Folge 10

13.2.90

Hallo Spectrum-User!

Wenn’s mit dem März-Beitrag zum Info noch was werden soll, wird es langsam Zeit, damit anzufangen! Deshalb will ich gleich loslegen! Thema dieses Beitrages: das Retten schon gelöschter Files beim BETA-Disk-System Version 5.03 (mit größter Wahrscheinlichkeit treffen die Angaben auch für ältere zu)!

Wem ist es nicht schon mal passiert, daß man versehentlich oder voreilig einen File mit ’ERASE "Name"’ gelöscht hat, den man dann doch noch mal braucht. Mir ist das schon öfter passiert, wenn ich im Laufe einer Programm-Entwicklung oder Änderung die alte Version schon gelöscht hatte und vergaß, die überarbeitete Version neu abzuspeichern oder bei einem der glücklicherweise seltenen Systemabstürze! Oft ist die Arbeit von Stunden hin und man wäre froh, wenigstens noch die gerade gelöschte alte Version zu haben. In der Regel ist die beim BETA-Disk-System unter bestimmten Voraussetzungen relativ einfach regenerierbar.

Erste Voraussetzung ist, daß die Disk nach dem Löschen noch nicht mit ’MOVE’ verdichtet wurde, denn danach sind alle gelöschten Programme und Daten wirklich und für immer verschwunden! Die zweite Voraussetzung gilt nur für den Sonderfall, daß der gelöschte File zufällig der letzte auf der betreffenden Disk war! Dann klappt die Regenerierung nur, wenn nicht inzwischen ein neuer File auf die Disk gespeichert wurde!

Zuerst muß man wissen, daß beim BETA-Disk-System ein neuer File immer hinter den vorhergehenden auf Disk geschrieben wird. ’Lücken’ die durch Löschen entstehen, werden nicht sofort aufgefüllt, wie es bei manchen anderen Disk-Systemen der Fall ist. Das hat den Vorteil, daß ein File nicht zerstückelt quer über die ganze Disk verteilt abgelegt ist.

Ein Nachteil dieser Methode ist, daß nach Löschung vieler Einträge viel Platz noch durch die eigentlich ja nicht mehr benötigten Files belegt bleibt. Das läßt sich erst mit ’MOVE’ ändern, einen Befehl, der von den andern erwähnten Systemen gar nicht gebraucht wird!

Wird ein ERASE-Befehl ausgeführt, sucht das TRDOS nach einigen Vorbereitungen zuerst im Disk-Katalog den angegebenen File-Namen und Typ. Dazu wird nur das erste Zeichen eines Namens verglichen und erst, wenn dieses mit dem Anfang des gesuchten identisch ist, das jeweils folgende, bis alle passen (auch der Filetyp)! Das spart Zeit beim Suchen. Wurde er nicht gefunden, geht’s zurück ins Programm oder ins TRDOS mit Fehlermeldung ’No file(s)’!

Ist die Suche von Erfolg gekrönt, wird das erste Zeichen des Namens mit dem Code $01 überschrieben. War es jedoch der zuletzt auf die Disk gespeicherte File-Name, wird dort der Code $00 eingesetzt! Damit ist der betreffende Name nicht mehr aufrufbar, das gesamte Programm aber noch vorhanden.

Warum nun diese Unterschiede und wie ermittelt das TRDOS überhaupt, welche Position im Katalog der Name hat? Letzteres geschieht ganz einfach, indem sich das TRDOS aus dem Organisationssektor (Track 0, Sektor 8) die Gesamtzahl der auf der Disk befindlichen Files merkt und davon die Anzahl der verglichenen Namen abzieht. Kommt dabei 0 heraus, weil ja genau so viele Namen verglichen werden mußten wie im Katalog standen, muß es sich um den letzten Eintrag handeln.

Weshalb wird er aber mit Code $00 markiert? Wie schon gesagt, bleibt das Programm ja auf Disk erhalten nach dem Löschen, den nur der Name im Katalog ist ja als gelöscht gekennzeichnet! Wer sich vorher und nachher die Zahl der freien Sektoren ansieht, wird feststellen, daß sich daran beim Löschen mitten im Katalog (siehe Beispiel ’CAT-1 und CAT-2’) nichts ändert! Allerdings wird die Fileanzahl geringer und die Zahl der gelöschten Files entsprechend erhöht. Die vom gelöschten Programm belegten Disk-Sektoren sind also noch nicht wieder verfügbar!

Ganz anders sehen die Katalogmeldungen dagegen nach Löschen des letzen Eintrags aus (siehe ’CAT-3’)! Da erhöht sich plötzlich die Zahl freier Sektoren um die Anzahl Blöcke, die vorher das entfernte Programm belegte, es ist aber immer noch vorhanden! Die Filezahl vermindert sich natürlich auch, aber diesmal bleibt die Angabe für gelöschte Files unverändert!

Vorerst unbemerkt, hat das TRDOS jetzt im Organisationssektor neue Werte für den 1. freien Track und Sektor eingetragen (siehe Org-Sek-1 - 3) und zwar steht dort nun die Nummer des Tracks und Sektors in dem unser gelöschtes Programm begann. Das dies Folgen beim SAVEn eines weiteren Programms hat, ist wohl leicht einzusehen, denn dies wird dann ja den alten File sofort überschreiben. Der ist folglich danach auch nicht mehr regenerierbar.

