET     Z               ;Wenn '0', zurueck
```

```
        CP      1fh             ;Ist es <CLEAR> ?
        RET     NZ              ;Wenn nein. zurueck
recall  CALL    0               ;Treiber call
        OR      A               ;FLAG's
        JR      Z,recall        ;Bis Taste
        CP      'a'             ;Ist es a
        LD      C,7bh           ;Code f. ae
        JR      Z,ende
        CP      'o'             ;Ist es o
        LD      C,7ch           ;Code f. oe
        JR      Z,ende
        CP      'u'             ;Ist es u
        LD      C,7dh           ;Code f. ue
        JR      Z,ende
        CP      's'             ;Ist es s
        LD      C,7eh           ;Code f. sz
        JR      Z,ende
        CP      'A'             ;Ist es 'A' ?
        LD      C,5bh           ;Code f. Ae
        JR      Z,ende
        CP      'O'             ;Ist es 'O'
        LD      C,5ch           ;Code f. Oe
        JR      Z,ende
        CP      'U'             ;Ist es 'U'
        LD      C,5dh           ;Code f. Ue
        JR      Z,ende
        CP      1fh             ;Ist es <CLEAR>
        LD      C,1fh           ;Code f. <CLEAR>
        JR      Z,ende
        LD      C,A             ;Code der Taste
ende    LD      A,C             ;Code in AKKU
end     RET                     ;Zurueck
byte    DB      0               ;Dummy f. relocat.
;
;       Ab hier Definitionen
;
topmem  EQU     4049h
basic   EQU     40b1h
kbdcb   EQU     4016h
vad     EQU     4467h
das     EQU     402dh
bereit  DM      0ah,'Der Umlauttreiber, (c) by Ralf Folkerts ist bereit'
,0ah,0dh
        END     start
```

ΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨΨ

Nochmal zum Thema 'Akustikkoppler':
Heinrich Thönnißen braucht bis

<u>spätestens 09.01.1985</u>

die Bestellungen. Es bleibt dann noch genügend Zeit, die genaue Anzahl im Januar-Info zu veröffentlichen. Jeder kann sich dann den genauen Preis ausrechnen.

<u>Stichtag: ----->> 09.01.1985 !!!!</u>

Und es geht doch: LPRINT CHR$(10)

Nachdem O. Stark und ich mit blindem Gottvertrauen behauptet hatten, mit LPRINT sei 0Ah auf den Drucker zu kriegen (womit wir uns bei dem Selbstdenkern unter Euch gründlich blamiert haben dürften), und nachdem ich in einem weiteren Beitrag (der frühestens im November-Info steht) Euch zur Suche nach dem casus knaxus animierte, habe ich ihn nun doch selber gefunden:

Der Druckertreiber ist wirklich vollkommen unschuldig. Er hat zwar seine Tücken, weil er bei ASCII 00 nur den Druckerstatus abfragt und aus einer vertikalen Tabulation und einem Blattvorschub lauter einzelne Zeilenvorschübe macht, aber dagegen ist das Kraut aus meinem Beitrag "LPRINT CHR$(irgendwas)" gewachsen. ASCII 0A kommt dort aber gar nicht erst an!

Eine BASIC-Zeile wie

```
10 LPRINT CHR$(10)
```

ist für den Interpreter eine relativ harte Nuß. Ohne mir jetzt die Mühe machen zu wollen, das zu überprüfen, schätze ich, daß zur Bearbeitung mindestens an die 30 Unterprogramme durchlaufen werden müssen. Dazu gehört auch das UP an 039C, das an der Ausgabe eines Zeichens auf den Drucker beteiligt ist. In dessen Verlauf findet sich auch diese Befehlsfolge:

