Folge 12

27.4.90

Hallo Spectrum User!

Mein heutiges Thema dreht sich um Verbesserungen des an sich schon guten Assembler/Disassemblers 'SYS' Vers. 2.2!

Die Erste betrifft den Disassembler 'Dismonl6/ 48'! Dort gibt es die auf Seite XX beschriebene LIST-Befehlerweiterung. Damit kann man Code von der Ladeadr. unabhängig mit der Original-Adr. disassemblieren, falls er z.B. auf Grund von Überschneidungen an eine andere als diese geladen werden mußte.

Gibt man die Routine laut Handbuch ein, so funktioniert sie leider nur bei LIST-, jedoch nicht bei DUMP. Das kann leicht geändert werden, da beide Befehle zur Adressausgabe die gleichen Routinen verwenden. Man braucht nur mit Hilfe der MONITOR-Funktion 2 Adressen ändern und zwar bei:

			Neue		alte Werte
"Dismonl6":	**ab**	**h72F7 –>**	**hA2 / h6C**		**(hD6 / h72)**
und	**ab**	**h73O5 –>**	**hD4 / h6C**		**(hA0 / h70)**
"Dismon48":	**ab**	**hF2F7 –>**	**hA2 / hEC**		**(hD6 / hF2)**
und	**ab**	**hF305 –>**	**hD4 / hEC**		**(hA0 / hF0)**

Jetzt wird auch bei DUMP neben Anfangs-/Endadr. eine Bezugsadr. erwartet und schon kann man sich alle MC-Programmteile mit korrekten Adressen ausgeben lassen. Es ist mir unverständl ich, warum diese Verbesserung im Handbuch nicht mit angeführt wird!

Nun die zweite Verbesserung! Sie betrifft den Assembler und gilt in erster Linie für alle BETA-Disk-User. Laut Handbuch (Seite 14) wird bei BETA und TIMEX der VERIFY-Befehl ignoriert, da das DOS ihn automatisch durchführt. Also habe ich mir überlegt, daß man ihn durch einen anderen ersetzen könnte, wie CAT oder ERASE!

Dazu mußte ich erst mal finden, wie und wo zwischen Normal- und Disk-Befehlen (die mit vorangestellten '*') unterschieden wird ('*CLEAR' fällt auch darunter, ist jedoch kein Disk-Befehl)!

Weil es vielleicht Einige interessiert, wie man so was angehen kann, will ich versuchen meine Überlegungen und das Vorgehen möglichst genau zu beschreiben!

Zuerst habe ich mir gedacht, daß es ja wohl irgendwo im MC-Programm eine Stelle geben müßte, an der auf das Symbol '*' (Code $2A) getestet wird! Da dies meist mit einem Vergleichsbefehl (cp = compare) geschieht, ist es naheliegend, darauf hin zu untersuchen. In unserem Fall lautet der Befehl also 'CP $2A', was die Zeichenfolge '$FE / $2A' gibt.

Die Sucharbeit habe ich aber dem Disassembler überlassen, dem man ja per EDIT die Zeichenfolge übergeben und anschließend mit SEARCH-Befehl im Bereich des 'Micass'-Codes (von 56626 bis 65535) suchen lassen kann«(An 4 Stellen wurde er bei meiner Version fündig und zwar bei Adr. $ECAC, $EDFC, $F744 und $F75B!

Deren Umgebung habe ich mir per DUMP näher angesehen, besonders den Bereich vor der gesuchten Zeichenfolge. Dort suche ich möglichst eine Stelle mit einem $C9 (= RET) oder $C3, $xx, $xx (= JUMP $xxxx)! Dahinter beginnt mit großer Sicherheit ein Unterprogramm, daß man mit 'LIST start,ende' disassembl ieren kann. So gibt's ziemlich sicher sinnvolle Listings.

Es zeigte sich, daß bei 5EDFC/FD die Codefolge zu 2 verschiedenen Befehlen gehörte, also kein 'CP $2A' ergab. Um $F744 und $F75B tauchen einige CP-Befehle auf,die hauptsächlich auf die Rechensymbole + - / * und $%"()& hin untersuchen. Auch nicht das Richtige!

Der entscheidende Befehl steht bei $ECAC! Dort trennen sich die Wege, falls ein '*'-Befehl gefunden wird! Alle weiteren Einzelheiten genau zu erklären, würde zu lang, daher nur das Wesentliche in Kurzform (siehe auch Assemblerlistings):

Zuerst wird das C-Reg. mit 5 geladen als Zähler für 5 gültige *-Befehle und dieser in der Systemvar. XPTR (Adr. $5C5F =23647) gerettet (durch 'LD (IY+ $25),C')! Nun wird das vorher ab $E542 abgelegte eingetippte Befehlswort mit denen in einer Tabelle ab $E7A3 verglichen (Kleinbuchstaben als große interpretiert).

Wurde in der Tabelle Wortende erkannt (Code $20), ohne daß das Wort mit dem gesuchten Befehl übereinstimmte, so wird das C-Reg. um 1 vermindert und weiter verglichen. Sonst Abbruch der Suche und in C steht eine Zahl, die anzeigt, das wievielte Wort dem gesuchten entsprach. Daraus wird nun auf komplizierte Weise ein Wert errechnet, der nach Addition mit dem IX-Reg. eine Adresse ergibt. Die wird über DE auf den Stapel gelegt und mit RET angesprungen.

Abhängig von der errechneten Adr. wird nun für den jeweiligen Diskbefehl eine unterschied1iche Schlüsselzahl ermittelt (außer bei '*clear'). Diese lautet bei:

***save -> $80, *load -> $81, *merge -> $82 und *verify -> $83**

und wird in $5C5F geladen. Die Schlüsselzahlen sind entscheidend für Verzweigungen zur befehlsspezifischen Bearbeitungsroutine!