Mit diesen Erkenntnissen wird auch sofort klar, wie man am Besten vorgeht, wenn eine ganze Reihe Files am Ende des Katalogs gelöscht werden sollen. Man beginnt beim letzten Eintrag und arbeitet sich von hinten nach vorn durch, denn weil dann ja jedesmal der augenblicklich letzte entfernt wird, stehen anschließend sofort die Sektoren für neue Programme zur Verfügung!

Nun aber zu einem praktischen Beispiel! Am einfachsten ist es, wir zäumen das Pferd von hinten auf, d.h. wir nehmen uns eine Disk vor, löschen schrittweise Files heraus und verfolgen dabei die Veränderungen im CAT-Ausdruck, im Disk-Katalog selbst und im Organisationssektor. Daraus lassen sich dann die notwendigen Maßnahmen zum Rekonstruieren von Katalog-Daten ableiten!

Ich habe das Programm "doctor" von der Utility-Disk zum Auslesen der entsprechenden Sektoren benutzt und eine Programmdiskette deren CAT-Ausdruck unter 'CAT-1' zu sehen ist! Daneben steht jeweils der CAT, wie er sich darstellt nachdem zuerst der Code-File "Symreass" und danach auch der Basic-File "Inst-sc1" mit 'ERASE' gelöscht wurde. Darunter ist noch der zugehörige entscheidende Ausschnitt der Disk-Katalogs und des Org.-Sektors abgedruckt!

Betrachtet man den 'CAT-2', so erkennt man sofort, daß ein File als gelöscht gemeldet ist (1 Del. File), und insgesamt einer weniger als bei 'CAT-1' angezeigt wird. Die Anzahl freier Sektoren hat sich nicht geändert. Im Disk-Kat.-2 findet man auch wie beschrieben, das erste Zeichen des gelöschten Namens "Symreass" mit einer $01 markiert wieder! Im Org.-Sektor ist nicht viel geschehen, nur als Anzahl für 'deleted Files' ist $01 eingetragen worden (Adr. $00F4)! In Adresse $E4 ist die Gesamtzahl aller Files der Disk vermerkt, aber daran ändert sich hier ja nichts, weil die Sektoren weiterhin blockiert bleiben!

Wollen wir nun den gelöschten File wieder retten, reicht es die $01 im Namen mit dem ursprünglichen Hex.-Wert zu überschreiben. Dazu die Utility-Disk einlegen und das "doctor"-Prog. aufrufen. Wenn das Menu erscheint sicherheitshalber zuerst

'8' (= Change Disk)

aufrufen, dann eventuell Track 0 und den erforderlichen Sektor anwählen und danach Menu-Punkt

'4' (= Read sector)

drücken. Die Frage "Read sector, sure? (Y/N)" mit 'Y' quittieren und schon ist der Sektor eingelesen ins RAM!

Menu-Punkt 6(= Display sector) zeigt nach Angabe der Startadresse eine Auflistung wie abgedruckt. Hat man die zu ändernde Stelle geortet, deren Adresse merken und per Menu-Punkt '7' (= Edit sector) diese aufrufen!

Nun braucht man nur noch den Hex-Wert des gewünschten ASCII-Codes (hier 'S'= $72) eingeben und nach Rückkehr ins Hauptmenu mit 'S' (= Write sector) den geänderten Sektor auf Disk zurückschreiben.

Nun kann man den File wieder im Katalog finden und auch schon laden oder kopieren. Falls der File nur vorübergehend gebraucht wird, kann man ihn nun auf eine andere Disk kopieren und anschließend das erste Zeichen wieder mit $01 überschreiben.

Denn wenn er auf Dauer auf der Disk erhalten bleiben soll, müssen wir ja noch die Anzahl der gelöschten Files im Org.-Sektor korrigieren, d.h. in unserem Fall um 1 vermindern, denn sonst stimmt dieser Zähler ja nicht mehr! Dazu braucht nur mit '2' (= Change sector) der Sektor 8 auf Track 0 eingestellt werden, der Rest läuft wie eben beschrieben ab! Damit ist dann alles erledigt und die Disk wieder voll funktionsfähig, denn nun stimmt der CAT, Disk-Katalog und Org.-Sektor wieder voll mit der ursprünglichen Version überein!

Etwas mehr Aufwand ist jedoch notwendig, falls (auch) der letzte File gelöscht wurde. In 'CAT-3' erkennt man, daß ein File weniger als im 'CAT-2' auf Disk ist, als gelöscht aber aus den schon beschriebenen Gründen auch nur einer angegeben wird. Die verfügbare Sektorenzahl hat sich um die 3 Blöcke des gelöschten Programms erhöht! In 'Disk-Kat-3' ist das erste Zeichen des Files "Inst-sc1" mit Code $00 überschrieben und bei "Symreass" mit Code $01!

