commodore

Serie 610 und 710

Hardware - Beschreibung

ROM-RAM-I/O

Assemblerlisting

ausführlich dokumentiert

2. verbesserte Auflage

R. Schineis · O.-M. Braun

V o r w o r t

Das vorliegende, kommentierte und strukturierte Assemblerlisting des Betriebssystems CBM 610 u. 710 wurde mit dem Hard + Soft 'Reassembler - Assemblersystem' unter Verwendung des Original-Betriebssystems mit folgender Kennung erstellt:

BASIC LOW	901243 - 02B	CBM II	128K
BASIC HI	901242 - 02	CBM II	128K
Kernal	901244 - 03B	CBM II	LP/HP

Diese ROMs sind in den Rechnern, Stand August/September 1983 eingebaut.

Das Buch ist in folgende fünf, verschiedenfarbige Teile gegliedert:

1. Hinweise zur Benutzung der Segment-Umschalteroutinen
2. Beschreibung der Hard- und Software der Schnittstelle RS 232 C
3. Speicherbelegung, ROM/RAM-Cartridge, Steckerbelegungen
4. Assembler-Listing BASIC lo/hi mit Crossreferenztabelle
5. Assembler-Listing Operating-System (Kernal) mit Crossreferenztabelle

Im ersten Teil finden Sie ein von Commodore zur Verfügung gestelltes Listing mit Beschreibung zur Benutzung der Segment-Umschaltroutinen. Damit ist es möglich, Maschinenunterprogramme in anderen 64K-Segmenten als dem gerade benutzten zu verwenden. Es besteht damit auch die Möglichkeit einer Segment- Umschaltung zum Segment $F (Betriebssystem) im Falle eines IRQ, NMI oder BRK.

Der zweite Teil behandelt ausführlich die Schnittstelle RS 232 C. Es wird darin Allgemeines über die Schnittstelle, der Befehlsaufbau, der Status und die Hardware beschrieben.

Im dritten Abschnitt finden Sie eine Übersicht der Speicheraufteilung, Besonderheiten und Handhabung des Cartridge-Anschlusses und eine Zusammenfassung aller Steckerbelegungen.

Im vierten und fünften Teil finden Sie das zweiteilige Assemblerlisting. Es ist jeweils durch eine Crossreferenz-Tabelle, in der alle Label mit ihren Adressen und den Zeilennummern in denen diese Label vorkommen, abgeschlossen. Bei der Erstellung wurden, soweit bekannt und möglich, die Original Commodore-Label verwendet. Gleiche Adressen können in beiden Teilen mit verschiedenen Namen bezeichnet sein.

Für dieses Buch ist nach Erscheinen der Rechner mit deutscher Tastatur ein voll kommentiertes Listing des geänderten Operating-Systems beim Herausgeber erhältlich. Außerdem wird für die Rechner CBM 620 u. 720 (deutsche Tastatur) wegen größerer Abweichungen ein eigenes Assemblerlisting herausgegeben. Das zweite Buch enthält bereits das oben erwähnte Operating-System mit deutscher Tastatur.

Der Buchrücken ist verleimt. Die Innenseiten sind für eine eventuelle Ringbuchablage vorgelocht. Es ist dadurch problemlos möglich, bei häufigem Gebrauch die einzelnen Seiten in einem Ringbuchordner abzulegen.

Der Inhalt des vorliegenden Buches wurde sorgfältig überprüft. Es ist jedoch nicht ganz auszuschließen, daß hier und da noch Druck- und andere Fehler vorliegen. Die Autoren sind für jeden schriftlichen Hinweis dankbar, um in kommenden Auflagen Ihre Erfahrungen einfließen lassen zu können. Außerdem sind alle weiteren Anregungen, sei es als Wunsch oder als Vorschlag, willkommen.

R. Schineis O. - M. Braun

I N H A L T S V E R Z E I C H N I S

1. Hinweise zur Benutzung der Segment-Umschalteroutinen

Die Segment-Umschalteroutinen dienen dem Programmierer als Hilfsmittel, das ihm erlaubt, Unterprogramme in anderen 64K-Segmenten (Bank) als dem gerade benutzten zu verwenden. Es werden dabei alle Register und Statusinformationen an das Unterprogramm übergeben. Nach der Rückkehr stehen die nun gültigen Registerwerte und Statusinformationen dem aufrufenden Programm zur Verfügung. Es besteht außerdem die Möglichkeit einer Segment-Umschaltung zum Segment $F (Betriebssystem Bank 15) im Falle eines Interrupts (IRQ), eines nicht maskierten Interrupts (NMI) und eines Break (BRK), wenn gerade ein Programm in einem anderen Segment ausgeführt wird.

Die Segment-Umschalteroutine besteht aus dem fest im Betriebssystem-ROM (Segment $F = Operating-System) abgelegten Teil und den Teilen in den jeweils benutzten RAM-Segmenten. Die Routinen in den RAM-Segmenten sind weitgehend identisch mit den ROM-Routinen; mit Ausnahme der Interruptbehandlung. Der Interrupt wird durch die Interrupt-Routinen in Segment $F bearbeitet, während die Interrupt-Routinen im RAM nur die Umschaltung auf das Segment $F vornehmen.

Die folgenden Beispielprogramme zeigen, wie Betriebsystem-Routinen von einem RAM Segment aufgerufen werden können. Bei der Programmierung in Assembler sollte das später gelistete Sourcecode-Modul 'TXRAM' an das Ende eines jeden Programmes gelegt werden. Es können dann die Betriebssystem-Routinen und -Vektoren verwendet werden.

Zur Umschaltung ist ein kurzes Programm in das RAM des Segmentes $F zu laden (Beispiel1 o. Beispiel2). Es initialisiert den Stack-Pointer im RAM- Segment und übergibt die Programmkontrolle an das gewünschte Segment. Beim Übergang der Programmkontrolle von einem Segment an das andere, wird der aktuelle Stack-Pointer in die Speicherstelle '$01FF' gerettet. Es muß deshalb bei der Initialisierung des Stack-Pointers eines jeden RAM-Segmentes der Wert '$FE' in die Speicherstelle '$01FF' geschrieben werden, um diese Speicherstelle zu schützen.

Die Nummer des Segmentes, an das die Programmkontrolle abgegeben wird, wird durch den Inhalt des 6509 Data-Indirection-Registers ($0001) bestimmt. Die obersten zwei Bytes auf dem Stack beinhalten die Adresse innerhalb des Segments. Dadurch ist es nach entsprechendem Setzen des Indirection-Registers möglich, das Unterprogramm an der gleichen Adresse aufzurufen, an der es sich im gewünschten Segment befindet. Das Unterprogramm im aktuellen Segment ruft die Unterroutine 'EXSUB' als Unterprogramm auf. 'EXSUB' benutzt die Rücksprung-Adresse abzgl. 2 als Sprung-Adresse im neuen Segment.

Das folgende 'Beispiel1' und die Betriebssystem-Aufrufe benutzen diese Methode zum Aufruf der Routinen im anderen Segment. Es ist dies die günstigste Methode. Sie setzt jedoch voraus, daß sich die aufrufende Routine an der gleichen Adresse befindet wie die Aufzurufende. Befinden sich die Routinen nicht an der gleichen Stelle, dann wird die Adresse der Routine +2 auf den Stack gelegt und danach die Routine 'EXSUB' direkt angesprungen, siehe 'Beispiel2'.

Beim Aufruf von Maschinen-Unterprogrammen von BASIC aus ist zu beachten, daß sich keine Speicherüberschneidungen ergeben. Es werden in den RAM-Segmenten zusätzlich zu dem von Daten und Programmen beanspruchten Speicherplatz die Seiten '$00XX, $01XX, $FEXX, $FFXX' benutzt. Zur Bereitstellung von RAM-Bereichen dienen die Routinen 'MEMTOP' und 'MEMBOT' (siehe Listing Operating-System).

```
segment-umsch. beispiel1   hard+soft ass/reass.....seite 0002

zeile adr.  obj.-code   source-code


0002  0000
0003  0000  ===> beispiel1 <===
0004  0000
0005  0000  dieses programm wird in das segment $f = bank 15 geschrieben.
0006  0000  im beispiel soll ein anwenderprogramm im segment $01 = bank 1
0007  0000  initialisiert und aufgerufen werden. das programm muß sich
0008  0000  an der gleichen stelle wie der aufruf befinden (adresse $078b)
0009  0000
0010  0000
0011  0000
0012  0000            i6509  = $0001          6509 indirection register
0013  0000            ipoint = $00ac          addrl ram indirekt-pointer
0014  0000            ipinit = $ff19          'ipoint' auf $0100, yr=$ff
0015  0000            exsub  = $feb2          execute-subr. auf vektor 'ipoint'
0016  0000
0017  0000
0018  0000                   * = $0780        anfangsadresse
0019  0780
0020  0780  a9 01            lda #$01         segmentnummer 1
0021  0782  85 01            sta i6509        6509 indirection register
0022  0784  20 19 ff         jsr ipinit       'ipoint' auf $0100, yr=$ff
0023  0787  a9 fe            lda #$fe         wert für stack-pointer
0024  0789  91 ac            sta (ipoint),y   $01ff = $fe
0025  078b  20 b2 fe         jsr exsub        umsch. zu dieser adresse in bank 1
0026  078e
0027  078e                   .end



fehler in pass 1 = 0000
fehler in pass 2 = 0000




 label-tabelle

 label    adr.

 exsub    feb2    i6509    0001    ipinit   ff19    ipoint   00ac


assemblierung ende
```

```
segment-umsch. beispiel2   hard+soft ass/reass.....seite 0002

zeile adr.  obj.-code   source-code


0002  0000
0003  0000  ===> beispiel2 <===
0004  0000
0005  0000  dieses programm wird in das segment $f = bank 15 geschrieben.
0006  0000  im beispiel soll ein anwenderprogramm im segment $01 = bank 1
0007  0000  inititialisiert und aufgerufen werden. das programm befindet sich
0008  0000  an einer beliebigen adresse (im beispiel bei 'start' = $078b).
0009  0000
0010  0000
0011  0000            i6509  = $0001          6509 indirection register
0012  0000            ipoint = $00ac          addrl ram indirekt-pointer
0013  0000            start  = $0200          startadresse in bank 1
0014  0000            starta = start+2        startadresse+2 für stack
0015  0000            ipinit = $ff19          'ipoint' auf $0100, yr=$ff
0016  0000            exsub  = $feb2          execute-subr. auf vektor 'ipoint'
0017  0000
0018  0000
0019  0000                   * = $0780        anfangsadresse
0020  0780
0021  0780  a9 01            lda #$01        segmentnummer 1
0022  0782  85 01            sta i6509       6509 indirection register
0023  0784  a0 01            ldy #$01        'ipoint' auf $0100
0024  0786  84 ad            sty ipoint+1    addrh ram indirekt-pointer
0025  0788  88               dey
0026  0789  84 ac            sty ipoint      addrl ram indirekt-pointer
0027  078b  88               dey             yr = $ff
0028  078c  a9 fe            lda #$fe        wert für stack-pointer
0029  078e  91 ac            sta (ipoint),y  $01ff = $fe
0030  0790  a9 02            lda #>starta    addrh startadresse+2
0031  0792  48               pha
0032  0793  a9 02            lda #<starta    addrl startadresse+2
0033  0795  48               pha
0034  0796  6c b2 fe         jmp exsub       umsch. zur stack-adresse in bank 1
0035  0799
0036  0799                   .end


fehler in pass 1 = 0000
fehler in pass 2 = 0000




label-tabelle

label    adr.

exsub    feb2    i6509    0001    ipinit   ff19    ipoint   00ac
start    0200    starta   0202

assemblierung ende
```

```
zeile adr.  obj.-code   source-code


0002  0000
0003  0000  ===> testprogramm für beispiel1 u. beispiel2 mit 'txram' <===
0004  0000
0005  0000  dieses programm befindet sich im segment $01 = bank 1
0006  0000  es wird durch 'beispiel1 oder 2' aus bank 15 angesprungen
0007  0000  nach ausführung erfolgt der ausdruck 'testumschaltung'
0008  0000  für 'beispiel2' ist die startadresse beliebig wählbar
0009  0000
0010  0000  an das eigentliche programm schließt die routine 'txram' an
0011  0000  diese routine ist hauptsächlich eine kopie aus dem 'kernal'
0012  0000
0013  0000
0014  0000
0015  0000  ===> ausführendes beispiel-programm 'start' <===
0016  0000
0017  0000                    * = $078b         ansprungadresse von segment $15
                                                (wichtig für 'beispiel1')
0018  078b
0019  078b  a0 00     start  ldy #$00
0020  078d  b9 a2 07  start1 lda text,y        ein zeichen aus text holen
0021  0790  20 d2 ff         jsr kbsout        ausgabe 1 char auf akt. kanal
0022  0793  c8               iny               offset erhöhen
0023  0794  c0 0e            cpy #tex-text     ende der zeichenkette ?
0024  0796  d0 f5            bne start1        nicht alle zeichen
0025  0798
0026  0798
0027  0798  20 e1 ff         jsr kstop         stop-taste lesen
0028  079b  d0 ee            bne start         nicht gedrückt, neuer textdruck
0029  079d  00               brk               ende mit break
0030  079e  ea               nop
0031  079f  4c 8b 07         jmp start         wenn weiter, neustart
0032  07a2
0033  07a2
0034  07a2  55 4d 53  text   .byt 'umschaltetest '
0034  07a5  43 48 41
0034  07a8  4c 54 45
0034  07ab  54 45 53
0034  07ae  54 20
0035  07b0            tex
```

```
zeile adr.  obj.-code   source-code


0037  07b0
0038  07b0  ===> anhang-routine 'txram' <===
0039  07b0
0040  07b0            e6509  = $0000          6509 execution register
0041  07b0            i6509  = $0001          6509 indirection register
0042  07b0            ipoint = $00ac          addrl ram indirekt-pointer
0043  07b0            stackp = $01ff          wert für stack-pointer
0044  07b0            exsub  = $feb2          execute-subr. auf vektor 'ipoint'
0045  07b0
0046  07b0                   * = exsub-78
0047  fe64
0048  fe64
0049  fe64  ---> ansprung irq/brk vom ram-segment <---
0050  fe64
0051  fe64  85 ac     txirq  sta ipoint       addrl ram indirekt-pointer
0052  fe66  68               pla              stackpointer in ac
0053  fe67  48               pha              zurück für 'rti'
0054  fe68  85 ad            sta ipoint+1     addrh ram indirekt-pointer
0055  fe6a  a5 01            lda i6509        6509 indirektion register
0056  fe6c  48               pha
0057  fe6d  a5 ad            lda ipoint+1     stackpointer in ac
0058  fe6f  20 99 fe         jsr txirq2       diese rts-adr. speichern
0059  fe72  85 ac            sta ipoint
0060  fe74  68               pla              i6509 wiederherstellen
0061  fe75  85 01            sta i6509        6509 indirektion register
0062  fe77  a5 ac            lda ipoint       ac wieder zurückholen
0063  fe79  40               rti              ausführung rti
0064  fe7a
0065  fe7a
0066  fe7a  ---> ansprung nmi vom ram-segment <---
0067  fe7a
0068  fe7a  48        txnmi  pha              ac retten
0069  fe7b  a5 ac            lda ipoint       addrl ram indirekt-vektor
0070  fe7d  48               pha              ipoint retten
0071  fe7e  a5 ad            lda ipoint+1     addrh ram indirekt-vektor
0072  fe80  48               pha              ipoint+1 retten
0073  fe81  ad ff 01         lda stackp       stackpointer
0074  fe84  48               pha              stackpointer retten
0075  fe85  a5 01            lda i6509        6509 indirektion register
0076  fe87  48               pha              i6509 retten
0077  fe88  20 35 ff         jsr exnmi        aufruf nmi als subr. in bank 15
0078  fe8b  68               pla              i6509 zurück
0079  fe8c  85 01            sta i6509        6509 indirektion register
0080  fe8e  68               pla              stackpointer zurück
0081  fe8f  8d ff 01         sta stackp       stackpointer
0082  fe92  68               pla              ipoint+1 zurück
0083  fe93  85 ad            sta ipoint+1     addrh ram indirekt-vektor
0084  fe95  68               pla              ipoint zurück
0085  fe96  85 ac            sta ipoint       addrl ram indirekt-vektor
0086  fe98  40               rti
0087  fe99
0088  fe99
0089  fe99  ---> 'rts' - adresse 'irq' brk' <---
0090  fe99
0091  fe99  29 10     txirq2 and #$10         test auf break
0092  fe9b  d0 05            bne exbrkx       sprung zu break
0093  fe9d
0094  fe9d  ---> ausführung 'irq' in segment $0f <---
```

```
zeile adr.  obj.-code   source-code


0095  fe9d
0096  fe9d  a5 ac            lda ipoint       addrl ram indirekt-vektor
0097  fe9f  4c 3c ff         jmp exirq        ausführung 'irq'
0098  fea2
0099  fea2  ---> ausfürung 'break' in segment $0f <---
0100  fea2
0101  fea2  a5 ac     exbrkx lda ipoint       addrl ram indirekt-vektor
0102  fea4  4c 39 ff         jmp exbrk        ausführung 'brk'
0103  fea7
0104  fea7
0105  fea7  ===> ausführung externe subroutine in voreingestellter bank <===
0106  fea7
0107  fea7  08        exsubf php              status retten
0108  fea8  48               pha              ac retten
0109  fea9  a9 0f            lda #$0f         segment $f = betriebssytem
0110  feab  85 01            sta i6509        6509 indirection register
0111  fead  68               pla              ac zurück
0112  feae  28               plp              status zurück
0113  feaf  4c b2 fe         jmp exsub        externe subroutine
0114  feb2
0115  feb2
0116  feb2  ===> externe subroutine <===
0117  feb2
0118  feb2                   * = exsub
0119  feb2
0120  feb2  08               php              status retten
0121  feb3  78               sei
0122  feb4  48               pha              ac retten
0123  feb5  8a               txa
0124  feb6  48               pha              xr retten
0125  feb7  98               tya
0126  feb8  48               pha              yr retten
0127  feb9  20 19 ff         jsr ipinit       'ipoint' init., stack vom xfer-seg.
                                              laden
0128  febc  a8               tay              yr = xfer-segment
0129  febd  a5 00            lda e6509        return-segment auf user-stack
0130  febf  20 2a ff         jsr putas        ac auf anderen stack
0131  fec2  a9 04            lda #<excrt2     xfer-seg. rts routine
0132  fec4  a2 ff            ldx #>excrt2     xfer-seg. rts routine
0133  fec6  20 24 ff         jsr putaxs       ac, xr auf xfer-seg. stack
0134  fec9  ba               tsx
0135  feca  bd 05 01         lda $0105,x      sp+5 = aktuelle rout. addrl
0136  fecd  38               sec
0137  fece  e9 03            sbc #3           -3 für 'jsr' dieser rout.
0138  fed0  48               pha              ac retten
0139  fed1  bd 06 01         lda $0106,x      sp+6 = aktuelle rout. addrh
0140  fed4  e9 00            sbc #00
0141  fed6  aa               tax              xr = addrh
0142  fed7  68               pla              ac = addrl
0143  fed8  20 24 ff         jsr putaxs       ac, xr auf xfer-seg. stack
0144  fedb  98               tya              xfer-seg. stack-pointer
0145  fedc
0146  fedc  38        excomm sec
0147  fedd  e9 04            sbc #04          4 bytes, yr, xr, ac, sp
0148  fedf  8d ff 01         sta stackp       xfer-seg. neuer stackpointer
0149  fee2  a8               tay
0150  fee3  a2 04            ldx #04          4 bytes, yr, xr, ac, sp
0151  fee5
```

```
zeile adr.  obj.-code   source-code

0152  fee5  68        exsu10 pla
0153  fee6  c8               iny
0154  fee7  91 ac            sta (ipoint),y   reg. von stack zu xfer-seg. stack
0155  fee9  ca               dex
0156  feea  d0 f9            bne exsu10
0157  feec  ac ff 01         ldy stackp       yr = stackpointer für xfer-seg.
0158  feef  a9 2d            lda #<expul2     addrl register u. rts-adr. vom stack
0159  fef1  a2 ff            ldx #>expul2     addrh register u. rts-adr. vom stack
0160  fef3  20 24 ff         jsr putaxs       ac, xr auf xfer-seg. stack
0161  fef6  68               pla
0162  fef7  68               pla
0163  fef8
0164  fef8  ba        exgbye tsx
0165  fef9  8e ff 01         stx stackp       lfd. stack-pointer von diesem
                                              segment
0166  fefc  98               tya              yr = stack-pointer für xfer-seg.
0167  fefd  aa               tax
0168  fefe  9a               txs              neuer stack für xfer-seg.
0169  feff  a5 01            lda i6509        6509 indirection register
0170  ff01  4c f6 ff         jmp gbye         'good bye'
0171  ff04
0172  ff04
0173  ff04  ---> rückkehradresse wenn 'rti' <---
0174  ff04
0175  ff04  ea               nop
0176  ff05
0177  ff05  ---> rückkehradresse wenn 'rts' <---
0178  ff05
0179  ff05  08        excrts php              status
0180  ff06  08               php
0181  ff07  78               sei
0182  ff08  48               pha              ac
0183  ff09  8a               txa
0184  ff0a  48               pha              xr
0185  ff0b  98               tya
0186  ff0c  48               pha              yr
0187  ff0d  ba               tsx
0188  ff0e  bd 06 01         lda $0106,x      sp+6 ist return-seg.
0189  ff11  85 01            sta i6509        6509 indirection register
0190  ff13  20 19 ff         jsr ipinit       'ipoint' init, stack vom xfer-seg.
0191  ff16  4c dc fe         jmp excomm
0192  ff19
0193  ff19
0194  ff19  ---> 'ipoint' initialisieren, stack vom xfer-seg. laden <---
0195  ff19
0196  ff19  a0 01     ipinit ldy #01
0197  ff1b  84 ad            sty ipoint+1     addrh ram indirekt-vektor
0198  ff1d  88               dey
0199  ff1e  84 ac            sty ipoint       addrl ram indirekt-vektor
0200  ff20  88               dey              yr = $ff
0201  ff21  b1 ac            lda (ipoint),y   stack-pointer von $01ff laden
0202  ff23  60               rts
0203  ff24
0204  ff24
0205  ff24  ---> ac, xr zu xfer-segment stack <---
0206  ff24
0207  ff24  48        putaxs pha              ac retten
0208  ff25  8a               txa
```

```
zeile adr.  obj.-code   source-code

0209  ff26  91 ac            sta (ipoint),y   addrh = xr
0210  ff28  88               dey
0211  ff29  68               pla              ac zurück
0212  ff2a
0213  ff2a  ---> ac auf anderen stack <---
0214  ff2a
0215  ff2a  91 ac     putas   sta (ipoint),y   addrl = ac
0216  ff2c  88               dey
0217  ff2d  60               rts
0218  ff2e
0219  ff2e
0220  ff2e  ---> register zurück <---
0221  ff2e
0222  ff2e  68         expull pla
0223  ff2f  a8               tay              yr zurück
0224  ff30  68               pla
0225  ff31  aa               tax              xr zurück
0226  ff32  68               pla              ac zurück
0227  ff33  28               plp              status zurück
0228  ff34  60               rts
0229  ff35
0230  ff35
0231  ff35  ---> externe interrupt 'nmi' 'brk' 'irq' segment $0f <---
0232  ff35
0233  ff35  20 a7 fe  exnmi  jsr exsubf       ext. nmi in segment $0f
0234  ff38
0235  ff38  ea               nop
0236  ff39
0237  ff39  20 a7 fe  exbrk  jsr exsubf       ext. brk in segment $0f
0238  ff3c
0239  ff3c  20 a7 fe  exirq  jsr exsubf       ext. irq in segment $0f
0240  ff3f
0241  ff3f
0242  ff3f            excrt2 = excrts-1
0243  ff3f            expul2 = expull-1
0244  ff3f
0245  ff3f
0246  ff3f
0247  ff3f  ===> vektor - tabelle <===
0248  ff3f
0249  ff3f                   * = $ff6f
0250  ff6f
0251  ff6f  20 a7 fe  kvrese jsr exsubf       netz ein/aus reset-vector
0252  ff72
0253  ff72  20 a7 fe  kipcgo jsr exsubf       mon-befehl 'z' umsch. auf
                                               coprozessor
0254  ff75
0255  ff75  20 a7 fe  kfunky jsr exsubf       vector für funktionstasten
0256  ff78
0257  ff78  20 a7 fe  kipcrq jsr exsubf       vector für ipc-anforderung
0258  ff7b
0259  ff7b  20 a7 fe  kioini jsr exsubf       initialisierung ein-/ausgabe
0260  ff7e
0261  ff7e  20 a7 fe  kcint  jsr exsubf       initialisierung bildschirm
0262  ff81
0263  ff81  20 a7 fe  kalloc jsr exsubf       allocation-routine
0264  ff84
0265  ff84  20 a7 fe  kvecto jsr exsubf       ein-/ausgabe-vector lesen/schreiben
```

```
zeile adr.  obj.-code   source-code

0266  ff87
0267  ff87  20 a7 fe  kresto jsr exsubf       standard ein-/ausgabe-vectoren
                                              setzen
0268  ff8a
0269  ff8a  20 a7 fe  klkups jsr exsubf       geräteadr. über sekundäradr. suchen
0270  ff8d
0271  ff8d  20 a7 fe  klkupl jsr exsubf       geräte-,sek.adr. über log. file#
                                              suchen
0272  ff90
0273  ff90  20 a7 fe  kstmsg jsr exsubf       meldung des operating-syst. ausgeben
0274  ff93
0275  ff93  20 a7 fe  ksecnd jsr exsubf       sekundäradr. nach listen auf iec-bus
0276  ff96
0277  ff96  20 a7 fe  ktksa  jsr exsubf       sekundäradr. nach talk auf iec-bus
0278  ff99
0279  ff99  20 a7 fe  kmemtp jsr exsubf       höchste speichergrenze
                                              lesen/schreiben
0280  ff9c
0281  ff9c  20 a7 fe  kmembt jsr exsubf       unterste speichergrenze
                                              lesen/schreiben
0282  ff9f
0283  ff9f  20 a7 fe  kscnky jsr exsubf       tastatur abfragen
0284  ffa2
0285  ffa2  20 a7 fe  ksetmo jsr exsubf       zeitüberwachung iec-bus ein
0286  ffa5
0287  ffa5  20 a7 fe  kacptr jsr exsubf       ein byte von iec-bus nach accu
0288  ffa8
0289  ffa8  20 a7 fe  kciout jsr exsubf       ein byte aus accu auf iec-bus
0290  ffab
0291  ffab  20 a7 fe  kuntlk jsr exsubf       untalk auf iec-bus ausgeben
0292  ffae
0293  ffae  20 a7 fe  kunlsn jsr exsubf       unlisten auf iec-bus ausgeben
0294  ffb1
0295  ffb1  20 a7 fe  klistn jsr exsubf       listen auf iec-bus ausgeben
0296  ffb4
0297  ffb4  20 a7 fe  ktalk  jsr exsubf       talk auf iec-bus ausgeben
0298  ffb7
0299  ffb7  20 a7 fe  kreast jsr exsubf       ein/ausgabe-status lesen / schreiben
0300  ffba
0301  ffba  20 a7 fe  kstlfs jsr exsubf       log. file#, geräte-, sek.adr.
                                              eintragen
0302  ffbd
0303  ffbd  20 a7 fe  kstnam jsr exsubf       länge und adr. des filenamens
                                              eintragen
0304  ffc0
0305  ffc0  20 a7 fe  kopen  jsr exsubf       log. file öffnen/befehl ausgeben
0306  ffc3
0307  ffc3  20 a7 fe  kclose jsr exsubf       logisches file schliessen
0308  ffc6
0309  ffc6  20 a7 fe  kchkin jsr exsubf       eingabekanal öffnen
0310  ffc9
0311  ffc9  20 a7 fe  kckout jsr exsubf       ausgabekanal öffnen
0312  ffcc
0313  ffcc  20 a7 fe  kclrch jsr exsubf       ein-/ausgabekanal schliessen
0314  ffcf
0315  ffcf  20 a7 fe  kbasin jsr exsubf       eingabe 1 char vom akt. kanal in ac
0316  ffd2
0317  ffd2  20 a7 fe  kbsout jsr exsubf       ausgabe 1 char auf akt. kanal
```

```
zeile adr.  obj.-code   source-code


0318  ffd5
0319  ffd5  20 a7 fe  kload  jsr exsubf       einlesen vom log. file
0320  ffd8
0321  ffd8  20 a7 fe  ksave  jsr exsubf       abspeichern auf log. file
0322  ffdb
0323  ffdb  20 a7 fe  ksttim jsr exsubf       interne uhr stellen
0324  ffde
0325  ffde  20 a7 fe  krdtim jsr exsubf       interne uhr ablesen
0326  ffe1
0327  ffe1  20 a7 fe  kstop  jsr exsubf       stop-taste lesen
0328  ffe4
0329  ffe4  20 a7 fe  kgetin jsr exsubf       eingabe 1 char vom aktiven kanal in
                                              ac
0330  ffe7
0331  ffe7  20 a7 fe  kclall jsr exsubf       alle logischen files schliessen
0332  ffea
0333  ffea  20 a7 fe  kudtim jsr exsubf       interne uhr bedienen
0334  ffed
0335  ffed  20 a7 fe  kscror jsr exsubf       bildschirm
0336  fff0
0337  fff0  20 a7 fe  kplot  jsr exsubf       cursor-koordinaten lesen/schreiben
0338  fff3
0339  fff3  20 a7 fe  iobase jsr exsubf       basisadr. i/o-gerät nach xr/yr
0340  fff6
0341  fff6
0342  fff6  ---> good-bye <---
0343  fff6
0344  fff6  85 00     gbye   sta e6509        execution register
0345  fff8  60               rts
0346  fff9
0347  fff9  ea               nop
0348  fffa
0349  fffa
0350  fffa  ---> 6509 hardware-vektoren <---
0351  fffa
0352  fffa  7a fe     hanmi  .word txnmi      ind. sprung nmi-vector (von $fffa)
0353  fffc
0354  fffc  8b 07     hares  .word start      system reset routine
0355  fffe
0356  fffe  64 fe     hairq  .word txirq      interrupt-routine (irq)
```

label-tabelle

label	adr.						
e6509	0000	exbrk	ff39	exbrkx	fea2	excomm	fedc
excrt2	ff04	excrts	ff05	exgbye	fef8	exirq	ff3c
exnmi	ff35	expul2	ff2d	expull	ff2e	exsu10	fee5
exsub	feb2	exsubf	fea7	gbye	fff6	hairq	fffe
hanmi	fffa	hares	fffc	i6509	0001	iobase	fff3
ipinit	ff19	ipoint	00ac	kacptr	ffa5	kalloc	ff81
kbasin	ffcf	kbsout	ffd2	kchkin	ffc6	kcint	ff7e
kciout	ffa8	kckout	ffc9	kclall	ffe7	kclose	ffc3
kclrch	ffcc	kfunky	ff75	kgetin	ffe4	kioini	ff7b
kipcgo	ff72	kipcrq	ff78	klistn	ffb1	klkupl	ff8d
klkups	ff8a	kload	ffd5	kmembt	ff9c	kmemtp	ff99
kopen	ffc0	kplot	fff0	krdtim	ffde	kreast	ffb7
kresto	ff87	ksave	ffd8	kscnky	ff9f	kscror	ffed
ksecnd	ff93	ksetmo	ffa2	kstlfs	ffba	kstmsg	ff90
kstnam	ffbd	kstop	ffe1	ksttim	ffdb	ktalk	ffb4
ktksa	ff96	kudtim	ffea	kunlsn	ffae	kuntlk	ffab
kvecto	ff84	kvrese	ff6f	putas	ff2a	putaxs	ff24
stackp	01ff	start	078b	start1	078d	tex	07b0
text	07a2	txirq	fe64	txirq2	fe99	txnmi	fe7a

assemblierung ende

2. Beschreibung der RS 232 C - Schnittstelle

2.1 Allgemeines

Die Rechner CBM 600/700 sind u.a. standardmäßig mit einer Schnittstelle ausgerüstet, die unter den Namen 'RS 232 C' bzw. 'V.24' bekannt ist. Diese Bezeichnungen haben ihren Ursprung in der Nomenklatur der Institute, welche diese Schnittstellen normten. Im amerikanischen Sprachraum ist dies die 'EIA' (Electronic Industries Association) für 'RS 232', in Europa ist es das 'CCITT' (Commitee Consultative Internationale de Telephone et Telegraphe) für V.24. In der BRD sind die Empfehlungen deser Schnittstelle in der DIN 66020 genormt.

Die Schnittstelle arbeitet asynchron und seriell. Seriell bedeutet, daß alle Informationen Bit für Bit übertragen werden. Asynchron bedeutet hierbei, daß die Übertragungsleitung, wenn keine Information zu übermitteln ist, auf einem bestimmten logischen Pegel gehalten wird; ganz im Gegensatz zur synchronen Datenübertragung, bei der ständig ein Füllbyte auf der Leitung gesendet wird.

Im seriellen Datenformat wird bei der Übertragung eines Byte zuerst ein Start-Bit gesendet, dann die Informations-Bits (LSB zuerst), eventuell ein Paritäts-Bit und ein oder zwei Stop-Bits. Da die Datenleitung im Grundzustand (d.h. wenn keine Information zu übertagen ist) im logischen Zustand 'high' ist, muß das Start-Bit ein logisches 'low' sein. An diesem Wechsel des Leitungspegels erkennt der Empfänger den Anfang einer Informationseinheit und synchronisiert sich selbst darauf. Der Zustand eines Stop-Bit ist logisch 'high', also gleich dem Grundzustand der Übertragungsleitung.