```
03A6  FE0A    CP    0AH          ;auf ASCII 0A prüfen
03A8  2003    JR    NZ,03ADH     ;dort weiter, falls anderer Wert
03AA  3E0D    LD    A,0DH        ;0A durch 0D ersetzen!
```

Bei der Suche nach dieser Gemeinheit erwies sich mal wieder der EG 64 MBA als lohnende Investition. Bei Sprungbefehlen das Ziel zu finden, ist simpel. Das Gegenteil erweist sich aber im ROM, wo man normalerweise nicht mit einem Monitor einen Breakpoint setzen oder auf andere Weise ein Kuckucksei legen kann, als Fleißarbeit. Wo der Interpreter aber soft vorliegt, läßt sich locker ein F7 (RST 30h, um DEBUG aufzurufen) einschreiben. Das DEBUG-Display zeigt nun u. a. den Stackpointer SP an. Er verweist auf die RET-Adresse und damit auf den CALLer. Dieser wird nun seinerseits mit einem F7 verwanzt usw., bis der Übeltäter gefunden ist. Das dauerte in diesem Falle keine drei Minuten.

Ohne Disco geht es nicht minder einfach. Statt RST 30h wird dann eben JP MONADR eingeschrieben, ein Sprungbefehl zu einem zuvor geladenen Monitor. Diese Adresse muß allerdings zuvor ermittelt werden: Man setzt an einer beliebigen Stelle im RAM einen Breakpoint. Mit Reset verläßt man nun den Monitor und schaut sich (mit PEEK oder wie auch immer) die drei Bytes ab Breakpoint an. Dort steht bei allen mir bekannten Monitoren ein CALL an die Adresse, wo Breakpoints bearbeitet werden. Das erste Byte dürfte daher CD lauten. Die beiden folgenden Bytes bilden die Adresse, die bei unserem künstlichen Breakpoint mit JP angesprungen wird. In der Anzeige erscheint dann SP mit seinem verräterischen Inhalt.

Natürlich soll dieser Fund nicht folgenlos bleiben. Wer seinen Microsoft-Interpreter durch drei EPROMs ersetzt, kann an den Stellen 03AA und 03AB zwei NOPs einschreiben und ist damit diese Sorge los. Bequemer geht es mit dem MBA (bitte nicht hauen, er wird in diesem Artikel nicht wieder erwähnt), denn der Interpreter liegt auf Wunsch ohnehin soft vor.

Arnulf Sopp

FDE - Sektoren

Der Rest des Direktorys ist belegt mit den FDE-Sektoren. Jeder Sektor enthält maximal 8 Einträge zu je 32 Bytes. Im Normalfall sind jedem Programm 32 Bytes zugeordnet.
Die verschiedenen Bits des 1. Bytes enthalten Informationen wie Ersteintrag oder Folgeeintrag, FDE frei oder belegt, usw.
Wird ein Eintrag durck "KILL" gelöscht, so wird nur das 4. Bit des 1. Bytes auf 0 gesetzt. So könnte also ein versehentlich gelöschter Eintrag mit dem COLZAP gerettet werden.
Byte 6 - 13 enthält den linksbündigen Namen, falls notwendig aufgefüllt mit Leerzeichen.
Byte 14 - 16 gibt den linksbündigen Filetyp (CMD, BAS, ..) an.
Die letzten beiden Bytes vor der "FF-Reihe" dienen zur Berechnung der relativen Sektornummer, an der der Eintrag beginnt.
Das erste dieser beiden Bytes gibt die dezimale Nummer des LUMPS an, die mal 15 gerechnet werden muß.
Die linken 4 Bits des nächsten Bytes können nur den Wert 0/1 2/3 oder 4/5 annehmen. Dieser Wert steht für den 0-ten, 5-ten oder den 10-ten Sektor des LUMPS, der nun zu dem vorigen Ergebnis zu addieren ist.

__Beispiel:__ 2. Eintrag des 1. FDE-Sektors (SINGLEST/BAS)

Die beiden Bytes sind 2C 40
2C hex ist 44 dez. 44 * 15 = 660
4 bedeutet den 10. Sektor → 660 + 10 = 670
Das Programm SINGLEST/BAS ist somit ab dem relativen Sektor Nummer 670 zu finden.

```
0000 5E083300004E435731393833204A484C
0010 607F1FB205000000FFFFFFFFFFFFFFFF
0020 102000DE0053494E474C455354424153   ← Name und Filetyp
0030 0000000003002C40FFFFFFFFFFFFFFFF
0040 00200000004550524F4D332020202020
0050 0000000000002D002F21FFFFFFFFFFFF
0060 00000000000000000000000000000000
0070 00000000000000000000000000000000
0080 00000000000000000000000000000000   } 8 Einträge zu je 32 Bytes
0090 00000000000000000000000000000000
00A0 00000000000000000000000000000000
00B0 00000000000000000000000000000000
00C0 00000000000000000000000000000000
00D0 00000000000000000000000000000000
00E0 00000000000000000000000000000000
00F0 00000000000000000000000000000000
Drive 0, DRS 0362, TRK 21, SEC 02, PROT
0000 ^.8..NCW1983 JHL`. ....
0020 . . ..SINGLESTBAS.......,$
0040 . ...EPROM3          .......-./!
0060 ................................
0080 ................................
00A0 ................................
00C0 ................................
00E0 ................................
```

Natürlich sind die von mir gemachten Angaben nicht vollständig. Für ein anfängliches Verstehen und Arbeiten dürfte das aber mehr als genug sein. In Bälde gibt es sicher auch für das Colourgenie-Floppy entsprechende Literatur, so daß ein Weiterstudium möglich ist.

Zum Schluß möchte ich noch auf eine Möglichkeit hinweisen, wie Disketten eines anderen Systemes lesbar werden, (z.B. NEWDOS 80).
Da das Colour-DOS keine direkte Beeinflussung der sogenannten PDRIVE-Daten hat, sind normalerweise Fremddisketten nicht lesbar. In den EPROMs des Betriebssystemes sind diese Daten allerdings fest eingebrannt. Nach einem Start werden sie dann an den Arbeitsbereich ab der HEX-Adresse 5A71 (für das 1. Laufwerk) geladen. Um nun eine Anpassung durchführen zu können, müssen die neuen Daten dahin gepoked werden.

Aktuelle Laufwerksinformationen des 1. LW

5A71	18	Track/Dir	neu 11
5A72	30		
5A73	53		
5A74	28	40 Spuren/Disk	
5A75	12	Sektoren/Track	
5A76	03		neu 02
5A77	00		
5A78	03		
5A79	05	Sektoren/Granule	
5A7A	03	Granules/Dir	neu 02

Mit nur drei pokes ist es nun möglich, Disketten des NEWDOS 80 zu lesen (POKE&H5A71,11 - POKE&H5A76,2 - POKE&H5A7A,2). Prinzipiell kann somit das Laufwerk auf jedes andere DISK-System angepaßt werden.
Die Veränderungen, um für dieses System Disketten schreiben zu können, habe ich leider noch nicht gefunden.
Für mich hat allerdings das Lesen von Disketten anderer Systeme bisher immer genügt. Denn zum Weiterbearbeiten sind die Programme dann ja auf die eigene Diskette zu saven.

Und nun umgekehrt! Auch das NEWDOS 80 kann Colourgenie-Disketten lesen, wenn die PDRIVE-Einstellung verändert wird. Der Arbeitsgang läuft folgendermaßen ab:

Eingabe: PDRIVE,0,1

Laufwerk mit Systemdiskette	Laufwerk mit Colourdiskette