Die gravierensten Veränderungen sind jedoch im Org.-Sektor geschehen. Dort taucht ein neuer Wert bei Adr. $E1 auf, statt vorher $03 steht da plötzlich $00! In dieser Adresse steht nämlich die Nummer des ersten freien Sektors auf der Disk und diese muß hier ja um 3 Sektoren (=Blocklänge von "Inst-scl") zurückverlegt werden. Es belegte also hier Sektor 0 bis 2 auf Track 8! Eventuell ändert sich gleichzeitig der Wert in Adr. $E2, der Nummer des Track, auf dem sich der erste freie Sektor befindet. Wäre unser gelöschter File hier z.B. 5 Sektoren lang gewesen, enthielte $E1 nun $0E (=14) und $E2 den Wert $07 (=7).

Adr. $E4 gibt uns die Gesamtzahl der Files inclusive der gelöschten an, hier nun $10 (= 16). Ein Vergleich mit dem 'CAT-3'-Daten zeigt die Richtigkeit (15 Files + 1 Del File =16)!

In den Adressen $E5 und $E6 findet man auch neue Werte ($80 =128 und $09 =9), denn hier steht die Anzahl freier Sektoren auf der Disk (in der üblichen low/high-Schreibweise. Zur Kontrolle mit den CAT-Angaben rechnen wir mal kurz nach:

9 * 256 + 128 « 2432, es stimmt also!

Die Anzahl gelöschter Files ist hier weiterhin 1 (Adr. $F4)!

Um den File "Inst-sc1" dauerhaft zu regenerieren, ist in diesem Fall also weit mehr zu korrigieren als im ersten Fall! Das Vorgehen mit Hilfe des "doctor"-Programms habe ich ja schon oben erläutert und welche Adressen mit welchen Daten zu korrigieren sind, sollte jetzt auch klar sein!

Wer wissen will, was in den hier nicht erwähnten Katalog- und Org.-Sektor-Daten verborgen ist, sollte das nochmal in den früheren Artikeln meiner Serie nachlesen (Folge 2 und 3 aus Info 8/89, Folge 4 aus Info 9/89)!

Man braucht oft jedoch gar nicht so viel Aufwand treiben, wenn man den gelöschten letzten File retten will! Auch hier kann man ihn nämlich schon nur durch Ersetzen des ersten Zeichen des Namens durch den ursprünglichen ASCII-Code wieder aufrufen.

Nachdem man ihn auf eine andere Disk kopiert hat, braucht dies Zeichen dann auf der alten Disk streng genommen nicht mal mit Code $00 überschrieben werden, da der Wert für den ersten freien Sektor ja nach wie vor auf den Anfang des gelöschten Programms zeigt. Beim nächsten SAVE wird es dann normalerweise überschrieben!

Gefährlich wäre allerdings, den File nun mit ERASE zu löschen, den dann würde er ja zum zweiten Mal als gelöscht im Org.-Sektor vermerkt und dort alle Angaben durcheinander bringen! Deshalb lieber auf Nummer sicher gehen und den ASCII-Code des ersten Zeichens mit $00 überschreiben! Man muß also immer höllisch aufpassen, daß man keinen Mist in den Disk-Katalog oder Org.-Sektor schreibt!

Das war's dann wieder für diesen Monat. Hoffentlich helfen diese Informationen dem Einen oder Anderen mal gelegentlich! Ob's mit einem Betrag im nächsten Monat was wird, ist noch nicht sicher.

Nachfolgend noch die im obigen Text öfter erwähnten CAT-Beispiel-Ausdrucke zum Vergleichen!

Übrigends, falls jemand an einer deutschen Übersetzung der BETA-Handbuchs Version 5.03 interessiert ist, kann ich weiterhelfen. Ich habe mir mal die Mühe gemacht und alles ins Deutsche übersetzt (ca. 32 DIN A 4-Seiten). Bis demnächst!

MfG

Wilhelm

(CAT-1)

Originalzustand

```
Title: Disk 38
17 File(s)
0 Del. File

A:Dismon16(B) 1:Dc16    (C) 20
A:Micass  (B) 2:MC2     (C) 33
A:Micass64(B) 2:MC64    (C) 35
A:Inst-drv(B) 1:Inst-dr1(C)  1
A:Inst-dr2(C) 6:Symreass(C)  4
A:Inst-scr(B) 1:Inst-sc2(C)  1
A:Inst-sc3(C) 2:Editas  (C)  1
A:Conf-isd(C) 1:Conf-isc(C)  1
A:Inst-sc1(B) 3

2429 Free
```

(CAT-2)

File "Symreass" gelöscht!

```
Title: Disk 38
16 File(s)
1 Del. File

A:Dismon16(B) 1:DC16    (C) 20
A:Micass  (B) 2:MC2     (C) 33
A:Micass64(B) 2:MC64    (C) 35
A:Inst-drv(B) 1:Inst-dr1(C)  1
A:Inst-dr2(C) 6:Inst-scr(B)  1
A:Inst-sc2(C) 1:Inst-sc3(C)  2
A:Editas  (C) 1:Conf-isd(C)  1
A:Conf-isc(C) 1:Inst-sc1(B)  3

2429 Free
```

(CAT-3)