Die besprochene Schnittstelle wird benutzt, um Peripheriegeräte zu bedienen oder zur Kommunikation mit anderen Rechnern über Modem. Im Hinblick auf die Datenfernübertragung mittels Modem ist die Schnittstelle so ausgelegt, daß ein Rechner CBM 600/700 als Datenterminal (Daten-Endeinrichtung) zu interpretieren ist.

2.2 BASIC 4.0 und RS 232 C

Die RS 232 C-Schnittstelle kann mit den normalen BASIC 4.0 Befehlen angesprochen werden; lediglich beim Eröffnen diesen Datenkanals sind einige Spezifikationen zu beachten. Es ist damit ein komfortables Handhaben gewährleistet.

Das Eröffnen geschieht mit dem BASIC-Befehl: 'open lu,pa,sa,fp'

- lu = Logical unit, logische Adresse
- pa = Primäradresse - für die RS 232 C-Schnittstelle = '2'
- sa = Sekundäradresse - gibt die Übertragungsrichtung an
- fp = File-Parameter - ein vier Byte langer Character-String, er enthält Angaben über Übertragungsgeschwindigkeit, Parität und Wortlänge

2.2.1 Sekundäradresse 'sa'

Die Sekundäradresse ist entsprechend der folgenden Tabelle anzugeben:

sa = 1: Rechner sendet nur, kein Datenempfang (z.B. Druckerausgabe)
sa = 2: Rechner empfängt nur, kein Senden (z.B. Meßwerterfassung)
sa = 3: Rechner sendet und empfängt Daten (z.B. Rechnerkommunikation)

Da der Zeichencode der CBM-Rechner (CBM-ASCII) in einigen Punkten von der ASCII-Norm abweicht, ist aus Kompatibilitätsgründen mit anderen Systemen vorgesehen, eine Code-Wandlung vorzunehmen. Bei der Ausgabe wird dadurch das CBM-ASCII in Standard-ASCII umgewandelt, beim Empfang entsprechend von Standard-ASCII in CBM-ASCII. Wird diese Konvertierung gewünscht, so ist der Wert der Sekundäradresse um 128 zu erhöhen. So ist z.B. als 'sa' bei bidirektionalem Datenverkehr mit Code-Wandlung der Wert '131' (3+128) anzugeben.

2.2.2 File-Parameter 'fp'

Die File-Parameter bestehen aus einem 4 Byte langen Character-String. Wichtig sind die ersten beiden Bytes, die letzten beiden spielen eine untergeordnete Rolle, müssen jedoch angegeben werden. Zunächst die Bedeutung des ersten Byte, mit dem stärksten Informationsgehalt. Es sind dort die Angaben über die Übertragungsgeschwindigkeit, Anzahl der Stop-Bits, sowie die Wortlänge enthalten. Den folgenden drei Tabellen ist jeweils der betreffende Wert zu entnehmen und diese drei Werte zu addieren. Der so ermittelte Wert stellt das erste Byte des Character-Strings dar.

1.Byte (fp) Tabelle Übertragungsgeschwindigkeit

Übertragungsgeschw.	Wert	
16 * Takt extern	0	(nur bei Empfang zulässig)
50 Baud	17	
75 Baud	18	
109,92 Baud	19	
134.58 Baud	20	
150 Baud	21	
300 Baud	22	
600 Baud	23	
1200 Baud	24	
1800 Baud	25	
2400 Baud	26	
3600 Baud	27	
4800 Baud	28	
7200 Baud	29	
9600 Baud	30	
19200 Baud	31	

1.Byte (fp) Tabelle Wortlänge

Wortlänge	Wert
8 Bit	0
7 Bit	32
6 Bit	64
5 Bit	96

1.Byte (fp) Tabelle Stop-Bits

Anzahl Stop-Bits	Wert
1	0
1 (falls Wortlänge = 8 mit Parität	128
1,5 (nur bei Wortlänge = 5 ohne Parität	128
2	128

2.Byte (fp)

Das zweite Byte baut sich nach einem ähnlichen Muster auf. Es wird aus zwei Tabellen errechnet. Um eine vollständige Leistungsbeschreibung zu bieten, wären fünf Tabellen notwendig. Die letzten drei sind nur für Sonderfälle vorgesehen. Es ist dazu ein umfassendes Verständnis des Betriebssystems, der Hardware des Rechners und der Datenübertragung Voraussetzung. Die Tabellen sind in der Folge der Vollständigkeit wegen aufgeführt; der BASIC-Benutzer sollte sie aber ignorieren. Der Wert des 2.Bytes ermittelt sich demnach aus der Summe jeweils eines Wertes aus den beiden folgenden Tabellen:

2.Byte (fp) Tabelle Modus

Modus	Wert	
Normal	0	
Echo	16	(nur Betriebsart Empfang)

Der Modus 'Echo' bewirkt, daß die Daten, die der Rechner empfängt, sofort wieder an den Sender zurückgesendet werden, um dort die richtige Datenübermittlung zu kontrollieren (Duplexbetrieb).

2.Byte (fp) Tabelle Parität

Parität		Wert
Empfangen	Senden	
keine	keine	0
ungerade	ungerade	32
gerade	gerade	96
keine	mark (log. 1)	160
keine	space (log. 0)	224

2.Byte (fp) Tabelle Data Terminal Ready (unwichtig)

Zustand		Wert	
Blockieren	Senden/Empfangen	0	(Default)
Freigeben	Senden/Empfangen	1	

2.Byte (fp) Tabelle Interrupt freigeben (unwichtig)

Interrupt	Wert	
Freigeben IRQ (Bit 0 Status)	0	(Default)
Sperren IRQ	1	

2.Byte (fp) Tabelle Sendekontrolle (unwichtig)

Sendeunterbrechung	(not RTS)	Zusatz	Wert	
Blockiert	high	-	0	(Default)
Freigegeben	low	-	4	
Blockiert	low	-	8	
Blockiert	low	Sende:BRK	16	

2.3 Statusvariable

Zur Kontrolle der Datenübertragung steht die Systemvariable 'ST' zur Verfügung. Der Inhalt dieser Variablen hat folgende Bedeutung:

Status	Bedeutung
0	alles in Ordnung
1	Paritätsfehler (nur beim Lesen !)
2	Formatfehler (nur beim Lesen, z.B. falsche Wortlänge)
4	Datenverlust (nur beim Lesen, die Daten werden schneller angeliefert, als sie ausgelesen wurden
16	Eingabepuffer leer (nur beim Lesen, beim letzten get# war noch kein gültiges Zeichen im Puffer
32	Fehler bei 'Data Carrier Detect'
64	Fehler bei 'Data Set Ready'

Unter Umständen können mehrere Fehler gleichzeitig auftreten. In diesem Fall sind die einzelnen Fehlermeldungen auszufiltern (Bitweises zerlegen der Statusvariablen).

Die nun richtig eröffnete RS 232 C-Schnittstelle kann mit den normalen BASIC-Befehlen gehandhabt werden (get#, print#, input#). Der Befehl 'input#' sollte jedoch mit Vorsicht verwendet werden, da dieser Befehl am Ende der Zeicheneingabe ein 'Carriage Return = chr$(13)' erwartet. Trifft dieses Zeichen nicht ein, so bleibt der Rechner in der Input-Schleife hängen. Man sollte diesen Befehl nur dann verwenden, wenn das Eintreffen des 'Carriage Return' mit Sicherheit gewährleistet ist.

ACHTUNG:

Beim ersten Auslesen der Variablen 'ST' wird ihr Wert auf null zurückgesetzt. Soll die Statusvariable später noch verwendet werden, so ist ihr Wert einer anderen Variablen zuzuweisen (kopieren).

2.4 Hardware RS 232 C

Die RS 232 C-Schnittstelle wird von dem Baustein 'ACIA 6551' = Asynchronous Communication Interface Adapter bedient. In diesem IC ist u.a. ein Baudratengenerator integriert, welcher Übertragungsraten von 50 bis 19.200 Baud ermöglicht. Bei Verwendung eines externen Taktes können in der Betriebsart 'Empfangen' nicht standardisierte Übertragungsgeschw. bis zu 125.000 Baud erreicht werden.
Der Baustein besitzt neben seinen internen Registern fünf weitere, von außen erreichbare, Register. Es sind dies:

Register	Adresse hex	Adresse dez.
Receiver - Register	$DD00	56576
Transmitter - Register	$DD00	56576
Status - Register	$DD01	56577
Control - Register	$DD02	56578
Command - Register	$DD03	56579

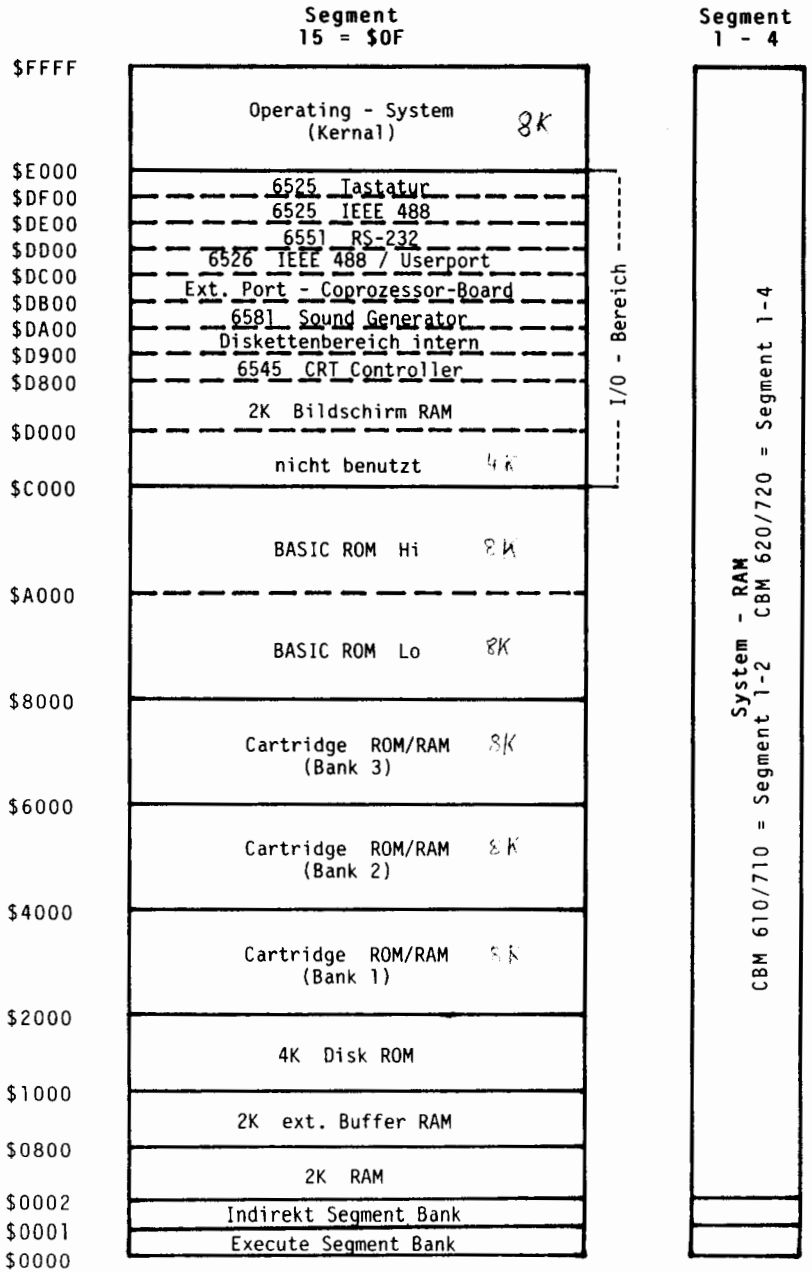
Über das Receiver-Register werden Daten eingelesen (lesender Zugriff), über das Transmitter-Register werden Daten gesendet (schreibender Zugriff). Das Status-Register übermittelt dem Benutzer durch Setzen der einzelnen Bits den Status des 6551-Bausteins. Im Control-Register ist der gewünschte Modus, wie Baud-Rate, Stop-Bit, Wortlänge sowie externer Takt, durch Setzen oder Löschen der einzelnen Bits, festgelegt. Im Command-Register werden die gewünschten Übertragungsmodes, wie Parität, Normal und Echo ect. festgelegt. Für eine ausführliche Beschreibung entnehmen Sie bitte die Daten dem '6551 Datenblatt'.
An der RS 232 C-Schnittstelle werden 10 Anschlüsse ausgewertet. Die Verbindung Rechner/Peripherie erfolgt über eine 25-polige Cannon-Buchse (Weibchen). Die verwendeten Anschlüsse sind im folgenden nochmals in einer Tabelle dokumentiert.

Pin Cannon	CCITT V.24	EIA RS232	DIN 66020	CBM Bez.	Beschreibung
1	101	AA	E1	protGND	Schutzerde, gemeinsame Masse
2	103	BA	D1	TxD	Sendedaten: Ltg. für abgehende Daten
3	104	BB	D2	RxD	Empfangsdaten: Ltg. ankommende Daten
4	105	CA	S2	RTS	Sendeteil ein: Mitteilung zum Modem, Daten sollen gesendet werden
5	106	CB	M2	CTS/RFS	Sendebereit: Antwort des Modems auf RTS
6	107	CC	M1	DSR	Betriebsbereit: Mitteilung des Modems ankommende Daten zu übernehmen
7	102	AB	E2	sig.GND	Signalerde, Betriebserde
8	109	CF	M5	DCD	Empfangssignalpegel: Mitteilung des Modems, Daten auf der Leitung
20	108.2	CD	S1.2	DTR	Terminal betriebsbereit: Mitteilung an Modem, Datenempfangsbereit
24	113	DA	T1	RxC	Takt für ext. Takt (nur Empfang)

Die Rechner CBM 600/700 erwarten an allen beschalteten Anschlüssen den richtigen logischen Pegel. Eine Ausnahme ist 'RxC' = Pin24. Er wird nur ausgewertet, wenn der externe Takt beim Eröffnen der Datenübertragungsleitung ausdrücklich erwünscht wird (näheres im vorherigen Kapitel 'Übertragungsgeschwindigkeit').
Da es sehr häufig vorkommt, daß das an den Rechner anzuschliessende Gerät nicht alle bedienten Anschlüsse auswertet und beantwortet, können zur Funktion folgende Lötbrücken am Stecker des Verbindungskabels hergestellt werden:

Anschlüsse '4 = RTS' mit '5 = CTS' verbinden und
Anschlüsse '6 = DSR' mit '8 = DCD' und '20 = DTR' verbinden

CBM 600 und 700 Speicheraufteilung



3.1 Speicheraufteilung

Bei den Geräten der Serie CBM 600/700 ist der gesamte adressierbare Speicherbereich in 16 Segmente 'Banks' unterteilt. Jedes Segment umfaßt einen Adressraum von 64 K-Byte. Der eingebaute Microprozessor '6509' kann 16 solcher Segmente (Bank 0-15) verwalten. Es sind also 16 * 64 K-Byte = 1 M-Byte adressierbar.

Unter BASIC haben einige Segmente, abhängig vom Ausbau, eine feste Bedeutung:

Modell	Ausbau	Segment 1	Segment 2	Segment 3	Segment 4
CBM 610/710	128 K-Byte	BASIC-Text	alle Variable		
CBM 620/720	256 K-Byte	BASIC-Text	Feld-Variable	einf. Variable	Strings
CBM 730	256 K-Byte	BASIC-Text	Feld-Variable	einf. Variable Disk-Betriebssyst.	Strings

Im Segment 15 befinden sich bei allen Modellen die Betriebssystem ROMs des Typs '2364 A' bzw. EPROMs '2764 A' (8 K-Byte) mit BASIC-Interpreter, Editor, Operatingsystem, Monitor, sowie das Bildschirm-RAM, das System-RAM (Zero-Page, Stack usw.) und die Adressen der Eingabe/Ausgabe-Bausteine. In einem weiteren Adressbereich ist Platz für das Disk-Betriebssystem (CBM 730) reserviert. Die restlichen Adressen sind über den später beschriebenen Cartridge-Anschluß nach außen geführt.

Die restlichen freien Segmente sind über die eingebaute 'Expansion Schnittstelle' extern mit RAM-Speicher belegbar.

Die einzelnen eingebauten RAM-Segmente sind mit 8 dynamischen RAM-Bausteinen '4864' aufgebaut. Jedes dieser RAMs hat eine Kapazität von 64 K-Bit. 8 RAMs mit 64K-Bit parallel ergeben 64 K-Byte. Für das Systemsegment-RAM wird ein statischer RAM-Baustein '2016' mit 2 K-Byte verwendet (Zero-Page, Stack).

Die folgende grafische Übersicht gibt Ihnen eine zusammenfassende Information über die Speicheraufteilung in den Geräten der Serien 600 und 700.

3.2 ROM/RAM - Cartridge

Die Geräte der Serien CBM 600/700 verfügen an der Geräterückseite über einen Anschluß für Steckmodule. Es lassen sich damit zusätzlich noch 24K-Byte ROM/RAM-/Eprom/IO in das Segment $0F = Bank 15 einfügen.

Wie bei den Geräten VC-20 und VC-64 besteht die Möglichkeit, einen Autostart des Modulprogramms durchzuführen. Nach dem Errichten des Stacks und dem Test, ob ein 'Reset' vorliegt (kein Kaltstart), wird nach dem Einschalten des Rechners der Bildschirm-Controller initialisiert. Dies ist notwendig um beim CBM 700 eine Beschädigung des Video-Teils zu vermeiden. Danach erfolgt die Prüfung ob ein Autostart-ROM eingesetzt ist.

Das Operating-System (Kernal) untersucht an den 4k-Byte-Grenzen '$1000 bis $8000' im Segment $0F (Bank 15) ob folgende Daten vorhanden sind:

Adresse	Wert	Bedeutung
$X006	'c'	hex $43='c' oder $C3='C' bei Standardinit.
$X007	$C2	ASCII 'B'
$X008	$CD	ASCII 'M'
$X009	'x'	'x' steht für 4K-Byte-Block 1-8 = $30-$38

Werden beim Test diese Daten gefunden, dann enthalten die Speicherstellen $X000 bis $X002 einen JMP $XXXX auf das Anwenderprogramm. Es erfolgt ein Autostart. Soll das Betriebssystem vor dem Starten des ROM-Programms eine normale Initialisierung durchführen, dann muß in der Speicherstelle $X006 der Wert 'c'+$80 stehen (höchstes Bit gesetzt).

Der jeweilige Warmstart-Einsprung ist $X003. Hier steht ein JMP $XXXX auf das Anwenderprogramm.

Wie in der Speicheraufteilung ersichtlich, liegen die eingebauten BASIC-ROMs von $8000 aufwärts. Die Sequenz für BASIC ab $8006 ist $C3, $C2, $CD, $38. Sind Floppy-Laufwerke eingebaut, so übernimmt das eingebaute Disk-ROM (ab Adresse $1000) die System-Kontrolle.

Sind kein Autostart- und keine BASIC-ROMs vorhanden, meldet sich der Rechner mit dem Maschinensprache-Monitor.

Allgemeiner Hinweis:

Die Geräte CBM 600/700 verfügen über einen hardwaremäßig erzeugten Cursor. Das Blinken des Cursors ist deshalb kein Beweis für einen 'lebenden' Rechner. Das heißt, der Rechner kann trotz blinkendem Cursor abgestürzt sein. Es muß danach ein Reset (Knopf an der Geräterückseite) durchgeführt werden. Dies gilt nicht für die Rechner CBM 8000.

3.3 Steckerbelegungen

3.3.1 IEEE Stecker

Pin	Bez.	Baustein	intern	Adresse	dez.
1	D1	CIA 6526	PRA 0	$DC00	56320
2	D2	CIA 6526	PRA 1	$DC00	56320
3	D3	CIA 6526	PRA 2	$DC00	56320
4	D4	CIA 6526	PRA 3	$DC00	56320
5	EOI	TPI 6525	PRA 5	$DE00	56832
6	DAV	TPI 6525	PRA 4	$DE00	56832
7	NRFD	TPI 6525	PRA 7	$DE00	56832
8	NDAC	TPI 6525	PRA 6	$DE00	56832
9	IFC	TPI 6525	PRB 0	$DE01	56833
10	SRQ	TPI 6525	PRB 1	$DE01	56833
11	ATN	TPI 6525	PRA 3	$DE00	56832
12	prot.GND	(Schutzerde, gemeinsame Masse)			
A	D5	CIA 6526	PRA 4	$DC00	56320
B	D6	CIA 6526	PRA 5	$DC00	56320
C	D7	CIA 6526	PRA 6	$DC00	56320
D	D8	CIA 6526	PRA 7	$DC00	56320
E	REN	TPI 6525	PRA 2	$DE00	56832
F-N	GND				

3.3.2 Keyboard Stecker

Pin	Bez.	Baustein	intern	Adresse	dez.
1	PA0	TPI 6525	PRA 0	$DF00	57088
2	PA2	TPI 6525	PRA 2	$DF00	57088
3	PA4	TPI 6525	PRA 4	$DF00	57088
4	PA6	TPI 6525	PRA 6	$DF00	57088
5	PB0	TPI 6525	PRB 0	$DF01	57089
6	PB1	TPI 6525	PRB 1	$DF01	57089
7	PB2	TPI 6525	PRB 2	$DF01	57089
8	PB3	TPI 6525	PRB 3	$DF01	57089
9	PB4	TPI 6525	PRB 4	$DF01	57089
10	PB5	TPI 6525	PRB 5	$DF01	57089
11	PB6	TPI 6525	PRB 6	$DF01	57089
12	PB7	TPI 6525	PRB 7	$DF01	57089
13	PC5	TPI 6525	PRC 5	$DF02	57090
14	PA1	TPI 6525	PRA 1	$DF00	57088
15	PA3	TPI 6525	PRA 3	$DF00	57088
16	PA5	TPI 6525	PRA 5	$DF00	57088
17	PA7	TPI 6525	PRA 7	$DF00	57088
18	PC0	TPI 6525	PRC 0	$DF02	57090
19	PC1	TPI 6525	PRC 1	$DF02	57090
20	PC2	TPI 6525	PRC 2	$DF02	57090
21	PC3	TPI 6525	PRC 3	$DF02	57090
22	GND				
23	GND				
24	GND				
25	PC4	TPI 6525	PRC 4	$DF02	57090

3.3.3 Cartridge Stecker

Pin	Bezeichnung	
1	A0	Adressbit 0
2	A1	Adressbit 1
3	A2	Adressbit 2
4	A3	Adressbit 3
5	A4	Adressbit 4
6	A5	Adressbit 5
7	A6	Adressbit 6
8	A7	Adressbit 7
9	A8	Adressbit 8
10	A9	Adressbit 9
11	A10	Adressbit 10
12	A11	Adressbit 11
13	A12	Adressbit 12
14	+ 5V DC	Spannung + 5V
15	+ 5V DC	Spannung + 5V
A	BD0	Datenbit 0
B	BD1	Datenbit 1
C	BD2	Datenbit 2
D	BD3	Datenbit 3
E	BD4	Datenbit 4
F	BD5	Datenbit 5
H	BD6	Datenbit 6
J	BD7	Datenbit 7
K	GND	Masse
L	GND	Masse
M	S R/W	Modus Read/Write
N	S02	Takt phi 2
P	NOT CSBANK 1	Chip Select Bank 1 (Segment 15)
R	NOT CSBANK 2	Chip Select Bank 2 (Segment 15)
S	NOT CSBANK 3	Chip Select Bank 3 (Segment 15)

3.3.4 Video Stecker

Pin	Bezeichnung	
1	VIDEO	Videosignal
2	GND	Masse
3	VERTICAL SYNC	Vertikal-Synchronisation
4	GND	Masse
5	HORIZONTAL SYNC	Horizontal-Synchronisation
6	KEY	Lichtgriffel
7	GND	Masse

3.3.5 User Stecker

Pin	Bez.	Baustein	intern	Adresse	dez.
1	GND				
2	PB2	TPI 6525	PRB 2	$DE01	56833
3	GND				
4	PB3	TPI 6525	PRB 3	$DE01	56833
5	NOT PC	CIA 6526	-PC	(Handshake PRB I/O, Output)	
6	NOT FL	Cass-Read	-FLAG	(Interrupt, Input)	
7	2D7	CIA 6526	PRB 7	$DC01	56321
8	2D6	CIA 6526	PRB 6	$DC01	56321
9	2D5	CIA 6526	PRB 5	$DC01	56321
10	2D4	CIA 6526	PRB 4	$DC01	56321
11	2D3	CIA 6526	PRB 3	$DC01	56321
12	2D2	CIA 6526	PRB 2	$DC01	56321
13	2D1	CIA 6526	PRB 1	$DC01	56321
14	2D0	CIA 6526	PRB 0	$DC01	56321
15	1D7	CIA 6526	PRA 7	$DC00	56320
16	1D6	CIA 6526	PRA 6	$DC00	56320
17	1D5	CIA 6526	PRA 5	$DC00	56320
18	1D4	CIA 6526	PRA 4	$DC00	56320
19	1D3	CIA 6526	PRA 3	$DC00	56320
20	1D2	CIA 6526	PRA 2	$DC00	56320
21	1D1	CIA 6526	PRA 1	$DC00	56320
22	1D0	CIA 6526	PRA 0	$DC00	56320
23	NOT CNT	CIA 6526	-CNT	$DC04	56324
		(Takt von Timer A)		$DC05	56325
24	+ 5V DC				
25	NOT IRQ	TPI 6525	PRC 5	$DE02	56834
26	SP	CIA 6526	SP	(Serial Port I/O)	

3.3.6 Audio Stecker

Pin	Bezeichnung
1	Lautsprecher (8 Ohm)
2	unbenutzt
3	Lautsprecher (8 Ohm)

3.3.7 Spannungs-Stecker

Pin	Bezeichnung
1	50 Hertz
2	- 12V DC
3	+ 12V DC
4	GND Masse
5	GND Masse
6	+ 5V DC

3.3.8 Co-Prozessor Stecker

Pin	Bezeichnung
1	EXTMA3
2	DRAMD0
3	EXTMA2
4	DRAMD1
5	EXTMA7
6	DRAMD2
7	EXTMA6
8	DRAMD3
9	EXTMA5
10	DRAMD4
11	EXTMA4
12	DRAMD5
13	EXTMA1
14	DRAMD6
15	EXTMA0
16	DRAMD7
17	GND
18	GND
19	GND
20	GND
21	GND
22	NOT BUSY 1
23	GND
24	NOT P2 REF REQ
25	GND
26	NOT P2 REF GRNT
27	GND
28	BP0
29	GND
30	BP1
31	GND
32	BP2
33	unbenutzt
34	BP3
35	NOT PROCRES
36	NOT BUSY 2
37	EXTBUF R/W
38	NOT ERAS
39	DRAM R/W
40	NOT ECAS

3.3.9 RS 232 C Stecker

Pin	Bezeichnung
1	prot. GND
2	TxD
3	RxD
4	RTS
5	CTS
6	DSR
7	GND
8	DCD
9	unbenutzt
10	unbenutzt
11	+ 5V DC
12	- 12V DC
13	unbenutzt
14	unbenutzt
15	unbenutzt
16	unbenutzt
17	unbenutzt
18	unbenutzt
19	unbenutzt
20	DTR
21	unbenutzt
22	unbenutzt
23	unbenutzt
24	RxC
25	unbenutzt

3.3.10 Expansion Stecker

Pin	Bez.	Baustein	intern	Adresse	dez.
1-4	+ 5V DC				
5-10	GND				
11	NOT BRAS		DRAM: Row Access		
12	IRQ 3	TPI 6525	PRC 3	$DE02	56834
13	- 12V DC				
14	NOT EXTRES		Reset-Durchla8		
15	+ 12V DC				
16	NOT SO	CPU 6509	SO		
17	NOT RES		System Reset		
18	LPEN	CRT 6845	Light Pen, Lichtgriffel		
19	S R/W		System Read/Write		
20	NOT EXTBUFCS		Adresse:	$0800-$0FFF	
21	TODCLK		Takt 50 Hertz		
22	NOT DISCROMCS		Adresse:	$1000-$1FFF	
23	BDOTCLK		Takt 18 MHz		
24	unbenutzt				
25	SO2		Takt phi 2		
26	NOT BCAS		DRAM: Column Access		
27	SO1		Takt phi 1		
28	NOT CS 1		Adresse:	$D900-$D9FF	
29	BD3		Datenbit 3		
30	NOT EXTPRTCS		Adresse:	$DB00-$DBFF	
31	BD4		Datenbit 4		
32	BD2		Datenbit 2		
33	BD5		Datenbit 5		
34	BD1		Datenbit 1		
35	BD7		Datenbit 7		
36	BD0		Datenbit 0		
37	BA13		Adressbit 13		
38	BD6		Datenbit 6		
39	BA14		Adressbit 14		
40	BA15		Adressbit 15		
41	BA1		Adressbit 1		
42	BA0		Adressbit 0		
43	BA2		Adressbit 2		
44	BA11		Adressbit 11		
45	BA3		Adressbit 3		
46	BA10		Adressbit 10		
47	BA12		Adressbit 12		
48	BA4		Adressbit 4		
49	BA9		Adressbit 9		
50	BA5		Adressbit 5		
51	BA8		Adressbit 8		
52	BA6		Adressbit 6		
53	BP0		Bank 0 (Segment 0)		
54	BA7		Adressbit 7		
55	BP1		Bank 1 (Segment 1)		
56	BP2		Bank 2 (Segment 2)		
57	NOT NMI	CPU 6509	Nicht maskierter Interrupt		
58	BP3		Bank 3 (Segment 3)		
59	RDY	CPU 6509	Ready		
60	NOT IRQ	CPU 6509	Interrupt Request		