Da VERIFY nur bei BETA- und TIMEX-Disk ignoriert wird, liegt der Schluß nahe, daß die weitere Behandlung dieses Befehls vom durch den File 'INSTAL.DRV' (enthält Anpassungsroutinen für ZX-Microdrive, Opus Discovery, Beta-Disk und Time-Floppy und andere) einkopierten BETA-Teil erfolgt!

Sieht man sich den 1. Teil dieser Anpassungsroutine ($E5E1 - $F142) an, so fällt bei aufmerksamer Durchsicht auf, daß obige Werte dort in Verbindung mit den entsprechenden Befehlscodes auftauchen:

$E530 $80, $F8 -> ($F8 = Code für SAVE)
$E532 $81, $EF -> ($EF = " " LOAD)
$E534 $82, $EF -> ($EF = M M LOAD)
$E536 $83, $D6 -> ($D6 = " " VERIFY)

Auffällig ist, daß Code $EF gleich 2 mal vorkommt. Dabei steht das 2. LOAD als Ersatz für MERGE, denn es handelt sich ja nur um ein LOAD mit neu errechneter Ladeadresse! Das BASIC-MERGE ist hier unbrauchbar, da nur für BASIC-Prog. tauglich.

Aus dem doppelten Auftreten des Codes $EF kann man schließen, daß nicht der Befehlscode selbst, sondern die davor stehende Nummer zur Unterscheidung der einzelnen Befehle mit herangezogen wird!

Dies bestätigt sich auch, wenn man nachsieht, wie nach der Schlüssel-zahlermittlung verzweigt wird (Adr. $EFBF-$EFC6). Dort wird erst mal geprüft, ob die Schlüsselzahl => $80 ist. Bei $80 ist es der '*savec oder *savet'-Befehl, der gesondert behandelt wird ($EFC7-$EFD7).

Danach oder bei einem sonstigen Disk-Befehl, wird erst der Namenspuffer ($E639-$E648) gelöscht, der Name hineinkopiert ($EFD8-$EFDC) und nach einigen Vorbereitungen mit Hilfe der Schlüsselzahl unterschieden, ob ein Kassetten- oder Disk-Befehl bearbeitet werden soll ($EFDD-$EFE2).

Da bei uns letzteres der Fall ist, wird mit dem Befehl 'JP M,$F091' in den zweiten Teil der Anpassungsroutine ($F091-$F142) gesprungen. Gleich bei $F091 steht 'CP $83' gefolgt bei $F093 von 'RET Z'? Damit ist auch klar, warum 'VERIFY' gar nicht erst abgearbeitet wird, denn bei Code $83 ist das Z-Flag gesetzt und damit der Rücksprung erzwungen.

Der dann folgende Programmteil generiert die bei BETA-Disk erforderliche REM-Zeile mit Befehl, Namen und CODE-Daten. Er wird bei allen anderen Disk-Befehlen durchlaufen! Zuletzt erfolgt die Ausführung durch den TRDOS-Aufruf bei Adr. $E5F9 mit anschließender Rückkehr in die Eingaberoutine!

Ersetzt man 'RET Z' durch 'NOP' (Code $00), so wird der Befehl 'VERIFY' ausgeführt, wie man schnell überprüfen kann, allerdings nicht durch Fehlermeldungen unterstützt, sodaß er wirklich unbrauchbar ist!

Aber jetzt haben wir den Schlüssel für neue, sinnvollere Befehle gefunden, denn wenn VERIFY nun abgearbeitet wird, muß das bei einem 'CAT' oder 'ERASE' auch funktionieren! Dazu sind allerdings noch 2 kleine Ergänzungen notwendig:

1.) In der Tabelle von $E630 - $E638 in der die Codes $80 - $83 den Befehls-codes zugeordnet sind, müssen wir den 'VERIFY'-Code $D6 durch den gewünschten ersetzen (bei 'CAT' = $CF oder bei 'ERASE' =$D2) bei Adresse $E638 (58936)!

2.) In der Tabelle mit den erlaubten Befehlen muß natürlich 'VERIFY' mit 'CAT' oder 'ERASE' überschrieben werden:

Original-Version mit **VERIFY**

$E7A3	$43	$4C	$45	$41	$52	$20	CLEAR
$E7A9	$53	$41	$56	$45	$20	$4C	SAVE L
$E7AF	$4F	$41	$44	$20	$56	$45	OAD VE
$E7B5	$52	$49	$46	$59	$20	$4D	RIFY M
$E7BB	$45	$52	$47	$45	$20	$D9	ERGE .

Version mit **CAT**

$E7AF	$4F	$41	$44	$20	**$43**	**$41**	OAD **CA**
$E7B5	**$54**	**$20**	**$4D**	**$45**	**$52**	**$47**	**T MERG**
$E7BB	**$45**	$20	$20	$20	$20	$D9	**E**

Version mit **ERASE**

$E7AF	$4F	$41	$44	$20	**$45**	**$52**	OAD **ER**
$E7B5	**$41**	**$53**	**$45**	$20	$4D	$45	**ASE ME**
$E7BB	**$52**	**$47**	**$45**	$20	$20	$D9	RGE .

Jetzt können wir den Assembler aufrufen und die neuen Befehle testen: Bei ’ERASE’ also eine Disk mit zu löschenden CODE-Files einschieben, *erase "Filename" <ENTER> eintippen und schon ist der CODE-File gelöscht. Bei anderen Filetypen klappt das aus dem Assembler natürlich nicht, aber das wäre ja auch nicht sinnvoll!

Ist die Wahl auf ’CAT’ gefallen, tippen wir *cat <ENTER> ein... und nichts passiert! Wie das? Nach der Befehlswortprüfung wird noch auf einen Filenamen in "" getestet und genau den vermißt der Assembler nun! Also Korrektur: *cat "a:" <ENTER> (oder *cat "") und schon erscheint der Katalog von Laufwerk A. Sehen wir uns die erzeugte Zeile 5 an, so sieht sie folgendermaßen aus:

5 REM: CAT "a:" CODE 32768,23599

Das ’CODE 32678,23599’ paßt eigentlich gar nicht zum CAT-Befehl, wird aber hier zwangsläufig erzeugt, weil auch das alte ’*verify’ nicht anderes als ein abgewandelter LOAD-Befehl war! Aber glücklicherweise stört es auch nicht! Man könnte statt ’CAT’ natürlich auch ’LIST’ einbauen, aber da gibt es schneller Probleme mit dem Speicherplatz. (Warum? Siehe meinen Artikel Folge 3 vom 15.7.89), wenn RAMTOP auf 25000 gesetzt ist.