File "Symreass" und "Inst-sc1" gelöscht!

```
Title: Disk 38
15 File(s)
1 Del. File

A:Dismon16(B) 1:DC16    (C) 20
A:Micass  (B) 2:MC2     (C) 33
A:Micass64(B) 2:MC64    (C) 35
A:Inst-drv(B) 1:Inst-dr1(C)  1
A:Inst-dr2(C) 6:Inst-scr(B)  1
A:Inst-sc2(C) 1:Inst-sc3(C)  2
A:Editas  (C) 1:Conf-isd(C)  1
A:Conf-isc(C) 1

2432 Free
```

(Disk-Kat.-1)

Track 0 / Sektor 0
CAT-Auszug, Originalzustand

```
0080  49 6E 73 74  I n s t
0084  2D 64 72 32  - d r 2
0088  43 A8 61 A6  C . a .
008C  05 06 0F 06  . . . .
0090  53 79 6D 72  S y m r
0094  65 61 73 73  e a s s
0098  43 A8 61 4F  C . a O
009C  03 04 05 07  . . . .
00A0  49 6E 73 74  I n s t
00A4  2D 73 63 72  - s c r
00A8  42 2A 00 2A  B * . *
00AC  00 01 09 07  . . . .
..............
..............
00F0  43 6F 6E 66  C o n f
00F4  2D 69 73 63  - i s c
00F8  43 2E E8 93  C . . .
00FC  00 01 0F 07  . . . .
0100  49 6E 73 74  I n s t
0104  2D 73 63 31  - s c 1
0108  42 AC 02 75  B . . u
010C  02 03 00 08  . . . .
0110  00 00 00 00  . . . .
```

(Disk-Kat.-2)

Track 0 / Sektor 0
CAT-Auszug, "Symreass" gelöscht

```
0080  46 6E 73 74  I n s t
0084  2D 64 72 32  - d r 2
0088  43 A8 61 A6  C . a .
008C  05 06 0F 06  . . . .
0090  01 79 6D 72  . y m r
0094  65 61 73 73  e a s s
0098  43 A8 61 4F  C . a O
009C  03 04 05 07  . . . .
00A0  49 6E 73 74  I n s t
00A4  2D 73 63 72  - s c r
00A8  42 2A 00 2A  B * . *
00AC  00 01 09 07  . . . .
..............
..............
00F0  43 6F 6E 66  C o n f
00F4  2D 69 73 63  - i s c
00F8  43 2E E8 93  C . . .
00FC  00 01 0F 07  . . . .
0100  49 6E 73 74  I n s t
0104  2D 73 63 31  - s c 1
0108  42 AC 02 75  B . . u
010C  02 03 00 08  . . . .
0110  00 00 00 80  . . . .
```

(Disk-Kat.-3)

Track 0 /Sektor 0
CAT-Auszug, "Symreass" und "Inst-sc1" gelöscht

```
0080  49 6E 73 74  I n s t
0084  2D 64 72 32  - d r 2
0088  43 A8 61 A6  C . a .
008C  05 06 0F 06  . . . .
0090  01 79 6D 72  . y m r
0094  65 61 73 73  e a s s
0098  43 A8 61 4F  C . a O
009C  03 04 05 07  . . . .
00A0  49 6E 73 74  I n s t
00A4  2D 73 63 72  - s c r
00A8  42 2A 00 2A  B * . *
00AC  00 01 09 07  . . . .
..............
..............
00F0  43 6F 6E 66  C o n f
00F4  2D 69 73 63  - i s c
00F8  43 2E E8 93  C . . .
00FC  00 01 0F 07  . . . .
0100  00 6E 73 74  . n s t
0104  2D 73 63 31  - s c 1
0108  42 AC 02 75  B . . u
010C  02 03 00 08  . . . .
0110  00 00 00 00  . . . .
```

Der Inhalt der zugehörigen Organisationssektoren sieht folgendermaßen aus:

(Org-Sek.-1)

Track 0 / Sektor 8

```
00E0  00 03 08 16  . . . .
00E4  11 7D 09 10  . ü . .
00E8  00 00 20 20  . . . .
00EC  20 20 20 20  . . . .
00F0  20 20 20 00  . . . .
00F4  00 44 69 73  . D i s
00F8  6B 20 33 38  k . 3 8
00FC  20 00 00 00  . . . .
```

(Org-Sek.-2)

Track 0 / Sektor 8

```
00E0  00 03 08 16  . . . .
00E4  11 7D 09 10  . ü . .
00E8  00 00 20 20  . . . .
00EC  20 20 20 20  . . . .
00F0  20 20 20 00  . . . .
00F4  01 44 69 73  . D i s
00F8  6B 20 33 38  k . 3 8
00FC  20 00 00 00  . . . .
```

(Org-Sek.-3)

Track 0 / Sektor 8

```
00E0  00 00 08 16  . . . .
00E4  10 80 09 10  . ü . .
00E8  00 00 20 20  . . . .
00EC  20 20 20 20  . . . .
00F0  20 20 20 00  . . . .
00F4  01 44 69 73  . D i s
00F8  6B 20 33 38  k . 3 8
00FC  20 00 00 00  . . . .
```

Fast hätte ich eine wichtige Information vergessen! Woher soll man eigentlich wissen, wie lang ein schon gelöschtes Programm war? Diese Angaben finden sich im Diskkatalog, beim File "Inst-sc1" z.B. in der Adr. $010D! Hier steht, daß dieser 3 Sektoren lang ist. In den Adr. $010E findet man übrigens den Sektor und in $010F den Track bei dem dies Programm beginnt! Dies Schema gilt natürlich für alle andern Files auch!