4. TEIL

ASSEMBLERLISTING 1

CBM 610/710

BASIC 4.0

```
b710bas.lib......seite 0001

zeile adr.  obj.-code   source-code


 pass 1


0004  0000                    .lib b710bas.lab
0224  0000                    .lib b710bas.con
0319  0000                    .lib tokens
0850  8550                    .lib contrl
1587  898d                    .lib bverbs1
2478  8e7a                    .lib bverbs2
3157  92a1                    .lib bverbs3
3570  9585                    .lib math1
4469  9af5                    .lib math2
5308  9fca                    .lib math3
5801  a280                    .lib math4
6570  a7c0                    .lib strng1
7001  aa7a                    .lib strng2
7810  af9a                    .lib delete
7897  b026                    .lib using
8538  b4b8                    .lib butes1
9117  b7bf                    .lib butes2
9616  ba93                    .lib init
9804  bbdd                    .lib syscal

 pass 2


0001  0000  610/710 library-file
0002  0000
0003  0000
0004  0000                    .lib b710bas.lab
```

```
zeile adr.  obj.-code   source-code


0006  0000
0007  0000  ===> externe zero-page-label <===
0008  0000
0009  0000            e6509  = $00          6509 execution register
0010  0000            i6509  = $01          6509 indirection register
0011  0000            usrpok = $02          jmp code für user-routine
0012  0000            tmhour = $05          zeit 'stunden' für ti$-berechnung
0013  0000            tmmin  = $06          zeit 'minuten' für ti$-berechnung
0014  0000            tmsec  = $07          zeit 'sekunden' für ti$-berechnung
0015  0000            tmten  = $08          zeit '1/10-sek.' für ti$-berechnung
0016  0000            form   = $09          addrl format-zeiger
0017  0000            charac = $0c          puffer für trennzeichen
0018  0000            endchr = $0d          puffer für trennzeichen
0019  0000            count  = $0e          allgemeiner zähler
0020  0000            xcnt   = $0f          dos loop-zähler
0021  0000            dimflg = $10          dim-flag für zeigersuche auf
                                            variable
0022  0000            valtyp = $11          flag für variablen typ (0=num,
                                            1=string)
0023  0000            intflg = $12          flag für integer-variable
0024  0000            dores  = $13          flag für token o. ascii/garbage-flag
0025  0000            subflg = $14          flag für indizierte variable
0026  0000            inpflg = $15          flag für eingabe (input,get,read)
0027  0000            dsdesc = $16          länge disc-status string
0028  0000            channl = $1a          speicherstelle für aktiven kanal
0029  0000            linnum = $1b          lo-byte akt. zeilennummer
0030  0000            temmpt = $1d          zeiger auf zeitw. stringdescriptor
                                            (rel. offset)
0031  0000            lastpt = $1e          addrl letzt benutzter string-descr.
0032  0000            tempst = $20          addrl für 3 temp.
                                            string-descriptoren
0033  0000            index1 = $22          addrl indirekter index #1
0034  0000            index2 = $25          addrl indirekter index #2
0035  0000            resho  = $28          zwischenergebnis multipl./division
0036  0000            resmoh = $29          zwischenergebnis multipl./division
0037  0000            resmo  = $2a          zwischenergebnis multipl./division
0038  0000            reslo  = $2b          zwischenergebnis multipl./division
0039  0000            txttab = $2d          addrl zeiger anfang  basic-text
0040  0000            txtend = $2f          addrl zeiger ende  basic-text
0041  0000            vartab = $31          addrl zeiger anfang einfache
                                            variable
0042  0000            arytab = $35          addrl zeiger anfang array-tabelle
0043  0000            aryend = $37          addrl zeiger ende array-tabelle
0044  0000            strend = $39          addrl ende benutzter ram-bereich
0045  0000            fretop = $3b          addrl top of string free space
0046  0000            frespc = $3d          addrl zeiger auf neuen string
0047  0000            memtop = $3f          addrl höchste ram-speicherstelle
0048  0000            curlin = $42          lo-byte akt. zeilennummer
0049  0000            oldlin = $44          lo-byte vorige zeilennummer (stop)
0050  0000            oldtxt = $46          addrl zeiger letzter basic-befehl
0051  0000            datlin = $49          addrl data-zeilennummer
0052  0000            datptr = $4b          addrl zeiger auf data nach 'read'
0053  0000            inpptr = $4d          addrl input-zeilennummer
0054  0000            varnam = $4f          erstes zeichen akt. variablenname
0055  0000            varpnt = $51          addrl zeiger auf variable im ram
0056  0000            lstpnt = $54          addrl zeiger letzter string
0057  0000            opptr  = $57          addrl zeiger akt. operator-routine
0058  0000            opmask = $5a          vom akt. operator erzeugte bit-maske
```

```
zeile adr.  obj.-code   source-code


0059  0000            defpnt = $5b            addrl zeiger funkt.-def./ temp.fac#3
0060  0000            dscpnt = $5e            addrl zeiger string
                                              descr./temp.fac#3
0061  0000            jmper  = $61            $4c wert für 'jmp' (functions-rout.)
0062  0000            highds = $64            addrl zielende (verschieben)/
                                              temp.fac#1
0063  0000            hightr = $67            addrl ursprungende (verschieben)/
                                              temp.fac#1
0064  0000            lowds  = $6a            addrl zielanfang (verschieben)/
                                              temp.fac#2
0065  0000            lowtr  = $6d            addrl ursprunganfang (verschieben)/
                                              temp.fac#2
0066  0000            dsctmp = $70            zeitweise descriptoren-ablage
0067  0000            facexp = $71            fac #1: exponent
0068  0000            facho  = $72            fac #1: mantisse
0069  0000            facmoh = $73            fac #1: mantisse
0070  0000            facmo  = $74            fac #1: mantisse
0071  0000            faclo  = $75            fac #1: mantisse
0072  0000            facsgn = $76            fac #1: vorzeichen
0073  0000            sgnflg = $77            fac #1: vorzeichenduplikat
0074  0000            bits   = $78            shift-zähler
0075  0000            argexp = $79            fac-arg ungepackt: exponent
0076  0000            argho  = $7a            fac-arg ungepackt: mantisse
0077  0000            argmoh = $7b            fac-arg ungepackt: mantisse
0078  0000            argmo  = $7c            fac-arg ungepackt: mantisse
0079  0000            arglo  = $7d            fac-arg ungepackt: mantisse
0080  0000            argsgn = $7e            fac-arg ungepackt: vorzeichen
0081  0000            arisgn = $7f            fac-arg ungepackt:
                                              vorzeichenduplikat
0082  0000            facov  = $80            fac overflow-byte
0083  0000            fbufpt = $82            addrl zeiger fac-puffer (fout-rout.)
0084  0000            txtptr = $85            addrl zeiger auf akt. term
0085  0000            buffpt = $88            addrl zeiger auf eingabe-puffer
0086  0000
0087  0000            parsts = $8b            dos analyse-byte 1
0088  0000
0089  0000  ---> bitbedeutung von dos analyse-byte 1 <---
0090  0000
0091  0000  bit 7  =  klammeraffe in dateiname gelesen
0092  0000  bit 6  =  'w' oder '1' gelesen
0093  0000  bit 5  =  2. laufwerk # gelesen
0094  0000  bit 4  =  1. laufwerk # gelesen
0095  0000  bit 3  =  geräte # gelesen
0096  0000  bit 2  =  logische file # gelesen
0097  0000  bit 1  =  2. dateiname gelesen
0098  0000  bit 0  =  1. dateiname gelesen
0099  0000
0100  0000            parstx = $8c            dos analyse-byte 2
0101  0000
0102  0000  ---> bitbedeutung von dos analyse-flag 2 <---
0103  0000
0104  0000  bit 7  =  unbenutzt
0105  0000  bit 6  =  unbenutzt
0106  0000  bit 5  =  unbenutzt
0107  0000  bit 4  =  unbenutzt
0108  0000  bit 3  =  unbenutzt
0109  0000  bit 2  =  dos-offset hi gelesen
0110  0000  bit 1  =  dos-offset lo gelesen
```

```
zeile adr.  obj.-code   source-code


0111  0000  bit 0  =  dos-bank gelesen
0112  0000
0113  0000            seedpt = $8d          addrl zeiger auf akt. zufallszahl
0114  0000            errnum = $8f          fehlernummer für 'trap'
0115  0000
0116  0000
0117  0000  ===> externe ram-label <===
0118  0000
0119  0000            stack  = $0100        6509 cpu-stack
0120  0000            buf    = $0200        basic input-puffer (-$2ff)
0121  0000            dosfl1 = $0210        dos filename 1 länge
0122  0000            dosds1 = $0211        dos disk drive 1
0123  0000            dosf1a = $0212        addrl dos filename 1
0124  0000            dosfb1 = $0214        bank dos filename 1
0125  0000            dosf2l = $0215        dos filename 2 länge
0126  0000            dosds2 = $0216        dos disk drive 2
0127  0000            dosf2a = $0217        addrl dos filename 2
0128  0000            dosfb2 = $0219        bank dos filename 2
0129  0000            dosbnk = $021a        dos bank nummer
0130  0000            dosofl = $021b        addrl dos offset untergrenze
                                            (bsave,bload)
0131  0000            dosofh = $021d        addrl dos offset obergrenze
                                            (bsave,bload)
0132  0000            dosla  = $021f        dos logische adresse
0133  0000            dosfa  = $0220        dos primär-adresse
0134  0000            dossa  = $0221        dos sekundär-adresse
0135  0000            dosrcl = $0222        dos record-länge
0136  0000            dosdid = $0223        dos erstes zeichen 'id'
0137  0000            didchk = $0225        dos did flag
0138  0000            dosstr = $0226        dos ausgabe string puffer
0139  0000            trmpos = $0255        cursor spalte auf bildschirm
0140  0000            eormsk = $0256        bitmaske für 'wait'
0141  0000            dfbank = $0257        vorgabe für bank-nummer
0142  0000            dolu   = $0258        vorgabe für ausgabe-adresse
0143  0000            tansgn = $0259        flag für 'tan' vorzeichen
0144  0000            ldaabs = $025a        lda abs routine
0145  0000            ldaadr = $025b        addrl modifiable adresse
0146  0000            hulp   = $025e        print using: zähler
0147  0000            bnr    = $025f        print using: zeiger auf beginn
0148  0000            enr    = $0260        print using: zeiger auf ende
0149  0000            dolr   = $0261        print using: dollar flag
0150  0000            flag   = $0262        print using: komma flag
0151  0000            swe    = $0263        print using: zähler
0152  0000            usgn   = $0264        print using: vorzeichen exponent
0153  0000            uexp   = $0265        print using: zeiger auf exponent
0154  0000            vn     = $0266        print using: anzahl stellen vor
                                            dezimalpunkt
0155  0000            chsn   = $0267        print using: justify flag
0156  0000            vf     = $0268        print using: # position vor dez.pkt.
0157  0000            nf     = $0269        print using: # position nach
                                            dez.pkt.
0158  0000            posp   = $026a        print using: +/- flag (feld)
0159  0000            fesp   = $026b        print using: exponent flag (feld)
0160  0000            etof   = $026c        print using: switch
0161  0000            cform  = $026d        print using: zeichen zähler (feld)
0162  0000            sno    = $026e        print using: vorzeichen nummer
0163  0000            blfd   = $026f        print using: blank/stern flag
0164  0000            begfd  = $0270        print using: zeiger auf anfang feld
```

```
zeile adr.  obj.-code   source-code

0165  0000            lfor   = $0271        print using: länge formatstrings
0166  0000            endfd  = $0272        print using: zeiger auf ende feld
0167  0000            pufill = $0273        print using: füllzeichen
0168  0000            pucoma = $0274        print using: kommazeichen
0169  0000            pudot  = $0275        print using: zeichen dez.punkt
0170  0000            pumony = $0276        print using: währungszeichen
0171  0000            ierror = $0280        addrl fehler routine
0172  0000            imain  = $0282        addrl basic-warteschleife
0173  0000            icrnch = $0284        addrl token-umwandlungsroutine
0174  0000            iqplop = $0286        addrl token ausgabe expander rout.
0175  0000            igone  = $0288        addrl dispatcher
0176  0000            ieval  = $028a        addrl beliebigen ausdruck auswerten
0177  0000            ifrmev = $028c        addrl beliebigen ausdruck auswerten
0178  0000            ichrgo = $028e        addrl chrgot routine
0179  0000            ichrge = $0290        addrl chrget routine
0180  0000            trapno = $0296        addrl zeiger error trap
0181  0000            errlin = $0298        lo-byte zeilennummer akt. fehler
0182  0000            errtxt = $029a        addrl basic-text zeiger bei
                                            fehlerbehandlung
0183  0000            oldstk = $029c        stack zeiger vor 'trap' ausführung
0184  0000            tmptrp = $029d        hi-byte zeilennummer 'trap, resume'
0185  0000            dsptmp = $029e        temp. speicher 'dispose'
0186  0000            oldtok = $029f        temp. speicher 'dispose'
0187  0000            tmpdes = $02a0        temp. speicher 'instr$'
0188  0000            highst = $02a6        lo-byte max. offset user bank
0189  0000
0190  0000
0191  0000  ===> externe rom-label <===
0192  0000
0193  0000            knwsys = $ff6c        transfer-of-execution jumper
0194  0000            kvrese = $ff6f        netz ein/aus reset-vector
0195  0000            kfunky = $ff75        vector für funktionstasten
0196  0000            kioini = $ff7b        initialisierung ein-/ausgabe
0197  0000            kcint  = $ff7e        initialisierung bildschirm
0198  0000            klkups = $ff8a        geräteadr. über sekundäradr.suchen
0199  0000            klkupl = $ff8d        geräte- sekundäradr. über logische
                                            file # suchen
0200  0000            kstmsg = $ff90        meldung des operating-syst. ausgeben
0201  0000            kmemtp = $ff99        höchste speichergrenze
                                            lesen/schreiben
0202  0000            kmembt = $ff9c        unterste speichergrenze
                                            lesen/schreiben
0203  0000            kreast = $ffb7        ein-/ausgabe-status lesen/schreiben
0204  0000            kstlfs = $ffba        log. filenummer, geräte- und
                                            sekundäradr. eintragen
0205  0000            kstnam = $ffbd        länge und adresse des filenamens
                                            eintragen
0206  0000            kopen  = $ffc0        log. file öffnen/befehl ausgeben
0207  0000            kclose = $ffc3        logisches file schliessen
0208  0000            kchkin = $ffc6        eingabekanal öffnen
0209  0000            kckout = $ffc9        ausgabekanal öffnen
0210  0000            kclrch = $ffcc        ein-/ausgabekanal schließen
0211  0000            kbasin = $ffcf        eingabe 1 char vom akt. kanal
0212  0000            kbsout = $ffd2        ausgabe 1 char auf akt. kanal
0213  0000            kload  = $ffd5        einlesen vom logischen file
0214  0000            ksave  = $ffd8        abspeichern auf logisches file
0215  0000            ksttim = $ffdb        interne uhr stellen
0216  0000            krdtim = $ffde        interne uhr ablesen
```

```
zeile adr.  obj.-code   source-code

0217  0000            kstop  = $ffe1          test stop-taste
0218  0000            kgetin = $ffe4          eingabe 1 char von tastaturpuffer
0219  0000            kclall = $ffe7          alle logischen files schliessen
0220  0000            kplot  = $fff0          x,y-pos. des cursor lesen/schreiben
0221  0000            hairq  = $fffe          6509 hardware-irq-vektor
0222  0000
0223  0000                   .end
0224  0000                   .lib b710bas.con
```

```
zeile adr.  obj.-code   source-code


0226  0000
0227  0000   ===>  ascii- und asc-codes  <===
0228  0000
0229  0000            lf      = $0a             ascii für 'line feed'
0230  0000            cr      = $0d             ascii für 'carriage return'
0231  0000            clrscr  = 147             asc für 'clear screen'
0232  0000            home    = 197             asc für 'cursor home'
0233  0000
0234  0000            eom     = $80             end-of-message-kennung
0235  0000
0236  0000
0237  0000   ===>  system-konstanten und parameter  <===
0238  0000
0239  0000            forsiz = 19
0240  0000            numlev = 26
0241  0000            bufsiz = 161
0242  0000            addprc = 1
0243  0000            addpr2 = addprc+addprc
0244  0000            addpr4 = addpr2+addpr2
0245  0000            addpr8 = addpr4+addpr4
0246  0000            stkend = $01fb            erstes byte für stack
0247  0000            clmwid = 10               spaltenbreite für tabulation mit ','
0248  0000            pi     = $ff              basic-token für 'pi'
0249  0000            strsiz = 4                anzahl speicherstellen für
                                                string-descriptor
0250  0000            ptrsiz = 3                anzahl speicherstellen für
                                                speicher-zeiger
0251  0000            numtmp = 3
0252  0000
0253  0000
0254  0000   ===>  ieee-konstanten und parameter  <===
0255  0000
0256  0000            dc      = $01             75160/75161 control line
0257  0000            te      = $02             75160/75161 control line
0258  0000            ren     = $04             remote enable
0259  0000            attn    = $08             attention
0260  0000            dav     = $10             data available
0261  0000            eoi     = $20             end or identify
0262  0000            ndac    = $40             not data accepted
0263  0000            nrfd    = $80             not ready for data
0264  0000            ifc     = $01             interface clear
0265  0000            srq     = $02             service request
0266  0000
0267  0000            rddb    = nrfd+ndac+te+dc+ren  direction for receiver
0268  0000            tddb    = eoi+dav+attn+te+dc+ren direction for transmit
0269  0000
0270  0000            eoist   = $40             eoi status test
0271  0000            tlkr    = $40             device is talker
0272  0000            lstnr   = $20             device is listener
0273  0000            utlkr   = $5f             device untalk
0274  0000            ulstn   = $3f             device unlisten
0275  0000
0276  0000            toout   = $01             timeout status on output
0277  0000            toin    = $02             timeout status on input
0278  0000            eoist   = $40             eoi on input
0279  0000            nodev   = $80             no device on bus
0280  0000            sperr   = $10             verify error
0281  0000
```

```
zeile adr.  obj.-code   source-code

0282  0000            slock  = $40            screen editor lock-out
0283  0000            dibf   = $80            data in output buffer
0284  0000
0285  0000
0286  0000  ===>  vorbelegte speicher-segmente  <===
0287  0000
0288  0000            sysbnk = $0f            nummer der system-bank
0289  0000            mxbank = sysbnk+1       erste nicht erreichbare bank
0290  0000            txtbnk = 1              bank für basic-programme
0291  0000            varbnk = 3              bank für einfache variablen
                                              (256k-version)
0292  0000            arybnk = 2              bank für variablenfelder
                                              (256k-version)
0293  0000            strbnk = 4              bank für strings (256k-version)
0294  0000
0295  0000
0296  0000  ===>  dos-interface-konstanten  <===
0297  0000
0298  0000            dosfnl = 16+2           länge von filenamen
0299  0000            dosdsk = 8              vorgabewert für disc-gerätenummer
0300  0000            doslfn = 14             dos internal logical file nummer
0301  0000            dosctl = 21
0302  0000            doslst = dosfnl+dosfnl+dosfnl+16
0303  0000
0304  0000
0305  0000  ===>  kernal coprozessor communications-variablen  <===
0306  0000
0307  0000            ipb    = $0800          basisadresse ipc-puffer
0308  0000            ipbsiz = 13             kernal inter process communication
                                              puffer-länge
0309  0000
0310  0000  --->  offsets im ipc-puffer  <---
0311  0000
0312  0000            ipccmd = 0              ipc befehl
0313  0000            ipcjmp = 1              ipc sprung-adresse
0314  0000            ipcin  = 3              anzahl ipc eingabe-bytes
0315  0000            ipcout = 4              anzahl ipc ausgabe-bytes
0316  0000            ipcdat = 5              ipc data-puffer (8 bytes)
0317  0000
0318  0000                   .end
0319  0000                   .lib tokens
```

```
zeile adr.  obj.-code   source-code


0321  0000
0322  0000                    * = $8000
0323  8000
0324  8000
0325  8000  ===> basic kalt-start <===
0326  8000
0327  8000  4c 27 bb          jmp init
0328  8003
0329  8003
0330  8003  ===> basic warm-start <===
0331  8003
0332  8003  4c cc bb          jmp warm          warm-start
0333  8006
0334  8006
0335  8006  ===> commodore rom-kennung 'cbm' <===
0336  8006
0337  8006  c3                .byte $c3, $c2, $cd
0337  8007  c2
0337  8008  cd
0338  8009
0339  8009
0340  8009  ===> adress-kennzeichnungs-byte für rom-beginn '$8' <===
0341  8009
0342  8009  38 20             .byte '8 '
0343  800b
0344  800b
0345  800b  ===> adress-tabelle <===
0346  800b
0347  800b  ba 96             .word eval1       umw. zahlenstring in gk-zahl
0348  800d  d0 96             .word eval2       folgenden ausdruck (keine ziffer)
                                                auswerten
0349  800f  1b a2             .word floats
0350  8011  55 a9             .word fretms
0351  8013  66 88             .word getstk      test stapelplatz
0352  8015  bc bb             .word initv       standard-vektoren ins ram kopieren
0353  8017  73 a9             .word inpcom
0354  8019  c8 89             .word prit4
0355  801b  5a 87             .word newstt      neues statement, stoptest
0356  801d  2c 99             .word ptrget      variable suchen/anlegen, deren name
                                                folgt
0357  801f  57 87             .word xeqdir      test gültiges befehlszeichen, weiter
0358  8021  87 8c             .word goto0
0359  8023  2a bb             .word init0
0360  8025  ef 80             .word reslst      basic-befehlswort text-tabelle
0361  8027
0362  8027
0363  8027  ===> adress-tabelle der basic-befehle <===
0364  8027
0365  8027  a9 8b      stmdsp .word end-1       basic-routine 'end'
0366  8029  93 8a             .word for-1       basic-routine 'for'
0367  802b  05 8b             .word next-1      basic-routine 'next'
0368  802d  de 8c             .word data-1      basic-routine 'data'
0369  802f  4a 8f             .word inputn-1    basic-routine 'input#'
0370  8031  65 8f             .word input-1     basic-routine 'input'
0371  8033  0b 91             .word dim-1       basic-routine 'dim'
0372  8035  e9 8f             .word read-1      basic-routine 'read'
0373  8037  89 8d             .word let-1       basic-routine 'let'
0374  8039  83 8c             .word goto-1      basic-routine 'goto'
```

```
zeile adr.  obj.-code   source-code

0375  803b
0376  803b  06 8c      stmds1 .word run-1       basic-routine 'run'
0377  803d  41 8c             .word if-1        basic-routine 'if'
0378  803f  78 8b             .word restor-1    basic-routine 'restore'
0379  8041  24 8c             .word gosub-1     basic-routine 'gosub'
0380  8043  b7 8c             .word return-1    basic-routine 'return'
0381  8045  76 8c             .word rem-1       basic-routine 'rem'
0382  8047  a7 8b             .word stop-1      basic-routine 'stop'
0383  8049  2a 8d             .word ongoto-1    basic-routine 'on - goto'
0384  804b  51 91             .word fnwait-1    basic-routine 'wait'
0385  804d  c7 91             .word cload-1     basic-routine 'load'
0386  804f  1a 92             .word csave-1     basic-routine 'save'
0387  8051  bb 91             .word cverf-1     basic-routine 'verify'
0388  8053  15 91             .word defn-1      basic-routine 'def'
0389  8055  45 91             .word poke-1      basic-routine 'poke'
0390  8057  79 8e             .word printn-1    basic-routine 'print#'
0391  8059  9c 8e             .word print-1     basic-routine 'print'
0392  805b  e8 8b             .word cont-1      basic-routine 'cont'
0393  805d  8c 89             .word list-1      basic-routine 'list'
0394  805f  44 8a             .word clear-1     basic-routine 'clr', zero=0 rts
0395  8061  7f 8e             .word cmd-1       basic-routine 'cmd'
0396  8063  e6 90             .word csys-1      basic-routine 'sys'
0397  8065  42 92             .word copen-1     basic-routine 'open'
0398  8067  96 92             .word cclos-1     basic-routine 'close'
0399  8069  14 8f             .word get-1       basic-routine 'get'
0400  806b  28 8a             .word scrath-1    basic-routine 'new'
0401  806d  7b 8c             .word go-1        basic-routine 'go to'
0402  806f  45 95             .word concat-1    disc-basic-routine 'concat'
0403  8071  6c 93             .word dopen-1     disc-basic-routine 'dopen'
0404  8073  a8 93             .word dclose-1    disc-basic-routine 'dclose'
0405  8075  9d 94             .word record-1    disc-basic-routine 'record'
0406  8077  26 94             .word format-1    disc-basic-routine 'header'
0407  8079  12 95             .word colect-1    disc-basic-routine 'collect'
0408  807b  5f 95             .word backup-1    disc-basic-routine 'backup'
0409  807d  29 95             .word dcopy-1     disc-basic-routine 'copy'
0410  807f  7d 93             .word append-1    disc-basic-routine 'append'
0411  8081  c2 93             .word dsave-1     disc-basic-routine 'dsave'
0412  8083  cd 93             .word dload-1     disc-basic-routine 'dload'
0413  8085  a0 92             .word dcat-1      disc-basic-rout. 'catalog-dir.'
0414  8087  51 95             .word rename-1    disc-basic-routine 'rename'
0415  8089  63 94             .word scratc-1    disc-basic-routine 'scratch'
0416  808b  a0 92             .word dcat-1      disc-basic-rout. 'catalog-dir.'
0417  808d  09 95             .word dclear-1    disc-basic-routine 'dclear'
0418  808f  dd 93             .word cnbank-1    basic-routine 'bank'
0419  8091  0d 94             .word bload-1     disc-basic-routine 'bload'
0420  8093  eb 93             .word bsave-1     disc-basic-routine 'bsave'
0421  8095  7e 91             .word fkey-1      basic-routine 'key'
0422  8097  9c af             .word delete-1    basic-routine 'delete'
0423  8099  76 8c             .word rem-1       basic-routine 'rem'
0424  809b  15 8d             .word trap-1      basic-routine 'trap'
0425  809d  c3 8d             .word resume-1    basic-routine 'resume'
0426  809f  23 8e             .word dispos-1    basic-routine 'dispose'
0427  80a1  bf a7             .word puctrl-1    basic-routine 'pudef'
0428  80a3  10 a2             .word sgn         basic-routine 'sgn' fac=sgn(fac)
0429  80a5  b1 a2             .word int         basic-routine 'int' fac=int(fac)
0430  80a7  2f a2             .word abs         basic-routine 'abs' abs=abs(fac)
0431  80a9  02 00             .word usrpok      jmp code für user-routine
0432  80ab  f5 9c             .word fre         basic-routine 'fre'
```

```
zeile adr.  obj.-code   source-code

0433  80ad  33 9d            .word pos         basic-routine 'pos'
0434  80af  37 a5            .word sqr         basic-routine 'sqr' fac=sqr(fac)
0435  80b1  59 a6            .word rnd         basic-routine 'rnd'
0436  80b3  ca 9f            .word log         basic-routine 'log'
0437  80b5  b3 a5            .word exp         basic-routine 'exp' fac=exp(fac)
0438  80b7  a6 a6            .word cos         basic-routine 'cos' fac=cos(fac)
0439  80b9  ad a6            .word sin         basic-routine 'sin'
0440  80bb  f8 a6            .word tan         basic-routine 'tan' fac=tan(fac)
0441  80bd  91 a7            .word atn         basic-routine 'atn' fac=atn(fac)
0442  80bf  07 9e            .word peek        basic-routine 'peek'
0443  80c1  8e ab            .word len         basic-routine 'len'
0444  80c3  db a7            .word strd        basic-routine 'str$'
0445  80c5  ae ab            .word val         basic-routine 'val'
0446  80c7  9d ab            .word asc         basic-routine 'asc'
0447  80c9  d1 aa            .word chrd        basic-routine 'chr$'
0448  80cb  eb aa            .word leftd       basic-routine 'left$'
0449  80cd  22 ab            .word rightd      basic-routine 'right$'
0450  80cf  42 ab            .word midd        basic-routine 'mid$'
0451  80d1
0452  80d1
0453  80d1  ===> adresse / prioritätsflags der math. rout. <===
0454  80d1
0455  80d1  79        optab  .byte $79
0456  80d2  4c 9e            .word faddt-1     adresse / prioritätsflags der math.
                                                rout.
0457  80d4  79               .byte $79         adresse / prioritätsflags der math.
                                                rout.
0458  80d5  2f 9e            .word fsubt-1     basic-routine '-' (dez. subtr.)
0459  80d7  7b               .byte $7b
0460  80d8  0a a0            .word fmultt-1    basic-routine '*' (dez. multipl.)
0461  80da  7b               .byte $7b
0462  80db  e8 a0            .word fdivt-1     basic-routine '/' (dez. division)
0463  80dd  7f               .byte $7f
0464  80de  40 a5            .word fpwrt-1     basic-routine 'dez. potenz'
0465  80e0  50               .byte $50
0466  80e1  6d 98            .word andop-1     basic operations-routine 'and'
0467  80e3  46               .byte $46
0468  80e4  6a 98            .word orop-1      basic operations-routine 'or'
0469  80e6  7d               .byte $7d
0470  80e7  79 a5            .word negop-1     basic-routine negation
0471  80e9  5a               .byte $5a
0472  80ea  f7 96            .word notop-1     basic operations-routine 'not'
0473  80ec  64               .byte $64
0474  80ed  a7 98            .word dorel-1     basic operations-routine '<=>'
0475  80ef
0476  80ef
0477  80ef  ===> basic-befehlswort text-tabelle <===
0478  80ef
0479  80ef  45 4e     reslst .byte 'en', $c4
0479  80f1  c4
0480  80f2  46 4f            .byte 'fo', $d2
0480  80f4  d2
0481  80f5  4e 45 58         .byte 'nex', $d4
0481  80f8  d4
0482  80f9  44 41 54         .byte 'dat', $c1
0482  80fc  c1