Wer nun glaubt, er müßte auch bei Rekorder-Betrieb auf den dort ja nicht unwichtigen VERIFY-Befehlverzichten, der irrt! Allerdings muß er mit ’cat’ "Filename" (oder ’erase "...") aufgerufen werden, denn ’verify "..."’ wird ignoriert, weil der Befehlsname in der Tabelle ja überschrieben wurde! Aber wieso funktioniert das überhaupt?

Entscheidend ist hier wieder die Schlüsselzahl (hier $03, siehe Listing Zeile 1800) denn nur sie wird bei der entscheidenden Verzweigung (bei Adr. $F0F3) abgefragt. Danach geht’s dann bald in die entsprechende ROM-Routine ($0556)!

Wer neugierig geworden ist, kann das Programm ja mal selbst mittels Disassembler erforschen! Dazu eignet sich hervorragend der TRACE-Modus in Verbindung mit der Möglichkeit, Breakpoints zu setzten!

Aber Achtung, das TRDOS läßt sich im TRACE-Modus nicht aufrufen, daher ist spätestens beim Befehl in Adresse $E5F9: ’CALL $3C03 (oder $3D03) Schluß zu machen! Genug für diesmal! Hoffentlich sind die Erläuterungen verständlich genug ausgefallen! Bis nächstes Mal!