Folge 11

25.3.90

Hallo Spectrum-User!

Ich weiß nicht, ob dieser Beitrag noch rechtzeitig für's April-Info fertig wird, denn urlaubsbedingt bin ich reichlich spät angefangen damit! Aber vielleicht klappt's ja noch!

Diesmal soll es auch wieder um eine Routine aus dem BETA-Disk-Betriebssystem (=TRD0S) gehen. Es handelt sich dabei um die Umwandlung und Ausgabe von Hexadazimal- in Dezimalzahlen. Wenn vom TRDOS beim CAT- oder LIST-Befehl Zahlen auf dem Screen ausgegeben werden, gibt es dafür mehrere Methoden mit deren Hilfe dies vorgenommen wird.

Verschiedentlich wird dazu die SPECTRUM-ROM-Routine ab $1A1B benutzt (vom TRDOS aus natürlich per RST $20-Aufruf (siehe Info 9/89)). Sie dient normalerweise u.a. zur Zeilennummernausgabe, wobei die Hex.-Zahl im BC-Reg. stehen muß. Daraus entsteht dann die gewünschte Dezimalzahl.

Leider funktioniert dies nur bei bis zu 4-stelligen Zahlen, deshalb braucht das TRDOS z.B. für die Darstellung 5-stelliger Werte, wie Startadresse oder Programmlänge beim LIST-Befehl eine eigene Routine. Diese wollen wir uns nun näher ansehen.Sie ist übrigens auch wesentlich übersichtlicher aufgebaut als die vergleichbare ROM-Routine und eignet sich daher besonders gut für eine Beschreibung und mit kleinen Ergänzungen auch für eigene MC-Programme.

Interessant ist auch ein Vergleich mit der genannten ROM-Routine, denn beide beschreiten etwas unterschiedliche Wege! Aber das würde den Rahmen dieses Beitrags sprengen. Ich Versuche die Beschreibung so ausführlich wie möglich zu gestalten, damit jeder Schritt leicht verständlich und nachvollziehbar wird. Aber Grundkenntnisse in MC-Programmierung sind schon zum Verständnis erforderlich!

Die TRDOS-Routine beginnt bei Adresse $115D im BETA-ROM. Sie wird durch einen CALL-Befehl für die Ausgabe der Startadr. von $12EC und für die Programmlänge von $12FA aufgerufen. Sieht man sich die Bereiche vor dem CALL xxxx genauer an, so findet man,daß der auszugebende Wert hier im HL-Reg. zu finden ist! Das ist wichtig zu wissen, damit man den Ablauf der Ausgabe und Umwandlung ab $115D versteht. Der Hex.-Wert im HL-Reg. wird hier natürlich vorher aus der Katalogkopie im erweiterten TRDOS-Puffer geholt (siehe Info 8/89 und 9/89).

Wenn man sich das Assemblerlisting ansieht, fällt sofort ins Auge, daß es aus 4, nahezu identischen Blöcken besteht, mit deren Hilfe nacheinander zuerst die Zehntausender-, dann die Tausender-, Hunderter- und Zehner-Stelle ermittelt und angezeigt wird. Nur die Ermittlung der Einer-Stelle fällt etwas kürzer aus!

Um das Ganze anschaulicher zu gestalten, ist es am Besten, ein Beispiel mit konkreten Werten durchzuspielen. Dazu nehmen wir mal an, das daß HL-Reg. zu Anfang z.B. die Zahl $FB31 (=64305) enthält.

Die größte mit dieser Routine darstellbare Zahl ist übrigens $FFFF (=65535), die kleinste $0000! Ist eine Zahl in einem Doppelregister wie HL größer als $FFFF (=65535) dann 'läuft es über', weil ja eine weitere Stelle notwendig wäre. 65536 ist in Hexadezimal nämlich $10000! Um den Überlauf zu vermerken, wird ein sogenanntes Flag im F-Register (dies ist Teil des AF-Reg.), das C-Flag (oder Carry-Flag, übertrags-F1ag) gesetzt. Eine Reihe von Befehlen nutzen den Zustand der Flags, um abhängig vom Zustand z.B. Sprünge auszuführen (z.B. JR c,xxxx oder CALL NC,xxxx).

Zu Beginn (in Zeile 80) einer Ste11enermitt1ung wird jedes mal das hier als Zähler verwendete A-Reg. und gleichzeitig damit das C-Flag (Übertrags-Flag) gelöscht. Durch den Befehl 'SBC HL,DE' wird dann versucht, von der Zahl im HL-Reg. zuerst 10000 abzuziehen.