0483  80fd  49 4e 50         .byte 'input', $a3
0483  8100  55 54
```

```
zeile adr.  obj.-code   source-code

0483  8102  a3
0484  8103  49 4e 50          .byte 'inpu', $d4
0484  8106  55
0484  8107  d4
0485  8108  44 49             .byte 'di', $cd
0485  810a  cd
0486  810b  52 45 41          .byte 'rea', $c4
0486  810e  c4
0487  810f  4c 45             .byte 'le', $d4
0487  8111  d4
0488  8112  47 4f 54          .byte 'got', $cf
0488  8115  cf
0489  8116  52 55             .byte 'ru', $ce
0489  8118  ce
0490  8119  49                .byte 'i', $c6
0490  811a  c6
0491  811b  52 45 53          .byte 'restor', $c5
0491  811e  54 4f 52
0491  8121  c5
0492  8122  47 4f 53          .byte 'gosu', $c2
0492  8125  55
0492  8126  c2
0493  8127  52 45 54          .byte 'retur', $ce
0493  812a  55 52
0493  812c  ce
0494  812d  52 45             .byte 're', $cd
0494  812f  cd
0495  8130  53 54 4f          .byte 'sto', $d0
0495  8133  d0
0496  8134  4f                .byte 'o', $ce
0496  8135  ce
0497  8136  57 41 49          .byte 'wai', $d4
0497  8139  d4
0498  813a  4c 4f 41          .byte 'loa', $c4
0498  813d  c4
0499  813e  53 41 56          .byte 'sav', $c5
0499  8141  c5
0500  8142  56 45 52          .byte 'verif', $d9
0500  8145  49 46
0500  8147  d9
0501  8148  44 45             .byte 'de', $c6
0501  814a  c6
0502  814b  50 4f 4b          .byte 'pok', $c5
0502  814e  c5
0503  814f  50 52 49          .byte 'print', $a3
0503  8152  4e 54
0503  8154  a3
0504  8155  50 52 49          .byte 'prin', $d4
0504  8158  4e
0504  8159  d4
0505  815a  43 4f 4e          .byte 'con', $d4
0505  815d  d4
0506  815e  4c 49 53          .byte 'lis', $d4
0506  8161  d4
0507  8162  43 4c             .byte 'cl', $d2
0507  8164  d2
0508  8165  43 4d             .byte 'cm', $c4
0508  8167  c4
```

```
zeile adr.  obj.-code   source-code

0509  8168  53 59           .byte 'sy', $d3
0509  816a  d3
0510  816b  4f 50 45        .byte 'ope', $ce
0510  816e  ce
0511  816f  43 4c 4f        .byte 'clos', $c5
0511  8172  53
0511  8173  c5
0512  8174  47 45           .byte 'ge', $d4
0512  8176  d4
0513  8177  4e 45           .byte 'ne', $d7
0513  8179  d7
0514  817a  54 41 42        .byte 'tab', $a8
0514  817d  a8
0515  817e  54              .byte 't', $cf
0515  817f  cf
0516  8180  46              .byte 'f', $ce
0516  8181  ce
0517  8182  53 50 43        .byte 'spc', $a8
0517  8185  a8
0518  8186  54 48 45        .byte 'the', $ce
0518  8189  ce
0519  818a  4e 4f           .byte 'no', $d4
0519  818c  d4
0520  818d  53 54 45        .byte 'ste', $d0
0520  8190  d0
0521  8191  ab              .byte $ab
0522  8192  ad              .byte $ad
0523  8193  aa              .byte $aa
0524  8194  af              .byte $af
0525  8195  de              .byte $de
0526  8196  41 4e           .byte 'an', $c4
0526  8198  c4
0527  8199  4f              .byte 'o', $d2
0527  819a  d2
0528  819b  be              .byte $be
0529  819c  bd              .byte $bd
0530  819d  bc              .byte $bc
0531  819e  53 47           .byte 'sg', $ce
0531  81a0  ce
0532  81a1  49 4e           .byte 'in', $d4
0532  81a3  d4
0533  81a4  41 42           .byte 'ab', $d3
0533  81a6  d3
0534  81a7  55 53           .byte 'us', $d2
0534  81a9  d2
0535  81aa  46 52           .byte 'fr', $c5
0535  81ac  c5
0536  81ad  50 4f           .byte 'po', $d3
0536  81af  d3
0537  81b0  53 51           .byte 'sq', $d2
0537  81b2  d2
0538  81b3  52 4e           .byte 'rn', $c4
0538  81b5  c4
0539  81b6  4c 4f           .byte 'lo', $c7
0539  81b8  c7
0540  81b9  45 58           .byte 'ex', $d0
0540  81bb  d0
0541  81bc  43 4f           .byte 'co', $d3
```

```
zeile adr.  obj.-code   source-code

0541  81be  d3
0542  81bf  53 49           .byte 'si', $ce
0542  81c1  ce
0543  81c2  54 41           .byte 'ta', $ce
0543  81c4  ce
0544  81c5  41 54           .byte 'at', $ce
0544  81c7  ce
0545  81c8  50 45 45        .byte 'pee', $cb
0545  81cb  cb
0546  81cc  4c 45           .byte 'le', $ce
0546  81ce  ce
0547  81cf  53 54 52        .byte 'str', $a4
0547  81d2  a4
0548  81d3  56 41           .byte 'va', $cc
0548  81d5  cc
0549  81d6  41 53           .byte 'as', $c3
0549  81d8  c3
0550  81d9  43 48 52        .byte 'chr', $a4
0550  81dc  a4
0551  81dd  4c 45 46        .byte 'left', $a4
0551  81e0  54
0551  81e1  a4
0552  81e2  52 49 47        .byte 'right', $a4
0552  81e5  48 54
0552  81e7  a4
0553  81e8  4d 49 44        .byte 'mid', $a4
0553  81eb  a4
0554  81ec  47              .byte 'g', $cf
0554  81ed  cf
0555  81ee  43 4f 4e        .byte 'conca', $d4
0555  81f1  43 41
0555  81f3  d4
0556  81f4  44 4f 50        .byte 'dope', $ce
0556  81f7  45
0556  81f8  ce
0557  81f9  44 43 4c        .byte 'dclos', $c5
0557  81fc  4f 53
0557  81fe  c5
0558  81ff  52 45 43        .byte 'recor', $c4
0558  8202  4f 52
0558  8204  c4
0559  8205  48 45 41        .byte 'heade', $d2
0559  8208  44 45
0559  820a  d2
0560  820b  43 4f 4c        .byte 'collec', $d4
0560  820e  4c 45 43
0560  8211  d4
0561  8212  42 41 43        .byte 'backu', $d0
0561  8215  4b 55
0561  8217  d0
0562  8218  43 4f 50        .byte 'cop', $d9
0562  821b  d9
0563  821c  41 50 50        .byte 'appen', $c4
0563  821f  45 4e
0563  8221  c4
0564  8222  44 53 41        .byte 'dsav', $c5
0564  8225  56
0564  8226  c5
```

```
zeile adr.  obj.-code   source-code

0565  8227  44 4c 4f         .byte 'dloa', $c4
0565  822a  41
0565  822b  c4
0566  822c  43 41 54         .byte 'catalo', $c7
0566  822f  41 4c 4f
0566  8232  c7
0567  8233  52 45 4e         .byte 'renam', $c5
0567  8236  41 4d
0567  8238  c5
0568  8239  53 43 52         .byte 'scratc', $c8
0568  823c  41 54 43
0568  823f  c8
0569  8240  44 49 52         .byte 'director', $d9
0569  8243  45 43 54
0569  8246  4f 52
0569  8248  d9
0570  8249  44 43 4c         .byte 'dclea', $d2
0570  824c  45 41
0570  824e  d2
0571  824f  42 41 4e         .byte 'ban', $cb
0571  8252  cb
0572  8253  42 4c 4f         .byte 'bloa', $c4
0572  8256  41
0572  8257  c4
0573  8258  42 53 41         .byte 'bsav', $c5
0573  825b  56
0573  825c  c5
0574  825d  4b 45            .byte 'ke', $d9
0574  825f  d9
0575  8260  44 45 4c         .byte 'delet', $c5
0575  8263  45 54
0575  8265  c5
0576  8266  45 4c 53         .byte 'els', $c5
0576  8269  c5
0577  826a  54 52 41         .byte 'tra', $d0
0577  826d  d0
0578  826e  52 45 53         .byte 'resum', $c5
0578  8271  55 4d
0578  8273  c5
0579  8274  44 49 53         .byte 'dispos', $c5
0579  8277  50 4f 53
0579  827a  c5
0580  827b  50 55 44         .byte 'pude', $c6
0580  827e  45
0580  827f  c6
0581  8280  55 53 49         .byte 'usin', $c7
0581  8283  4e
0581  8284  c7
0582  8285  45 52 52         .byte 'err', $a4
0582  8288  a4
0583  8289  49 4e 53         .byte 'inst', $d2, $00
0583  828c  54
0583  828d  d2
0583  828e  00
0584  828f
0585  828f
0586  828f  ===> ind. sprungtabelle basic-/system fehlermeldungen <===
0587  828f
```

```
zeile adr.  obj.-code   source-code

0588  828f  e7 82     ebase  .word ams0       text: 'stop key detected'
0589  8291  f9 82            .word ams1       text: 'too many files'
0590  8293  08 83            .word ams2       text: 'file open'
0591  8295  12 83            .word ams3       text: 'file not open'
0592  8297  20 83            .word ams4       text: 'file not found'
0593  8299  2f 83            .word ams5       text: 'device not present'
0594  829b  42 83            .word ams6       text: 'not input file'
0595  829d  51 83            .word ams7       text: 'not output file'
0596  829f  61 83            .word ams8       text: 'missing file name'
0597  82a1  73 83            .word ams9       text: 'illegal device number'
0598  82a3  89 83            .word ams30      text: 'are you sure ?'
0599  82a5  99 83            .word ams31      text: 'bad disk '
0600  82a7  de 84            .word reddy      text: 'ready.'
0601  82a9  d9 84            .word aintxt     text: ' in '
0602  82ab  e7 84            .word abrktx     text: 'break'
0603  82ad  ee 84            .word aexi       text: 'extra ignored'
0604  82af  fd 84            .word atry       text: 'redo from start'
0605  82b1  00 02            .word buf        basic input-puffer (-$2ff)
0606  82b3  0e 85            .word aremsg     text: 'more'
0607  82b5  81 bb            .word signon     text: '*** cbm basic 128,v4.0 ***'
0608  82b7  a5 83            .word aernf      text: 'next without for'
0609  82b9  b6 83            .word aersn      text: 'syntax error'
0610  82bb  c3 83            .word aerrg      text: 'return without gosub'
0611  82bd  d8 83            .word aerod      text: 'out of data'
0612  82bf  e4 83            .word aerfc      text: 'illegal quantity'
0613  82c1  f5 83            .word aerov      text: 'overflow'
0614  82c3  fe 83            .word aerom      text: 'out of memory'
0615  82c5  0c 84            .word aerus      text: 'undefined statement'
0616  82c7  20 84            .word aerbs      text: 'bad subscript'
0617  82c9  2e 84            .word aerdd      text: 'redim''d array'
0618  82cb  3c 84            .word aerdvo     text: 'division by zero'
0619  82cd  4d 84            .word aerid      text: 'illegal direct'
0620  82cf  5c 84            .word aertm      text: 'type mismatch'
0621  82d1  6a 84            .word aerls      text: 'string too long'
0622  82d3  7a 84            .word aerbd      text: 'file data'
0623  82d5  84 84            .word aerst      text: 'formula too complex'
0624  82d7  98 84            .word aercn      text: 'cannot continue'
0625  82d9  a8 84            .word aeruf      text: 'undefined function'
0626  82db  bb 84            .word aerld      text: '?load error'
0627  82dd  c9 84            .word aervr      text: '?verify error'
0628  82df  14 85            .word aeros      text: 'out of stack'
0629  82e1  2d 85            .word aercr      text: 'unable to resume'
0630  82e3  3e 85            .word aerdi      text: 'unable to dispose'
0631  82e5  21 85            .word aerot      text: 'out of text'
0632  82e7
0633  82e7
0634  82e7  ===> text: 'stop key detected' <===
0635  82e7
0636  82e7  53 54 4f  ams0   .byte 'stop key detected', $00
0636  82ea  50 20 4b
0636  82ed  45 59 20
0636  82f0  44 45 54
0636  82f3  45 43 54
0636  82f6  45 44
0636  82f8  00
0637  82f9
0638  82f9
0639  82f9  ===> text: 'too many files' <===
```

```
zeile adr.  obj.-code   source-code

0640  82f9
0641  82f9  54 4f 4f  ams1   .byte 'too many files', $00
0641  82fc  20 4d 41
0641  82ff  4e 59 20
0641  8302  46 49 4c
0641  8305  45 53
0641  8307  00
0642  8308
0643  8308
0644  8308  ===> text: 'file open' <===
0645  8308
0646  8308  46 49 4c  ams2   .byte 'file open', $00
0646  830b  45 20 4f
0646  830e  50 45 4e
0646  8311  00
0647  8312
0648  8312
0649  8312  ===> text: 'file not open' <===
0650  8312
0651  8312  46 49 4c  ams3   .byte 'file not open', $00
0651  8315  45 20 4e
0651  8318  4f 54 20
0651  831b  4f 50 45
0651  831e  4e
0651  831f  00
0652  8320
0653  8320
0654  8320  ===> text: 'file not found' <===
0655  8320
0656  8320  46 49 4c  ams4   .byte 'file not found', $00
0656  8323  45 20 4e
0656  8326  4f 54 20
0656  8329  46 4f 55
0656  832c  4e 44
0656  832e  00
0657  832f
0658  832f
0659  832f  ===> text: 'device not present' <===
0660  832f
0661  832f  44 45 56  ams5   .byte 'device not present', $00
0661  8332  49 43 45
0661  8335  20 4e 4f
0661  8338  54 20 50
0661  833b  52 45 53
0661  833e  45 4e 54
0661  8341  00
0662  8342
0663  8342
0664  8342  ===> text: 'not input file' <===
0665  8342
0666  8342  4e 4f 54  ams6   .byte 'not input file', $00
0666  8345  20 49 4e
0666  8348  50 55 54
0666  834b  20 46 49
0666  834e  4c 45
0666  8350  00
0667  8351
0668  8351
```

```
zeile adr.  obj.-code   source-code

0669  8351  ===> text: 'not output file' <===
0670  8351
0671  8351  4e 4f 54  ams7   .byte 'not output file', $00
0671  8354  20 4f 55
0671  8357  54 50 55
0671  835a  54 20 46
0671  835d  49 4c 45
0671  8360  00
0672  8361
0673  8361
0674  8361  ===> text: 'missing file name' <===
0675  8361
0676  8361  4d 49 53  ams8   .byte 'missing file name', $00
0676  8364  53 49 4e
0676  8367  47 20 46
0676  836a  49 4c 45
0676  836d  20 4e 41
0676  8370  4d 45
0676  8372  00
0677  8373
0678  8373
0679  8373  ===> text: 'illegal device number' <===
0680  8373
0681  8373  49 4c 4c  ams9   .byte 'illegal device number', $00
0681  8376  45 47 41
0681  8379  4c 20 44
0681  837c  45 56 49
0681  837f  43 45 20
0681  8382  4e 55 4d
0681  8385  42 45 52
0681  8388  00
0682  8389
0683  8389
0684  8389  ===> text: 'are you sure ?' <===
0685  8389
0686  8389  0d        ams30  .byte $0d
0687  838a  41 52 45         .byte 'are you sure ?', $00
0687  838d  20 59 4f
0687  8390  55 20 53
0687  8393  55 52 45
0687  8396  20 3f
0687  8398  00
0688  8399
0689  8399
0690  8399  ===> text: 'bad disk ' <===
0691  8399
0692  8399  0d        ams31  .byte $0d
0693  839a  42 41 44         .byte 'bad disk ', $0d, $00
0693  839d  20 44 49
0693  83a0  53 4b 20
0693  83a3  0d
0693  83a4  00
0694  83a5
0695  83a5
0696  83a5  ===> text: 'next without for' <===
0697  83a5
0698  83a5  4e 45 58  aernf  .byte 'next without for', $00
0698  83a8  54 20 57
```

```
zeile adr.  obj.-code   source-code


0698  83ab  49 54 48
0698  83ae  4f 55 54
0698  83b1  20 46 4f
0698  83b4  52
0698  83b5  00
0699  83b6
0700  83b6
0701  83b6  ===> text: 'syntax error' <===
0702  83b6
0703  83b6  53 59 4e  aersn  .byte 'syntax error', $00
0703  83b9  54 41 58
0703  83bc  20 45 52
0703  83bf  52 4f 52
0703  83c2  00
0704  83c3
0705  83c3
0706  83c3  ===> text: 'return without gosub' <===
0707  83c3
0708  83c3  52 45 54  aerrg  .byte 'return without gosub', $00
0708  83c6  55 52 4e
0708  83c9  20 57 49
0708  83cc  54 48 4f
0708  83cf  55 54 20
0708  83d2  47 4f 53
0708  83d5  55 42
0708  83d7  00
0709  83d8
0710  83d8
0711  83d8  ===> text: 'out of data' <===
0712  83d8
0713  83d8  4f 55 54  aerod  .byte 'out of data', $00
0713  83db  20 4f 46
0713  83de  20 44 41
0713  83e1  54 41
0713  83e3  00
0714  83e4
0715  83e4
0716  83e4  ===> text: 'illegal quantity' <===
0717  83e4
0718  83e4  49 4c 4c  aerfc  .byte 'illegal quantity', $00
0718  83e7  45 47 41
0718  83ea  4c 20 51
0718  83ed  55 41 4e
0718  83f0  54 49 54
0718  83f3  59
0718  83f4  00
0719  83f5
0720  83f5
0721  83f5  ===> text: 'overflow' <===
0722  83f5
0723  83f5  4f 56 45  aerov  .byte 'overflow', $00
0723  83f8  52 46 4c
0723  83fb  4f 57
0723  83fd  00
0724  83fe
0725  83fe
0726  83fe  ===> text: 'out of memory' <===
0727  83fe
```

```
zeile adr.  obj.-code   source-code

0728  83fe  4f 55 54  aerom  .byte 'out of memory', $00
0728  8401  20 4f 46
0728  8404  20 4d 45
0728  8407  4d 4f 52
0728  840a  59
0728  840b  00
0729  840c
0730  840c
0731  840c  ===> text: 'undefined statement' <===
0732  840c
0733  840c  55 4e 44  aerus  .byte 'undefined statement', $00
0733  840f  45 46 49
0733  8412  4e 45 44
0733  8415  20 53 54
0733  8418  41 54 45
0733  841b  4d 45 4e
0733  841e  54
0733  841f  00
0734  8420
0735  8420
0736  8420  ===> text: 'bad subscript' <===
0737  8420
0738  8420  42 41 44  aerbs  .byte 'bad subscript', $00
0738  8423  20 53 55
0738  8426  42 53 43
0738  8429  52 49 50
0738  842c  54
0738  842d  00
0739  842e
0740  842e
0741  842e  ===> text: 'redim''d array' <===
0742  842e
0743  842e  52 45 44  aerdd  .byte 'redim''d array', $00
0743  8431  49 4d 27
0743  8434  44 20 41
0743  8437  52 52 41
0743  843a  59
0743  843b  00
0744  843c
0745  843c
0746  843c  ===> text: 'division by zero' <===
0747  843c
0748  843c  44 49 56  aerdvo .byte 'division by zero', $00
0748  843f  49 53 49
0748  8442  4f 4e 20
0748  8445  42 59 20
0748  8448  5a 45 52
0748  844b  4f
0748  844c  00
0749  844d
0750  844d
0751  844d  ===> text: 'illegal direct' <===
0752  844d
0753  844d  49 4c 4c  aerid  .byte 'illegal direct', $00
0753  8450  45 47 41
0753  8453  4c 20 44
0753  8456  49 52 45
0753  8459  43 54
```

```
zeile adr.  obj.-code   source-code

0753  845b  00
0754  845c
0755  845c
0756  845c  ===> text: 'type mismatch' <===
0757  845c
0758  845c  54 59 50  aertm  .byte 'type mismatch', $00
0758  845f  45 20 4d
0758  8462  49 53 4d
0758  8465  41 54 43
0758  8468  48
0758  8469  00
0759  846a
0760  846a
0761  846a  ===> text: 'string too long' <===
0762  846a
0763  846a  53 54 52  aerls  .byte 'string too long', $00
0763  846d  49 4e 47
0763  8470  20 54 4f
0763  8473  4f 20 4c
0763  8476  4f 4e 47
0763  8479  00
0764  847a
0765  847a
0766  847a  ===> text: 'file data' <===
0767  847a
0768  847a  46 49 4c  aerbd  .byte 'file data', $00
0768  847d  45 20 44
0768  8480  41 54 41
0768  8483  00
0769  8484
0770  8484
0771  8484  ===> text: 'formula too complex' <===
0772  8484
0773  8484  46 4f 52  aerst  .byte 'formula too complex', $00
0773  8487  4d 55 4c
0773  848a  41 20 54
0773  848d  4f 4f 20
0773  8490  43 4f 4d
0773  8493  50 4c 45
0773  8496  58
0773  8497  00
0774  8498
0775  8498
0776  8498  ===> text: 'cannot continue' <===
0777  8498
0778  8498  43 41 4e  aercn  .byte 'cannot continue', $00
0778  849b  4e 4f 54
0778  849e  20 43 4f
0778  84a1  4e 54 49
0778  84a4  4e 55 45
0778  84a7  00
0779  84a8
0780  84a8
0781  84a8  ===> text: 'undefined function' <===
0782  84a8
0783  84a8  55 4e 44  aeruf  .byte 'undefined function', $00
0783  84ab  45 46 49
0783  84ae  4e 45 44
```

```
zeile adr.  obj.-code   source-code

0783  84b1  20 46 55
0783  84b4  4e 43 54
0783  84b7  49 4f 4e
0783  84ba  00
0784  84bb
0785  84bb
0786  84bb  ===> text: '?load error' <===
0787  84bb
0788  84bb  0d        aerld  .byte $0d
0789  84bc  3f 4c 4f         .byte '?load error', $0d, $00
0789  84bf  41 44 20
0789  84c2  45 52 52
0789  84c5  4f 52
0789  84c7  0d
0789  84c8  00
0790  84c9
0791  84c9
0792  84c9  ===> text: '?verify error' <===
0793  84c9
0794  84c9  0d        aervr  .byte $0d
0795  84ca  3f 56 45         .byte '?verify error', $0d, $00
0795  84cd  52 49 46
0795  84d0  59 20 45
0795  84d3  52 52 4f
0795  84d6  52
0795  84d7  0d
0795  84d8  00
0796  84d9
0797  84d9
0798  84d9  ===> text: ' in ' <===
0799  84d9
0800  84d9  20 49 4e  aintxt .byte ' in ', $00
0800  84dc  20
0800  84dd  00
0801  84de
0802  84de
0803  84de  ===> text: 'ready.' <===
0804  84de
0805  84de  0d        reddy  .byte $0d
0806  84df  52 45 41         .byte 'ready.', $0d, $00
0806  84e2  44 59 2e
0806  84e5  0d
0806  84e6  00
0807  84e7
0808  84e7
0809  84e7  ===> text: 'break' <===
0810  84e7
0811  84e7  0d        abrktx .byte $0d
0812  84e8  42 52 45         .byte 'break', $00
0812  84eb  41 4b
0812  84ed  00
0813  84ee
0814  84ee
0815  84ee  ===> text: 'extra ignored' <===
0816  84ee
0817  84ee  45 58 54  aexi   .byte 'extra ignored', $0d, $00
0817  84f1  52 41 20
0817  84f4  49 47 4e
```

```
zeile adr.  obj.-code   source-code

0817  84f7  4f 52 45
0817  84fa  44
0817  84fb  0d
0817  84fc  00
0818  84fd
0819  84fd
0820  84fd  ===> text: 'redo from start' <===
0821  84fd
0822  84fd  52 45 44  atry   .byte 'redo from start', $0d, $00
0822  8500  4f 20 46
0822  8503  52 4f 4d
0822  8506  20 53 54
0822  8509  41 52 54
0822  850c  0d
0822  850d  00
0823  850e
0824  850e
0825  850e  ===> text: 'more' <===
0826  850e
0827  850e  4d 4f 52  aremsg .byte 'more', $0d, $00
0827  8511  45
0827  8512  0d
0827  8513  00
0828  8514
0829  8514
0830  8514  ===> text: 'out of stack' <===
0831  8514
0832  8514  4f 55 54  aeros  .byte 'out of stack', $00
0832  8517  20 4f 46
0832  851a  20 53 54
0832  851d  41 43 4b
0832  8520  00
0833  8521
0834  8521
0835  8521  ===> text: 'out of text' <===
0836  8521
0837  8521  4f 55 54  aerot  .byte 'out of text', $00
0837  8524  20 4f 46
0837  8527  20 54 45
0837  852a  58 54
0837  852c  00
0838  852d
0839  852d
0840  852d  ===> text: 'unable to resume' <===
0841  852d
0842  852d  55 4e 41  aercr  .byte 'unable to resume', $00
0842  8530  42 4c 45
0842  8533  20 54 4f
0842  8536  20 52 45
0842  8539  53 55 4d
0842  853c  45
0842  853d  00
0843  853e
0844  853e
0845  853e  ===> text: 'unable to dispose' <===
0846  853e
0847  853e  55 4e 41  aerdi  .byte 'unable to dispose', $00
0847  8541  42 4c 45
```

```
zeile adr.  obj.-code   source-code


0847  8544  20 54 4f
0847  8547  20 44 49
0847  854a  53 50 4f
0847  854d  53 45
0847  854f  00
0848  8550
0849  8550                    .end
0850  8550                    .lib contrl
```

```
zeile adr.  obj.-code   source-code

0852  8550
0853  8550  ===> ausgabe '?out of memory error', ready <===
0854  8550
0855  8550  a2 34     omerr  ldx #$34
0856  8552  6c 80 02  error  jmp (ierror)     addrl fehler routine
0857  8555
0858  8555
0859  8555  ===> error-routine <===
0860  8555
0861  8555  86 8f     nerror stx errnum       fehlernummer für 'trap'
0862  8557  e0 36            cpx #$36
0863  8559  f0 37            beq errigd       fehlermeldung ausgeben
0864  855b  20 57 9d         jsr tstdir       test direktmodus, hi-byt zeile=$ff
0865  855e  f0 32            beq errigd       fehlermeldung ausgeben
0866  8560  ac 97 02         ldy trapno+1     addrh zeiger error trap
0867  8563  c8               iny
0868  8564  f0 2c            beq errigd       fehlermeldung ausgeben
0869  8566  88               dey
0870  8567  84 1c            sty linnum+1     hi-byte akt. zeilennummer
0871  8569  8c 9d 02         sty tmptrp       hi-byte zeilennummer 'trap, resume'
0872  856c  ac 96 02         ldy trapno       addrl zeiger error trap
0873  856f  84 1b            sty linnum       lo-byte akt. zeilennummer
0874  8571  a0 ff            ldy #$ff
0875  8573  8c 97 02         sty trapno+1     addrh zeiger error trap
0876  8576  a2 01            ldx #$01
0877  8578
0878  8578
0879  8578  ===> fehlerzeilen # abspeichern <===
0880  8578
0881  8578  b5 42     sots   lda curlin,x     lo-byte akt. zeilennummer
0882  857a  9d 98 02         sta errlin,x     lo-byte zeilennummer akt. fehler
0883  857d  b5 46            lda oldtxt,x     addrl zeiger letzter basic-befehl
0884  857f  9d 9a 02         sta errtxt,x     addrl basic-text zeiger bei
                                               fehlerbehandlung
0885  8582  ca               dex
0886  8583  10 f3            bpl sots         fehlerzeilen # abspeichern
0887  8585  ae 9c 02         ldx oldstk       stack zeiger vor 'trap' ausführung
0888  8588  9a               txs
0889  8589  20 8c ba         jsr mapusr       umsch. auf bank # 1
0890  858c  20 a0 8c         jsr luk4it       zeilen # > akt. suchen/abarb.
0891  858f  4c 80 87         jmp nstt9
0892  8592
0893  8592
0894  8592  ===> fehlermeldung ausgeben <===
0895  8592
0896  8592  a5 1a     errigd lda channl       speicherstelle für aktiven kanal
0897  8594  f0 07            beq error2
0898  8596  20 cc ff         jsr kclrch       ein-/ausgabekanal schließen
0899  8599  a9 00            lda #$00
0900  859b  85 1a            sta channl       speicherstelle für aktiven kanal
0901  859d  20 c8 8e  error2 jsr ocrlf        ausgabe cr (+lf) auf akt. kanal
0902  85a0  20 38 b5         jsr outqst       ausgabe '?'
0903  85a3  a6 8f            ldx errnum       fehlernummer für 'trap'
0904  85a5  20 c3 a3         jsr msg          textzeiger + xr, textausgabe
0905  85a8  20 75 8a         jsr stkini       reset stack- u. basicbefehl-zeiger
0906  85ab
0907  85ab
0908  85ab  ===> im prg-mode zeilen#-ausgabe, ready <===
```

```
zeile adr.  obj.-code   source-code


0909  85ab
0910  85ab  20 8c ba  errfin jsr mapusr       umsch. auf bank # 1
0911  85ae  a4 43            ldy curlin+1     hi-byte akt. zeilennummer
0912  85b0  c0 fa            cpy #$fa
0913  85b2  b0 0c            bcs ready        ausgabe 'ready', ready-status
0914  85b4  a2 1a            ldx #$1a
0915  85b6  20 c3 a3         jsr msg          textzeiger + xr, textausgabe
0916  85b9  a5 43            lda curlin+1     hi-byte akt. zeilennummer
0917  85bb  a6 42            ldx curlin       lo-byte akt. zeilennummer
0918  85bd  20 b4 a3         jsr linprt       umw. hex/dez, lo=xr, hi=ac, druck
                                               zeilen #
0919  85c0
0920  85c0
0921  85c0  ===> ausgabe 'ready', ready-status <===
0922  85c0
0923  85c0  a2 18      ready  ldx #$18
0924  85c2  20 c3 a3         jsr msg          textzeiger + xr, textausgabe
0925  85c5  a9 80            lda #$80
0926  85c7  20 90 ff         jsr kstmsg       meldung des operating-syst. ausgeben
0927  85ca  6c 82 02  main   jmp (imain)      addrl basic-warteschleife
0928  85cd
0929  85cd
0930  85cd  ===> hauptroutine basic-interpreter <===
0931  85cd
0932  85cd  a2 ff      nmain  ldx #$ff
0933  85cf  86 43            stx curlin+1     hi-byte akt. zeilennummer
0934  85d1  8e 98 02         stx errlin       lo-byte zeilennummer akt. fehler
0935  85d4  8e 99 02         stx errlin+1     hi-byte zeilennummer akt. fehler
0936  85d7  86 8f            stx errnum       fehlernummer für 'trap'
0937  85d9  20 e3 86         jsr inlin        eingabe einer zeile
0938  85dc  85 85            sta txtptr       addrl zeiger auf akt. term
0939  85de  84 86            sty txtptr+1     addrh zeiger auf akt. term
0940  85e0  86 87            stx txtptr+2     bank zeiger auf akt. term
0941  85e2  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
0942  85e5  aa               tax
0943  85e6  f0 e2            beq main         hauptroutine basic-interpreter
0944  85e8  90 09            bcc main1        programmzeilen löschen/einfügen
0945  85ea  20 bf 88         jsr crunch       ind.jmp umw. text in basic-tokens
0946  85ed  20 29 ba         jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
0947  85f0  4c 57 87         jmp xeqdir       test gültiges befehlszeichen, weiter
0948  85f3
0949  85f3
0950  85f3  ===> programmzeilen löschen/einfügen <===
0951  85f3
0952  85f3  20 4e 8d  main1  jsr linget       zeilennummer lesen, in adressformat
                                               umwandeln
0953  85f6  20 bf 88         jsr crunch       ind.jmp umw. text in basic-tokens
0954  85f9  84 0e            sty count        allgemeiner zähler
0955  85fb  20 1f 87         jsr fndlin       startadr. von prg.zeile berechnen
0956  85fe  90 47            bcc nodel        test: programmzeile einfügen
0957  8600  a0 01            ldy #$01
0958  8602  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                               temp.fac#2
0959  8604  85 23            sta index1+1     addrh indirekter index #1
0960  8606  a5 2f            lda txtend       addrl zeiger ende  basic-text
0961  8608  85 22            sta index1       addrl indirekter index #1
0962  860a  a5 6e            lda lowtr+1      addrh ursprunganfang (verschieben)
                                               temp.fac#2
```

```
zeile adr.  obj.-code   source-code

0963  860c  85 26           sta index2+1     addrh indirekter index #2
0964  860e  88              dey
0965  860f  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
0966  8611  18              clc
0967  8612  e5 6d           sbc lowtr        addrl ursprunganfang (verschieben)/
                                             temp.fac#2
0968  8614  49 ff           eor #$ff
0969  8616  18              clc
0970  8617  65 2f           adc txtend       addrl zeiger ende  basic-text
0971  8619  85 2f           sta txtend       addrl zeiger ende  basic-text
0972  861b  85 25           sta index2       addrl indirekter index #2
0973  861d  a5 30           lda txtend+1     addrh zeiger ende  basic-text
0974  861f  69 ff           adc #$ff
0975  8621  85 30           sta txtend+1     addrh zeiger ende  basic-text
0976  8623  e5 6e           sbc lowtr+1      addrh ursprunganfang (verschieben)
                                             temp.fac#2
0977  8625  aa              tax
0978  8626  38              sec
0979  8627  a5 6d           lda lowtr        addrl ursprunganfang (verschieben)/
                                             temp.fac#2
0980  8629  e5 2f           sbc txtend       addrl zeiger ende  basic-text
0981  862b  a8              tay
0982  862c  b0 03           bcs qdect1
0983  862e  e8              inx
0984  862f  c6 26           dec index2+1     addrh indirekter index #2
0985  8631  18        qdect1 clc
0986  8632  65 22           adc index1       addrl indirekter index #1
0987  8634  90 03           bcc mloop        verschiebeschleife
0988  8636  c6 23           dec index1+1     addrh indirekter index #1
0989  8638  18              clc
0990  8639
0991  8639
0992  8639  ===> verschiebeschleife <===
0993  8639
0994  8639  b1 22     mloop  lda (index1),y   addrl indirekter index #1
0995  863b  91 25           sta (index2),y   addrl indirekter index #2
0996  863d  c8              iny
0997  863e  d0 f9           bne mloop        verschiebeschleife
0998  8640  e6 23           inc index1+1     addrh indirekter index #1
0999  8642  e6 26           inc index2+1     addrh indirekter index #2
1000  8644  ca              dex
1001  8645  d0 f2           bne mloop        verschiebeschleife
1002  8647
1003  8647
1004  8647  ===> test: programmzeile einfügen <===
1005  8647
1006  8647  20 47 8a  nodel  jsr clearc       basic-routine 'clr'
1007  864a  20 a4 86         jsr lnkprg       linkadressen berechnen
1008  864d  a0 00           ldy #$00
1009  864f  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
1010  8651  d0 03           bne nodele       programmzeile einfügen
1011  8653  4c ca 85         jmp main         hauptroutine basic-interpreter
1012  8656
1013  8656
1014  8656  ===> programmzeile einfügen <===
1015  8656
1016  8656  18        nodele clc
```

```
zeile adr.  obj.-code   source-code

1017  8657  a5 2f           lda txtend       addrl zeiger ende  basic-text
1018  8659  a4 30           ldy txtend+1     addrh zeiger ende  basic-text
1019  865b  85 67           sta hightr       addrl ursprungende (verschieben)/
                                              temp.fac#1
1020  865d  84 68           sty hightr+1     addrh ursprungende (verschieben)/
                                              temp.fac#1
1021  865f  65 0e           adc count        allgemeiner zähler
1022  8661  90 01           bcc nodel1
1023  8663  c8             iny
1024  8664  18       nodel1 clc
1025  8665  69 04           adc #$04
1026  8667  90 01           bcc nodelc
1027  8669  c8             iny
1028  866a  85 64      nodelc sta highds       addrl zielende (verschieben)/
                                              temp.fac#1
1029  866c  84 65           sty highds+1     addrh zielende (verschieben/
                                              temp.fac#1
1030  866e  20 15 88        jsr bltut        blockverschiebe-rout. für programm
1031  8671  a0 00           ldy #$00
1032  8673  a9 01           lda #$01
1033  8675  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1034  8677  c8             iny
1035  8678  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1036  867a  c8             iny
1037  867b  a5 1b           lda linnum       lo-byte akt. zeilennummer
1038  867d  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1039  867f  a5 1c           lda linnum+1     hi-byte akt. zeilennummer
1040  8681  c8             iny
1041  8682  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1042  8684  c8             iny
1043  8685  98             tya
1044  8686  18             clc
1045  8687  65 6d           adc lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1046  8689  85 6d           sta lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1047  868b  90 02           bcc main2