```
PDRIVE,0,1,TD=E,TC=40,SPT=18,
           TRS=0,GPL=3,DDSL=24,
           DDGA=3
PDRIVE,0,A
```

Wenn nur ein Laufwerk vorhanden ist, muß natürlich die Systemdiskette gegen die Colourgeniediskette nach Anweisung ausgetauscht werden. Diese Angaben hat mein Freund Christian Röhrs auf seinem TRS 80 herausgefunden.
Und nun wünsche ich allen, die das ausprobieren wollen viel Erfolg beim Diskettentausch.

Dieter Gerblinger
Alpenkorpsstrasse 23
8102 Mittenwald
Tel.: 08823 - 2017

Wie ist Genietext aufgebaut?

Antwort auf die Frage von K.J.Mühlenbein warum Genietext die rechte Randbegrenzung nicht mehr mitmacht, wenn ein sog. "umgewandeltes Zeichen" (z.B.: #a) verwendet wird.

Genietext folgt einem bestimmten mathematischen Algorithmus, um den Randausgleich zu berechnen. Es zählt wieviel Wörter in eine Zeile passen und dann fügt es "Blanks" zwischen den Wörtern ein, bis die Zeile die endgültige Länge erreicht hat. Erst wenn Genietext den Randausgleich einer Zeile errechnet hat, geht er zum Drucken über.

Jetzt kommt aber der große Fehler von Genietext! Beim Berechnen des Randausgleichs, werden die Sonderzeichen ($#) mitgezählt. Beim Drucken werden diese Zeichen aber nicht ausgedruckt. Damit schiebt sich jede Zeile jeweils um ein Zeichen nach links. Der Randausgleich funktioniert nicht mehr. Um es zu verändern, müßte man mit Maschinensprache an das Programm ran. Meine Version von Genietext wandelt die Sonderzeichen beim Drucken in Blanks um. Das ist auch ein lästiger Effekt.

Die Tatsache, daß Genietext selbstbootend ist, hat bestimmt viele von uns abgeschrekt, sich an das Programm ranzumachen. Jetzt kommt aber der Clou!!! Genietext bassiert auf NewDos80. Bootet mal eine NewDos80 Diskette. Gebt dann <DIR $U> ein und tut dann die Genietext Diskette ins Laufwerk ein. Ihr seht jetzt das Directory von Genietext. Leider erscheint nicht alles auf das Directory, was in Wirklichkeit drauf ist. (Kleiner Trick von Zender!) Es handelt sich um die Programme BASIC/CMD, SYS8/CMD und SYS9/CMD.

Wer die Inhalte der anderen Sys-Files genau wissen will, dem rate ich vorher Ramsys zu laden, da Genietext alle überflüßige Files gekillt hat. Ich habe es getan, und bin zu folgende überraschende Ergebnisse gekommen:

1) BASIC/CMD ist nicht das Basic, sondern das eigentliche Genietextprogramm. Man kann es aber leider nur auf der Genietextdiskette zum Laufen bringen.

2) SYS8/CMD ist das Basic. Wozu braucht aber Genietext Basic? Das wird im nächsten Punkt erklärt.

3) SYS5/SYS ist ein Basic-Programm!!! Es wird von Genietext aufgerufen, wenn man das Kommando 'U' eingibt. Die Unterkommandos werden also vom Basic aus bedient. (Leicht zu kopieren und zu verändern!). Wie werden aber die Änderungen durchgeführt? Sie werden mit Superzap gemacht.

4) SYS9/CMD ist Superzap, und wird vom Basic aus aufgerufen.

Die Änderungen sind leicht durchzuführen. Wenn jemand weiterkommt, bitte im Info bescheidgeben.

(© Alfonso Sanz)

Die Systemoptik ausgetrickst!

System- oder programminterne (was wohl im Prinzip dasselbe ist) Bedingungen erschweren von Fall zu Fall den Versuch, auf dem Bildschirm oder der Hardcopy ein optisch ansprechendes Erscheinungsbild zu erzeugen. So ist beispielsweise ein Diskname (ebenso das Datum) nicht mit dazwischenliegenden Blanks (Leerstellen) einzugeben. Und EDTASM, der wohl weitestverbreitete Editor-Assembler, läßt nicht die Eingabe von Steuercodes für den Drucker zu. Und im File SYS28/SYS, in das die Zeichen für den FORM-Befehl (G-DOS) eingegeben werden, sinddie Codes 03 und 0Dh nicht zulässig, denn sie werden als Endzeichen interpretiert.

Häufig hilft die CTRL-Taste, mit der man Codes unter ASCII 32 (Blank) erzeugen kann. Wird sie gleichzeitig mit einer Zeichentaste gedrückt, steht im I/O-Puffer der um (zumeist) 64 verminderte Wert. Aber genau gleichzeitig schafft man es nie, so daß in der Regel zunächst ein LF (line feed = Zeilenvorschub) ausgegeben wird.

Wenn nichts mehr hilft, kann man zumeist problemlos mitten in die fertig vorliegende Datei (Programm, Daten, Text, Quelltext usw.) hineinschreiben. Es versteht sich, daß für die später beabsichtigten Steuerzeichen für den Bildschirm oder den Drucker Platz reserviert werden muß. Man kann bei der Eingabe an der betreffenden Stelle ein beliebiges, möglichst leicht wiederzufindendes Zeichen eingeben. Anschließend wird mit einem Monitorprogramm oder durch schlichtes PEEKen diese Stelle aufgesucht und der endgültige Code eingeschrieben.