Das Ergebnis findet sich jedes mal im HL-Reg. wieder. Das geschieht so oft, bis sich ein Wert <0 ergibt. Dann wird sofort das C-Flag gesetzt und damit vermerkt, daß diesmal die erste Zahl (Minuend) kleiner war als der Subtrahend. Bei jeder neuen Subtraktion wurde durch 'INC A' auch jedesmal der Inhalt des A-Reg. erhöht, so daß dort letztendlich steht, wie oft die abgezogene Zahl (z.B. 10000) im ursprünglichen Wert enthalten war.

Nun eine Tabelle mit den nacheinander im HL- (oben Hexadezimal, darunter Dezimal) und A-Reg. enthaltenen Werten nach jedem Durchlauf:

HL:	$FB31	$D421	$SAD11	$8601	$5EF1	$37E1	$10D1	$E9C1
	64305	54305	44305	34305	24305	14305	4305	59841
A:	0	1	2	3	4	5	6	C-Flag

Die Zahl $E9C1 (=59841) zum Schluß kommt wie folgt zustande:

```
  $110Dl    ->      69841    (=65536+4035)
- $2710     ->     -10000
--------            -------
  $E9Cl     ->     -59841
```

Die 5. Stelle der Hex.-Zahl $110Dl muß sich der Prozessor 'ausleihen', damit er die Rechenoperation überhaupt vornehmen kann. Er setzt darum das C-Flag! Im Grunde wird dadurch zum HL-Reg.-Inhalt $10000 (=65536) 'addiert'. Im A-Reg. steht nun eine 6, bevor das C-Flag gesetzt wurde.

Letzteres bewirkt nun einen Sprung ('JR C,$1168) in ein Unterprogramm, wo zuerst einmal mit 'ADD $30' zum Wert im A-Reg. die Zahl 48 addiert wird. Hier ist das Ergebnis $36 (=54), was bekanntlich den ASCII-Code der Zahl 6 ergibt! Diese wird mit Hilfe eines weiteren Unterprogramms (aufgerufen mit 'CALL $11A8') ausgegeben und anschließend zum noch im HL-Reg. befindlichen Wert (hier $E9C1 = 59841) mit 'ADD HL,DE' hier 10000 zu addiert.

Das ergibt natürlich eine Zahl >$FFFF, in diesem Beispiel wieder 69841 ($110Dl). Da das HL-Reg. aber nicht mehr als 4 Stellen faßt, bleibt davon in HL nur $10Dl übrig (=4305)! Den übertrag zeigt wieder das C-Flag an.

Jetzt geht das Spiel wieder von vorne los mit der Ermittlung der Tausender-Stelle (ab Zeile 280)! Zuerst wieder A-Reg. samt C-Flag löschen, DE mit 1000 laden usw.! Welche Werte diesmal bei unserem Beispiel im HL- und A-Reg. erscheinen zeigt wieder die nachstehende Tabelle:

HL:	$10Dl	$0CE9	$0901	$0509	$0131	$FD49	
	4305	3305	2305	1305	305	64841	
A :	0	1	2	3	4	C-Flag	gesetzt!

Die Zahl $FD49 (=64841) zum Schluß kommt wie folgt zustande:

```
  $10131    ->      65841    (=65536+305)
 -$03E8     →     - 1000
--------            -------
  $FD49     ->      64841
```

Die 4 im A-Reg. wird nun wie beschrieben in den entsprechenden ASCII-Code umgewandelt und ausgegeben. Dann wird wie gehabt mit 'ADD HL,DE' der positive Rest wieder ins HL-Reg. zurückgeholt und weiter geht's mit der Hunderter-Stelle (ab Zeile 400). Auch dazu die Liste mit den Registerinhalten bei der Subtraktion:

HL:	$0131	$OOCD	$0069	$0005	$FFA1
	305	205	105	5	65441
A :	0	1	2	3	C-Flag gesetzt!

Die Zahl $FFA1 (=65441) zum Schluß kommt wie folgt zustande:

```
  $10005    ->      65541   (=65536+5)
- $0064     ->        -100
--------            ------
  $FFA1     ->      65441
```

Auch hier wird nun wieder die 3 im A-Reg. in den ASCII-Code umgewandelt und auf den Bildschirm und der verbliebene Rest wieder ins HL-Reg. gebracht. Die Zehner-Stelle kommt nun an die Reihe (ab Zeile 520). Die Liste dazu ist diesmal sehr kurz:

```
HL:     $0005    $FFFB
          5      65531
A :       0      C-Flag   gesetzt!
```

```
Die Zahl $FFFB (=65531) zum Schluß kommt wie folgt zustande:
                      $10005    ->         65541      (=65536+5)
                     -$000A     ->           -10
                     --------               -------
                       $FFFB    ->         65531
```

Im A-Reg. steht diesmal eine 0, weil sich von 5 die 10 kein mal ohne übertrag abziehen ließ. Diese wird auch wieder ausgegeben und nachdem auch hier der Restbetrag wieder ins HL-Reg. gebracht wurde, kann die Einer-Stelle drankommen (ab Zeile 640. Zu deren Ermittlung ist nun kein großer Aufwand notwendig, denn sie befindet sich schon im HL-Reg. als letzter Rest, genau genommen im L-Reg.! Deshalb braucht dessen Inhalt (hier 5) nur noch mit 'ADD $30' in den richtigen ASCII-Code verwandelt und auf den Bildschirm gebracht zu werden. Nun steht dort unsere Beispielzahl '64305' komplett!