MfG
Wilhelm

```
E542        0010      ORG $E542 ;EIN6ABEPUFFERBEREICH (mit Beispiel):
            0020 ;
E542        0030      DB   $2a, $5c, $6f, $61, $64, $20    ;*load
E548        0040      DB   $22, $46, $69, $6c, $65, $64    ;*Filen
E54E        0050      DB   $61, $64, $65, $22, $00, $00    ;ame"
E554        0060      END
            0070 ;================================================================
            0080 ;
E630        0090      ORG $E630 ;TABELLE MIT SCHLÜSSELZAHLEN UND BEFEHLSCODES!
            0100 ;
E630        0110      DB   $80, $f8, $81, $ef     ;$F8 = save, $EF = load
E634        0120      DB   $82, $ef, $fiB, $B6    ;$EF = load (lerge),$D6  = verify
E638        0130      END
            0140 ;================================================================
            0150 ;
E7A3        0160      ORG $E7A3 STABELLE MIT *-BEFEHLEn!
            0170 ;
E7A3        0180      DB   $43, $4c, $45, $41, $52, $20 ;C L E A R
E7A9        0190      DB   $53, $41, $56, $45, $20, $4c ;S A V E L
E7AF        0200      DB   $4f, $41, $44, $20, $56, $45 ;0 A D V E
E7B5        0210      DB   $52, $49, $46, $59, $20, $44 ;R 1 F Y M
E7BB        0220      DB   $45, $52, $47, $45, $20, $D9 ;E R G E .
E7C1        0230      END
```

```
                 0260 ;HL- zeigt auf Eingabepufferanfang ($E542) und B-Reg. enthält $2013.
                 0270 ;1. Zeichen <>$20 aus Puffer ist in A, HL zeigt auf dessen Pufferadr.!
                 0280 ;
ECAC             0290          org    $ECAC        ;BEGINN  DER UNTERSCHEIDUNG ZWISCHEN NORMAL-  UND '*'-BEFEHLEN
                 0300 ;
ECAC  FE2A       0310          CP     $2a          ;Ist das erste Zeichen des Befehls '*'?
ECAE  2807       0320          JR     Z,Lecb7      ;Ja? Dann weiter bei $ECB7!
ECBO  FE30       0330          CP     $30          ;Oder ist das Zeichen => '0'?
ECB2  300D       0340          JR     NC,Lecc1     ;Dann weiter prüfen bei $ECC1!
ECB4  C3BAEB     0350          JP     $ebba        ;Weiter bei $EBBA (Wert unzulässig)!
ECB7  0E05       0360    Lecb7 LD     C,$05        ;C-Reg. mit  5 laden als Zähler für '*'-Befehle)
ECB9  11A3E7     0370          LD     DE,$e7a3     ;Anfang der Tabelle der zulässigen '*'-Befehle laden.
ECBC  CD3BEC     0380          CALL    $ec3b       ;Nächstes Zeichen <> $20 aus Puffer ins A-Reg. bringen.
ECBF  1808       0390          JR     Lecc9        ;Weiter bei $ECC9!
ECC1  FE3A       0400    Lecc1 CP     $3a          ;Ist der Code <$3A?
ECC3  DAC6F1     0410          JP     C,$f1c6      ;Dann ist es eine Zahl (0-9)! Weiter bei $F1C6.
ECC6  1150E7     0420          LD     DE,$e750     ;Anfangsadr. der Tabelle aller erlaubten Befehle laden!
ECC9  FD7125     0430    Lecc9 LD     (IY+$25),C   ;Stand des Befehlzählers in $5C5F laden.
                 0440 ;
                 0450 ;Wenn kein "*'-Befehl, dann enthält C = $13 (=19 mogliche Befehle).
                 0460 ;
ECCC  E5         0470    Leccc PUSH   HL           ;Puffer-Adr. des ersten Zeichens retten.
ECCD  1A         0480    Leccd LD     A,(DE)       ;Zeichen aus Befehlswort-Tab. in A-Reg. holen und
ECCE  B8         0490          CP     B            ;Testen, ob es $20 (Befehlswortende) ist (B enthält auch $20)!
ECCF  281E       0500          JR     Z,Lecef      ;Dann ist Befehl gefunden! Weiter bei $ECEF!
ECD1  AE         0510          XOR (HL)            ;Sonst durch XOR-Verknüpfung von A- und HL-Reg.-Inhalt!
                 0520 ;
                 0530 ;Dadurch Unterscheidung Klein- / Großschrift / sonstige  Zeichen! Beispiel:
                 0540 ;Obere Zeile: das Zeichen im A-Reg. (hier $44 =*D'),
                 0550 ;nächste Zeile: diverse Zeichen aus HL (=Eingabepuffer),
                 0560 ;untere Zeile: Ergebnis in A-Reg. (ungleiche Bits ergeben eine 1).
                 0570 ;
                 0580 ;'D'=$44 = 01000100              'D'=$44 = 01000100        'D'=$44 = 01000100
                 0590 ;'D'=$44 = 01000100              'd'=$64 = 01100100        '.'=$2E = 00101110
                 0600 ; ---------------------------------- ------------------------------------ ----------------------------------
                 0610 ;     $00 = 00000000                   $20 = 00l00000             $6A = 01101010
                 0620 ;
ECD2  2817       0630          JR     Z,Leceb      ;Korrekte  Großschrift? Dann Z-Flag gesetzt, weiter > $ECEB!
ECD4  B8         0640          CP     B            ;Oder steht im A-Reg. nun $20 (wie in B!)?
ECD5  2814       0650          JR     Z,Leceb      ;Dann korrekte Kleinschrift! Zum nächsten Zeichenvergleich!
ECD7  7E         0660          LD     A,(HL)       ;Keins davon? Dann ursprünglichen Code aus Puffer holen
ECD8  B8         0670          CP     B            ;und prüfen, ob es ein Code $20 (Leerzeichen) ist!
ECD9  2815       0680          JR     Z,Lecf0      ;Dann weitermachen bei $ECF0!
ECDB  FE2E       0690          CP     $2e          ;oder ist es ein "."? (abgekürzter Befehl)
ECDD  2811       0700          JR     Z,LecfO      ;Dann auch weiter bei $ECF0, da Vergleich beendet!
ECDF  1A         0710 Lecdf LD        A,(DE)       ;Sonst nächstes Zeichen aus Tabelle holen,
ECE0  13         0720          INC    DE           ;DE-Reg. auf nächste Tabellenadr. erhöhen
ECE1  B8         0730          CP     B            ;und prüfen, ob das Zeichen Code $20 ist (Wortende).
ECE2  20FB       0740          JR     NZ,Lecdf     ;Nein? Dann zurück und weiter danach suchen!
ECE4  El         0750          POP HL              ;Wenn Wortende gefunden, Pufferanfangsadr. vom Stapel holen!
ECE5  0D         0760          DEC    C            ;Befehlswortzähler um 1 erniedrigen
ECE6  20E4       0770          JR     NZ,Leccc     ;und falls <> 0, nächsten Befehl mit Eingabe vergleichen!
ECE8  C3B9EB     0780          JP     $ebb9        ;Sonst Abbruch, da Eingabe falsch!
                 0790 ;
ECEB  13         0800    Leceb INC    DE           ;Tabellenadresse
ECEC  23         0810          INC    HL           ;Eingabepufferadresse jeweils um 1 erhöhen!
ECED  18DE       0820          JR     Leccd        ;und weiter zum nächsten Zeichenvergleich!
                 0830 ;
ECEF  2B         0840    Lecef DEC    HL           ;HL auf vorletztes Zeichen im Eingabepuffer zurücksetzen.
ECF0  DD2114ED   0850    Lecf0 LD     IX,$ed14     ;IX mit Basisadr. für Sprungadr.-Berechnung laden und
ECF4  0600       0860          LD B,$00            ;B mit 0 (beides wird nur bei Nicht-'*'-Befehlen gebraucht).
ECF6  FDCB2556   0870          BIT    2,(IY+$25)   ;Bit 2 von $5C5F ist nur bei Nicht-'*'-Befehlen gesetzt