1048  868d  e6 6e           inc lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
1049  868f  a4 0e      main2  ldy count        allgemeiner zähler
1050  8691  88             dey
1051  8692  b1 85      stolop lda (txtptr),y   addrl zeiger auf akt. term
1052  8694  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1053  8696  88             dey
1054  8697  c6 0e           dec count        allgemeiner zähler
1055  8699  d0 f7           bne stolop
1056  869b
1057  869b
1058  869b  ===> linkadr. berechnen, ready <===
1059  869b
1060  869b  20 a4 86  fini   jsr lnkprg       linkadressen berechnen
1061  869e  20 40 8a        jsr runc         basic-zeiger inititial., 'clr'
1062  86a1  4c ca 85        jmp main         hauptroutine basic-interpreter
```

```
zeile adr.  obj.-code   source-code


1063  86a4
1064  86a4
1065  86a4  ===> linkadressen berechnen <===
1066  86a4
1067  86a4  a5 2d     lnkprg lda txttab       addrl zeiger anfang  basic-text
1068  86a6  a4 2e            ldy txttab+1     addrh zeiger anfang  basic-text
1069  86a8  85 22            sta index1       addrl indirekter index #1
1070  86aa  84 23            sty index1+1     addrh indirekter index #1
1071  86ac  18               clc
1072  86ad  a0 00     chead  ldy #$00
1073  86af  b1 22            lda (index1),y   addrl indirekter index #1
1074  86b1  d0 05            bne chea3
1075  86b3  c8               iny
1076  86b4  b1 22            lda (index1),y   addrl indirekter index #1
1077  86b6  f0 1c            beq lnkrts
1078  86b8  a0 04     chea3  ldy #$04
1079  86ba  c8        czloop iny
1080  86bb  b1 22            lda (index1),y   addrl indirekter index #1
1081  86bd  d0 fb            bne czloop
1082  86bf  c8               iny
1083  86c0  98               tya
1084  86c1  65 22            adc index1       addrl indirekter index #1
1085  86c3  aa               tax
1086  86c4  a0 00            ldy #$00
1087  86c6  91 22            sta (index1),y   addrl indirekter index #1
1088  86c8  98               tya
1089  86c9  65 23            adc index1+1     addrh indirekter index #1
1090  86cb  c8               iny
1091  86cc  91 22            sta (index1),y   addrl indirekter index #1
1092  86ce  86 22            stx index1       addrl indirekter index #1
1093  86d0  85 23            sta index1+1     addrh indirekter index #1
1094  86d2  90 d9            bcc chead
1095  86d4  18        lnkrts clc
1096  86d5  a5 22            lda index1       addrl indirekter index #1
1097  86d7  a4 23            ldy index1+1     addrh indirekter index #1
1098  86d9  69 02            adc #$02
1099  86db  90 01            bcc lmkrt1
1100  86dd  c8               iny
1101  86de  85 2f     lmkrt1 sta txtend       addrl zeiger ende  basic-text
1102  86e0  84 30            sty txtend+1     addrh zeiger ende  basic-text
1103  86e2  60               rts
1104  86e3
1105  86e3
1106  86e3  ===> eingabe einer zeile <===
1107  86e3
1108  86e3  a5 88     inlin  lda buffpt       addrl zeiger auf eingabe-puffer
1109  86e5  a4 89            ldy buffpt+1     addrh zeiger auf eingabe-puffer
1110  86e7  a2 01            ldx #$01
1111  86e9  85 22            sta index1       addrl indirekter index #1
1112  86eb  84 23            sty index1+1     addrh indirekter index #1
1113  86ed  86 24            stx index1+2     bank indirekter index #1
1114  86ef  a0 00            ldy #$00
1115  86f1  84 0e     inlinc sty count        allgemeiner zähler
1116  86f3  20 ee bb         jsr basin        eingabe 1 char vom akt. kanal
                                              (chn-vektor)
1117  86f6  c9 0d            cmp #$0d
1118  86f8  f0 0c            beq fiminl       00 in puffer, ausgabe cr+(lf)
1119  86fa  a4 0e            ldy count        allgemeiner zähler
```

```
zeile adr.  obj.-code   source-code

1120  86fc  91 22           sta (index1),y   addrl indirekter index #1
1121  86fe  c8              iny
1122  86ff  c0 a1           cpy #$a1
1123  8701  90 ee           bcc inlinc
1124  8703  4c 85 b7        jmp errlen       ausgabe '?string too long', ready
1125  8706
1126  8706
1127  8706  ===> 00 in puffer, ausgabe cr+(lf) <===
1128  8706
1129  8706  a4 0e     fiminl ldy count        allgemeiner zähler
1130  8708  a9 00           lda #$00
1131  870a  91 22           sta (index1),y   addrl indirekter index #1
1132  870c  a5 1a           lda channl       speicherstelle für aktiven kanal
1133  870e  d0 03           bne fimin1
1134  8710  20 c8 8e        jsr ocrlf        ausgabe cr (+lf) auf akt. kanal
1135  8713  a4 23     fimin1 ldy index1+1     addrh indirekter index #1
1136  8715  a6 22           ldx index1       addrl indirekter index #1
1137  8717  d0 01           bne fimin2
1138  8719  88              dey
1139  871a  ca        fimin2 dex
1140  871b  8a              txa
1141  871c  a2 01           ldx #$01
1142  871e  60              rts
1143  871f
1144  871f
1145  871f  ===> startadr. von prg.zeile berechnen <===
1146  871f
1147  871f  a5 2d     fndlin lda txttab       addrl zeiger anfang  basic-text
1148  8721  a6 2e           ldx txttab+1     addrh zeiger anfang  basic-text
1149  8723
1150  8723
1151  8723  ===> startadr. der zeile in 'lowtr' <===
1152  8723
1153  8723  85 6d     fndlnc sta lowtr        addrl ursprunganfang (verschieben)//
                                             temp.fac#2
1154  8725  86 6e           stx lowtr+1      addrh ursprunganfang (verschieben)
                                             temp.fac#2
1155  8727  a0 00           ldy #$00
1156  8729  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)//
                                             temp.fac#2
1157  872b  d0 05           bne fndln3
1158  872d  c8              iny
1159  872e  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)//
                                             temp.fac#2
1160  8730  f0 0a           beq flnrt
1161  8732  a0 03     fndln3 ldy #$03
1162  8734  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)//
                                             temp.fac#2
1163  8736  c5 1c           cmp linnum+1     hi-byte akt. zeilennummer
1164  8738  f0 04           beq fndl20
1165  873a  90 0b           bcc affrts
1166  873c  18        flnrt  clc
1167  873d  60        flnrts rts
1168  873e
1169  873e
1170  873e  88        fndl20 dey
1171  873f  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)//
                                             temp.fac#2
```

```
zeile adr.  obj.-code   source-code

1172  8741  c5 1b           cmp linnum      lo-byte akt. zeilennummer
1173  8743  f0 f8           beq flnrts
1174  8745  b0 f5           bcs flnrt
1175  8747  a0 01    affrts ldy #$01
1176  8749  b1 6d           lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                            temp.fac#2
1177  874b  aa              tax
1178  874c  88              dey
1179  874d  b1 6d           lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                            temp.fac#2
1180  874f  90 d2           bcc fndlnc      startadr. der zeile in 'lowtr'
1181  8751  6c 88 02  xeqcm  jmp (igone)     addrl dispatcher
1182  8754
1183  8754
1184  8754  ===> dispatcher-routine <===
1185  8754
1186  8754  20 26 ba  ngone  jsr chrget      nächstes zeichen nach ac (ind.jmp)
1187  8757
1188  8757
1189  8757  ===> test gültiges befehlszeichen, weiter <===
1190  8757
1191  8757  20 a8 87  xeqdir jsr xeqcm3      basic-befehl (token in ac) ausführen
1192  875a
1193  875a
1194  875a  ===> neues statement, stoptest <===
1195  875a
1196  875a  20 e1 ff  newstt jsr kstop       test stop-taste
1197  875d  d0 1c           bne nstt1
1198  875f  38              sec
1199  8760  4c af 8b        jmp stopc       ansprung bei gedr. stop-taste, test
                                            'trap' 'break'
1200  8763
1201  8763
1202  8763  b1 85    sav42  lda (txtptr),y  addrl zeiger auf akt. term
1203  8765  d0 08           bne nstt4
1204  8767  c8              iny
1205  8768  b1 85           lda (txtptr),y  addrl zeiger auf akt. term
1206  876a  d0 03           bne nstt4
1207  876c  4c c0 85  nstt3  jmp ready       ausgabe 'ready', ready-status
1208  876f
1209  876f
1210  876f  a0 03    nstt4  ldy #$03
1211  8771  b1 85           lda (txtptr),y  addrl zeiger auf akt. term
1212  8773  85 42           sta curlin      lo-byte akt. zeilennummer
1213  8775  c8              iny
1214  8776  b1 85           lda (txtptr),y  addrl zeiger auf akt. term
1215  8778  85 43           sta curlin+1    hi-byte akt. zeilennummer
1216  877a  60              rts
1217  877b
1218  877b
1219  877b  20 57 9d  nstt1  jsr tstdir      test direktmodus, hi-byt zeile=$ff
1220  877e  f0 0c           beq nstt2
1221  8780  a5 85    nstt9  lda txtptr      addrl zeiger auf akt. term
1222  8782  a4 86           ldy txtptr+1    addrh zeiger auf akt. term
1223  8784  85 46           sta oldtxt      addrl zeiger letzter basic-befehl
1224  8786  84 47           sty oldtxt+1    addrh zeiger letzter basic-befehl
1225  8788  ba              tsx
1226  8789  8e 9c 02        stx oldstk      stack zeiger vor 'trap' ausführung
```

```
zeile adr.  obj.-code   source-code

1227  878c  a0 00     nstt2  ldy #$00
1228  878e  b1 85            lda (txtptr),y   addrl zeiger auf akt. term
1229  8790  d0 3f            bne morsts
1230  8792  20 57 9d         jsr tstdir       test direktmodus, hi-byt zeile=$ff
1231  8795  f0 d5            beq nstt3
1232  8797  a0 01            ldy #$01
1233  8799  20 63 87         jsr sav42
1234  879c  98                tya
1235  879d  18                clc
1236  879e  65 85            adc txtptr       addrl zeiger auf akt. term
1237  87a0  85 85            sta txtptr       addrl zeiger auf akt. term
1238  87a2  90 ad            bcc xeqcm
1239  87a4  e6 86            inc txtptr+1     addrh zeiger auf akt. term
1240  87a6  d0 a9            bne xeqcm
1241  87a8
1242  87a8
1243  87a8  ===> basic-befehl (token in ac) ausführen <===
1244  87a8
1245  87a8  f0 6a     xeqcm3 beq ffrts
1246  87aa  c9 ff     xeqcm2 cmp #$ff
1247  87ac  f0 2a            beq snerr1       ausgabe '?syntax error', ready
1248  87ae  38                sec
1249  87af  e9 80            sbc #$80
1250  87b1  90 1b            bcc glet         sprung zur basicrout. 'let'
1251  87b3  c9 23            cmp #$23
1252  87b5  90 0a            bcc nstt6        adr. der basic-routine auf stapel,
                                              ausführen
1253  87b7  c9 4b            cmp #$4b
1254  87b9  90 1d            bcc snerr1       ausgabe '?syntax error', ready
1255  87bb  c9 67            cmp #$67
1256  87bd  b0 19            bcs snerr1       ausgabe '?syntax error', ready
1257  87bf  e9 27            sbc #$27
1258  87c1
1259  87c1
1260  87c1  ===> adr. der basic-routine auf stapel, ausführen <===
1261  87c1
1262  87c1  0a        nstt6  asl a
1263  87c2  a8               tay
1264  87c3  b9 28 80         lda stmdsp+1,y   adress-tabelle der basic-befehle
1265  87c6  48               pha
1266  87c7  b9 27 80         lda stmdsp,y     adress-tabelle der basic-befehle
1267  87ca  48               pha
1268  87cb  4c 26 ba         jmp chrget       nächstes zeichen nach ac (ind.jmp)
1269  87ce
1270  87ce
1271  87ce  4c 8a 8d  glet   jmp let          sprung zur basicrout. 'let'
1272  87d1
1273  87d1
1274  87d1  c9 3a     morsts cmp #$3a
1275  87d3  d0 03            bne snerr1       ausgabe '?syntax error', ready
1276  87d5  4c 51 87         jmp xeqcm
1277  87d8
1278  87d8
1279  87d8  4c 4f 97  snerr1 jmp snerr        ausgabe '?syntax error', ready
1280  87db
1281  87db
1282  87db  ===> stapel-suchroutine for-next/goto <===
1283  87db
```

```
zeile adr.  obj.-code   source-code

1284  87db  ba        fndfor tsx
1285  87dc  e8               inx
1286  87dd  e8               inx
1287  87de  e8               inx
1288  87df  e8               inx
1289  87e0
1290  87e0
1291  87e0  ===> nächstes stapel-byte <===
1292  87e0
1293  87e0  bd 01 01  ffloop lda stack+1,x    6509 cpu-stack
1294  87e3  c9 81             cmp #$81
1295  87e5  d0 2d             bne ffrts
1296  87e7  a5 56             lda lstpnt+2    bank zeiger letzter string
1297  87e9  10 0f             bpl cmpfor      mit 'for-next'-zeiger vergleichen
1298  87eb  bd 02 01          lda stack+2,x   6509 cpu-stack
1299  87ee  85 54             sta lstpnt      addrl zeiger letzter string
1300  87f0  bd 03 01          lda stack+3,x   6509 cpu-stack
1301  87f3  85 55             sta lstpnt+1    addrh zeiger letzter string
1302  87f5  bd 04 01          lda stack+4,x   6509 cpu-stack
1303  87f8  85 56             sta lstpnt+2    bank zeiger letzter string
1304  87fa
1305  87fa
1306  87fa  ===> mit 'for-next'-zeiger vergleichen <===
1307  87fa
1308  87fa  dd 04 01  cmpfor cmp stack+4,x    6509 cpu-stack
1309  87fd  d0 0e             bne addfrs      stapelzeiger um 19 erhöhen
1310  87ff  a5 55             lda lstpnt+1    addrh zeiger letzter string
1311  8801  dd 03 01          cmp stack+3,x   6509 cpu-stack
1312  8804  d0 07             bne addfrs      stapelzeiger um 19 erhöhen
1313  8806  a5 54             lda lstpnt      addrl zeiger letzter string
1314  8808  dd 02 01          cmp stack+2,x   6509 cpu-stack
1315  880b  f0 07             beq ffrts
1316  880d
1317  880d
1318  880d  ===> stapelzeiger um 19 erhöhen <===
1319  880d
1320  880d  8a        addfrs txa
1321  880e  18               clc
1322  880f  69 13            adc #$13
1323  8811  aa               tax
1324  8812  d0 cc            bne ffloop      nächstes stapel-byte
1325  8814  60        ffrts  rts
1326  8815
1327  8815
1328  8815  ===> blockverschiebe-rout. für programm <===
1329  8815
1330  8815  20 b0 88  bltut  jsr reasnt      test speicherplatz für programm
1331  8818  85 2f             sta txtend     addrl zeiger ende  basic-text
1332  881a  84 30             sty txtend+1   addrh zeiger ende  basic-text
1333  881c  90 0a             bcc bltu       blockverschiebe-rout. akt. bank
1334  881e
1335  881e
1336  881e  ===> blockverschiebe-rout. für variable <===
1337  881e
1338  881e  20 77 88  bltuv  jsr reason      test speicherplatz für variable
1339  8821  85 39             sta strend     addrl ende benutzter ram-bereich
1340  8823  84 3a             sty strend+1   addrh ende benutzter ram-bereich
1341  8825  20 87 ba          jsr mapvar     umsch. auf bank # 2
```

```
zeile adr.  obj.-code   source-code

1342  8828
1343  8828
1344  8828  ===> blockverschiebe-rout. akt. bank <===
1345  8828
1346  8828  38        bltu   sec
1347  8829  a5 67             lda hightr       addrl ursprungende (verschieben)/
                                               temp.fac#1
1348  882b  e5 6d             sbc lowtr        addrl ursprunganfang (verschieben)/
                                               temp.fac#2
1349  882d  85 22             sta index1       addrl indirekter index #1
1350  882f  a8               tay
1351  8830  a5 68             lda hightr+1     addrh ursprungende (verschieben)/
                                               temp.fac#1
1352  8832  e5 6e             sbc lowtr+1      addrh ursprunganfang (verschieben)
                                               temp.fac#2
1353  8834  aa               tax
1354  8835  e8               inx
1355  8836  98               tya
1356  8837  f0 23            beq decblt
1357  8839  a5 67             lda hightr       addrl ursprungende (verschieben)/
                                               temp.fac#1
1358  883b  38               sec
1359  883c  e5 22             sbc index1       addrl indirekter index #1
1360  883e  85 67             sta hightr       addrl ursprungende (verschieben)/
                                               temp.fac#1
1361  8840  b0 03            bcs blt1
1362  8842  c6 68             dec hightr+1     addrh ursprungende (verschieben)/
                                               temp.fac#1
1363  8844  38               sec
1364  8845  a5 64     blt1    lda highds       addrl zielende (verschieben)/
                                               temp.fac#1
1365  8847  e5 22             sbc index1       addrl indirekter index #1
1366  8849  85 64             sta highds       addrl zielende (verschieben)/
                                               temp.fac#1
1367  884b  b0 08            bcs moren1
1368  884d  c6 65             dec highds+1     addrh zielende (verschieben/
                                               temp.fac#1
1369  884f  90 04            bcc moren1
1370  8851
1371  8851
1372  8851  ===> verschiebeschleife <===
1373  8851
1374  8851  b1 67     bltlp   lda (hightr),y   addrl ursprungende (verschieben)/
                                               temp.fac#1
1375  8853  91 64             sta (highds),y   addrl zielende (verschieben)/
                                               temp.fac#1
1376  8855  88        moren1  dey
1377  8856  d0 f9            bne bltlp        verschiebeschleife
1378  8858  b1 67            lda (hightr),y   addrl ursprungende (verschieben)/
                                               temp.fac#1
1379  885a  91 64            sta (highds),y   addrl zielende (verschieben)/
                                               temp.fac#1
1380  885c  c6 68     decblt  dec hightr+1     addrh ursprungende (verschieben)/
                                               temp.fac#1
1381  885e  c6 65             dec highds+1     addrh zielende (verschieben/
                                               temp.fac#1
1382  8860  ca               dex
1383  8861  d0 f2            bne moren1
```

```
zeile adr.  obj.-code   source-code

1384  8863  4c 8c ba          jmp mapusr       umsch. auf bank # 1
1385  8866
1386  8866
1387  8866  ===> test stapelplatz <===
1388  8866
1389  8866  0a         getstk asl a
1390  8867  69 34             adc #$34
1391  8869  b0 07             bcs oserr        ausgabe '?out of stack error', ready
1392  886b  85 22             sta index1       addrl indirekter index #1
1393  886d  ba                tsx
1394  886e  e4 22             cpx index1       addrl indirekter index #1
1395  8870  b0 3d             bcs rearts
1396  8872
1397  8872
1398  8872  ===> ausgabe '?out of stack error', ready <===
1399  8872
1400  8872  a2 50      oserr  ldx #$50
1401  8874  4c 52 85          jmp error        ind. jmp zur fehlerroutine
1402  8877
1403  8877
1404  8877  ===> test speicherplatz für variable <===
1405  8877
1406  8877  c4 3c      reason cpy fretop+1      addrh top of string free space
1407  8879  90 34             bcc rearts
1408  887b  d0 04             bne trymor
1409  887d  c5 3b             cmp fretop       addrl top of string free space
1410  887f  90 2e             bcc rearts
1411  8881  48         trymor pha
1412  8882  a2 09             ldx #$09
1413  8884  98                tya
1414  8885  48         reasav pha
1415  8886  b5 63             lda jmper+2,x    addrh sprung zu functions-rout.
1416  8888  ca                dex
1417  8889  10 fa             bpl reasav
1418  888b  20 b5 ad          jsr garba2       garbage collect (müll entfernen)
1419  888e  a2 f7             ldx #$f7
1420  8890  68         reasto pla
1421  8891  95 6d             sta lowtr,x      addrl ursprunganfang (verschieben)/
                                               temp.fac#2
1422  8893  e8                inx
1423  8894  30 fa             bmi reasto
1424  8896  68                pla
1425  8897  a8                tay
1426  8898  68                pla
1427  8899  c4 3c             cpy fretop+1     addrh top of string free space
1428  889b  90 12             bcc rearts
1429  889d  d0 04             bne omerrc       ausgabe '?out of memory error',
                                               ready
1430  889f  c5 3b             cmp fretop       addrl top of string free space
1431  88a1  90 0c             bcc rearts
1432  88a3  4c 50 85  omerrc  jmp omerr        ausgabe '?out of memory error',
                                               ready
1433  88a6
1434  88a6
1435  88a6  20 2c 99  sav73   jsr ptrget       variable suchen/anlegen, deren name
                                               folgt
1436  88a9  85 54     sav74   sta lstpnt       addrl zeiger letzter string
1437  88ab  84 55             sty lstpnt+1     addrh zeiger letzter string
```

```
zeile adr.  obj.-code   source-code

1438  88ad  86 56           stx lstpnt+2     bank zeiger letzter string
1439  88af  60       rearts rts
1440  88b0
1441  88b0
1442  88b0  ===> test speicherplatz für programm <===
1443  88b0
1444  88b0  c4 89     reasnt cpy buffpt+1     addrh zeiger auf eingabe-puffer
1445  88b2  90 fb            bcc rearts
1446  88b4  d0 04            bne omerrt
1447  88b6  c5 88            cmp buffpt       addrl zeiger auf eingabe-puffer
1448  88b8  90 f5            bcc rearts
1449  88ba  a2 56     omerrt ldx #$56
1450  88bc  4c 52 85         jmp error        ind. jmp zur fehlerroutine
1451  88bf
1452  88bf
1453  88bf  6c 84 02  crunch jmp (icrnch)     addrl token-umwandlungsroutine
1454  88c2
1455  88c2
1456  88c2  ===> umwandlung text in basic-tokens <===
1457  88c2
1458  88c2  a5 85     ncnnch lda txtptr       addrl zeiger auf akt. term
1459  88c4  48               pha
1460  88c5  a5 86            lda txtptr+1     addrh zeiger auf akt. term
1461  88c7  48               pha
1462  88c8  20 29 ba  crun05 jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
1463  88cb  4c d1 88         jmp crun20
1464  88ce
1465  88ce
1466  88ce  20 26 ba  crun10 jsr chrget       nächstes zeichen nach ac (ind.jmp)
1467  88d1  90 fb     crun20 bcc crun10
1468  88d3  c9 00            cmp #$00
1469  88d5  f0 59            beq crun90       token-umwandlung ende
1470  88d7  c9 3a            cmp #$3a
1471  88d9  f0 f3            beq crun10
1472  88db  c9 3f            cmp #$3f
1473  88dd  d0 08            bne crun30
1474  88df  a9 99            lda #$99
1475  88e1  a0 00            ldy #$00
1476  88e3  91 85            sta (txtptr),y   addrl zeiger auf akt. term
1477  88e5  f0 e7            beq crun10
1478  88e7  c9 80     crun30 cmp #$80
1479  88e9  90 0b            bcc crun40       tokenroutine, test '"'
1480  88eb  c9 ff            cmp #$ff
1481  88ed  f0 df            beq crun10
1482  88ef  a0 01            ldy #$01
1483  88f1  20 3f 89         jsr kloop
1484  88f4  f0 d2            beq crun05
1485  88f6
1486  88f6
1487  88f6  ===> tokenroutine, test '"' <===
1488  88f6
1489  88f6  c9 22     crun40 cmp #$22
1490  88f8  d0 0d            bne crun60
1491  88fa  20 26 ba  crun50 jsr chrget       nächstes zeichen nach ac (ind.jmp)
1492  88fd  c9 00            cmp #$00
1493  88ff  f0 2f            beq crun90       token-umwandlung ende
1494  8901  c9 22            cmp #$22
```

```
zeile adr.  obj.-code   source-code

1495  8903  f0 c9           beq crun10
1496  8905  d0 f3           bne crun50
1497  8907  20 57 89  crun60 jsr reser       akt text mit befehlsworttabelle
                                             vergleichen
1498  890a  90 c2           bcc crun10
1499  890c  c0 00           cpy #$00
1500  890e  f0 03           beq crun70
1501  8910  20 3f 89        jsr kloop
1502  8913  a5 0e     crun70 lda count       allgemeiner zähler
1503  8915  a0 00           ldy #$00
1504  8917  91 85           sta (txtptr),y   addrl zeiger auf akt. term
1505  8919  c9 8f           cmp #$8f
1506  891b  f0 0d           beq crun80
1507  891d  c9 83           cmp #$83
1508  891f  d0 ad           bne crun10
1509  8921  20 26 ba        jsr chrget       nächstes zeichen nach ac (ind.jmp)
1510  8924  20 df 8c        jsr data         basic-routine 'data'
1511  8927  4c c8 88        jmp crun05
1512  892a
1513  892a
1514  892a  20 26 ba  crun80 jsr chrget       nächstes zeichen nach ac (ind.jmp)
1515  892d  20 77 8c        jsr rem          basic-routine 'rem'
1516  8930
1517  8930
1518  8930  ===> token-umwandlung ende <===
1519  8930
1520  8930  a6 85     crun90 ldx txtptr       addrl zeiger auf akt. term
1521  8932  68               pla
1522  8933  85 86           sta txtptr+1     addrh zeiger auf akt. term
1523  8935  68               pla
1524  8936  85 85           sta txtptr       addrl zeiger auf akt. term
1525  8938  38               sec
1526  8939  8a               txa
1527  893a  e5 85           sbc txtptr       addrl zeiger auf akt. term
1528  893c  a8               tay
1529  893d  c8               iny
1530  893e  60               rts
1531  893f
1532  893f
1533  893f  18        kloop  clc
1534  8940  98               tya
1535  8941  65 85           adc txtptr       addrl zeiger auf akt. term
1536  8943  85 22           sta index1       addrl indirekter index #1
1537  8945  a5 86           lda txtptr+1     addrh zeiger auf akt. term
1538  8947  69 00           adc #$00
1539  8949  85 23           sta index1+1     addrh indirekter index #1
1540  894b  a0 00           ldy #$00
1541  894d
1542  894d
1543  894d  ===> zeiger 'index1' nach 'txtptr' <===
1544  894d
1545  894d  b1 22     kloop2 lda (index1),y   addrl indirekter index #1
1546  894f  91 85           sta (txtptr),y   addrl zeiger auf akt. term
1547  8951  c8               iny
1548  8952  c9 00           cmp #$00
1549  8954  d0 f7           bne kloop2       zeiger 'index1' nach 'txtptr'
1550  8956  60               rts
1551  8957
```

```
zeile adr.  obj.-code   source-code

1552  8957
1553  8957  ===> akt text mit befehlsworttabelle vergleichen <===
1554  8957
1555  8957  a9 80     reser  lda #$80
1556  8959  a0 ef            ldy #$ef
1557  895b  85 23            sta index1+1     addrh indirekter index #1
1558  895d  84 22            sty index1       addrl indirekter index #1
1559  895f  a0 00            ldy #$00
1560  8961  84 0e            sty count        allgemeiner zähler
1561  8963  88               dey
1562  8964  c8        rese10 iny
1563  8965  b1 85     rese20 lda (txtptr),y   addrl zeiger auf akt. term
1564  8967  38               sec
1565  8968  f1 22            sbc (index1),y   addrl indirekter index #1
1566  896a  f0 f8            beq rese10
1567  896c  c9 80            cmp #$80
1568  896e  f0 18            beq rese60
1569  8970  20 64 ba  rese30 jsr ldi1y        lade (index1),y in ac
1570  8973  30 03            bmi rese40
1571  8975  c8               iny
1572  8976  d0 f8            bne rese30
1573  8978  c8        rese40 iny
1574  8979  e6 0e            inc count        allgemeiner zähler
1575  897b  18               clc
1576  897c  98               tya
1577  897d  20 19 ab         jsr sav14
1578  8980  18               clc
1579  8981  a0 00            ldy #$00
1580  8983  20 64 ba         jsr ldi1y        lade (index1),y in ac
1581  8986  d0 dd            bne rese20
1582  8988  05 0e     rese60 ora count        allgemeiner zähler
1583  898a  85 0e            sta count        allgemeiner zähler
1584  898c  60               rts
1585  898d
1586  898d                   .end
1587  898d                   .lib bverbs1
```

```
zeile adr.  obj.-code   source-code

1589  898d
1590  898d  ===> basic-routine 'list' <===
1591  898d
1592  898d  20 f4 af  list   jsr range        zeilenbereich einlesen, test gültig
1593  8990  d0 01            bne lstend       pla,pla, test auf basic-textende
1594  8992  60              rts
1595  8993
1596  8993
1597  8993  ===> pla,pla, test auf basic-textende <===
1598  8993
1599  8993  68        lstend pla
1600  8994  68               pla
1601  8995
1602  8995
1603  8995  ===> test basic-textende erreicht <===
1604  8995
1605  8995  a0 01     list4  ldy #$01
1606  8997  84 13           sty dores        flag für token o. ascii/garbage-flag
1607  8999  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
1608  899b  d0 05           bne list44       stop-taste abfragen/ausführen
1609  899d  88              dey
1610  899e  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
1611  89a0  f0 4c           beq grody        sprung nach 'ready'
1612  89a2
1613  89a2
1614  89a2  ===> stop-taste abfragen/ausführen <===
1615  89a2
1616  89a2  a0 01     list44 ldy #$01
1617  89a4  20 e1 ff         jsr kstop        test stop-taste
1618  89a7  d0 04            bne list5        ausführung 'list'
1619  89a9  18               clc
1620  89aa  4c af 8b         jmp stopc        ansprung bei gedr. stop-taste, test
                                             'trap' 'break'
1621  89ad
1622  89ad
1623  89ad  ===> ausführung 'list' <===
1624  89ad
1625  89ad  20 c8 8e  list5  jsr ocrlf        ausgabe cr (+lf) auf akt. kanal
1626  89b0  c8               iny
1627  89b1  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
1628  89b3  aa               tax
1629  89b4  c8               iny
1630  89b5  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
1631  89b7  c5 1c            cmp linnum+1     hi-byte akt. zeilennummer
1632  89b9  d0 04            bne tstdun       carry=1 ready, sonst test ob letzte
                                             prg.zeile (list)
1633  89bb  e4 1b            cpx linnum       lo-byte akt. zeilennummer
1634  89bd  f0 02            beq typlin       test ob letzte prg.zeile (list)
1635  89bf
1636  89bf
1637  89bf  ===> carry=1 ready, sonst test ob letzte prg.zeile (list) <===
1638  89bf
1639  89bf  b0 2d     tstdun bcs grody        sprung nach 'ready'
1640  89c1
```

```
zeile adr.  obj.-code   source-code

1641  89c1
1642  89c1  ===> test ob letzte prg.zeile (list) <===
1643  89c1
1644  89c1  84 54     typlin sty lstpnt      addrl zeiger letzter string
1645  89c3  20 b4 a3         jsr linprt      umw. hex/dez, lo=xr, hi=ac, druck
                                              zeilen #
1646  89c6  a9 20            lda #$20
1647  89c8  a4 54     prit4  ldy lstpnt      addrl zeiger letzter string
1648  89ca  29 7f            and #$7f
1649  89cc
1650  89cc
1651  89cc  ===> ausgabe 1 zeichen von ac, test '"' <===
1652  89cc
1653  89cc  20 3a b5  ploop  jsr ochr        ausgabe 1 zeichen (in ac)
1654  89cf  c9 22            cmp #$22
1655  89d1  d0 06            bne ploop1      test ob zeilenende erreicht
1656  89d3  a5 13            lda dores       flag für token o. ascii/garbage-flag
1657  89d5  49 ff            eor #$ff
1658  89d7  85 13            sta dores       flag für token o. ascii/garbage-flag
1659  89d9
1660  89d9
1661  89d9  ===> test ob zeilenende erreicht <===
1662  89d9
1663  89d9  c8        ploop1 iny
1664  89da  f0 12            beq grody       sprung nach 'ready'
1665  89dc  b1 6d            lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1666  89de  d0 11            bne qplop       umwandlung basic-tokens in text
1667  89e0  a8               tay
1668  89e1  b1 6d            lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1669  89e3  aa               tax
1670  89e4  c8               iny
1671  89e5  b1 6d            lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1672  89e7  86 6d            stx lowtr       addrl ursprunganfang (verschieben)/
                                              temp.fac#2
1673  89e9  85 6e            sta lowtr+1     addrh ursprunganfang (verschieben)
                                              temp.fac#2
1674  89eb  4c 95 89         jmp list4       test basic-textende erreicht
1675  89ee
1676  89ee
1677  89ee  4c c0 85  grody  jmp ready       sprung nach 'ready'
1678  89f1
1679  89f1
1680  89f1  6c 86 02  qplop  jmp (iqplop)    addrl token ausgabe expander rout.
1681  89f4
1682  89f4
1683  89f4  ===> umwandlung basic-tokens in text <===
1684  89f4