Ich hoffe, daß meine Beschreibung in Verbindung mit dem Assemblerlisting auch für Nicht-Experten den Ablauf der Routine verständlich macht! Wer aufgepaßt hat, wird bemerkt haben, daß diese Routine bei Dezimal-Zahlen mit weniger als 5 Stellen die ersten Stellen mit 0 auffüllt (wie auch beim LIST-Befehl zu beobachten)! Das abzustellen würde etwas mehr Aufwand bedeuten, da ja auch zwischen führenden und sonstigen Nullen unterschieden werden müßte! Daher soll uns das jetzt nicht weiter stören.

Wenn man diese Routine in eigenen MC-Programmen einsetzen will, muß man natürlich den TRDOS-spezifischen Teil von Zeile 720-830 entfernen und die 5 'CALL $11A8'-Befehle ersetzt man am Besten durch ein 'RST $10'. Dieser Befehl wurde ja auch über einige notwendige Umwege vorher vom TRDOS aufgerufen. Er bewirkt letztendlich die Ausgabe des Codes im A-Reg. auf den Bildschirm an der aktuellen PRINT-Position.

Drei weitere kleine Ergänzungen sind noch erforderlich, bevor die Routine vollständig ist. Zuerst muß die Startadresse in den RAM-Bereich verlegt werden. Dazu in Zeile 10 einen neuen Wert für 'org' angeben.

Nun muß für die Zeichenausgabe ein Kanal geöffnet werden mit:

```
20 LD A,2              ;Kanalnummer ins A-Reg. (2=Screen, 3=Drucker)
30 CALL $1601          ;SPECTRUM-ROM-Routine 'Kanal öffnen' aufrufen
```

Dann muß natürlich auch noch die auszugebende Zahl ins HL-Reg. gebracht werden. Im einfachsten Fall geschieht dies durch einen simplen Ladebefehl:

```
40 LD HL,$xxxx
```

Damit ist die Routine in eigenen MC-Programmen lauffähig. Zur Eingabe ist natürlich ein Assemblerprogramm empfehlenswert, falls man nicht mühsam alle Mnemonics (so heißen die Befehlskürzel wirklich!) mit einem Hex.-Lader als Hexadezimalzahlen eingeben will. Letzteres ist immer sehr fehlerträchtig und mühsam, besonders wenn Ergänzungen eingeschoben werden müssen.

Bei der Ermittlung der Registerinhalte hat mir mein Disassembler gute Dienste geleistet. Er erlaubt das Abarbeiten von MC-Programmen im Einzelschritt-Modus (Trace-Modus) und zeigt bei jedem Schritt alle Änderungen der Register, Flags usw. an. So kann man sich leicht die Wirkung einzelner Befehle vor Augen führen und genau erkennen, wo's zum Beispiel in einer selbst gestrickten Routine noch hakt!

Nun muß für diesmal wieder Schluß sein. Es ist schon der 27.3. und ich glaube nicht, daß es für's April-Info noch reicht! Wahrscheinlich nicht, aber ich will mich überraschen lassen. Bis demnächst!