ECFA 2806        0880          JR     Z,Led02      ;Bei denen ist Z-Flag auch gesetzt! Dann weiter bei $ED02!
ECFC  DD213A6D   0890          LD     IX,$6d3a     ;Bei '*'-Befehl $6D3A als Basis für Sprungadr.-Berechnung
ED00  0680       0900          LD     B,$80        ;ins IX- und $80 ins B-Reg. laden!
                 0910 ;
                 0920 ;Im B- steht nun $80, im C-Reg. die Nummer des eingegebenen Befehls,
                 0930 ;d.h. bei *clear = 5, *save « 4, *load = 3, *verify = 2, *merge = 1!
                 0940 ;
ED02  D9         0950    Led02 EXX                  ;Erst- und Zweit-Registersatz vertauschen!
ED03  68         0960          LD     L,B          ;B- nach L-Reg. umladen
ED04  D9         0970          EXX                 ;und Registersätze wieder vertauschen!
ED05  CB21       0980          SLA    C            ;Nun C-Reg.-Inhalt verdoppeln durch Linksschieben der Bits in C!
                 0990 ;
                 1000 ;Z.B. bei C-Inhalt 3   =>    00000011  ->    nach SLA C    ->   00000110 = 6
                 1010 ;
ED07  DD09       1020          ADD IX,BC           ;Nun hier IX * BC, Ergebnis in IX = Adresszeiger
                 1030 ;
                 1040 ;Bei  *clear : IX + BC      =        $6D3A  +  $800A = $ED44   ->   zeigt auf Adr.   $EED5
                 1050 ;   "  *save        "       "        S6D3A  +  $8008  =$ED42   ->    "       "        $EFBF
                 1060 ;   "  *load        "       "        $6D3A  +  $8006  =$ED40   ->    "       "        $EFBE
                 1070 ;   "  *verify      "       "        $6D3A  +  $8004  =$ED3E   ->    "       "        $EFBC
                 1080 ;   "  *merge       "       "        $6D3A  +  $8002  =$ED3C   ->    "       "        $EFBD
                 1090 ;
ED09  Dl         1100          POP    DE           ;Alten Wert von DE-Reg. vom Stapel nehmen, da nun überflüssig!
ED0A 118CEC      1110          LD     DE,$ec8c     ;$EC8C als Rücksprungadr. in Eingabeschleife laden
ED0D  D5         1120          PUSH   DE           ;und für später retten!
ED0E  DD5E00     1130          LD     E,(IX+$00)   ;Nun Wert der Adr. auf die IX zeigt in DE laden.
EB11  DD5601     1140          LD     D,(IX+$01)   ;DE enthält damit für jeden Befehl eine andere Adresse!
ED14  D5         1150 Led14    PUSH   DE           ;Diese auf dem Stapel ablegen und
ED15  C9         1160          RET                 ;gleich wieder vom Stapel nehmen und anspringen.
ED16             1170          END
```

```
ED3C          1200          ORG  $ed3c          ;TABELLE  MIT  SPRUNGADRESSEN  ZUR SCHLÜSSELZAHLERMITTLUNG!
              1210 ;
ED3C          1220          DB    $bd,    $ef    ;Diese Adr. ($EFBD) wird bei  '*'merge' angesprungen!
ED3E          1230          DB    $bc,    $ef    ;      "          ($EFDC)    "   '*verify'       "
ED40          1240          DB    $be,    $ef    ;      "          ($EFBE)    "   '*load'         "
EB42          1250          DB    $bf,    $ef    ;      "          ($EFBF)    "   '*save'         "
ED44          1260          DB    $d5,    $ee    ;      "          ($EED5)    "   '*clear'        "
ED46          1270          END
              1280 ;
              1290 ;
EFBC          1300          ORG    $EFBC         (SCHLÜSSELZAHLERMITTLUNG UND VERZVEIGUNGZU *-BEFEHLEN!
              1310;
EFBC 04       1320          INC    B             ;Hierhin erfolgt Sprung bei   '*verify'!
EFBD 04       1330          INC    B             ;      "            "          '*merge'!
EFBE 04       1340          INC    B             ;      "            "          '*load'!
EFBF FD7025   1350          LD     (IY+$25),B    ;      "            "          '*save'!
              1360 ;
              1370 ;Schlüsselzahl in B ist nur bei *save unverändert $80, bei *load $81,
              1380 ;*merge $82 und *verify $83! Wert in (IY+$25) = $5C5F ablegen!
              1390 ;
EFC2 3E7F     1400          LD     A,$7f         ;A-Reg. mit 127 (binär 01111111) laden und damit Wert aus
EFC4 A0       1410          AND    B             ;B maskieren. Nur bei B = $80 (10000000) ist das Ergebnis 0!
              1420 ;
              1430 ;AND-Verknüpfung: Beide Bits 1? Dann Ergebnis ebenso 1! Sonst 0!
              1440 ;Verknüpfungsergebnis landet in A! Nur bei A = 0 ist Z-Flag gesetzt!
              1450 ;
              1460    ;$7F  =  01111111    $7F     = 01111111     $7F   =01111111     $7F     = 01111111
              1470    ;$80  =  10000000    $81     =10000001      $82   =10000010     $83     = 10000011
              1480 ;          ------------ ----------------------- ------------------------ -------------------------------
              1490    ;$00  =  00000000    $01     = 00000001     $02   =00000010     $03     = 00000011
              1500 ;
EFC5 2011     1510          JR     NZ,Lefd8      ;Nicht gesetzt? Also kein *save-Befehl? Weiter bei $EFD8!
EFC7 CD3BEC   1520          CALL     $ec3b       ;Doch? Dann nächstes Zeichen <>$20 von Puffer in A laden
EFCA F620     1530          OR     $20           ;und alle Buchstaben wie kleine behandeln!
              1540 ;
              1550 ;OR-Verknüpfung: Eins oder beide Bits 1? Dann Ergebnis 1! Sonst 0!
              1560 ;Verknüpfungsergebnis landet in A! Nur bei A = 0 ist Z-Flag gesetzt!
              1570 ;
              1580 ;$20 = "  "  = 00100000          $20 = "  " = 00100000          $20 = "  " = 00100000
              1590 ;$43 = "C" = 01000011          $54 = 'T" = 01010100          $63 = "t"  = 01110100
              1600 ; ---------------------------------- -..... ..... .....
              1610 ;$63 = "c"  = 01100011          $74 = "t"  = 01110100          $63 = "t   = 01110100
              1620 ;
EFCC FE63     1630          CP     $63           ;Ist es "c"? Dann wurde '*savec' eingegeben und
EFCE 2805     1640          JR     Z.LefdS       ;es soll Objektcode gespeichert werden! Weiter bei $EFD5!
EFD0FE74      1650          CP     $74           ;Oder ist es "t"? Dann wurde '*savet' eingegeben (Quelltext)!
EF02 C2B9EB   1660          JP     NZ,$ebb9      ;Keins von beiden? Dann Eingabe nicht korrekt. Abbruch!
EFD5 32605C   1670 LefdS    LD     ($5c60),A     ;Sonst den Code von "c" oder "t" in $5C60 retten!
EFD8 C5       1680 Lefd8    PUSH   BC            ;BC-Reg. retten (B enthält die errechnete Schlüsselzahl!)
EFD9 CD37EB   1690          CALL     $eb37       ;Puffer loschen (16 Zeichen ab $E639). Namen hineinkopieren
              1700 ;