Die Lösung ist für das o. g. FORM-Problem am einfachsten, deshalb dies zuerst: Für die meisten Drucker ist das höchstwertige, das Bit 7 des übertragenen Codes erst wichtig, wenn der Rest 20h (32d) überschreitet, wenn also der Gesamtwert mindestens ASCII A0h (160d) beträgt. So kann 0Dh leicht als 8Dh und eine 3 als 83h eingetippt werden, ohne daß für den Drucker ein Unterschied bestünde.

Etwas umständlicher, im Prinzip aber ebenso einfach geht es, in einem BASIC- oder EDTASM-Quelltext die Anweisungen zur Druckerformatierung unterzubringen. Im Folgenden sollen für BASIC und Assembler je ein Fallbeispiel gezeigt werden:

1. Problem: REM-Statements sollen hervorgehoben werden.

Das Befehlswort REM kann auch durch das Hochkomma <'> ersetzt werden, was hier geschieht, damit bei LLIST nur ein Zeichen übermittelt wird. Und dies soll zunächst zum Verschwinden gebracht werden. Hierzu dient der DEL-Code (delete, ASCII 7Fh = 127d) für den Drucker. Er bewirkt, daß das vorangegangene Zeichen getilgt wird und nicht zum Ausdruck kommt. Sodann soll die REM-Zeile unterstrichen erscheinen. Dies geschieht bei meinem Drucker mit der Zeichenfolge <ESC> <-> ASCII 1. Insgesamt müssen also vier Dummy-Zeichen zwischen dem <'> und dem REM-Text erscheinen.

Um die Unterstreichung wieder zu löschen, wird normalerweise <ESC> <-> NUL (ASCII 0) eingegeben. Da jedoch die Null als Zeilenende interpretiert wird, was hier fatale Folgen hätte, kann man auf die Neuinitialisierung (bei meinem Gerät <ESC> <§>) ausweichen. Auch hierfür sind nach dem REM-Text zwei Dummy-Zeichen vorzusehen.

Die Abb. 1 und 2 zeigen denselben BASIC-Text, mit Dummy-Zeichen und nach dem Ersetzen durch die beabsichtigten Codes. Ich habe es mir leichtgemacht, indem ich DEL mit D andeutete, ESC mit $ usw. So konnte ich die betreffenden Speicherstellen leicht wiederfinden. Wie, das kommt später.

Selbstredend muß man darauf achten, daß solche Manipulationen nur da erlaubt sind, wo sie keinen Syntax-Error verursachen können, also nur nach REM oder zwischen <">. Daß man mit der gleichen Methode auch die Zeilennummern in eine phantasievolle Reihenfolge verbiegen und sonst noch allerhand anstellen kann, sei hier nur erwähnt. Das Bildschirmlisting mag kraus aussehen - das darf hier nicht stören, wenn es darum geht, "für die Akten" eine ansprechende Hardcopy zu erzeugen.

2. Problem: Gutaussehender Header einer EDTASM-Source

Es ist gern geübter Brauch, den Kopf eines Assembler-Quelltextes mit <*> vom eigentlichen Programmtext optisch zu trennen. Weit mehr ist möglich, und zwar mit der gleichen Methode, die oben für BASIC-Texte beschrieben ist. Die Abb. 3 und 4 zeigen das "Vorher-Nachher". Für dieses Beispiel habe ich den Header einer Maschinenroutine gewählt, die ich vor längerer Zeit entwickelte, um Maschinenprogramme mit dem SYSTEM-Befehl auch mit sedezimaler Adresse starten zu können (natürlich nur für Level 2 interessant, denn Disk-BASIC kann das auch).

Hier ist zu beachten, daß EDTASM beim H- bzw. A/LF-Befehl das Bit 7 des übertragenen Codes mißachtet. Es ist daher leider nicht möglich, die Graphikzeichen >80h bzw. >A0h auf den Drucker zu bringen, ohne zuvor EDTASM umzukrempeln. Aber was anstandslos funktioniert, ist genug:

In unserem Beispiel werden die Semikola <;> gelöscht, die dieselbe Funktion wie REM in BASIC haben, Breitschrift wird eingeschaltet und der Programmname wird unterstrichen. Hier habe ich der Demonstration halber ein wenig zuviel des Guten getan, denn man stelle sich vor, jemand tippt eine Kommentarzeile ohne Semikolon ab!

Wie schon beim BASIC-Beispiel hat auch hier die Neuinitialisierung, um die Unterstreichung zu löschen, die Folge eines LF. Je nach Druckertyp kann mit der hier beschriebenen Methode wohl auch dieser Schönheitsfehler beseitigt werden.

Bei Manipulationen dieser Art gilt es natürlich zunächst, den Text aufzufinden, bevor man ihn verändern kann. Auf BASIC-Programmtexte weist ein Zeiger in 40A4/40A5h (16548/16549d). Man findet die Stelle mit der Befehlsfolge