MfG
Wilhelm

```
115D            0010            org $115D ;Startadr. in TRDOS (AUSGABE  EINER  5-STELLIGEN  DEZIMAL-ZAHL)
                0020 ;
                0030 ;Der Hex-Wert der umzuwandelnden Zahl muß hier im HL-Reg. stehen!
                0040 ;
                0050 ;
                0060 ;Zuerst wird die ZEHNTAUSENDER-Stelle ermittelt!
                0070 ;
115D AF         0080            XOR    A            ;A-Reg. (und damit auch gleichzeitig das C-Flag  löschen)
115E 111027     0090            LD     DE,$2710     ;und DE-Reg. mit 10000 laden!
1161 EB52       0100    L1161 SBC     HL,DE        ;Von momentanem Wert in HL nun 10000 abziehen
                0110 ;
                0120 ;Das Ergebnis der Subtraktion findet man anschließend im HL-Reg.!
                0130 ;
1163 3803       0140            JR     C,L1168      ;Wenn C-Flag gesetzt ist, bei $1168 weiter!
1165  3C        0150            INC    A            ;Sonst A-Reg.-Inhalt um 1 erhöhen!
1166 18F9       0160            JR     L1161        ;Wiederholen, bis Wert in HL <0 ist (also quasi negativ)
                0170 ;
                0180 ;Dann wird das C-Flag (Übertrags-Flag) gesetzt.
                0190 ;
1168 C630       0200    L1168 A99     $30          ;A-Reg.-Inhalt *          $30 * Code der l0000er-Stelle!
116A C6A811     0210
                0220            CALLL11A8          ;Nun zur Ausgabe        des gefundenen Codes springen.
1169 19         0220            ADD    HL,DE        ;Zum 'negativen' Wert in HL wieder 10000 addieren.
                0230 ;
                0240 ;Der verbliebene positive Rest befindet sich somit wieder im HL-Reg.!
                0250 ;
                0260 ;Nun die TAUSENDER-Stelle ermitteln!
                0270 ;
116E AF         0280            XOR    A            ;A-Reg. (und damit auch gleichzeitig das C-Flag löschen)
116F 11E803     0290            LD     DE,$03E8     ;und DE-Reg. mit 1000 laden!
1172 EB52       0300 L1172      S9C    HL, DE       ;Vom momentanem Wert in HL nun 1000 abziehen
1174 3803       0310            JR     C,L1179      ;Wenn C-Flag gesetzt ist, bei $1179 weiter!
1176 3C         0320            INC    A            ;(Sonst A-Reg.-Inhalt um 1 erhöhen!
1177 18F9       0330            JR     L1172        ;Wiederholen, bis Wert in HL <0 ist (also quasi negativ)
1179 C630       0340 L1179      ADD    $30          ;A-Reg.-Inhalt + $30 = Code der l000er-Stelle!
117B CDA811     0350            CALL  11A8          ;Nun zur Ausgabe des gefundenen Codes springen.
117E 19         0360            ADD   HL,DE         ;Zum 'negativen' Wert in HL wieder 1000 addieren.
                0370 ;
                0380 ;Nun die HUNDERTER-Stelle ermitteln!
                0390 ;
117F AF 0400                    XOR A               ;(A-Reg. (und damit auch gleichzeitig das C-Flag löschen)
1179 116400     0410            LD     DE,$0064     ;und SE-Reg. mit 100 laden!
1183 EB52       0420 L1183      SBC    HL,9E        ;Vom momentanem Wert in HL nun 100 abziehen
1185 3803       0430            JR     C,L118A      ;Wenn C-Flag gesetzt ist, bei $118A weiter!
1187 3C         0440            INC    A            ;Sonst A-Reg.-Inhalt um 1 erhöhen!
1188 18F9       0450            JR     L1183        ;Wiederholen, bis Wert in HL <0 ist (also quasi negativ)
118A C630       0460 L118A      ADD    $30          ;A-Reg.-Inhalt + $30 = Code der l00er-Stelle!
118C CDA811     0470            CALL    L11A8       ;Nun zur Ausgabe des gefundenen Codes springen.
118F 19         0480            ADD    HL,DE        ;Zum 'negativen' Wert in HL wieder 100 addieren.
                0490 ;
                0500 ;Nun die ZEHNER-Stelle ermitteln!
                0510 ;
1190 AF         0520            XOR    A            ;A-Reg. (und damit auch gleichzeitig das C-Flag löschen)
1191 110A00     0530            LD     DE,$000A     ;und DE-Reg. mit 10 laden!
1194 E952       0540 L1194      SBC    HL,DE        ;Vom momentanem Wert in HL nun 10 abziehen
1196 3803       0550            JR     C,L119B      ;Wenn C-Flag gesetzt ist, bei $1199 weiter!
1198 3C         0560            INC    A            ;Sonst A-Reg.-Inhalt um 1 erhöhen!
1199 18F9       0570            JR     L1194        ;Wiederholen, bis Wert in HL <0 ist (also quasi negativ)
119B C630        0580 L119B     ADD    $30          ;A-Reg.-Inhalt + $30 = Code der l0er-Stelle!
119D CBA811     0590            CALL    L11A8       ;Nun zur Ausgabe des gefundenen Codes springen.
UA0 19          0600            ASS    HL,DE        ;Zum 'negativen' Wert in HL wieder 10 addieren.
                0610 ;
                0620 ;Nun bleibt als Rest noch die EINER-Stelle!
                0630 ;
11Al 79         0640            LD     A,L          ;Rest (= ler-Stelle)          ins A-Reg.    bringen!
11A2 C630       0650            ADD    $30          ;A-Reg.-Inhalt * $30          = Code der    ler-Stelle!
11A4 CBA811     0660            CALL    L11A8       ;Nun zur Ausgabe des        gefundenen      Codes springen.
11A0 C9         0670            RET                 ;Ende der Routine und zuruck!
                0680 ;
                0690 ;
                0700 ;UNTERPROG. ZUR AUSGABE EINES ZEICHENS!
                0710 ;
11A8 E5         0720 L11A8      PUSH   HL           ;HL- und
11A9 95         0730            PUSH   9E           ;DE-Reg. retten.
11AA CD823D     0740            CALL   $3D82        ;Auf Interface 1 prüfen und ROM-Routine RST $10 aufrufen!
                0750 ;
                0760 ;Ein ROM-Routinenaufruf geschieht aus den TRDOS heraus bekanntlich per
                0770 ;RST $20-Aufruf (siehe Info 9/89 Seite 4). Nachdem das Zeichen aus dem
                0780 ;A-Reg. auf den Bildschirm oder Drucker ausgegeben wurde, erfolgt die
                0790 ;Rückkehr ins TRDOS!
                0800 ;
11AD D1         0810            POP    DE           ;DE- und
11AE El         0820            POP    HL           ;HL-Reg.-Werte zurückholen.
11AF C9         0830            RET                 ;Zurück ins Hauptprogramm!

#       61A8  L1161   1161  L1168    1168  L1172   1172  L1179   1179  L1183   1183  L118A   118A  L1194  1194  L119B   119B
L11A8   11A8
```