              1710 ;Dabei wird in $5C5B (=IY+$21) Anzahl der nicht genutzten Zeichen des
              1720 ;Namens (max. 12, z.B.: "a: Filename") abgelegt!
              1730 ;
EFDC C1       1740          POP    BC            ;BC-Reg. zurückholen!
EFDD 3EFF     1750          LD     A,$f f        ;A-Reg. mit $FF laden und mit Wert aus B-Reg.
EFBF A0       1760          AND    B             ;AND-Verknüpfung vornehmen.
              1770 ;
              1780 ;Hierbei steht im A- anschließend der gleiche Wert wie im  B-Reg.! Bei
              1790 ;AND wird auch das S-Flag gesetzt, wenn das Ergebnis =>$80 ist!
              1800 ;
EFE0 FA91F0   1810          JP     N,$f091       ; S-Flag gesetzt? Dann war's ein *-Befehl. Weiter bei $F091
              1820 ;
              1830 ;Ab hier werden Kassettenbefehle weiterbehandelt! Schlüsselzahlen
              1840 ;für Kassettenbefehle: SAVE =$00, LOAD =$01, MER6E =$02, VERIFY $03!
EFE3          1850          END
              1860;=============================================================================================
              1870 ;
F091          1880          0RG    $F091         ;'*VERIFY'-ABVEISUNG
              1890 ;
F091 FE83     1900          CP     $83           ;Ist die ermittelte Schlüsselzahl $83?
F093 C8       1910          RET    Z             ;Dann Abbruch der Bearbeitung und Rückkehr in Eingaberoutine!
              1920 ;
              1930 ;HIER BEGINNT  FÜR  ALLE  ÜBRIGEN  DISK-BEFEHLE  DIE  REM-ZEILEN-ERZEUGUNG!
              1940 ;
F094 2A5F5C   1950          LD     HL,($5c5f)    ;Schlüsselzahl (und bei *save Code "t" bzw "c") in HL
F097 22765C   1960          LD     ($5c76),HL    ;und dann in Systemvariable $5C76 ablegen!
F09A ED7B3D5C 1970          LD     SP,($5c3d)    ;Stapelzeiger auf Adr. aus ERR SP setzen und
F09E 210313   1980          LD     11,$1303      ;Adr. ROM-Rout. "Rückkehr nach Zeilenausführung" laden und
F0A1 E3       1990          EX     (SP),HL       ;als Rücksprungadr. nach REM-Zeilenausführung auf Stapel!
F0A2 FD3600FF 2000          LD (IY+$00),$ff      ;ERR NR auf $FF (= Wert für 'kein Fehler') setzen!
F0A6 216F15   2010          LD     HL,$156f      ;Nun einen Teil der ROM-Routine "Zeile in Progr. einfügen"
F0A9 1142E5   2020          LD     DE,$e542      ;(von ,$156F-$15AA) in Eingabepuffer ab $E542 kopieren.
F0AC 013C00   2030          LD     BC,$003c      ;(Routinenlänge = 60 Zeichen)
F0AF EDB0     2040          LDIR
F0B1 3EC9     2050          LD     A,$c9         ;Code für RET
F0B3 12       2060          LD     (DE),A        ;ans Ende der Kopie schreiben.
F0B4 3E1D     2070          LD     A,$1d         ;Nun max. Länge der zu erzeugenden Zeile festlegen (=29) und
F0B6 FD9621   2080          SUB     (IY+$21)     ;davon Anzahl der nicht genutzten Namenszeichen abziehen.
F0B9 6F       2090          LD     L,A           ;Das Ergebnis (= tatsächliche Zeilenlänge) in L bringen
```

```
F0BA 60          2100          LD    H,B          ; und H-Reg. löschen!
F0BB 0E05        2110          LD    C,$05        ;C-Reg.mit der vorgesehenen Zeilennmer laden und
F0BD EB43495C    2120          LB    ($5c49),BC   ;lin E PPC (Nummer der laufenden Prog.-Zeile) ablegen.
F0C1 E5          2130          PUSH   HL          ;HL-Reg. (Zeilenlänge!) und
F0C2 C5          2140          PUSH   BC          ;BC-Reg. (Zeilennummer) retten!
F0C3 CD42E5      2150          CALL    $e542      ;ROM-Rout.- Kopie aufrufen und Platz für Zeile 5 schaffen!
F0C6 El          2160          POP    HL          ;Nun die Zeilennr. von Stapel in HL holen und dann ROM-Rout.
F0C7 CD6E19      2170          CALL    $196e      ;*Zeilenanfang suchen* aufrufen, Nummer und Länge einfügen!
F0CA 23          2180          INC    HL          ;HL-Reg. zeigt auf Zeilenanfang, deshalb durch viermaliges
F0CB 23          2190          INC    HL          ;hochzählen Zeilennummer (2 Stellen) und
F0CC 23          2200          INC    HL          ;Zeilenlängenangabe (2 Stellen) überspringen.
F0CD 23          2210          INC    HL          ;HL-Reg. zeigt danach auf Adr. für's Befehlswort!
F0CE 225D5C      2220          LD    ($5c5d),HL   ;Diesen Wert in CH ADD (= nächstes zu interpr. Zeichen).
F0D1 36EA        2230          LD    (HL),$ea     ;In diese Adr. den Befehlswort-Code von 'REM' schreiben,
F0D3 23          2240          INC    HL          ;HL eins weiter,
F0D4 363A        2250          LD    (HL),$3a     ;den ':'einfügen und
F0D6 23          2260          INC    HL          ;HL-Reg. nochmals erhöhen!
F0D7 E5          2270          PUSH   HL          ;Diese aktuelle Zeilen-Adresse erst mal auf Stapel retten!
F0D8 3A765C      2280          LD    A,($5c76)   ;Nun Schlüsselzahl ins A-Reg. holen
F0DB 4F          2290          LD    C,A          ;und nach C umladen.
F0DC 2130E6      2300          LD    HL,$e630     ;Nun Anfangsadr. der Schlüsselzahlen- / Befehlscode-Tabelle
F0DF EDB1        2310          CPIR               ;laden und diese auf Code im A-Reg. (=Schlüsselzahl) durchsuchen!
F0E1 7E          2320          LD    A,(HL)       ;HL zeigt nun auf zugehörigen Befehlscode, diesen ins A-Reg.
F0E2 E1          2330          POP    HL          ;Aktuelle Zeilenadr. vom Stapel holen und
F0E3 77          2340          LD    (HL),A       ;den Befehls-Code in Zeile einfügen ($F8, $EF oder $D6)!
F0E4 08          2350          EX    AF,AF'       ;Befehls-Code erst mal im Alternativ-Reg. zwischenspeichern
F0E5 23          2360          INC    HL          ;und HL auf nächste Zeilenadr. setzen.
F0E6 3622        2370          LD    (HL),$22     ;Das Anführungszeichen einfügen,
F0E8 23          2380          INC    HL          ;HL wieder erhöhen,
F0E9 C1          2390          POP    BC          ;tatsächliche Zeilenlänge nun ins BC-Reg. holen
F0EA 79          2400          LD    A,C          ;und ins A-Reg. umladen! Davon den Wert für min. Zeilenlänge
F0EB D612        2410          SUB    $12         ;abziehen (=18 = Zeilenlänge  mit ' ' als Namen).
F0ED 4F          2420          LD    C,A          ;Ergebnis (=Namenslänge) in C zurückschreiben!
F0EE 2807        2430          JR    Z,Lf0f7      ;War es 0? Dann kein Name (= ' ')! Weiter bei $F0F7!
F0F0 1139E6      2440          LD    DE,$e639     ;Sonst Anfangsadr. des Programmnamen-Puffers in DE laden!
F0F3 EB          2450          EX    DE,HL        ;Reg.-Tausch vornehmen und den Namen anschließend in die
F0F4 EDBO        2460          LDIR               ;Zeile umkopieren!
F0F6 EB          2470          EX    DE,HL        ;Erneut Reg.-Tausch. HL zeigt nun auf Adr. hinterm Namen!