```
PRINT PEEK(16548)+256*PEEK(16549)
```

Mit einer FOR-NEXT-Schleife, einem Monitorprogramm oder wie auch immer kann man nun das Programm nach den vorbereiteten Dummy-Zeichen durchsuchen und die so aufgespürten Speicherstellen neu beschreiben. Die dergestalt veränderten Programmzeilen lassen sich ohne weiteres edieren (EDIT), solange die Codes <80h (128d) sind.

Bei EDTASM wird es insofern etwas schwieriger, als derartige Utilities gerne für den eigenen Bedarf verändert werden. Bei der mir vorliegenden Disk-Version beginnt der Quelltext bei 7700h (30464d). In Level-2-Versionen läßt er sich mit einem Monitor oder mit der BASIC-Befehlsfolge

```
FOR I%=17129 TO 65535: PRINT CHR$(PEEK(I%));: NEXT
```

auffinden. Irgendwann erscheint auf dem Bildschirm etwas, das man als den Anfang seiner EDTASM-Source wiedererkennt. Dann kann man mit <BREAK> und PRINT I% feststellen, wo gerade gesucht wurde. Mit einer kleineren FOR-NEXT-Schleife wird dann die genaue Stelle gefunden. Hierbei ist zu beachten, daß die Level-2-Version von EDTASM ziemlich tief residiert. Es ist daher riskant, programmgesteuert zu suchen. BASIC könnte EDTASM zuschaufeln. Direkte Befehle ohne Zeilennummern sind unbedingt vorzuziehen. Die Zählvariablen (zumal von Integertyp V%) für FOR-NEXT allein reichen im RAM noch nicht bis zu EDTASM hinauf. Die Variable sollte aber immer denselben Namen haben, damit nicht mehrere Variable zusammen so viel RAM besetzen, daß EDTASM die weiße Fahne schwenkt.

Das für diesen Artikel verwendete Textverarbeitungsprogramm TSCRIPS hat sich bisher leider meinen Versuchen widersetzt, dergleichen auch mit ihm zu veranstalten. Z. B. wollte ich meinen Briefkopf mit Graphikelementen verschönern, leider bisher ohne Erfolg. Wer TSCRIPS intimer kennt als ich und mir helfen möchte, es zu verändern, findet meine Adresse in Abb. 3/4. Vielen Dank!