1685  89f4  10 d6     nqglop bpl ploop       ausgabe 1 zeichen von ac, test '"'
1686  89f6  c9 ff            cmp #$ff
1687  89f8  f0 d2            beq ploop       ausgabe 1 zeichen von ac, test '"'
1688  89fa  24 13            bit dores       flag für token o. ascii/garbage-flag
1689  89fc  30 ce            bmi ploop       ausgabe 1 zeichen von ac, test '"'
1690  89fe  aa               tax
1691  89ff  84 54            sty lstpnt      addrl zeiger letzter string
1692  8a01  a0 80            ldy #$80
```

```
zeile adr.  obj.-code   source-code

1693  8a03  84 23           sty index1+1     addrh indirekter index #1
1694  8a05  a0 ef           ldy #$ef
1695  8a07  84 22           sty index1       addrl indirekter index #1
1696  8a09  a0 00           ldy #$00
1697  8a0b  0a              asl a
1698  8a0c  f0 11           beq prit3b
1699  8a0e
1700  8a0e
1701  8a0e  ===> text zu token aus 'reslst' suchen <===
1702  8a0e
1703  8a0e  ca        resrch dex
1704  8a0f  10 0d           bpl prit3        textausgabe zu token aus 'reslst'
1705  8a11  e6 22     rescr1 inc index1       addrl indirekter index #1
1706  8a13  d0 02           bne restr2
1707  8a15  e6 23           inc index1+1     addrh indirekter index #1
1708  8a17  20 64 ba  restr2 jsr ldi1y       lade (index1),y in ac
1709  8a1a  10 f5           bpl rescr1
1710  8a1c  30 f0           bmi resrch       text zu token aus 'reslst' suchen
1711  8a1e
1712  8a1e
1713  8a1e  ===> textausgabe zu token aus 'reslst' <===
1714  8a1e
1715  8a1e  c8        prit3  iny
1716  8a1f  20 64 ba  prit3b jsr ldi1y        lade (index1),y in ac
1717  8a22  30 a4           bmi prit4
1718  8a24  20 3a b5        jsr ochr         ausgabe 1 zeichen (in ac)
1719  8a27  d0 f5           bne prit3        textausgabe zu token aus 'reslst'
1720  8a29
1721  8a29
1722  8a29  ===> basic-routine 'new' <===
1723  8a29
1724  8a29  d0 64     scrath bne stkrts
1725  8a2b  a9 00     scrtch lda #$00
1726  8a2d  a8               tay
1727  8a2e  91 2d            sta (txttab),y   addrl zeiger anfang  basic-text
1728  8a30  c8               iny
1729  8a31  91 2d            sta (txttab),y   addrl zeiger anfang  basic-text
1730  8a33  18               clc
1731  8a34  a5 2d            lda txttab       addrl zeiger anfang  basic-text
1732  8a36  69 02            adc #$02
1733  8a38  85 2f            sta txtend       addrl zeiger ende  basic-text
1734  8a3a  a5 2e            lda txttab+1     addrh zeiger anfang  basic-text
1735  8a3c  69 00            adc #$00
1736  8a3e  85 30            sta txtend+1     addrh zeiger ende  basic-text
1737  8a40
1738  8a40
1739  8a40  ===> basic-zeiger inititial., 'clr' <===
1740  8a40
1741  8a40  20 b8 b4  runc   jsr stxtpt       programmzeiger auf start
1742  8a43  a9 00            lda #$00
1743  8a45
1744  8a45
1745  8a45  ===> basic-routine 'clr', zero=0 rts <===
1746  8a45
1747  8a45  d0 48     clear  bne stkrts
1748  8a47
1749  8a47
1750  8a47  ===> basic-routine 'clr' <===
```

```
zeile adr.  obj.-code   source-code

1751  8a47
1752  8a47  20 18 bb  clearc jsr settop
1753  8a4a  a9 ff           lda #$ff
1754  8a4c  8d 97 02        sta trapno+1      addrh zeiger error trap
1755  8a4f  a2 03           ldx #$03
1756  8a51  bd 90 8a  clrpu  lda pudefs,x      printusing zeichen
1757  8a54  9d 73 02        sta pufill,x      print using: füllzeichen
1758  8a57  ca              dex
1759  8a58  10 f7           bpl clrpu
1760  8a5a  a9 0f           lda #$0f
1761  8a5c  8d 57 02        sta dfbank        vorgabe für bank-nummer
1762  8a5f  a9 00           lda #$00
1763  8a61  85 16           sta dsdesc        länge disc-status string
1764  8a63  20 16 bc        jsr clall         alle logischen files schliessen
1765  8a66  a5 31           lda vartab        addrl zeiger anfang einfache
                                              variable
1766  8a68  a4 32           ldy vartab+1      addrh zeiger anfang einfache
                                              variable
1767  8a6a  85 39           sta strend        addrl ende benutzter ram-bereich
1768  8a6c  84 3a           sty strend+1      addrh ende benutzter ram-bereich
1769  8a6e  85 35           sta arytab        addrl zeiger anfang array-tabelle
1770  8a70  84 36           sty arytab+1      addrh zeiger anfang array-tabelle
1771  8a72
1772  8a72
1773  8a72  ===> reset 'read'-,stack-,befehl-zeiger <===
1774  8a72
1775  8a72  20 97 8b  fload  jsr resto1        'read'-zeiger auf basicanfang
1776  8a75
1777  8a75
1778  8a75  ===> reset stack- u. basicbefehl-zeiger <===
1779  8a75
1780  8a75  a2 00     stkini ldx #$00
1781  8a77  86 1d           stx temmpt        zeiger auf zeitw. stringdescriptor
                                              (rel. offset)
1782  8a79  68              pla
1783  8a7a  a8              tay
1784  8a7b  68              pla
1785  8a7c  a2 fe           ldx #$fe
1786  8a7e  9a              txs
1787  8a7f  48              pha
1788  8a80  98              tya
1789  8a81  48              pha
1790  8a82  a9 00           lda #$00
1791  8a84  85 46           sta oldtxt        addrl zeiger letzter basic-befehl
1792  8a86  85 47           sta oldtxt+1      addrh zeiger letzter basic-befehl
1793  8a88  85 1a           sta channl        speicherstelle für aktiven kanal
1794  8a8a  8d 58 02        sta dolu          vorgabe für ausgabe-adresse
1795  8a8d  85 14           sta subflg        flag für indizierte variable
1796  8a8f  60        stkrts rts
1797  8a90
1798  8a90
1799  8a90  ===> printusing zeichen <===
1800  8a90
1801  8a90  20 2c 2e  pudefs .byte ' ,.$'
1801  8a93  24
1802  8a94
1803  8a94
1804  8a94  ===> basic-routine 'for' <===
```

```
zeile adr.  obj.-code   source-code

1805  8a94
1806  8a94  a9 80      for     lda #$80
1807  8a96  85 14              sta subflg      flag für indizierte variable
1808  8a98  20 8a 8d           jsr let         basic-routine 'let'
1809  8a9b  20 db 87           jsr fndfor      stapel-suchroutine for-next/goto
1810  8a9e  d0 05              bne notol
1811  8aa0  8a                 txa
1812  8aa1  69 10              adc #$10
1813  8aa3  aa                 tax
1814  8aa4  9a                 txs
1815  8aa5  68          notol  pla
1816  8aa6  68                 pla
1817  8aa7  a9 09              lda #$09
1818  8aa9  20 66 88           jsr getstk      test stapelplatz
1819  8aac  20 ed 8c           jsr datan       trennzeichen ':', zeilenende '0'
                                               suchen, offset yr
1820  8aaf  18                 clc
1821  8ab0  98                 tya
1822  8ab1  65 85              adc txtptr      addrl zeiger auf akt. term
1823  8ab3  48                 pha
1824  8ab4  a5 86              lda txtptr+1    addrh zeiger auf akt. term
1825  8ab6  69 00              adc #$00
1826  8ab8  48                 pha
1827  8ab9  a5 43              lda curlin+1    hi-byte akt. zeilennummer
1828  8abb  48                 pha
1829  8abc  a5 42              lda curlin      lo-byte akt. zeilennummer
1830  8abe  48                 pha
1831  8abf  a9 a4              lda #$a4
1832  8ac1  20 32 97           jsr synchr      test:folgt ascii o. token im akt.
                                               text, sonst error
1833  8ac4  20 04 b5           jsr chknum      prüfen, ob numerische variable
1834  8ac7  20 01 b5           jsr frmnum      numerischen ausdruck holen
1835  8aca  a5 76              lda facsgn      fac #1: vorzeichen
1836  8acc  09 7f              ora #$7f
1837  8ace  25 72              and facho       fac #1: mantisse
1838  8ad0  85 72              sta facho       fac #1: mantisse
1839  8ad2  a9 dd              lda #$dd
1840  8ad4  a0 8a              ldy #$8a
1841  8ad6  85 22              sta index1      addrl indirekter index #1
1842  8ad8  84 23              sty index1+1    addrh indirekter index #1
1843  8ada  4c 6d 96           jmp forpsh
1844  8add
1845  8add
1846  8add  ===> step mit 1 vorbesetzen <===
1847  8add
1848  8add  a9 9c      ldfone lda #$9c
1849  8adf  a0 9f             ldy #$9f
1850  8ae1  20 66 a1           jsr movfm       vari adr.lo=ac, hi=yr nach fac #1
1851  8ae4  20 29 ba           jsr chrgot      letztes zeichen erneut nach ac
                                               (indirekter sprung)
1852  8ae7  c9 a9              cmp #$a9
1853  8ae9  d0 06              bne oneon       vorzeichen, string-zeiger,
                                               'for'-token auf stapel
1854  8aeb  20 26 ba           jsr chrget      nächstes zeichen nach ac (ind.jmp)
1855  8aee  20 01 b5           jsr frmnum      numerischen ausdruck holen
1856  8af1
1857  8af1
1858  8af1  ===> vorzeichen, string-zeiger, 'for'-token auf stapel <===
```

```
zeile adr.  obj.-code   source-code

1859  8af1
1860  8af1  20 02 a2  oneon  jsr sign         vorzeichentest fac #1
1861  8af4  20 5e 96         jsr pushf
1862  8af7  a5 56            lda lstpnt+2     bank zeiger letzter string
1863  8af9  48               pha
1864  8afa  a5 55            lda lstpnt+1     addrh zeiger letzter string
1865  8afc  48               pha
1866  8afd  a5 54            lda lstpnt       addrl zeiger letzter string
1867  8aff  48               pha
1868  8b00  a9 81            lda #$81
1869  8b02  48               pha
1870  8b03  4c 5a 87         jmp newstt       neues statement, stoptest
1871  8b06
1872  8b06
1873  8b06  ===> basic-routine 'next' <===
1874  8b06
1875  8b06  d0 04     next   bne getfor       basic-routine 'next' mit variable
1876  8b08  a2 ff            ldx #$ff
1877  8b0a  30 03            bmi stxfor       basic-routine 'next' mit wert
1878  8b0c
1879  8b0c
1880  8b0c  ===> basic-routine 'next' mit variable <===
1881  8b0c
1882  8b0c  20 2c 99  getfor jsr ptrget       variable suchen/anlegen, deren name
                                               folgt
1883  8b0f
1884  8b0f
1885  8b0f  ===> basic-routine 'next' mit wert <===
1886  8b0f
1887  8b0f  20 a9 88  stxfor jsr sav74
1888  8b12  20 db 87         jsr fndfor       stapel-suchroutine for-next/goto
1889  8b15  f0 05            beq havfor
1890  8b17  a2 28            ldx #$28
1891  8b19  4c 52 85         jmp error        ind. jmp zur fehlerroutine
1892  8b1c
1893  8b1c
1894  8b1c  9a        havfor txs
1895  8b1d  8a               txa
1896  8b1e  18               clc
1897  8b1f  69 05            adc #$05
1898  8b21  48               pha
1899  8b22  69 06            adc #$06
1900  8b24  85 25            sta index2       addrl indirekter index #2
1901  8b26  68               pla
1902  8b27  a0 01            ldy #$01
1903  8b29  20 66 a1         jsr movfm        vari adr.lo=ac, hi=yr nach fac #1
1904  8b2c  ba               tsx
1905  8b2d  bd 0a 01         lda stack+10,x   6509 cpu-stack
1906  8b30  85 76            sta facsgn       fac #1: vorzeichen
1907  8b32  a5 54            lda lstpnt       addrl zeiger letzter string
1908  8b34  a4 55            ldy lstpnt+1     addrh zeiger letzter string
1909  8b36  a6 56            ldx lstpnt+2     bank zeiger letzter string
1910  8b38  20 5e a0         jsr ucnupk
1911  8b3b  20 4d 9e         jsr faddt        basic-routine '+' (dez. add.)
1912  8b3e  20 9f a1         jsr movvf        gk-zahl aus fac in variable
1913  8b41  a0 01            ldy #$01
1914  8b43  20 34 a2         jsr fcompn
1915  8b46  ba               tsx
```

```
zeile adr.  obj.-code   source-code

1916  8b47  38              sec
1917  8b48  fd 0a 01        sbc stack+10,x   6509 cpu-stack
1918  8b4b  f0 17           beq loopdn       daten vom stack, folgt ',' bearb.
                                             nächstes for-next
1919  8b4d  bd 10 01        lda stack+16,x   6509 cpu-stack
1920  8b50  85 42           sta curlin       lo-byte akt. zeilennummer
1921  8b52  bd 11 01        lda stack+17,x   6509 cpu-stack
1922  8b55  85 43           sta curlin+1     hi-byte akt. zeilennummer
1923  8b57  bd 13 01        lda stack+19,x   6509 cpu-stack
1924  8b5a  85 85           sta txtptr       addrl zeiger auf akt. term
1925  8b5c  bd 12 01        lda stack+18,x   6509 cpu-stack
1926  8b5f  85 86           sta txtptr+1     addrh zeiger auf akt. term
1927  8b61  4c 5a 87  newsgo jmp newstt      neues statement, stoptest
1928  8b64
1929  8b64
1930  8b64  ===> daten vom stack, folgt ',' bearb. nächstes for-next <===
1931  8b64
1932  8b64  8a        loopdn txa
1933  8b65  69 12           adc #$12
1934  8b67  aa              tax
1935  8b68  9a              txs
1936  8b69  20 29 ba        jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
1937  8b6c  c9 2c           cmp #$2c
1938  8b6e  d0 f1           bne newsgo
1939  8b70  20 26 ba        jsr chrget       nächstes zeichen nach ac (ind.jmp)
1940  8b73  20 0c 8b        jsr getfor       basic-routine 'next' mit variable
1941  8b76  4c 01 b5        jmp frmnum       numerischen ausdruck holen
1942  8b79
1943  8b79
1944  8b79  ===> basic-routine 'restore' <===
1945  8b79
1946  8b79  f0 1c     restor beq resto1       'read'-zeiger auf basicanfang
1947  8b7b  20 e5 b4        jsr getpin       dezimal-zahl in klammer nach linnum
                                             einlesen
1948  8b7e  84 1b           sty linnum       lo-byte akt. zeilennummer
1949  8b80  85 1c           sta linnum+1     hi-byte akt. zeilennummer
1950  8b82  20 8c ba        jsr mapusr       umsch. auf bank # 1
1951  8b85  20 1f 87        jsr fndlin       startadr. von prg.zeile berechnen
1952  8b88  90 1b           bcc reserr
1953  8b8a  a5 6d           lda lowtr        addrl ursprunganfang (verschieben)/
                                             temp.fac#2
1954  8b8c  e9 01           sbc #$01
1955  8b8e  85 4b           sta datptr       addrl zeiger auf data nach 'read'
1956  8b90  a5 6e           lda lowtr+1      addrh ursprunganfang (verschieben)
                                             temp.fac#2
1957  8b92  e9 00           sbc #$00
1958  8b94  85 4c           sta datptr+1     addrh zeiger auf data nach 'read'
1959  8b96  60              rts
1960  8b97
1961  8b97
1962  8b97  ===> 'read'-zeiger auf basicanfang <===
1963  8b97
1964  8b97  38        resto1 sec
1965  8b98  a5 2d           lda txttab       addrl zeiger anfang  basic-text
1966  8b9a  e9 01           sbc #$01
1967  8b9c  85 4b           sta datptr       addrl zeiger auf data nach 'read'
1968  8b9e  a5 2e           lda txttab+1     addrh zeiger anfang  basic-text
```

```
zeile adr.  obj.-code   source-code

1969  8ba0  e9 00           sbc #$00
1970  8ba2  85 4c           sta datptr+1     addrh zeiger auf data nach 'read'
1971  8ba4  60       runm10 rts
1972  8ba5
1973  8ba5
1974  8ba5  4c cd 8c  reserr jmp userr       ausgabe '?undefined statement
                                             error', ready
1975  8ba8
1976  8ba8
1977  8ba8  ===> basic-routine 'stop' <===
1978  8ba8
1979  8ba8  b0 01     stop   bcs stpz2
1980  8baa
1981  8baa
1982  8baa  ===> basic-routine 'end' <===
1983  8baa
1984  8baa  18       end    clc
1985  8bab  d0 59     stpz2  bne contx
1986  8bad  f0 14           beq stopd
1987  8baf
1988  8baf
1989  8baf  ===> ansprung bei gedr. stop-taste, test 'trap' 'break' <===
1990  8baf
1991  8baf  08       stopc  php
1992  8bb0  ac 97 02        ldy trapno+1     addrh zeiger error trap
1993  8bb3  c8              iny
1994  8bb4  f0 0a           beq sa           ausführung 'break'
1995  8bb6
1996  8bb6
1997  8bb6  ===> bei 'trap' warten auf stop, error-routine <===
1998  8bb6
1999  8bb6  20 e1 ff  unstop jsr kstop       test stop-taste
2000  8bb9  f0 fb           beq unstop       bei 'trap' warten auf stop,
                                             error-routine
2001  8bbb  a2 1c           ldx #$1c
2002  8bbd  4c 52 85        jmp error        ind. jmp zur fehlerroutine
2003  8bc0
2004  8bc0
2005  8bc0  ===> ausführung 'break' <===
2006  8bc0
2007  8bc0  28       sa     plp
2008  8bc1  48              pha
2009  8bc2  48              pha
2010  8bc3  08       stopd  php
2011  8bc4  20 57 9d        jsr tstdir       test direktmodus, hi-byt zeile=$ff
2012  8bc7  f0 10           beq nsxx4        plp, pla, pla, carry =0 ready, sonst
                                             ausgabe 'break'
2013  8bc9  a5 85           lda txtptr       addrl zeiger auf akt. term
2014  8bcb  a4 86           ldy txtptr+1     addrh zeiger auf akt. term
2015  8bcd  85 46           sta oldtxt       addrl zeiger letzter basic-befehl
2016  8bcf  84 47           sty oldtxt+1     addrh zeiger letzter basic-befehl
2017  8bd1  a5 42           lda curlin       lo-byte akt. zeilennummer
2018  8bd3  a4 43           ldy curlin+1     hi-byte akt. zeilennummer
2019  8bd5  85 44           sta oldlin       lo-byte vorige zeilennummer (stop)
2020  8bd7  84 45           sty oldlin+1     hi-byte vorige zeilennummer (stop)
2021  8bd9
2022  8bd9
2023  8bd9  ===> plp, pla, pla, carry =0 ready, sonst ausgabe 'break' <===
```

```
zeile adr.  obj.-code   source-code

2024  8bd9
2025  8bd9  28        nsxx4  plp
2026  8bda  68               pla
2027  8bdb  68               pla
2028  8bdc  90 08            bcc nsxx6        sprung nach 'ready'
2029  8bde  a2 1c            ldx #$1c
2030  8be0  20 c3 a3         jsr msg          textzeiger + xr, textausgabe
2031  8be3  4c ab 85         jmp errfin       im prg-mode zeilen#-ausgabe, ready
2032  8be6
2033  8be6
2034  8be6  4c c0 85  nsxx6  jmp ready        sprung nach 'ready'
2035  8be9
2036  8be9
2037  8be9  ===> basic-routine 'cont' <===
2038  8be9
2039  8be9  d0 1b     cont   bne contx
2040  8beb  a4 47            ldy oldtxt+1     addrh zeiger letzter basic-befehl
2041  8bed  d0 09            bne contz3
2042  8bef  a5 46            lda oldtxt       addrl zeiger letzter basic-befehl
2043  8bf1  d0 07            bne contz4
2044  8bf3  a2 48            ldx #$48
2045  8bf5  4c 52 85         jmp error        ind. jmp zur fehlerroutine
2046  8bf8
2047  8bf8
2048  8bf8  a5 46     contz3 lda oldtxt       addrl zeiger letzter basic-befehl
2049  8bfa  85 85     contz4 sta txtptr       addrl zeiger auf akt. term
2050  8bfc  84 86            sty txtptr+1     addrh zeiger auf akt. term
2051  8bfe  a5 44            lda oldlin       lo-byte vorige zeilennummer (stop)
2052  8c00  a4 45            ldy oldlin+1     hi-byte vorige zeilennummer (stop)
2053  8c02  85 42            sta curlin       lo-byte akt. zeilennummer
2054  8c04  84 43            sty curlin+1     hi-byte akt. zeilennummer
2055  8c06  60        contx  rts
2056  8c07
2057  8c07
2058  8c07  ===> basic-routine 'run' <===
2059  8c07
2060  8c07  20 17 8c  run    jsr runmod       test direkt/run-modus, run eintr.
2061  8c0a  20 29 ba         jsr chrgot       letztes zeichen erneut nach ac
                                              (indirekter sprung)
2062  8c0d  d0 03            bne run2         'run' mit zeilennummer
2063  8c0f  4c 40 8a         jmp runc         basic-zeiger initial., 'clr'
2064  8c12
2065  8c12
2066  8c12  ===> 'run' mit zeilennummer <===
2067  8c12
2068  8c12  20 47 8a  run2   jsr clearc       basic-routine 'clr'
2069  8c15  f0 22            beq runc2
2070  8c17
2071  8c17
2072  8c17  ===> test direkt/run-modus, run eintr. <===
2073  8c17
2074  8c17  20 57 9d  runmod jsr tstdir       test direktmodus, hi-byt zeile=$ff
2075  8c1a  d0 88            bne runm10
2076  8c1c  a9 fe            lda #$fe
2077  8c1e  85 43            sta curlin+1     hi-byte akt. zeilennummer
2078  8c20  a9 00            lda #$00
2079  8c22  4c 90 ff         jmp kstmsg       meldung des operating-syst. ausgeben
2080  8c25
```

```
zeile adr.  obj.-code   source-code


2081  8c25
2082  8c25  ===> basic-routine 'gosub' <===
2083  8c25
2084  8c25  a9 03     gosub  lda #$03
2085  8c27  20 66 88         jsr getstk       test stapelplatz
2086  8c2a  a5 86            lda txtptr+1     addrh zeiger auf akt. term
2087  8c2c  48               pha
2088  8c2d  a5 85            lda txtptr       addrl zeiger auf akt. term
2089  8c2f  48               pha
2090  8c30  a5 43            lda curlin+1     hi-byte akt. zeilennummer
2091  8c32  48               pha
2092  8c33  a5 42            lda curlin       lo-byte akt. zeilennummer
2093  8c35  48               pha
2094  8c36  a9 8d            lda #$8d
2095  8c38  48               pha
2096  8c39  20 29 ba  runc2  jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
2097  8c3c  20 84 8c         jsr goto         basic-routine 'goto'
2098  8c3f  4c 5a 87         jmp newstt       neues statement, stoptest
2099  8c42
2100  8c42
2101  8c42  ===> basic-routine 'if' <===
2102  8c42
2103  8c42  20 c1 95  if     jsr frmevl       auswertung beliebiger ausdruck
2104  8c45  24 11            bit valtyp       flag für variablen typ (0=num,
                                               1=string)
2105  8c47  10 03            bpl if50
2106  8c49  20 e9 a8         jsr frefac       stringverwaltung, freier string
2107  8c4c  20 29 ba  if50   jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
2108  8c4f  c9 89            cmp #$89
2109  8c51  f0 05            beq okgoto       'goto' folgt auf erfülltes 'if'
2110  8c53  a9 a7            lda #$a7
2111  8c55  20 32 97         jsr synchr       test:folgt ascii o. token im akt.
                                               text, sonst error
2112  8c58
2113  8c58
2114  8c58  ===> 'goto' folgt auf erfülltes 'if' <===
2115  8c58
2116  8c58  a5 71     okgoto lda facexp       fac #1: exponent
2117  8c5a  d0 13            bne docond
2118  8c5c
2119  8c5c
2120  8c5c  ===> test auf 'else' nach 'if' <===
2121  8c5c
2122  8c5c  20 df 8c  lkelse jsr data         basic-routine 'data'
2123  8c5f  a0 00            ldy #$00
2124  8c61  b1 85            lda (txtptr),y   addrl zeiger auf akt. term
2125  8c63  f0 12            beq rem          basic-routine 'rem'
2126  8c65  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
2127  8c68  c9 e1            cmp #$e1
2128  8c6a  d0 f0            bne lkelse       test auf 'else' nach 'if'
2129  8c6c  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
2130  8c6f  20 29 ba  docond jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
2131  8c72  90 10            bcc goto         basic-routine 'goto'
2132  8c74  4c a8 87         jmp xeqcm3       basic-befehl (token in ac) ausführen
2133  8c77
```

```
zeile adr.  obj.-code   source-code

2134  8c77
2135  8c77  ===> basic-routine 'rem' <===
2136  8c77
2137  8c77  20 f0 8c  rem     jsr remn          basic-zeilenende suchen, offset yr
2138  8c7a  f0 66             beq addon         textzeiger + yr
2139  8c7c
2140  8c7c
2141  8c7c  ===> basic-routine 'go to' <===
2142  8c7c
2143  8c7c  20 29 ba  go      jsr chrgot        letztes zeichen erneut nach ac
                                                (indirekter sprung)
2144  8c7f  a9 a4             lda #$a4
2145  8c81  20 32 97          jsr synchr        test:folgt ascii o. token im akt.
                                                text, sonst error
2146  8c84
2147  8c84
2148  8c84  ===> basic-routine 'goto' <===
2149  8c84
2150  8c84  20 4e 8d  goto    jsr linget        zeilennummer lesen, in adressformat
                                                umwandeln
2151  8c87  20 f0 8c  goto0   jsr remn          basic-zeilenende suchen, offset yr
2152  8c8a  38                sec
2153  8c8b  a5 42             lda curlin        lo-byte akt. zeilennummer
2154  8c8d  e5 1b             sbc linnum        lo-byte akt. zeilennummer
2155  8c8f  a5 43             lda curlin+1      hi-byte akt. zeilennummer
2156  8c91  e5 1c             sbc linnum+1      hi-byte akt. zeilennummer
2157  8c93  b0 0b             bcs luk4it        zeilen # > akt. suchen/abarb.
2158  8c95  98                tya
2159  8c96  38                sec
2160  8c97  65 85             adc txtptr        addrl zeiger auf akt. term
2161  8c99  a6 86             ldx txtptr+1      addrh zeiger auf akt. term
2162  8c9b  90 07             bcc lukall        zeilen # < akt. suchen/abarb.
2163  8c9d  e8                inx
2164  8c9e  b0 04             bcs lukall        zeilen # < akt. suchen/abarb.
2165  8ca0
2166  8ca0
2167  8ca0  ===> zeilen # > akt. suchen/abarb. <===
2168  8ca0
2169  8ca0  a5 2d     luk4it lda txttab         addrl zeiger anfang  basic-text
2170  8ca2  a6 2e            ldx txttab+1       addrh zeiger anfang  basic-text
2171  8ca4
2172  8ca4
2173  8ca4  ===> zeilen # < akt. suchen/abarb. <===
2174  8ca4
2175  8ca4  20 23 87  lukall jsr fndlnc         startadr. der zeile in 'lowtr'
2176  8ca7  90 24             bcc userr         ausgabe '?undefined statement
                                                error', ready
2177  8ca9  a5 6d             lda lowtr         addrl ursprunganfang (verschieben)/
                                                temp.fac#2
2178  8cab  e9 01             sbc #$01
2179  8cad  85 85             sta txtptr        addrl zeiger auf akt. term
2180  8caf  a5 6e             lda lowtr+1       addrh ursprunganfang (verschieben)
                                                temp.fac#2
2181  8cb1  e9 00             sbc #$00
2182  8cb3  85 86             sta txtptr+1      addrh zeiger auf akt. term
2183  8cb5  4c 17 8c          jmp runmod        test direkt/run-modus, run eintr.
2184  8cb8
2185  8cb8
```

```
zeile adr.  obj.-code   source-code


2186  8cb8  ===> basic-routine 'return' <===
2187  8cb8
2188  8cb8  d0 32     return bne addrts
2189  8cba  a9 10            lda #$10
2190  8cbc  85 56            sta lstpnt+2     bank zeiger letzter string
2191  8cbe  a9 ff            lda #$ff
2192  8cc0  85 55            sta lstpnt+1     addrh zeiger letzter string
2193  8cc2  20 db 87         jsr fndfor       stapel-suchroutine for-next/goto
2194  8cc5  9a               txs
2195  8cc6  c9 8d            cmp #$8d
2196  8cc8  f0 08            beq retu1        zeilen- u. programm-zeiger vom
                                              stapel, weiter 'data'
2197  8cca  a2 2c            ldx #$2c
2198  8ccc  2c               .byte $2c
2199  8ccd
2200  8ccd
2201  8ccd  ===> ausgabe '?undefined statement error', ready <===
2202  8ccd
2203  8ccd  a2 36     userr  ldx #$36
2204  8ccf  4c 52 85         jmp error        ind. jmp zur fehlerroutine
2205  8cd2
2206  8cd2
2207  8cd2  ===> zeilen- u. programm-zeiger vom stapel, weiter 'data' <===
2208  8cd2
2209  8cd2  68        retu1  pla
2210  8cd3  68               pla
2211  8cd4  85 42            sta curlin       lo-byte akt. zeilennummer
2212  8cd6  68               pla
2213  8cd7  85 43            sta curlin+1     hi-byte akt. zeilennummer
2214  8cd9  68               pla
2215  8cda  85 85            sta txtptr       addrl zeiger auf akt. term
2216  8cdc  68               pla
2217  8cdd  85 86            sta txtptr+1     addrh zeiger auf akt. term
2218  8cdf
2219  8cdf
2220  8cdf  ===> basic-routine 'data' <===
2221  8cdf
2222  8cdf  20 ed 8c  data   jsr datan        trennzeichen ':', zeilenende '0'
                                              suchen, offset yr
2223  8ce2
2224  8ce2
2225  8ce2  ===> textzeiger + yr <===
2226  8ce2
2227  8ce2  98        addon  tya
2228  8ce3  18               clc
2229  8ce4  65 85            adc txtptr       addrl zeiger auf akt. term
2230  8ce6  85 85            sta txtptr       addrl zeiger auf akt. term
2231  8ce8  90 02            bcc addrts
2232  8cea  e6 86            inc txtptr+1     addrh zeiger auf akt. term
2233  8cec  60        addrts rts
2234  8ced
2235  8ced
2236  8ced  ===> trennzeichen ':', zeilenende '0' suchen, offset yr <===
2237  8ced
2238  8ced  a2 3a     datan  ldx #$3a
2239  8cef  2c               .byte $2c
2240  8cf0
2241  8cf0
```

```
zeile adr.  obj.-code   source-code


2242  8cf0  ===> basic-zeilenende suchen, offset yr <===
2243  8cf0
2244  8cf0  a2 00     remn   ldx #$00
2245  8cf2  86 0c            stx charac       puffer für trennzeichen
2246  8cf4  a0 00            ldy #$00
2247  8cf6  84 0d            sty endchr       puffer für trennzeichen
2248  8cf8  a5 0d     exchqt lda endchr       puffer für trennzeichen
2249  8cfa  a6 0c            ldx charac       puffer für trennzeichen
2250  8cfc  85 0c            sta charac       puffer für trennzeichen
2251  8cfe  86 0d            stx endchr       puffer für trennzeichen
2252  8d00  a5 87     remer  lda txtptr+2     bank zeiger auf akt. term
2253  8d02  85 01            sta i6509        6509 indirection register
2254  8d04  b1 85            lda (txtptr),y   addrl zeiger auf akt. term
2255  8d06  f0 0b            beq remtxt
2256  8d08  c5 0d            cmp endchr       puffer für trennzeichen
2257  8d0a  f0 07            beq remtxt
2258  8d0c  c8               iny
2259  8d0d  c9 22            cmp #$22
2260  8d0f  d0 ef            bne remer
2261  8d11  f0 e5            beq exchqt
2262  8d13  4c 8c ba  remtxt jmp mapusr       umsch. auf bank # 1
2263  8d16
2264  8d16
2265  8d16  ===> basic-routine 'trap' <===
2266  8d16
2267  8d16  20 4a 9d  trap   jsr errdir       test direkt, ausgabe '?illegal
                                               direct error', ready
2268  8d19  20 29 ba         jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
2269  8d1c  f0 07            beq trap1
2270  8d1e  20 e5 b4         jsr getpin       dezimal-zahl in klammer nach linnum
                                               einlesen
2271  8d21  8c 96 02         sty trapno       addrl zeiger error trap
2272  8d24  2c               .byte $2c
2273  8d25  a9 ff     trap1  lda #$ff
2274  8d27  8d 97 02         sta trapno+1     addrh zeiger error trap
2275  8d2a  60               rts
2276  8d2b
2277  8d2b
2278  8d2b  ===> basic-routine 'on - goto' <===
2279  8d2b
2280  8d2b  20 d6 b4  ongoto jsr getbyt       zahl < 256 aus akt. text nach xr
2281  8d2e  48               pha
2282  8d2f  c9 8d            cmp #$8d
2283  8d31  f0 07            beq onglop
2284  8d33  c9 89            cmp #$89
2285  8d35  f0 03            beq onglop
2286  8d37  4c 4f 97  snerr3 jmp snerr        ausgabe '?syntax error', ready
2287  8d3a
2288  8d3a
2289  8d3a  c6 75     onglop dec faclo        fac #1: mantisse
2290  8d3c  d0 04            bne onglp1
2291  8d3e  68               pla
2292  8d3f  4c aa 87         jmp xeqcm2
2293  8d42
2294  8d42
2295  8d42  20 26 ba  onglp1 jsr chrget       nächstes zeichen nach ac (ind.jmp)
2296  8d45  20 4e 8d         jsr linget       zeilennummer lesen, in adressformat
                                               umwandeln
```

```
zeile adr.  obj.-code   source-code

2297  8d48  c9 2c            cmp #$2c
2298  8d4a  f0 ee            beq onglop
2299  8d4c  68               pla
2300  8d4d  60        ongrts rts
2301  8d4e
2302  8d4e
2303  8d4e  ===> zeilennummer lesen, in adressformat umwandeln <===
2304  8d4e
2305  8d4e  a2 00     linget ldx #$00
2306  8d50  86 1b            stx linnum       lo-byte akt. zeilennummer
2307  8d52  86 1c            stx linnum+1     hi-byte akt. zeilennummer
2308  8d54  b0 f7     morlin bcs ongrts
2309  8d56  e9 2f            sbc #$2f
2310  8d58  85 0c            sta charac       puffer für trennzeichen
2311  8d5a  a5 1c            lda linnum+1     hi-byte akt. zeilennummer
2312  8d5c  85 22            sta index1       addrl indirekter index #1
2313  8d5e  c9 19            cmp #$19
2314  8d60  b0 d5            bcs snerr3
2315  8d62  a5 1b            lda linnum       lo-byte akt. zeilennummer
2316  8d64  0a               asl a
2317  8d65  26 22            rol index1       addrl indirekter index #1
2318  8d67  0a               asl a
2319  8d68  26 22            rol index1       addrl indirekter index #1