F0F7 3622        2480   Lf0f7  LD    (HL),$22     ;An diese Adr. den Code von '*' schreiben
F0F9 23          2490          INC    HL          ;und HL auf nächste Adr. einstellen!
F0FA 36AF        2500          LD    (HL),$af     ;Dahin den Befehl 'CODE' (= $AF) schreiben.
F0FC E5          2510          PUSH   HL          ;Dann aktuelle Zeilenadr. erst mal retten
F0FD 08          2520          EX    AF,AF'       ;und Befehlscode aus Alternativreg. zurückholen!
F0FE FEF8        2530          CP    $f8          ;War es ein 'save'-Befehl?
F100 3810        2540          JR    C,Lf11f      ;Nein? Dann weiter bei $F11F!
F102 3A775C      2550          LD    A,($5c77)    ;Sonst hier abgelegten Code holen ("c"oder  "t") und
F105 0F          2560          RRCA               ;'Rechts durch C-Flag rotieren', also Bit 0 ins C-Flag! Bei 'c'
F106 3809        2570          JR    C,Lf111      ;(= $63 = 01100011) nach $F111 zur Adr.- u. Längenberechnung,
F108 CD43F1      2580          CALL Lf143         ;bei 't' (= $74 = 01110100) dagegen nach   $F143!
F10B 22E6E5      2590          LD    ($e5e6),HL   ;Ermittelte  Quelltext-Anfangsadr. hier   ablegen!
F10E EB          2600          EX     DE,HL       ;Vertauschen! Nun HL = Quelltext-Länge, DE = Quelltext-Anfang.
F10F 182D        2610          JR      Lf13e      ;Ermittelte Werte in  Zeile einbauen und Zeile aufrufen!
F111             2620          END

                 2640 ========================================================================
                 2650 ;
F110             2660          ORG $F110          ;STARTADR. UND LÄNGENBERECHNUNG  BEI 'SAVEC, LOAD, MERGE'!
                 2670 ;
F110 2D          2680          DEC    L
F111 2A54FF      2690 Lf111 LD       HL,($ff54)   ;Bei 'savec': Anfang des Objektcodepuffers in HL
F114 22E6E5      2700          LD    ($e5e6),HL   ;und in $E5E6 ablegen!
F117 EB          2710          EX    DE,HL        ;Durch Reg.-Tausch Adr. des Objektcode-Anfangs in DE bringen
F118 2A69E6      2720          LD    HL,($e669)   ;und HL mit dessen Endadresse laden!
F11B ED52        2730          SBC    HL,DE       ;Jetzt aus beiden Werten die Objektcode-Länge berechnen
F11D 181E        2740          JR    Lf13d        ;und zum Einbau der Adr. in die Zeile und Ausführung!
                 2750 ;
F11F CD43F1      2760 Lf11f CALL        Lf143      ;Bei load / merge: erst Quelltext-Anfang / Ende berechnen!
F122 FBCB3C46    2770          BIT    0,(IY+$3c)   ;Test ob Bit 0 von Schlüsselzahl in $5C76 »0! Nicht 0?
F126 2005        2780          JR    NZ,Lf12d     ;Dann war Bit 0=1, da Schlüsselzahl bei 'load' = $81!
                 2790 ;
F128 19          2800          ADD    HL,DE       ;Sonst 'merge'! Dann Anfang+Länge = Quelltext-Ende errechnen
F129 2B          2810          DEC    HL          ;und davon 1 abziehen, da Ende vorher auf Ende-Marker zeigte!
F12A 327FE6      2820          LD    ($e67f),A    ;Code des Ende-Markers (=$FF) in $E67F ablegen!
                 2830 ;
F12D 227DE6      2840 Lf12d    LD    ($e67d),HL   ;Bei 'load': Anfangsadr. des Quelltextes, bzw. bei
F130 22E6E5      2850          LD    ($e5e6), HL  ;'merge' Endadr. = Ladeadresse hier 21 ablegen!
F133 EB          2860          EX    DE,HL        ;Durch Reg.-Tausch die Ladeadresse in DE bringen
F134 2A63E6      2870          LD    HL,($e663)   ;und HL mit der Anfangsadr. des Assemblers laden!
F137 25          2880          DEC    H           ;Davon 256 abziehen, um etwas Platz zu behalten.
F138 ED52        2890          SBC    HL,DE       ;Aus dieser Adr. - Ladeadr. den restlichen Platz errechnen!
F13A 3001        2900          JR    NC,Lf13d     ;Ist noch Platz für Quelltext? Dann Zeile fertigstellen!
F13C 24          2910          INC    H           ;Sonst steht in H-Reg. $FF, auf 0 setzen! Nun Wert im HL <$FF!
F13D 23          2920 Lf13d    INC    HL          ;Zur Verfügung stehender Platz + 1!
F13E C3E1E5      2930 Lf13e    JP    $e5e1        ;Ladeadr. und Länge in Zeile einbauen und ausführen!
F141 00          2940          NOP
F142 00          2950          NOP
F143 2A50FF      2960 Lf143    LD    HL,($ff50)   ;Anfangsadr. des Quelltextpuffers in HL holen
F146 E5          2970          PUSH   HL          ;und diesen Wert auf den Stapel retten!
F147 AF          2980          XOR    A           ;A-Reg. löschen {
F148 47          2990          LD    B,A          ;ebenso B-Reg.
```

```
F146 E5       2970      PUSH HL         ;und diesen Wert auf den Stapel retten!
F147 AF       2980      XOR  A          ;A-Reg. Löschen,
F148 47       2990      LD   B,A        ;ebenso B-
F149 4F       3000      LD   C,A        ;und C-Register!
F14A 3D       3010      DEC  A          ;0 - 1 = 255 ($FF) = Code für Textende-Marker!
F14B EDB1     3020      CPIR            ;Diesen im Que11text suchen. Gefunden? Dessen Adr. dann in HL!
F14D D1       3030      POP  DE         ;Nun die Adr. des Quelltext-Anfangs von Stapel in DE holen
F14E ED52     3040      SBC  HL,DE      ;und Ende-Marker-Adr. - Anfang = Länge! Ergebnis in HL.
F150 EB       3050      EX   DE,HL      ;Nach Tausch: DE = Quelltext-Länge, HL = Quelltext-Anfang!
F151 C9       3060      RET             ;Fertig? Deshalb zurück nach $F10B oder $F122!
F152          3070      END
              3080 ;
              3100 ;
E5E1          3110      ORG $E5E1       ;STARTADR. UND LÄNGE IN ZEILE EINBAUEN UND DIESE AUSFÜHREN!
              3120 ;
E5E1 22EEE5   3130 Le5e1 LD  ($e5ee),HL  ;Code- oder Text-Länge in $E5EE ablegen.
E5E4 E1       3140      POP  HL         ;Nun aktuelle Zeilenadr. wieder vom Stapel holen undin DE-Reg.
E5E5 110080   3150      LD   DE,$8000   ;Quelltext- (32768) oder Objektcode-Anfang (25000)laden
E5E8 CD0FE6   3160      CALL $e60f      ;Jeweilige Zahl in die Zeile einbauen!
E5EB 362C     3170      LD (HL).$2c     ;Ein Komma dahinter setzen!
E5ED 112F5C   3180      LD DE,$5c2f     ;Eben hier abgelegte Code-/Text-Länge (z.9.23599) in OE!
E5F0 CD0FE6   3190      CALL $e60f      ;Auch diese Angabe in die Zeile einfügen!
E5F3 360D     3200      LD (HL),$0d     ;Als Zeilenende den 'ENTER'-Code = $0D dahinter setzen!
E5F5 215827   3210      LD HL,$2758     ;HL mit 10072 laden! Sinn ist mir z.Z. noch nicht klar!!!
E5F8 D9       3220      EXX             ;Alle Reg. mit des Alternativ-Reg.-Satz vertauschen
E5F9 CD033D   3230      CALL $3d03      ;und TRDOS Vers.5.xx aufrufen (bei Vers. 4.xx =$3C03)!
E5FC C369EC   3240      JP $ec69        ;Nach ordnungsgemäßem beenden zurück in die Eingaberoutine!
E5FF          3250      END
```