```
10 'D$-1Variablenzuordnung:$§
20 A=5: B$="Testprogramm": C1=&H42E9
```

Abb. 1

```
10 Variablenzuordnung:

20 A=5: B$="Testprogramm": C1=&H42E9
```

Abb. 2

```
00100 :DB**************************
00110
00120 :DB    $-1S  Y  S  H  E  X$§
00130
00140 :D Eingabe der Startadresse von System-Format-Pro=
00150    grammen wahlweise dezimal oder hexadezimal
00160
00170 Dezimale Eingabe wie gewohnt mit       "/ddddd",
00180 Hexeingabe stattdessen mit             ":hhhh".
00190
00200 :DB**************************
00210 :DC 1983 by A. Sopp, Wakenitzstr. 8, 2400 Lübeck 1
```

Abb. 3

```
00100 ************************************
00110
00120          S  Y  S  H  E  X

00130
00140  Eingabe der Startadresse von System-Format-Pro=
00150    grammen wahlweise dezimal oder hexadezimal
00160
00170 Dezimale Eingabe wie gewohnt mit       "/ddddd",
00180 Hexeingabe stattdessen mit             ":hhhh".
00190
00200 ************************************
00210 C 1983 by A. Sopp, Wakenitzstr. 8, 2400 Lübeck 1
```

Abb. 4

Arnulf Sopp

Convert

Das Programm 'CONVERT' dient der Umwandlung von SCRIPSIT/SUSCRIP Text Files in ASCII Files. Hierdurch kann man z.B. mit dem Textprogramm ein BASIC Programm schreiben, mit dem Programm convert in ASCII Format umwandeln und danach als BASIC Programm laden. Nach einer kleinen Modifikation (vertauschen zweier Zeilen) ist auch der umgekehrte Weg -möglich. Es kann dann ein BASIC Programm 'RE-convertIERT' und mit dem Textprogramm editiert werden. Das Programm ist in der vorliegenden Form für das DOS 'MULTIDOS' geschrieben, kann jedoch nach einer kleinen änderung auch unter NEWDOS 'gefahren' werden. Auch eine Anpassung an andere Textprogramme ist möglich.

Funktion:

In den Zeilen 120 - 440 werden die beiden Filespecs 'geholt'. Hierzu werden die Filenamen, die in Klammern (mit beliebig vielen Leerzeichen von Befehl 'CONV' getrennt, jedoch OHNE Leerzeichen in der Klammer) stehen zuerst in die Zwischenspeicher 'QFILE' und 'ZFILE' geschoben, und jeweils mit einem 0D abgeschlossen.

In den Zeilen 460 - 510 werden die File DCB's gebildet. Dies geschieht dadurch, daß die Filenamen in 'QFILE' und 'ZFILE' stehen, durch einen Aufruf der DOS Routine ab 441CH in die Speicher 'QCONV' und 'ZCONV' verschoben werden.

Dann werden in den Zeilen 540 - 680 die Files geöffnet. Hierbei ist zu beachten, daß bei NEWDOS in einer Zeile 555 'LD B,0' stehen muß, da Multidos die LRL aus der Directory entnimmt, während er bei NEWDOS angegeben werden muß.

In den Zeilen ab 700 werden dann die einzelnen Quellfile Sectoren in den Zwischenspeicher 'QDAT' gelesen, dann in den Speicher 'ZDAT' kopiert, wobei auch die Konvertierung geschieht. Der Speicher 'ZDAT' wird dann in den Zielfile Sector geschrieben.

Benutzung:

Quellfile (zu konvertierender SCRIPSIT File) = QFILE
Zielfile (ASCII File, Ergebnis der Konvertierung) = ZFILE

Aufruf mit: 'CONV (QFILE=ZFILE) <NEW LINE>
Zu beachten ist, daß in der Klammer kein Leerzeichen stehen darf, während zwischen Klammer und 'CONV' beliebig viele Blanks stehen dürfen. Wenn keine Klammer vorhanden ist, oder die Filespecs falsch in der Klammer stehen meldet sich daß System mit 'DOS ERROR = 2F UNKNOWN ERROR' (dies ist die Meldung 'BAD PARAMETER(S)' bei NEWDOS).

Um das Programm auch umgekehrt laufen zu lassen (ASCII in SCRIPSIT) sind die Zeilen 810 und 830 einfach zu vertauschen. Dies Programm kann man dann 'RECONV' nennen. Zu beachten ist natürlich, daß ein Basic Programm als ASCII File gespeichert werden muß.

Anpassung an andere Textprogramme:

Hierzu ist in der Zeile 810 das 8DH in das ASCII Zeichen zu verändern, welches das verwendete Textprogramm als 'CR' verwendet (bei TRSTEXT ist dies z.B....).

Da sich das Programm in den Reservierten DOS Bereich ab 5200H lädt, ist ein setzen von HIMEM oder TOPMEM nich nötig.