2320  8d6a  65 1b            adc linnum       lo-byte akt. zeilennummer
2321  8d6c  85 1b            sta linnum       lo-byte akt. zeilennummer
2322  8d6e  a5 22            lda index1       addrl indirekter index #1
2323  8d70  65 1c            adc linnum+1     hi-byte akt. zeilennummer
2324  8d72  85 1c            sta linnum+1     hi-byte akt. zeilennummer
2325  8d74  06 1b            asl linnum       lo-byte akt. zeilennummer
2326  8d76  26 1c            rol linnum+1     hi-byte akt. zeilennummer
2327  8d78  a5 1b            lda linnum       lo-byte akt. zeilennummer
2328  8d7a  65 0c            adc charac       puffer für trennzeichen
2329  8d7c  85 1b            sta linnum       lo-byte akt. zeilennummer
2330  8d7e  90 02            bcc nxtlgc
2331  8d80  e6 1c            inc linnum+1     hi-byte akt. zeilennummer
2332  8d82  20 26 ba  nxtlgc jsr chrget       nächstes zeichen nach ac (ind.jmp)
2333  8d85  4c 54 8d         jmp morlin
2334  8d88
2335  8d88
2336  8d88  d0 ad     ressnr bne snerr3
2337  8d8a
2338  8d8a
2339  8d8a  ===> basic-routine 'let' <===
2340  8d8a
2341  8d8a  20 a6 88  let    jsr sav73
2342  8d8d  a9 b2            lda #$b2
2343  8d8f  20 32 97         jsr synchr       test:folgt ascii o. token im akt.
                                              text, sonst error
2344  8d92  a5 12            lda intflg       flag für integer-variable
2345  8d94  48               pha
2346  8d95  a5 11            lda valtyp       flag für variablen typ (0=num,
                                              1=string)
2347  8d97  48               pha
2348  8d98  20 c1 95         jsr frmevl       auswertung beliebiger ausdruck
2349  8d9b  68               pla
2350  8d9c  2a               rol a
2351  8d9d  20 07 b5         jsr chkval       prüft ob richtiger variablentyp
2352  8da0  d0 1e            bne copstr
```

```
zeile adr.  obj.-code   source-code

2353  8da2  68              pla
2354  8da3  10 18     qintgr bpl copflt       gk-zahl aus fac in variable
2355  8da5  20 f2 a1         jsr round        fac #1 rundung
2356  8da8  20 13 9b         jsr ayint        umwandlung real nach integer
2357  8dab  a5 56             lda lstpnt+2     bank zeiger letzter string
2358  8dad  85 01             sta i6509        6509 indirection register
2359  8daf  a0 00             ldy #$00
2360  8db1  a5 74             lda facmo        fac #1: mantisse
2361  8db3  91 54             sta (lstpnt),y   addrl zeiger letzter string
2362  8db5  c8                iny
2363  8db6  a5 75             lda faclo        fac #1: mantisse
2364  8db8  91 54             sta (lstpnt),y   addrl zeiger letzter string
2365  8dba  4c 8c ba          jmp mapusr       umsch. auf bank # 1
2366  8dbd
2367  8dbd
2368  8dbd  4c 9f a1  copflt  jmp movvf        gk-zahl aus fac in variable
2369  8dc0
2370  8dc0
2371  8dc0  68        copstr  pla
2372  8dc1  4c 73 a9          jmp inpcom
2373  8dc4
2374  8dc4
2375  8dc4  ===> basic-routine 'resume' <===
2376  8dc4
2377  8dc4  20 4a 9d  resume  jsr errdir       test direkt, ausgabe '?illegal
                                                direct error', ready
2378  8dc7  ae 99 02          ldx errlin+1     hi-byte zeilennummer akt. fehler
2379  8dca  e8                inx
2380  8dcb  f0 52             beq rescnt
2381  8dcd  20 29 ba          jsr chrgot       letztes zeichen erneut nach ac
                                                (indirekter sprung)
2382  8dd0  f0 2d             beq resswp
2383  8dd2  90 1d             bcc resnum
2384  8dd4  c9 82             cmp #$82
2385  8dd6  d0 b0      gooos  bne ressnr
2386  8dd8  20 ff 8d          jsr resswp
2387  8ddb  20 8c ba          jsr mapusr       umsch. auf bank # 1
2388  8dde  a0 00             ldy #$00
2389  8de0  b1 85             lda (txtptr),y   addrl zeiger auf akt. term
2390  8de2  d0 07             bne resum2
2391  8de4  c8                iny
2392  8de5  20 63 87          jsr sav42
2393  8de8  20 e2 8c          jsr addon        textzeiger + yr
2394  8deb  20 26 ba  resum2  jsr chrget       nächstes zeichen nach ac (ind.jmp)
2395  8dee  4c df 8c          jmp data         basic-routine 'data'
2396  8df1
2397  8df1
2398  8df1  20 e5 b4  resnum  jsr getpin       dezimal-zahl in klammer nach linnum
                                                einlesen
2399  8df4  85 1c             sta linnum+1     hi-byte akt. zeilennummer
2400  8df6  20 0e 8e          jsr resend
2401  8df9  20 8c ba          jsr mapusr       umsch. auf bank # 1
2402  8dfc  4c a0 8c          jmp luk4it       zeilen # > akt. suchen/abarb.
2403  8dff
2404  8dff
2405  8dff  a2 01     resswp  ldx #$01
2406  8e01  bd 98 02  resum0  lda errlin,x     lo-byte zeilennummer akt. fehler
2407  8e04  95 42             sta curlin,x     lo-byte akt. zeilennummer
```

```
zeile adr.  obj.-code   source-code

2408  8e06  bd 9a 02         lda errtxt,x     addrl basic-text zeiger bei
                                              fehlerbehandlung
2409  8e09  95 85            sta txtptr,x     addrl zeiger auf akt. term
2410  8e0b  ca               dex
2411  8e0c  10 f3            bpl resum0
2412  8e0e  a2 ff     resend ldx #$ff
2413  8e10  86 0f            stx xcnt         dos loop-zähler
2414  8e12  8e 98 02         stx errlin       lo-byte zeilennummer akt. fehler
2415  8e15  8e 99 02         stx errlin+1     hi-byte zeilennummer akt. fehler
2416  8e18  ae 9d 02         ldx tmptrp       hi-byte zeilennummer 'trap, resume'
2417  8e1b  8e 97 02         stx trapno+1     addrh zeiger error trap
2418  8e1e  60               rts
2419  8e1f
2420  8e1f
2421  8e1f  a2 52     rescnt ldx #$52
2422  8e21  4c 52 85         jmp error        ind. jmp zur fehlerroutine
2423  8e24
2424  8e24
2425  8e24  ===> basic-routine 'dispose' <===
2426  8e24
2427  8e24  c9 81     dispos cmp #$81
2428  8e26  f0 04            beq dispo0
2429  8e28  c9 8d            cmp #$8d
2430  8e2a  d0 aa            bne gooos
2431  8e2c  8d 9f 02  dispo0 sta oldtok       temp. speicher 'dispose'
2432  8e2f  20 4a 9d         jsr errdir       test direkt, ausgabe '?illegal
                                              direct error', ready
2433  8e32  ba               tsx
2434  8e33  e8               inx
2435  8e34  e8               inx
2436  8e35  8a               txa
2437  8e36  a8               tay
2438  8e37  98        dispo1 tya
2439  8e38  aa               tax
2440  8e39  c0 ff            cpy #$ff
2441  8e3b  f0 38            beq diserr
2442  8e3d  bd 01 01         lda stack+1,x    6509 cpu-stack
2443  8e40  c9 81            cmp #$81
2444  8e42  d0 06            bne dispo2
2445  8e44  8a               txa
2446  8e45  69 12            adc #$12
2447  8e47  4c 4e 8e         jmp dispo3
2448  8e4a
2449  8e4a
2450  8e4a  8a        dispo2 txa
2451  8e4b  18               clc
2452  8e4c  69 07            adc #$07
2453  8e4e  b0 25     dispo3 bcs diserr
2454  8e50  a8               tay
2455  8e51  bd 01 01         lda stack+1,x    6509 cpu-stack
2456  8e54  cd 9f 02         cmp oldtok       temp. speicher 'dispose'
2457  8e57  d0 de            bne dispo1
2458  8e59  f0 03            beq dispo5
2459  8e5b  ae 9e 02  dispo4 ldx dsptmp       temp. speicher 'dispose'
2460  8e5e  bd 00 01  dispo5 lda stack,x      6509 cpu-stack
2461  8e61  99 00 01         sta stack,y      6509 cpu-stack
2462  8e64  88               dey
2463  8e65  ca               dex
```

```
zeile adr.  obj.-code   source-code


2464  8e66  8e 9e 02         stx dsptmp       temp. speicher 'dispose'
2465  8e69  ba               tsx
2466  8e6a  ec 9e 02         cpx dsptmp       temp. speicher 'dispose'
2467  8e6d  d0 ec            bne dispo4
2468  8e6f  98               tya
2469  8e70  aa               tax
2470  8e71  9a               txs
2471  8e72  4c 26 ba         jmp chrget       nächstes zeichen nach ac (ind.jmp)
2472  8e75
2473  8e75
2474  8e75  a2 54     diserr ldx #$54
2475  8e77  4c 52 85         jmp error        ind. jmp zur fehlerroutine
2476  8e7a
2477  8e7a                   .end
2478  8e7a                   .lib bverbs2
```

```
zeile adr.  obj.-code   source-code

2480  8e7a
2481  8e7a  ===> basic-routine 'print#' <===
2482  8e7a
2483  8e7a  20 80 8e  printn jsr cmd          basic-routine 'cmd'
2484  8e7d  4c 59 8f         jmp iodone       akt. kanal clr, tast. aktivieren
2485  8e80
2486  8e80
2487  8e80  ===> basic-routine 'cmd' <===
2488  8e80
2489  8e80  20 d6 b4  cmd    jsr getbyt       zahl < 256 aus akt. text nach xr
2490  8e83  f0 03            beq cmnd2
2491  8e85  20 30 97         jsr chkcom       test 'komma' sonst fehler + ready
2492  8e88  08        cmnd2  php
2493  8e89  48               pha
2494  8e8a  8e 58 02         stx dolu         vorgabe für ausgabe-adresse
2495  8e8d  20 86 95         jsr patch1       system-status schreiben, iec listen
2496  8e90  85 1a            sta channl       speicherstelle für aktiven kanal
2497  8e92  68               pla
2498  8e93  28               plp
2499  8e94  4c 9d 8e         jmp print        basic-routine 'print'
2500  8e97
2501  8e97
2502  8e97  20 15 b5  strdon jsr strprt       ausgabe string -$00 (addr. in
                                               zeropage)
2503  8e9a  20 29 ba  newchr jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
2504  8e9d
2505  8e9d
2506  8e9d  ===> basic-routine 'print' <===
2507  8e9d
2508  8e9d  f0 29     print  beq ocrlf         ausgabe cr (+lf) auf akt. kanal
2509  8e9f  c9 e6            cmp #$e6
2510  8ea1  d0 03            bne printc       test ob 'tab' 'spc' ',' ';' folgt
2511  8ea3  4c 26 b0         jmp using        basic-routine 'printusing'
2512  8ea6
2513  8ea6
2514  8ea6  ===> test ob 'tab' 'spc' ',' ';' folgt <===
2515  8ea6
2516  8ea6  f0 31     printc beq ocrlfx
2517  8ea8  c9 a3            cmp #$a3
2518  8eaa  f0 3e            beq taber        tabulator ausführen
2519  8eac  c9 a6            cmp #$a6
2520  8eae  18               clc
2521  8eaf  f0 39            beq taber        tabulator ausführen
2522  8eb1  c9 2c            cmp #$2c
2523  8eb3  f0 25            beq comprt       10-tabulierung bei print (,)
2524  8eb5  c9 3b            cmp #$3b
2525  8eb7  f0 53            beq notabr       nächstes zeichen lesen, drucken
2526  8eb9  20 c1 95         jsr frmevl       auswertung beliebiger ausdruck
2527  8ebc  24 11            bit valtyp       flag für variablen typ (0=num,
                                               1=string)
2528  8ebe  30 d7            bmi strdon
2529  8ec0  20 be a3         jsr outfac
2530  8ec3  20 2e b5         jsr ospc         bei bs cursor rechts, sonst space
2531  8ec6  d0 d2            bne newchr
2532  8ec8
2533  8ec8
2534  8ec8  ===> ausgabe cr (+lf) auf akt. kanal <===
```

```
zeile adr.  obj.-code   source-code

2535  8ec8
2536  8ec8  a9 0d     ocrlf  lda #$0d
2537  8eca  20 3a b5         jsr ochr        ausgabe 1 zeichen (in ac)
2538  8ecd  ad 58 02         lda dolu        vorgabe für ausgabe-adresse
2539  8ed0  10 05            bpl crfin
2540  8ed2  a9 0a            lda #$0a
2541  8ed4  20 3a b5         jsr ochr        ausgabe 1 zeichen (in ac)
2542  8ed7  49 ff     crfin  eor #$ff
2543  8ed9  60        ocrlfx rts
2544  8eda
2545  8eda
2546  8eda  ===> 10-tabulierung bei print (,) <===
2547  8eda
2548  8eda  38        comprt sec
2549  8edb  20 f0 ff         jsr kplot       x,y-pos. des cursor lesen/schreiben
2550  8ede  98               tya
2551  8edf  38               sec
2552  8ee0  e9 0a     morco1 sbc #$0a
2553  8ee2  b0 fc            bcs morco1
2554  8ee4  49 ff            eor #$ff
2555  8ee6  69 01            adc #$01
2556  8ee8  d0 18            bne aspac       funktion 'tab'
2557  8eea
2558  8eea
2559  8eea  ===> tabulator ausführen <===
2560  8eea
2561  8eea  08        taber  php
2562  8eeb  38               sec
2563  8eec  20 f0 ff         jsr kplot       x,y-pos. des cursor lesen/schreiben
2564  8eef  8c 55 02         sty trmpos      cursor spalte auf bildschirm
2565  8ef2  20 d3 b4         jsr gtbytc      charget, 1 byte-zahl in xr
2566  8ef5  c9 29            cmp #$29
2567  8ef7  d0 19            bne snerr2
2568  8ef9  28               plp
2569  8efa  90 07            bcc xspac       funktion 'space'
2570  8efc  8a               txa
2571  8efd  ed 55 02         sbc trmpos      cursor spalte auf bildschirm
2572  8f00  90 0a            bcc notabr      nächstes zeichen lesen, drucken
2573  8f02
2574  8f02
2575  8f02  ===> funktion 'tab' <===
2576  8f02
2577  8f02  aa        aspac  tax
2578  8f03
2579  8f03
2580  8f03  ===> funktion 'space' <===
2581  8f03
2582  8f03  e8        xspac  inx
2583  8f04  ca        xspac2 dex
2584  8f05  f0 05            beq notabr      nächstes zeichen lesen, drucken
2585  8f07  20 2e b5         jsr ospc        bei bs cursor rechts, sonst space
2586  8f0a  d0 f8            bne xspac2
2587  8f0c
2588  8f0c
2589  8f0c  ===> nächstes zeichen lesen, drucken <===
2590  8f0c
2591  8f0c  20 26 ba  notabr jsr chrget      nächstes zeichen nach ac (ind.jmp)
2592  8f0f  4c a6 8e         jmp printc      test ob 'tab' 'spc' ',' ';' folgt
```

```
zeile adr.  obj.-code   source-code

2593  8f12
2594  8f12
2595  8f12  4c 4f 97  snerr2 jmp snerr        ausgabe '?syntax error', ready
2596  8f15
2597  8f15
2598  8f15  ===> basic-routine 'get' <===
2599  8f15
2600  8f15  20 4a 9d  get    jsr errdir       test direkt, ausgabe '?illegal
                                              direct error', ready
2601  8f18  20 29 ba         jsr chrgot       letztes zeichen erneut nach ac
                                              (indirekter sprung)
2602  8f1b  c9 23             cmp #$23
2603  8f1d  d0 0b             bne get010
2604  8f1f  20 d3 b4          jsr gtbytc       charget, 1 byte-zahl in xr
2605  8f22  20 30 97          jsr chkcom       test 'komma' sonst fehler + ready
2606  8f25  20 8c 95          jsr patch2       system-status schreiben, iec-talk
2607  8f28  85 1a             sta channl       speicherstelle für aktiven kanal
2608  8f2a  a9 01     get010  lda #$01
2609  8f2c  a4 89             ldy buffpt+1     addrh zeiger auf eingabe-puffer
2610  8f2e  a6 88             ldx buffpt       addrl zeiger auf eingabe-puffer
2611  8f30  e8                inx
2612  8f31  d0 01             bne get020
2613  8f33  c8                iny
2614  8f34  86 22     get020  stx index1       addrl indirekter index #1
2615  8f36  84 23             sty index1+1     addrh indirekter index #1
2616  8f38  85 24             sta index1+2     bank indirekter index #1
2617  8f3a  a0 00             ldy #$00
2618  8f3c  98                tya
2619  8f3d  91 22             sta (index1),y   addrl indirekter index #1
2620  8f3f  a4 23             ldy index1+1     addrh indirekter index #1
2621  8f41  a9 40             lda #$40
2622  8f43  20 f3 8f          jsr inpco1
2623  8f46  a6 1a             ldx channl       speicherstelle für aktiven kanal
2624  8f48  d0 11             bne iorele       in ac kanal clr, tast. aktivieren
2625  8f4a  60                rts
2626  8f4b
2627  8f4b
2628  8f4b  ===> basic-routine 'input#' <===
2629  8f4b
2630  8f4b  20 d6 b4  inputn jsr getbyt        zahl < 256 aus akt. text nach xr
2631  8f4e  20 30 97          jsr chkcom       test 'komma' sonst fehler + ready
2632  8f51  20 8c 95          jsr patch2       system-status schreiben, iec-talk
2633  8f54  85 1a             sta channl       speicherstelle für aktiven kanal
2634  8f56  20 75 8f          jsr notqti
2635  8f59
2636  8f59
2637  8f59  ===> akt. kanal clr, tast. aktivieren <===
2638  8f59
2639  8f59  a5 1a     iodone lda channl        speicherstelle für aktiven kanal
2640  8f5b
2641  8f5b
2642  8f5b  ===> in ac kanal clr, tast. aktivieren <===
2643  8f5b
2644  8f5b  20 cc ff  iorele jsr kclrch        ein-/ausgabekanal schließen
2645  8f5e  a2 00             ldx #$00
2646  8f60  86 1a             stx channl       speicherstelle für aktiven kanal
2647  8f62  8e 58 02          stx dolu         vorgabe für ausgabe-adresse
2648  8f65  60                rts
```

```
zeile adr.  obj.-code   source-code

2649  8f66
2650  8f66
2651  8f66  ===> basic-routine 'input' <===
2652  8f66
2653  8f66  c9 22     input  cmp #$22
2654  8f68  d0 0b            bne notqti
2655  8f6a  20 e7 96         jsr strtxt      string in '"' erfassen
2656  8f6d  a9 3b            lda #$3b
2657  8f6f  20 32 97         jsr synchr      test:folgt ascii o. token im akt.
                                             text, sonst error
2658  8f72  20 15 b5         jsr strprt      ausgabe string -$00 (addr. in
                                             zeropage)
2659  8f75  20 4a 9d  notqti jsr errdir      test direkt, ausgabe '?illegal
                                             direct error', ready
2660  8f78  20 a8 8f  getagn jsr qinlin      druck '?', eingabeschleife
2661  8f7b  a5 1a            lda channl      speicherstelle für aktiven kanal
2662  8f7d  f0 0d            beq bufful
2663  8f7f  20 dd bb         jsr readst      system-status in ac lesen
2664  8f82  29 03            and #$03
2665  8f84  f0 06            beq bufful
2666  8f86  20 59 8f         jsr iodone      akt. kanal clr, tast. aktivieren
2667  8f89  4c df 8c         jmp data        basic-routine 'data'
2668  8f8c
2669  8f8c
2670  8f8c  a0 00     bufful ldy #$00
2671  8f8e  b1 22            lda (index1),y  addrl indirekter index #1
2672  8f90  f0 06            beq buffu2
2673  8f92  20 3c 9e  buffu1 jsr sav75
2674  8f95  4c f1 8f         jmp inpcon
2675  8f98
2676  8f98
2677  8f98  a5 1a     buffu2 lda channl      speicherstelle für aktiven kanal
2678  8f9a  d0 03            bne buffu3
2679  8f9c  4c df 8c         jmp data        basic-routine 'data'
2680  8f9f
2681  8f9f
2682  8f9f  20 dd bb  buffu3 jsr readst      system-status in ac lesen
2683  8fa2  29 40            and #$40
2684  8fa4  f0 d2            beq getagn
2685  8fa6  d0 ea            bne buffu1
2686  8fa8
2687  8fa8
2688  8fa8  ===> druck '?', eingabeschleife <===
2689  8fa8
2690  8fa8  a5 1a     qinlin lda channl      speicherstelle für aktiven kanal
2691  8faa  d0 06            bne ginlin
2692  8fac  20 38 b5         jsr outqst      ausgabe '?'
2693  8faf  20 2e b5         jsr ospc        bei bs cursor rechts, sonst space
2694  8fb2  4c e3 86  ginlin jmp inlin       eingabe einer zeile
2695  8fb5
2696  8fb5
2697  8fb5  ===> bearbeitung 'read' <===
2698  8fb5
2699  8fb5  20 ed 8c  datlop jsr datan       trennzeichen ':', zeilenende '0'
                                             suchen, offset yr
2700  8fb8  c8               iny
2701  8fb9  aa               tax
2702  8fba  d0 1e            bne nowlin
```

```
zeile adr.  obj.-code   source-code

2703  8fbc  a5 87           lda txtptr+2     bank zeiger auf akt. term
2704  8fbe  85 01           sta i6509        6509 indirection register
2705  8fc0  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
2706  8fc2  d0 0a           bne dtlp0
2707  8fc4  c8              iny
2708  8fc5  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
2709  8fc7  d0 06           bne dtlp1
2710  8fc9  a2 2e           ldx #$2e
2711  8fcb  4c 52 85        jmp error        ind. jmp zur fehlerroutine
2712  8fce
2713  8fce
2714  8fce  c8        dtlp0 iny
2715  8fcf  c8        dtlp1 iny
2716  8fd0  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
2717  8fd2  85 49           sta datlin       addrl data-zeilennummer
2718  8fd4  c8              iny
2719  8fd5  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
2720  8fd7  c8              iny
2721  8fd8  85 4a           sta datlin+1     addrh data-zeilennummer
2722  8fda  20 8c ba  nowlin jsr mapusr      umsch. auf bank # 1
2723  8fdd  20 e2 8c        jsr addon        textzeiger + yr
2724  8fe0  20 29 ba        jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
2725  8fe3  aa              tax
2726  8fe4  e0 83           cpx #$83
2727  8fe6  d0 cd           bne datlop       bearbeitung 'read'
2728  8fe8  f0 4f           beq datbk1
2729  8fea
2730  8fea
2731  8fea  ===> basic-routine 'read' <===
2732  8fea
2733  8fea  a6 4b     read  ldx datptr       addrl zeiger auf data nach 'read'
2734  8fec  a4 4c           ldy datptr+1     addrh zeiger auf data nach 'read'
2735  8fee  a9 98           lda #$98
2736  8ff0  2c              .byte $2c
2737  8ff1  a9 00     inpcon lda #$00
2738  8ff3  85 15     inpco1 sta inpflg      flag für eingabe (input,get,read)
2739  8ff5  86 4d           stx inpptr       addrl input-zeilennummer
2740  8ff7  84 4e           sty inpptr+1     addrh input-zeilennummer
2741  8ff9  20 a6 88  inloop jsr sav73
2742  8ffc  a5 85           lda txtptr       addrl zeiger auf akt. term
2743  8ffe  a4 86           ldy txtptr+1     addrh zeiger auf akt. term
2744  9000  85 57           sta opptr        addrl zeiger akt. operator-routine
2745  9002  84 58           sty opptr+1      addrh zeiger akt. operator-routine
2746  9004  a6 4d           ldx inpptr       addrl input-zeilennummer
2747  9006  a4 4e           ldy inpptr+1     addrh input-zeilennummer
2748  9008  86 85           stx txtptr       addrl zeiger auf akt. term
2749  900a  84 86           sty txtptr+1     addrh zeiger auf akt. term
2750  900c  20 29 ba        jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
2751  900f  d0 28           bne datbk1
2752  9011  24 15           bit inpflg       flag für eingabe (input,get,read)
2753  9013  50 13           bvc qdata
2754  9015  20 e8 bb        jsr getin        eingabe 1 char vom akt. kanal
                                             (queue-vektor)
2755  9018  20 3c 9e        jsr sav75
2756  901b  86 22           stx index1       addrl indirekter index #1
2757  901d  84 23           sty index1+1     addrh indirekter index #1
```

```
zeile adr.  obj.-code   source-code

2758  901f  a0 01           ldy #$01
2759  9021  91 22           sta (index1),y   addrl indirekter index #1
2760  9023  a4 23           ldy index1+1     addrh indirekter index #1
2761  9025  4c 35 90        jmp datbk
2762  9028
2763  9028
2764  9028  30 8b     qdata  bmi datlop       bearbeitung 'read'
2765  902a  a5 1a           lda channl       speicherstelle für aktiven kanal
2766  902c  d0 03           bne getnth
2767  902e  20 38 b5        jsr outqst       ausgabe '?'
2768  9031  20 a8 8f  getnth jsr qinlin       druck '?', eingabeschleife
2769  9034  aa             tax
2770  9035  86 85     datbk  stx txtptr       addrl zeiger auf akt. term
2771  9037  84 86           sty txtptr+1     addrh zeiger auf akt. term
2772  9039  20 26 ba  datbk1 jsr chrget       nächstes zeichen nach ac (ind.jmp)
2773  903c  24 11           bit valtyp       flag für variablen typ (0=num,
                                             1=string)
2774  903e  10 37           bpl numins
2775  9040  24 15           bit inpflg       flag für eingabe (input,get,read)
2776  9042  50 09           bvc setaut
2777  9044  e8             inx
2778  9045  86 85           stx txtptr       addrl zeiger auf akt. term
2779  9047  a9 00           lda #$00
2780  9049  85 0c           sta charac       puffer für trennzeichen
2781  904b  f0 0c           beq resetc
2782  904d  85 0c     setaut sta charac       puffer für trennzeichen
2783  904f  c9 22           cmp #$22
2784  9051  f0 07           beq nowget
2785  9053  a9 3a           lda #$3a
2786  9055  85 0c           sta charac       puffer für trennzeichen
2787  9057  a9 2c           lda #$2c
2788  9059  18       resetc clc
2789  905a  85 0d     nowget sta endchr       puffer für trennzeichen
2790  905c  20 6b 90        jsr sav30        bank akt. text nach xr, wenn carry=1
                                             textzeiger increment
2791  905f  20 27 a8        jsr strlt2
2792  9062  20 f3 ab        jsr st2txt
2793  9065  20 73 a9        jsr inpcom
2794  9068  4c 7f 90        jmp strdn2
2795  906b
2796  906b
2797  906b  ===> bank akt. text nach xr, wenn carry=1 textzeiger increment <===
2798  906b
2799  906b  a5 85     sav30  lda txtptr       addrl zeiger auf akt. term
2800  906d  a4 86           ldy txtptr+1     addrh zeiger auf akt. term
2801  906f  69 00           adc #$00
2802  9071  90 01           bcc nowge1
2803  9073  c8             iny
2804  9074  a6 87     nowge1 ldx txtptr+2     bank zeiger auf akt. term
2805  9076  60             rts
2806  9077
2807  9077
2808  9077  20 d8 a2  numins jsr fin          umw. zahlenstring in gk-zahl
2809  907a  a5 12           lda intflg       flag für integer-variable
2810  907c  20 a3 8d        jsr qintgr
2811  907f  20 29 ba  strdn2 jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
2812  9082  f0 04           beq trmok
```

```
zeile adr.  obj.-code   source-code

2813  9084  c9 2c           cmp #$2c
2814  9086  d0 1b           bne trmnok
2815  9088  a5 85     trmok lda txtptr      addrl zeiger auf akt. term
2816  908a  a4 86           ldy txtptr+1    addrh zeiger auf akt. term
2817  908c  85 4d           sta inpptr      addrl input-zeilennummer
2818  908e  84 4e           sty inpptr+1    addrh input-zeilennummer
2819  9090  a5 57           lda opptr       addrl zeiger akt. operator-routine
2820  9092  a4 58           ldy opptr+1     addrh zeiger akt. operator-routine
2821  9094  85 85           sta txtptr      addrl zeiger auf akt. term
2822  9096  84 86           sty txtptr+1    addrh zeiger auf akt. term
2823  9098  20 29 ba        jsr chrgot      letztes zeichen erneut nach ac
                                           (indirekter sprung)
2824  909b  f0 2e           beq endvar
2825  909d  20 30 97        jsr chkcom      test 'komma' sonst fehler + ready
2826  90a0  4c f9 8f        jmp inloop
2827  90a3
2828  90a3
2829  90a3  a5 15     trmnok lda inpflg     flag für eingabe (input,get,read)
2830  90a5  f0 0d           beq trmno1      behandlung eingabefehler
2831  90a7  10 08           bpl snerr6
2832  90a9  a5 49           lda datlin      addrl data-zeilennummer
2833  90ab  a4 4a           ldy datlin+1    addrh data-zeilennummer
2834  90ad  85 42           sta curlin      lo-byte akt. zeilennummer
2835  90af  84 43           sty curlin+1    hi-byte akt. zeilennummer
2836  90b1  4c 4f 97  snerr6 jmp snerr      ausgabe '?syntax error', ready
2837  90b4
2838  90b4
2839  90b4  ===> behandlung eingabefehler <===
2840  90b4
2841  90b4  a5 1a     trmno1 lda channl     speicherstelle für aktiven kanal
2842  90b6  f0 05           beq doagin      ausgabe '?redo from start', zeiger
                                           'oldtxt' nach 'txtptr'
2843  90b8  a2 44           ldx #$44
2844  90ba  4c 52 85        jmp error       ind. jmp zur fehlerroutine
2845  90bd
2846  90bd
2847  90bd  ===> ausgabe '?redo from start', zeiger 'oldtxt' nach 'txtptr' <===
2848  90bd
2849  90bd  a2 20     doagin ldx #$20
2850  90bf  20 c3 a3        jsr msg         textzeiger + xr, textausgabe
2851  90c2  a5 46           lda oldtxt      addrl zeiger letzter basic-befehl
2852  90c4  a4 47           ldy oldtxt+1    addrh zeiger letzter basic-befehl
2853  90c6  85 85           sta txtptr      addrl zeiger auf akt. term
2854  90c8  84 86           sty txtptr+1    addrh zeiger auf akt. term
2855  90ca  60              rts
2856  90cb
2857  90cb
2858  90cb  a5 4d     endvar lda inpptr     addrl input-zeilennummer
2859  90cd  a4 4e           ldy inpptr+1    addrh input-zeilennummer
2860  90cf  a6 15           ldx inpflg      flag für eingabe (input,get,read)
2861  90d1  10 05           bpl vary0
2862  90d3  85 4b           sta datptr      addrl zeiger auf data nach 'read'
2863  90d5  84 4c           sty datptr+1    addrh zeiger auf data nach 'read'
2864  90d7  60        inprts rts
2865  90d8
2866  90d8
2867  90d8  a0 00     vary0  ldy #$00
2868  90da  b1 4d           lda (inpptr),y  addrl input-zeilennummer
```

```
zeile adr.  obj.-code   source-code

2869  90dc  f0 f9           beq inprts
2870  90de  a5 1a           lda channl      speicherstelle für aktiven kanal
2871  90e0  d0 f5           bne inprts
2872  90e2  a2 1e           ldx #$1e
2873  90e4  4c c3 a3        jmp msg         textzeiger + xr, textausgabe
2874  90e7
2875  90e7
2876  90e7  ===> basic-routine 'sys' <===
2877  90e7
2878  90e7  20 e5 b4  csys   jsr getpin      dezimal-zahl in klammer nach linnum
                                             einlesen
2879  90ea  a5 01           lda i6509       6509 indirection register
2880  90ec  48              pha
2881  90ed  ad 57 02        lda dfbank      vorgabe für bank-nummer
2882  90f0  c9 0f           cmp #$0f
2883  90f2  f0 0b           beq fligm
2884  90f4  a6 1b           ldx linnum      lo-byte akt. zeilennummer
2885  90f6  a4 1c           ldy linnum+1    hi-byte akt. zeilennummer
2886  90f8  20 6c ff        jsr knwsys      transfer-of-execution jumper
2887  90fb
2888  90fb
2889  90fb  ===> bank# vor 'sys' nach i6509 <===
2890  90fb
2891  90fb  68         csysrz pla
2892  90fc  85 01           sta i6509       6509 indirection register
2893  90fe  60              rts
2894  90ff
2895  90ff
2896  90ff  a9 90      fligm  lda #$90
2897  9101  48              pha
2898  9102  a9 fa            lda #$fa
2899  9104  48              pha
2900  9105  6c 1b 00        jmp (linnum)    lo-byte akt. zeilennummer
2901  9108
2902  9108
2903  9108  60              rts
2904  9109
2905  9109
2906  9109  20 30 97  dim3   jsr chkcom      test 'komma' sonst fehler + ready
2907  910c
2908  910c
2909  910c  ===> basic-routine 'dim' <===
2910  910c
2911  910c  aa         dim    tax
2912  910d  20 31 99           jsr ptrgt1
2913  9110  20 29 ba           jsr chrgot      letztes zeichen erneut nach ac
                                             (indirekter sprung)
2914  9113  d0 f4              bne dim3
2915  9115  60                 rts
2916  9116
2917  9116
2918  9116  ===> basic-routine 'def' <===
2919  9116
2920  9116  20 5c 9d  defn     jsr getfnm      'fn' variable suchen / anlegen
2921  9119  20 4a 9d           jsr errdir      test direkt, ausgabe '?illegal
                                             direct error', ready
2922  911c  20 2d 97           jsr chkopn      test '(', sonst fehler + ready
2923  911f  a9 80              lda #$80
```

```
zeile adr.  obj.-code   source-code

2924  9121  85 14           sta subflg      flag für indizierte variable
2925  9123  20 2c 99        jsr ptrget      variable suchen/anlegen, deren name
                                            folgt
2926  9126  20 04 b5        jsr chknum      prüfen, ob numerische variable
2927  9129  20 2a 97        jsr chkcls      test')', sonst fehler + ready
2928  912c  a9 b2           lda #$b2
2929  912e  20 32 97        jsr synchr      test:folgt ascii o. token im akt.
                                            text, sonst error
2930  9131  a5 53           lda varpnt+2    bank zeiger auf variable im ram
2931  9133  48              pha
2932  9134  a5 52           lda varpnt+1    addrh zeiger auf variable im ram
2933  9136  48              pha
2934  9137  a5 51           lda varpnt      addrl zeiger auf variable im ram
2935  9139  48              pha
2936  913a  a5 86           lda txtptr+1    addrh zeiger auf akt. term
2937  913c  48              pha
2938  913d  a5 85           lda txtptr      addrl zeiger auf akt. term
2939  913f  48              pha
2940  9140  20 df 8c        jsr data        basic-routine 'data'
2941  9143  4c eb 9d        jmp deffin      funktionen-zeiger vom stack holen
2942  9146
2943  9146
2944  9146  ===> basic-routine 'poke' <===
2945  9146