29.4.90

Nun noch ein kleiner Nachtrag, obwohl der Beitrag schon länger ist, als ursprünglich geplant! Wer nämlich genau hingesehen hat, der wird sich vielleicht fragen, wieso denn überhaupt die von der Routine ab Zeile 1950 gebildete REM-Zeile abgearbeitet werden kann? Denn normalerweise müßte sie ja in der korrekten BETA-Syntax (hier -für Vers. 5.03) z.B. so aussehen:

5 RANDOMIZE USR 15619: REM: LOAD -Filename" CODE 32768,23599

Der RANDOMIZE-Aufruf fehlt jedoch in der gebildeten Zeile! Das der Aufruf trotzdem funktioniert liegt daran, daß wir in Zeile 2220 (siehe Assemb1er 1isting) die Adresse des REM-Befehl in die Systemvariable CH ADD ($5C5D) gerettet haben.

Wird jetzt mit CALL $3D03 (=15619) in Zeile 3230 das TRDOS aufgerufen, so wird nämlich auf diese Variable CH ADD zugegriffen, denn sie zeigt ja auf das nächste zu interpretierende Zeichen. Ab da wird dann zuerst das 'REM' gesucht, alles auf richtige Syntax getestet und falls fehlerfrei, der Befehl ausgeführt!

Im Grunde ist hier nur die gleiche Situation geschaffen worden, wie sie beim Aufruf der vollständigen Zeile nach Ausführung des 'RANDOMIZE USR 15619' herrschen würde, denn auch dann zeigt CH ADD auf das Zeichen ':' dahinter und der Rest läuft wie gehabt!

Diesmal war mein Beitrag nun sehr speziell, aber ich hoffe, daß auch der eine oder andere Maschinensprache-Interessierte, jedoch 'Nicht'-BETA-Disk oder 'SYS'-Besitzer, etwas damit anfangen kann! Deshalb auch die ausführlichen Kommentare! Also nochmals, bis demnächst!

MfG

Wilhelm