```
                        00010 ;         CONV/OBJ
                        00020 ;         KONVERTIERT SCRISIT TEST FILES IN ASCII FORMAT
                        00030 ;         03/84 BY
                        00040 ;         RALF FOLKERTS
                        00050 ;         NUTZHORNER STRASSE 9
                        00060 ;         2875  BOOKHOLZBERG
                        00070 ;         *****
                        00080 ;         START
                        00090 ;         *****
5200                    00100           ORG     5200H
                        00110 ;         ***     5200 IST START VON DOS OVERLAY AREA
5200 211843             00120 START     LD      HL,4318H
                        00130 ;         ***     4318H IST START DES 'BEFEHLSSPEICHERS'
5203 7E                 00140 SEARCH    LD      A,(HL)          ;HL AUF BEFEHLSPEICHER
5204 FE0D               00150           CP      0DH             ;IST ES CR ?
5206 CA3255             00160           JP      Z,PAERR         ;WENN JA, PARAMETER ERROR
5209 FE28               00170           CP      '('             ;IST ES '(' ?
520B 23                 00180           INC     HL              ;ERHOEHE ZAEHLER
520C C20352             00190           JP      NZ,SEARCH       ;WIEDERH. BIS '('
520F 11C252             00200           LD      DE,QFILE        ;ADDR. FUER QUELLE IN DE
5212 7E                 00210 GETQ      LD      A,(HL)          ;AKKU AUS HL
5213 FE0D               00220           CP      0DH             ;IST ES CR ?
5215 CA3255             00230           JP      Z,PAERR         ;WENN CR, DANN PA. ERR.
5218 FE3D               00240           CP      '='             ;IST ES '=' ?
521A CA2352             00250           JP      Z,GETZ          ;WENN JA, DANN ZIELFILE
521D 12                 00260           LD      (DE),A          ;ABSPEICHERN FUER QUELLE
521E 13                 00270           INC     DE              ;POINTER <= POINTER + 1
521F 23                 00280           INC     HL              ;---'--- <= ---'--- ' '
5220 C31252             00290           JP      GETQ            ;WIEDERHOLE BIS GES.QFILE
5223 3E0D               00300 GETZ      LD      A,0DH           ;CR IN AKKU
5225 12                 00310           LD      (DE),A          ;CR AN FILESPEC
                        00320 ;         ***     JETZT IST QUELLFILE FERTIG
5226 11DA52             00330           LD      DE,ZFILE        ;ADDR. FUER ZIEL IN DE
5229 23                 00340 GETZ2     INC     HL              ;POINTER <= POINTER + 1
522A 7E                 00350           LD      A,(HL)          ;HOLE NAECHSTES ZEICHEN
522B FE29               00360           CP      ')'             ;IST ES ')'
522D CA3A52             00370           JP      Z,REFILE        ;WENN JA, WEITER
5230 FE0D               00380           CP      0DH             ;IST ES CR ?
5232 CA3A52             00390           JP      Z,REFILE        ;WENN JA, WEITER
5235 12                 00400           LD      (DE),A          ;ABSPEICHERN FUER ZIEL
5236 13                 00410           INC     DE              ;POINTER <= POINTER + 1
5237 C32952             00420           JP      GETZ2           ;WIEDERHOLE BIS ZIEL
523A 3E0D               00430 REFILE    LD      A,0DH           ;CR IN AKKU
523C 12                 00440           LD      (DE),A          ;CR AN FILESPEC.
                        00450 ;         ***     JETZT IST ZIELFILE FERTIG
523D 21C252             00460           LD      HL,QFILE        ;QUELL ADDR. IN HL
5240 11F252             00470           LD      DE,QCONV        ;QUELL CONV. ADDR. IN DE
5243 CD1C44             00480           CALL    441CH           ;CONVERTIERE
5246 21DA52             00490           LD      HL,ZFILE        ;ZIEL ADDR. IN HL
5249 111253             00500           LD      DE,ZCONV        ;ZIEL CONV. ADDR. IN DE
524C CD1C44             00510           CALL    441CH           ;CONVERTIERE
                        00520 ;         ***     DIE KONVERTIERTEN ADDRESSEN STEHEN JETZT
                        00530 ;         ***     IN QCONV UND ZCONV.
524F 11F252             00540           LD      DE,QCONV        ;QUELL FILE IN DE
5252 213253             00550           LD      HL,QDAT         ;DATENSPEICHER
```

```
5255 CD2444   00560          CALL    4424H          ;OEFFNE FILE
              00570 ;        ***     A C H T U N G
              00580 ;        ***     BEI ANDEREN DOS'ES MUSS LRL ANGEGEBEN
              00590 ;        ***     WERDEN (MULTIDOS ERMITTELT SIE SELBST)
5258 FE00     00600          CP      0              ;FEHLERCODE IN AKKU ?
525A C23A55   00610          JP      NZ,DOSERR      ;WENN JA, ABBRUCH
              00620 ;        ***     QUELLFILE IST JETZT OFFEN
525D 111253   00630          LD      DE,ZCONV       ;ZIEL FILE IN DE
5260 213254   00640          LD      HL,ZDAT        ;DATENSPEICHER
5263 0600     00650          LD      B,0            ;LRL VON 256
5265 CD2044   00660          CALL    4420H          ;OPEN / INIT FILE
5268 FE00     00670          CP      0              ;FEHLERCODE IN AKKU ?
526A C23A55   00680          JP      NZ,DOSERR      ;WENN JA, ABBRUCH
              00690 ;        ***     ZIELFILE IST JETZT GEOFFNET
526D 11F252   00700 GETSEC   LD      DE,QCONV       ;QUELLFILE
5270 CD3644   00710          CALL    4436H          ;READ
5273 FE00     00720          CP      0              ;FEHLERCODE IN AKKU ?
5275 CA8052   00730          JP      Z,NOERR        ;WENN NEIN, WEITER
5278 FE1D     00740          CP      1DH            ;EOF ?
527A CAA952   00750          JP      Z,CLOSE        ;WENN JA, CLOSE
527D C33A55   00760          JP      DOSERR         ;SONST ABBRUCH
5280 0600     00770 NOERR    LD      B,0            ;ZAEHLER AUF 0
5282 113254   00780          LD      DE,ZDAT        ;ZIELDATENADDR. IN DE
5285 213253   00790          LD      HL,QDAT        ;QUELLDATENADDR.IN DE
5288 7E       00800 CONV     LD      A,(HL)         ;LADE ZEICHEN
5289 FE8D     00810          CP      8DH            ;IST ES 8DH ?
528B C29052   00820          JP      NZ,NOCHNG      ;WENN NEIN, NO CHANGE
528E 3E0D     00830          LD      A,0DH          ;ASCII CR
5290 12       00840 NOCHNG   LD      (DE),A         ;SPEICHERN AUF ZIEL
5291 13       00850          INC     DE             ;ZIEL + 1
5292 23       00860          INC     HL             ;QUELLE + 1
5293 04       00870          INC     B              ;COUNTER + 1
5294 78       00880          LD      A,B            ;B IN AKKU
5295 B7       00890          OR      A              ;FALGS SETZEN
5296 FE00     00900          CP      0              ;IST ER 0 (=256) ?
5298 C28852   00910          JP      NZ,CONV        ;WENN NEIN, WEITER
529B 111253   00920          LD      DE,ZCONV       ;ZIELFILE IN DE
529E CD3C44   00930          CALL    443CH          ;WRITE UND VERIFY
52A1 FE00     00940          CP      0              ;FEHLERCODE IN AKKU ?
52A3 C23A55   00950          JP      NZ,DOSERR      ;WENN JA, ABBRUCH
52A6 C36D52   00960          JP      GETSEC         ;SONST NAECHSTEN SEKTOR
52A9 11F252   00970 CLOSE    LD      DE,QCONV       ;QUELLFILE IN DE
52AC CD2844   00980          CALL    4428H          ;CLOSE QUELLFILE
52AF FE00     00990          CP      0              ;FEHLERCODE IN AKKU ?
52B1 C23A55   01000          JP      NZ,DOSERR      ;WENN JA, ABBRUCH
52B4 111253   01010          LD      DE,ZCONV       ;ZIELFILE IN DE
52B7 CD2844   01020          CALL    4428H          ;CLOSE ZIELFILE
52BA FE00     01030          CP      0              ;FEHLERCODE IN AKKU ?
52BC C23A55   01040          JP      NZ,DOSERR      ;WENN JA, ABBRUCH
52BF C34055   01050          JP      END

52C2          01060 QFILE    EQU     $              ;START QUELL TAB.
52DA          01070          ORG     $+24           ;LAENGE = 23 BYTES
52DA          01080 ZFILE    EQU     $              ;START ZIEL TAB.
52F2          01090          ORG     $+24           ;LAENGE = 23 BYTES
52F2          01100 QCONV    EQU     $              ;CONVERTIERTE QUELLE
```

```
5312            01110           ORG     $+20H           ;LAENGE
5312            01120 ZCONV     EQU     $               ;CONVERTIERTE ZFILE
5332            01130           ORG     $+20H           ;LAENGE
5332            01140 QDAT      EQU     $               ;START QUELL DAT.
5432            01150           ORG     $+256           ;LAENGE
5432            01160 ZDAT      EQU     $               ;START ZIEL DAT.
5532            01170           ORG     $+256           ;LAENGE
5532 3E2F       01180 PAERR     LD      A,2FH           ;FEHLERCODE
5534 CD0944     01190           CALL    4409H           ;AUSGABE VON ERRORCODE
5537 C32D40     01200           JP      402DH           ;ZUM DOS
553A CD0944     01210 DOSERR    CALL    4409H           ;DISPLAY ERROR
553D C32D40     01220           JP      402DH           ;ZUM DOS
5540 C32D40     01230 END       JP      402DH           ;ZUM DOS
5200            01240           END     START
00000 TOTAL ERRORS

CL  E   52A9
CO      5288
D  ERR  553A
END     5540
GETQ    5212
GETSEC  526D
GETZ    5223
GETZ2   5229
NOCHNG  5290
NOERR   5280
PAERR   5532
QCONV   52F2
QDAT    5332
QFILE   52C2
REFILE  523A
SEARCH  5203
START   5200
ZCONV   5312
ZDAT    5432
ZF      52DA
```