2946  9146  20 ca b4  poke   jsr getnum     2 byte-zahl in 'linnum', kommatest,
                                            1 byte-zahl in xr
2947  9149  8a              txa
2948  914a  20 75 91        jsr sav32       i6509 in xr, vorgabe-bank # in
                                            i6509, yr=0
2949  914d  91 1b           sta (linnum),y  lo-byte akt. zeilennummer
2950  914f  86 01           stx i6509       6509 indirection register
2951  9151  60              rts
2952  9152
2953  9152
2954  9152  ===> basic-routine 'wait' <===
2955  9152
2956  9152  20 ca b4  fnwait jsr getnum     2 byte-zahl in 'linnum', kommatest,
                                            1 byte-zahl in xr
2957  9155  8e 55 02        stx trmpos      cursor spalte auf bildschirm
2958  9158  a2 00           ldx #$00
2959  915a  20 29 ba        jsr chrgot      letztes zeichen erneut nach ac
                                            (indirekter sprung)
2960  915d  f0 03           beq stordo
2961  915f  20 cd b4        jsr combyt      kommatest, holt 1byte in xr
2962  9162  8e 56 02  stordo stx eormsk     bitmaske für 'wait'
2963  9165  20 75 91        jsr sav32       i6509 in xr, vorgabe-bank # in
                                            i6509, yr=0
2964  9168  b1 1b     waiter lda (linnum),y  lo-byte akt. zeilennummer
2965  916a  4d 56 02        eor eormsk      bitmaske für 'wait'
2966  916d  2d 55 02        and trmpos      cursor spalte auf bildschirm
2967  9170  f0 f6           beq waiter
2968  9172  86 01           stx i6509       6509 indirection register
2969  9174  60              rts
2970  9175
2971  9175
2972  9175  ===> i6509 in xr, vorgabe-bank # in i6509, yr=0 <===
2973  9175
2974  9175  a6 01     sav32  ldx i6509       6509 indirection register
```

```
zeile adr.  obj.-code   source-code

2975  9177  ac 57 02        ldy dfbank      vorgabe für bank-nummer
2976  917a  84 01           sty i6509       6509 indirection register
2977  917c  a0 00           ldy #$00
2978  917e  60              rts
2979  917f
2980  917f
2981  917f  ===> basic-routine 'key' <===
2982  917f
2983  917f  d0 04     fkey   bne key20
2984  9181  a0 00           ldy #$00
2985  9183  f0 26           beq keygo
2986  9185  20 d6 b4  key20  jsr getbyt      zahl < 256 aus akt. text nach xr
2987  9188  8a              txa
2988  9189  d0 03           bne key40
2989  918b  4c a7 9b  fcerr3 jmp fcerr       ausgabe '?illegal quantity error',
                                             ready
2990  918e
2991  918e
2992  918e  c9 15     key40  cmp #$15
2993  9190  b0 f9           bcs fcerr3
2994  9192  48              pha
2995  9193  20 30 97        jsr chkcom      test 'komma' sonst fehler + ready
2996  9196  20 b6 91        jsr sav13
2997  9199  85 64           sta highds      addrl zielende (verschieben)/
                                             temp.fac#1
2998  919b  a5 22           lda index1      addrl indirekter index #1
2999  919d  a6 23           ldx index1+1    addrh indirekter index #1
3000  919f  a4 24           ldy index1+2    bank indirekter index #1
3001  91a1  85 65           sta highds+1    addrh zielende (verschieben/
                                             temp.fac#1
3002  91a3  86 66           stx highds+2    bank zielende (verschieben/
                                             temp.fac#1
3003  91a5  84 67           sty hightr      addrl ursprungende (verschieben)/
                                             temp.fac#1
3004  91a7  68              pla
3005  91a8  a8              tay
3006  91a9  a9 64           lda #$64
3007  91ab  20 75 ff  keygo  jsr kfunky      vector für funktionstasten
3008  91ae  90 05           bcc bakrts
3009  91b0  a2 34           ldx #$34
3010  91b2  6c 80 02        jmp (ierror)    addrl fehler routine
3011  91b5
3012  91b5
3013  91b5  60        bakrts rts
3014  91b6
3015  91b6
3016  91b6  20 c1 95  sav13  jsr frmevl      auswertung beliebiger ausdruck
3017  91b9  4c e6 a8        jmp frestr      alten string entfernen
3018  91bc
3019  91bc
3020  91bc  ===> basic-routine 'verify' <===
3021  91bc
3022  91bc  a9 4e     cverf  lda #$4e
3023  91be  48              pha
3024  91bf  a9 80           lda #$80
3025  91c1  20 00 92        jsr ldver
3026  91c4  f0 20           beq cvf1
3027  91c6  68              pla
```

```
zeile adr.  obj.-code   source-code

3028  91c7  60              rts
3029  91c8
3030  91c8
3031  91c8  ===> basic-routine 'load' <===
3032  91c8
3033  91c8  a9 00     cload  lda #$00
3034  91ca  20 00 92         jsr ldver
3035  91cd  f0 09     loadck beq clf1
3036  91cf  20 a4 86         jsr lnkprg      linkadressen berechnen
3037  91d2  20 b8 b4         jsr stxtpt      programmzeiger auf start
3038  91d5  4c 72 8a         jmp fload       reset 'read'-,stack-,befehl-zeiger
3039  91d8
3040  91d8
3041  91d8  ad 5b 02  clf1   lda ldaadr      addrl modifiable adresse
3042  91db  c9 01            cmp #$01
3043  91dd  d0 1a            bne cld4
3044  91df  86 2f            stx txtend      addrl zeiger ende  basic-text
3045  91e1  84 30            sty txtend+1    addrh zeiger ende  basic-text
3046  91e3  a9 4c            lda #$4c
3047  91e5  48               pha
3048  91e6  20 dd bb  cvf1   jsr readst      system-status in ac lesen
3049  91e9  29 10            and #$10
3050  91eb  d0 04            bne cld2
3051  91ed  68               pla
3052  91ee  a9 18            lda #$18
3053  91f0  48               pha
3054  91f1  68        cld2   pla
3055  91f2  aa               tax
3056  91f3  20 c3 a3  cld3   jsr msg         textzeiger + xr, textausgabe
3057  91f6  4c 9b 86         jmp fini        linkadr. berechnen, ready
3058  91f9
3059  91f9
3060  91f9  a6 56     cld4   ldx lstpnt+2    bank zeiger letzter string
3061  91fb  d0 f6            bne cld3
3062  91fd  48        loadnp pha
3063  91fe  f0 04            beq lvrl
3064  9200  48        ldver  pha
3065  9201  20 b2 b9         jsr plsv
3066  9204  20 e1 ff  lvrl   jsr kstop       test stop-taste
3067  9207  f0 fb            beq lvrl
3068  9209  68               pla
3069  920a  09 01            ora #$01
3070  920c  a6 2d            ldx txttab      addrl zeiger anfang  basic-text
3071  920e  a4 2e            ldy txttab+1    addrh zeiger anfang  basic-text
3072  9210  20 06 bc         jsr load        einlesen vom logischen file
3073  9213  b0 29            bcs ldsver
3074  9215  8d 5b 02         sta ldaadr      addrl modifiable adresse
3075  9218  4c 57 9d         jmp tstdir      test direktmodus, hi-byt zeile=$ff
3076  921b
3077  921b
3078  921b  ===> basic-routine 'save' <===
3079  921b
3080  921b  20 b2 b9  csave  jsr plsv
3081  921e  a6 2d     savenp ldx txttab      addrl zeiger anfang  basic-text
3082  9220  a4 2e            ldy txttab+1    addrh zeiger anfang  basic-text
3083  9222  a9 01            lda #$01
3084  9224  86 64            stx highds      addrl zielende (verschieben)/
                                             temp.fac#1
```

```
zeile adr.  obj.-code   source-code

3085  9226  84 65           sty highds+1     addrh zielende (verschieben/
                                             temp.fac#1
3086  9228  85 66           sta highds+2     bank zielende (verschieben/
                                             temp.fac#1
3087  922a  a6 2f           ldx txtend       addrl zeiger ende  basic-text
3088  922c  a4 30           ldy txtend+1     addrh zeiger ende  basic-text
3089  922e
3090  922e
3091  922e  ===> teil der save-routine <===
3092  922e
3093  922e  86 67     savenb stx hightr       addrl ursprungende (verschieben)/
                                             temp.fac#1
3094  9230  84 68           sty hightr+1     addrh ursprungende (verschieben)/
                                             temp.fac#1
3095  9232  85 69           sta hightr+2     bank ursprungende (verschieben)
3096  9234  a2 64           ldx #$64
3097  9236  a0 67           ldy #$67
3098  9238  20 0c bc         jsr save         abspeichern auf logisches file
3099  923b  b0 01           bcs ldsver
3100  923d  60              rts
3101  923e
3102  923e
3103  923e  0a       ldsver asl a
3104  923f  aa              tax
3105  9240  4c 52 85        jmp error        ind. jmp zur fehlerroutine
3106  9243
3107  9243
3108  9243  ===> basic-routine 'open' <===
3109  9243
3110  9243  a2 00     copen  ldx #$00
3111  9245  86 8b           stx parsts       dos analyse-byte 1
3112  9247  8e 10 02        stx dosf1l       dos filename 1 länge
3113  924a  8e 21 02        stx dossa        dos sekundär-adresse
3114  924d  e8              inx
3115  924e  8e 20 02        stx dosfa        dos primär-adresse
3116  9251  20 04 ba        jsr plsv32
3117  9254  20 d6 b4        jsr getbyt       zahl < 256 aus akt. text nach xr
3118  9257  8e 1f 02        stx dosla        dos logische adresse
3119  925a  20 87 92        jsr cops
3120  925d  8e 20 02        stx dosfa        dos primär-adresse
3121  9260  a0 00           ldy #$00
3122  9262  e0 03           cpx #$03
3123  9264  90 01           bcc cop1
3124  9266  88              dey
3125  9267  8c 21 02  cop1   sty dossa        dos sekundär-adresse
3126  926a  20 87 92        jsr cops
3127  926d  8e 21 02        stx dossa        dos sekundär-adresse
3128  9270  20 29 ba        jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
3129  9273  f0 06           beq copx
3130  9275  20 01 ba        jsr plsv30
3131  9278  20 9c b9        jsr sav77
3132  927b  20 d6 b9  copx   jsr plsvx
3133  927e  20 cc ff  globtt jsr kclrch       ein-/ausgabekanal schließen
3134  9281  20 e1 bb        jsr open         log. file öffnen/befehl ausgeben
3135  9284  4c 50 93        jmp dcat0
3136  9287
3137  9287
```

```
zeile adr.  obj.-code   source-code

3138  9287  20 29 ba  cops   jsr chrgot       letztes zeichen erneut nach ac
                                              (indirekter sprung)
3139  928a  d0 05            bne copg
3140  928c  68               pla
3141  928d  68               pla
3142  928e  4c 7b 92         jmp copx
3143  9291
3144  9291
3145  9291  20 01 ba  copg   jsr plsv30
3146  9294  4c d6 b4         jmp getbyt       zahl < 256 aus akt. text nach xr
3147  9297
3148  9297
3149  9297  ===> basic-routine 'close' <===
3150  9297
3151  9297  20 04 ba  cclos  jsr plsv32
3152  929a  20 d6 b4         jsr getbyt       zahl < 256 aus akt. text nach xr
3153  929d  8a               txa
3154  929e  4c 12 bc         jmp close        carry=1, log. file schliessen
3155  92a1
3156  92a1                   .end
3157  92a1                   .lib bverbs3
```

```
zeile adr.  obj.-code   source-code

3159  92a1
3160  92a1  ===> disc-basic-rout. 'catalog-dir.' <===
3161  92a1
3162  92a1  20 47 b5  dcat   jsr dospar       ersatzwerte vor disk-befehl init.
3163  92a4  a5 8b            lda parsts       dos analyse-byte 1
3164  92a6  29 e6            and #$e6
3165  92a8  f0 03            beq dcat00
3166  92aa  4c 4f 97         jmp snerr        ausgabe '?syntax error', ready
3167  92ad
3168  92ad
3169  92ad  a0 01      dcat00 ldy #$01
3170  92af  a2 01             ldx #$01
3171  92b1  a5 8b             lda parsts       dos analyse-byte 1
3172  92b3  29 11             and #$11
3173  92b5  f0 06             beq dcat2
3174  92b7  4a                lsr a
3175  92b8  90 02             bcc dcat1
3176  92ba  e8                inx
3177  92bb  e8                inx
3178  92bc  e8         dcat1  inx
3179  92bd  8a         dcat2  txa
3180  92be  20 12 b8          jsr sendp        dos befehlsstring zusammensetzen u.
                                               ausgeben
3181  92c1  a9 15             lda #$15
3182  92c3  85 0f             sta xcnt         dos loop-zähler
3183  92c5  a0 60             ldy #$60
3184  92c7  20 5b 93          jsr ochanl
3185  92ca  a0 03             ldy #$03
3186  92cc  84 7b      dcat3  sty argmoh       fac-arg ungepackt: mantisse
3187  92ce  a2 0e             ldx #$0e
3188  92d0  20 fa bb          jsr chkin        eingabekanal öffnen, fehlertest
3189  92d3  20 ee bb          jsr basin        eingabe 1 char vom akt. kanal
                                               (chn-vektor)
3190  92d6  85 7c             sta argmo        fac-arg ungepackt: mantisse
3191  92d8  20 dd bb          jsr readst       system-status in ac lesen
3192  92db  d0 6b             bne dcat10
3193  92dd  20 ee bb          jsr basin        eingabe 1 char vom akt. kanal
                                               (chn-vektor)
3194  92e0  85 7d             sta arglo        fac-arg ungepackt: mantisse
3195  92e2  20 dd bb          jsr readst       system-status in ac lesen
3196  92e5  d0 61             bne dcat10
3197  92e7  a4 7b             ldy argmoh       fac-arg ungepackt: mantisse
3198  92e9  88                dey
3199  92ea  d0 e0             bne dcat3
3200  92ec  20 50 93          jsr dcat0
3201  92ef  a6 7c             ldx argmo        fac-arg ungepackt: mantisse
3202  92f1  a5 7d             lda arglo        fac-arg ungepackt: mantisse
3203  92f3  20 b4 a3          jsr linprt       umw. hex/dez, lo=xr, hi=ac, druck
                                               zeilen #
3204  92f6  a9 20             lda #$20
3205  92f8  20 f4 bb          jsr bsout        ausgabe 1 char auf akt. kanal
3206  92fb  20 cc ff   dcat4  jsr kclrch       ein-/ausgabekanal schließen
3207  92fe  a2 0e             ldx #$0e
3208  9300  20 fa bb          jsr chkin        eingabekanal öffnen, fehlertest
3209  9303  20 ee bb          jsr basin        eingabe 1 char vom akt. kanal
                                               (chn-vektor)
3210  9306  48                pha
3211  9307  20 dd bb          jsr readst       system-status in ac lesen
```

```
zeile adr.  obj.-code   source-code

3212  930a  d0 3b           bne dcat9
3213  930c  20 50 93        jsr dcat0
3214  930f  68              pla
3215  9310  f0 05           beq dcat5
3216  9312  20 f4 bb        jsr bsout       ausgabe 1 char auf akt. kanal
3217  9315  90 e4           bcc dcat4
3218  9317  c6 0f     dcat5 dec xcnt        dos loop-zähler
3219  9319  a9 0d           lda #$0d
3220  931b  20 f4 bb        jsr bsout       ausgabe 1 char auf akt. kanal
3221  931e  20 cc ff        jsr kclrch      ein-/ausgabekanal schließen
3222  9321  20 e1 ff        jsr kstop       test stop-taste
3223  9324  f0 22           beq dcat10
3224  9326  a5 0f           lda xcnt        dos loop-zähler
3225  9328  d0 19           bne dcat8
3226  932a  a9 15           lda #$15
3227  932c  85 0f           sta xcnt        dos loop-zähler
3228  932e  20 57 9d        jsr tstdir      test direktmodus, hi-byt zeile=$ff
3229  9331  d0 10           bne dcat8
3230  9333  a6 1a           ldx channl      speicherstelle für aktiven kanal
3231  9335  e0 00           cpx #$00
3232  9337  d0 0a           bne dcat8
3233  9339  a2 24           ldx #$24
3234  933b  20 c3 a3        jsr msg         textzeiger + xr, textausgabe
3235  933e  20 e8 bb  dcat7 jsr getin       eingabe 1 char vom akt. kanal
                                            (queue-vektor)
3236  9341  f0 fb           beq dcat7
3237  9343  a0 02     dcat8 ldy #$02
3238  9345  d0 85           bne dcat3
3239  9347  68        dcat9 pla
3240  9348  20 cc ff  dcat10 jsr kclrch      ein-/ausgabekanal schließen
3241  934b  a9 0e           lda #$0e
3242  934d  20 12 bc        jsr close       carry=1, log. file schliessen
3243  9350  20 cc ff  dcat0  jsr kclrch      ein-/ausgabekanal schließen
3244  9353  ae 58 02        ldx dolu        vorgabe für ausgabe-adresse
3245  9356  f0 4b           beq goorat
3246  9358  4c 00 bc        jmp chkout      ausgabekanal öffnen, fehlertest
3247  935b
3248  935b
3249  935b  ae 20 02  ochanl ldx dosfa       dos primär-adresse
3250  935e  d0 02           bne och110
3251  9360  a2 08           ldx #$08
3252  9362  a9 0e     och110 lda #$0e
3253  9364  20 ba ff        jsr kstlfs      log. filenummer, geräte- und
                                            sekundäradr. eintragen
3254  9367  20 cc ff        jsr kclrch      ein-/ausgabekanal schließen
3255  936a  4c e1 bb        jmp open        log. file öffnen/befehl ausgeben
3256  936d
3257  936d
3258  936d  ===> disc-basic-routine 'dopen' <===
3259  936d
3260  936d  a9 22     dopen  lda #$22
3261  936f  20 8e 93        jsr sav9
3262  9372  a0 02           ldy #$02
3263  9374  a2 04           ldx #$04
3264  9376  24 8b           bit parsts      dos analyse-byte 1
3265  9378  50 0d           bvc dop2
3266  937a  a2 08           ldx #$08
3267  937c  d0 09           bne dop2
```

```
zeile adr.  obj.-code   source-code

3268  937e
3269  937e
3270  937e  ===> disc-basic-routine 'append' <===
3271  937e
3272  937e  a9 e2     append lda #$e2
3273  9380  20 8e 93         jsr sav9
3274  9383  a0 03            ldy #$03
3275  9385  a2 05            ldx #$05
3276  9387  8a        dop2   txa
3277  9388  20 12 b8         jsr sendp        dos befehlsstring zusammensetzen u.
                                               ausgeben
3278  938b  4c 7e 92         jmp globtt
3279  938e
3280  938e
3281  938e  20 49 b5  sav9   jsr dosprs
3282  9391  20 19 b9         jsr chk6
3283  9394  a0 61            ldy #$61
3284  9396  c8        fsca10 iny
3285  9397  c0 6f            cpy #$6f
3286  9399  f0 09            beq fsca20
3287  939b  20 8a ff         jsr klkups       geräteadr. über sekundäradr.suchen
3288  939e  90 f6            bcc fsca10
3289  93a0  8c 21 02         sty dossa        dos sekundär-adresse
3290  93a3  60        goorat rts
3291  93a4
3292  93a4
3293  93a4  a2 02     fsca20 ldx #$02
3294  93a6  4c 52 85         jmp error        ind. jmp zur fehlerroutine
3295  93a9
3296  93a9
3297  93a9  ===> disc-basic-routine 'dclose' <===
3298  93a9
3299  93a9  a9 f3     dclose lda #$f3
3300  93ab  20 49 b5         jsr dosprs
3301  93ae  20 94 b9         jsr oldclr       länge disk.status =0, system-status
                                               schreiben
3302  93b1  a5 8b            lda parsts       dos analyse-byte 1
3303  93b3  29 04            and #$04
3304  93b5  f0 06            beq dclall       dos-ger;t log. file schliessen
3305  93b7  ad 1f 02         lda dosla        dos logische adresse
3306  93ba  4c 12 bc         jmp close        carry=1, log. file schliessen
3307  93bd
3308  93bd
3309  93bd  ===> dos-ger;t log. file schliessen <===
3310  93bd
3311  93bd  ad 20 02  dclall lda dosfa        dos primär-adresse
3312  93c0  4c 92 95         jmp patch3       alle logischen files schliessen
3313  93c3
3314  93c3
3315  93c3  ===> disc-basic-routine 'dsave' <===
3316  93c3
3317  93c3  a9 66     dsave  lda #$66
3318  93c5  20 49 b5         jsr dosprs
3319  93c8  20 0b b8         jsr sav20
3320  93cb  4c 1e 92         jmp savenp
3321  93ce
3322  93ce
3323  93ce  ===> disc-basic-routine 'dload' <===
```

```
zeile adr.  obj.-code   source-code

3324  93ce
3325  93ce  a9 e6     dload  lda #$e6
3326  93d0  20 49 b5         jsr dosprs
3327  93d3  20 0b b8         jsr sav20
3328  93d6  a9 00            lda #$00
3329  93d8  20 fd 91         jsr loadnp
3330  93db  4c cd 91         jmp loadck
3331  93de
3332  93de
3333  93de  ===> basic-routine 'bank' <===
3334  93de
3335  93de  20 d6 b4  cnbank jsr getbyt       zahl < 256 aus akt. text nach xr
3336  93e1  e0 10            cpx #$10
3337  93e3  b0 04            bcs xbkerr
3338  93e5  8e 57 02         stx dfbank       vorgabe für bank-nummer
3339  93e8  60        bsvrts rts
3340  93e9
3341  93e9
3342  93e9  4c a7 9b  xbkerr jmp fcerr        ausgabe '?illegal quantity error',
                                               ready
3343  93ec
3344  93ec
3345  93ec  ===> disc-basic-routine 'bsave' <===
3346  93ec
3347  93ec  a9 66     bsave  lda #$66
3348  93ee  a2 f8            ldx #$f8
3349  93f0  20 4b b5         jsr dosprx
3350  93f3  20 0b b8         jsr sav20
3351  93f6  ae 1b 02         ldx dosofl       addrl dos offset untergrenze
                                               (bsave,bload)
3352  93f9  ac 1c 02         ldy dosofl+1     addrh dos offset untergrenze
                                               (bsave,bload)
3353  93fc  ad 1a 02         lda dosbnk       dos bank nummer
3354  93ff  86 64            stx highds       addrl zielende (verschieben)/
                                               temp.fac#1
3355  9401  84 65            sty highds+1     addrh zielende (verschieben/
                                               temp.fac#1
3356  9403  85 66            sta highds+2     bank zielende (verschieben/
                                               temp.fac#1
3357  9405  ae 1d 02         ldx dosofh       addrl dos offset obergrenze
                                               (bsave,bload)
3358  9408  ac 1e 02         ldy dosofh+1     addrh dos offset obergrenze
                                               (bsave,bload)
3359  940b  4c 2e 92         jmp savenb       teil der save-routine
3360  940e
3361  940e
3362  940e  ===> disc-basic-routine 'bload' <===
3363  940e
3364  940e  a9 e6     bload  lda #$e6
3365  9410  a2 fc            ldx #$fc
3366  9412  20 4b b5         jsr dosprx
3367  9415  20 0b b8         jsr sav20
3368  9418  ad 1a 02         lda dosbnk       dos bank nummer
3369  941b  18               clc
3370  941c  ae 1b 02         ldx dosofl       addrl dos offset untergrenze
                                               (bsave,bload)
3371  941f  ac 1c 02         ldy dosofl+1     addrh dos offset untergrenze
                                               (bsave,bload)
```

```
zeile adr.  obj.-code   source-code

3372  9422  20 06 bc          jsr load          einlesen vom logischen file
3373  9425  90 c1             bcc bsvrts
3374  9427
3375  9427
3376  9427  ===> disc-basic-routine 'header' <===
3377  9427
3378  9427  20 47 b5  format jsr dospar        ersatzwerte vor disk-befehl init.
3379  942a  20 f3 b8         jsr chk1          syntaxprüfung disk-basic
3380  942d  29 11            and #$11
3381  942f  c9 11            cmp #$11
3382  9431  d0 68            bne rec5
3383  9433  20 bd 93         jsr dclall        dos-ger;t log. file schliessen
3384  9436  20 5b b9         jsr rusure        bearbeitung 'are you sure ?'
3385  9439  b0 ad            bcs bsvrts
3386  943b  a0 04            ldy #$04
3387  943d  a9 04            lda #$04
3388  943f  ae 23 02         ldx dosdid        dos erstes zeichen 'id'
3389  9442  f0 02            beq frmt6
3390  9444  a9 06            lda #$06
3391  9446  20 78 95  frmt6  jsr trans
3392  9449  20 22 b9         jsr errchl        statusstring über sek.adr. 15
                                               einlesen
3393  944c  20 57 9d         jsr tstdir        test direktmodus, hi-byt zeile=$ff
3394  944f  d0 97            bne bsvrts
3395  9451  20 7d ba         jsr mapstr        umsch. auf bank # 2
3396  9454  a0 00            ldy #$00
3397  9456  b1 17            lda (dsdesc+1),y addrl disc-status string
3398  9458  c9 32            cmp #$32
3399  945a  b0 03            bcs frmt8
3400  945c  4c 8c ba         jmp mapusr        umsch. auf bank # 1
3401  945f
3402  945f
3403  945f  a2 16     frmt8  ldx #$16
3404  9461  4c 52 85         jmp error         ind. jmp zur fehlerroutine
3405  9464
3406  9464
3407  9464  ===> disc-basic-routine 'scratch' <===
3408  9464
3409  9464  20 47 b5  scratc jsr dospar        ersatzwerte vor disk-befehl init.
3410  9467  20 f3 b8         jsr chk1          syntaxprüfung disk-basic
3411  946a  20 5b b9         jsr rusure        bearbeitung 'are you sure ?'
3412  946d  b0 2b            bcs zxit
3413  946f  a0 0a            ldy #$0a
3414  9471  a9 04            lda #$04
3415  9473  20 78 95         jsr trans
3416  9476  20 22 b9         jsr errchl        statusstring über sek.adr. 15
                                               einlesen
3417  9479  20 57 9d         jsr tstdir        test direktmodus, hi-byt zeile=$ff
3418  947c  d0 1c            bne zxit
3419  947e  a9 0d            lda #$0d
3420  9480  20 f4 bb         jsr bsout         ausgabe 1 char auf akt. kanal
3421  9483  20 7d ba         jsr mapstr        umsch. auf bank # 2
3422  9486  a0 00            ldy #$00
3423  9488  b1 17     sctc1  lda (dsdesc+1),y addrl disc-status string
3424  948a  f0 06            beq sctc2
3425  948c  20 f4 bb         jsr bsout         ausgabe 1 char auf akt. kanal
3426  948f  c8               iny
3427  9490  d0 f6            bne sctc1
```

```
zeile adr.  obj.-code   source-code

3428  9492  a9 0d     sctc2  lda #$0d
3429  9494  20 f4 bb         jsr bsout       ausgabe 1 char auf akt. kanal
3430  9497  20 8c ba         jsr mapusr      umsch. auf bank # 1
3431  949a  60        zxit   rts
3432  949b
3433  949b
3434  949b  4c 4f 97  rec5   jmp snerr       ausgabe '?syntax error', ready
3435  949e
3436  949e
3437  949e  ===> disc-basic-routine 'record' <===
3438  949e
3439  949e  a9 01     record lda #$01
3440  94a0  8d 22 02         sta dosrcl      dos record-länge
3441  94a3  20 29 ba         jsr chrgot      letztes zeichen erneut nach ac
                                             (indirekter sprung)
3442  94a6  a9 23            lda #$23
3443  94a8  20 32 97         jsr synchr      test:folgt ascii o. token im akt.
                                             text, sonst error
3444  94ab  20 8d b7         jsr gtvl2
3445  94ae  e0 00            cpx #$00
3446  94b0  f0 16            beq rec4
3447  94b2  8e 1f 02         stx dosla       dos logische adresse
3448  94b5  20 30 97         jsr chkcom      test 'komma' sonst fehler + ready
3449  94b8  f0 e1            beq rec5
3450  94ba  90 14            bcc rec1
3451  94bc  20 2d 97         jsr chkopn      test '(', sonst fehler + ready
3452  94bf  20 e5 b4         jsr getpin      dezimal-zahl in klammer nach linnum
                                             einlesen
3453  94c2  20 2a 97         jsr chkcls      test')', sonst fehler + ready
3454  94c5  4c d3 94         jmp rec2
3455  94c8
3456  94c8
3457  94c8  4c 1b b7  rec4   jmp qtyerr      ausgabe 'illegal quantity error',
                                             ready
3458  94cb
3459  94cb
3460  94cb  a2 06     rec6   ldx #$06
3461  94cd  4c 52 85         jmp error       ind. jmp zur fehlerroutine
3462  94d0
3463  94d0
3464  94d0  20 e5 b4  rec1   jsr getpin      dezimal-zahl in klammer nach linnum
                                             einlesen
3465  94d3  20 29 ba  rec2   jsr chrgot      letztes zeichen erneut nach ac
                                             (indirekter sprung)
3466  94d6  f0 18            beq rec3
3467  94d8  20 30 97         jsr chkcom      test 'komma' sonst fehler + ready
3468  94db  f0 be            beq rec5
3469  94dd  20 8d b7         jsr gtvl2
3470  94e0  e0 00            cpx #$00
3471  94e2  f0 e4            beq rec4
3472  94e4  e0 ff            cpx #$ff
3473  94e6  f0 e0            beq rec4
3474  94e8  8e 22 02         stx dosrcl      dos record-länge
3475  94eb  20 29 ba         jsr chrgot      letztes zeichen erneut nach ac
                                             (indirekter sprung)
3476  94ee  d0 ab     aaaa   bne rec5
3477  94f0  ad 1f 02  rec3   lda dosla       dos logische adresse
3478  94f3  20 8d ff         jsr klkupl      geräte- sekundäradr. über logische
                                             file # suchen
```

```
zeile adr.  obj.-code   source-code

3479  94f6  b0 d3           bcs rec6
3480  94f8  8c 21 02        sty dossa       dos sekundär-adresse
3481  94fb  8e 20 02        stx dosfa       dos primär-adresse
3482  94fe  20 ba ff        jsr kstlfs      log. filenummer, geräte- und
                                            sekundäradr. eintragen
3483  9501  20 94 b9        jsr oldclr      länge disk.status =0, system-status
                                            schreiben
3484  9504  a0 0b           ldy #$0b
3485  9506  a9 04           lda #$04
3486  9508  d0 6e           bne trans
3487  950a
3488  950a
3489  950a  ===> disc-basic-routine 'dclear' <===
3490  950a
3491  950a  20 47 b5  dclear jsr dospar      ersatzwerte vor disk-befehl init.
3492  950d  a0 00           ldy #$00
3493  950f  a9 02           lda #$02
3494  9511  d0 65           bne trans
3495  9513
3496  9513
3497  9513  ===> disc-basic-routine 'collect' <===
3498  9513
3499  9513  20 47 b5  colect jsr dospar      ersatzwerte vor disk-befehl init.
3500  9516  20 05 b9        jsr chk3
3501  9519  20 bd 93        jsr dclall      dos-ger;t log. file schliessen
3502  951c  a0 05           ldy #$05
3503  951e  a2 01           ldx #$01
3504  9520  a5 8b           lda parsts      dos analyse-byte 1
3505  9522  29 10           and #$10
3506  9524  f0 01           beq clct2
3507  9526  e8              inx
3508  9527  8a        clct2  txa
3509  9528  d0 4e           bne trans
3510  952a
3511  952a
3512  952a  ===> disc-basic-routine 'copy' <===
3513  952a
3514  952a  20 47 b5  dcopy  jsr dospar      ersatzwerte vor disk-befehl init.
3515  952d  29 30           and #$30
3516  952f  c9 30           cmp #$30
3517  9531  d0 06           bne dcpy2
3518  9533  a5 8b           lda parsts      dos analyse-byte 1
3519  9535  29 c7           and #$c7
3520  9537  f0 07           beq dcpy4
3521  9539  a5 8b     dcpy2 lda parsts      dos analyse-byte 1
3522  953b  20 0a b9        jsr chk4
3523  953e  a5 8b           lda parsts      dos analyse-byte 1
3524  9540  a0 07     dcpy4 ldy #$07
3525  9542  a9 08           lda #$08
3526  9544  d0 32           bne trans
3527  9546
3528  9546
3529  9546  ===> disc-basic-routine 'concat' <===
3530  9546
3531  9546  20 47 b5  concat jsr dospar      ersatzwerte vor disk-befehl init.
3532  9549  20 0a b9        jsr chk4
3533  954c  a0 08           ldy #$08
3534  954e  a9 0c           lda #$0c
```

```
zeile adr.  obj.-code   source-code

3535  9550  d0 26            bne trans
3536  9552
3537  9552
3538  9552  ===> disc-basic-routine 'rename' <===
3539  9552
3540  9552  a9 e4     rename lda #$e4
3541  9554  20 49 b5        jsr dosprs
3542  9557  20 10 b9        jsr chk5
3543  955a  a0 09           ldy #$09
3544  955c  a9 08           lda #$08
3545  955e  d0 18           bne trans
3546  9560
3547  9560
3548  9560  ===> disc-basic-routine 'backup' <===
3549  9560
3550  9560  a9 c7     backup lda #$c7
3551  9562  20 49 b5        jsr dosprs
3552  9565  29 30           and #$30
3553  9567  c9 30           cmp #$30
3554  9569  d0 83           bne aaaa
3555  956b  20 5b b9        jsr rusure      bearbeitung 'are you sure ?'
3556  956e  90 01           bcc bup1
3557  9570  60              rts
3558  9571
3559  9571
3560  9571  20 bd 93  bup1  jsr dclall      dos-ger;t log. file schliessen
3561  9574  a0 06           ldy #$06
3562  9576  a9 04           lda #$04
3563  9578  20 12 b8  trans jsr sendp       dos befehlsstring zusammensetzen u.
                                            ausgeben
3564  957b  20 cc ff        jsr kclrch      ein-/ausgabekanal schließen
3565  957e  38              sec
3566  957f  20 e2 bb        jsr opn10
3567  9582  4c 50 93        jmp dcat0
3568  9585
3569  9585                  .end
3570  9585                  .lib math1
```

```
zeile adr.  obj.-code   source-code

3572  9585
3573  9585  ===> prüfbyte rom $8000-$9fff <===
3574  9585
3575  9585  22        forfix .byte $22
3576  9586
3577  9586
3578  9586  ===> system-status schreiben, iec listen <===
3579  9586
3580  9586  20 94 b9  patch1 jsr oldclr       länge disk.status =0, system-status
                                              schreiben
3581  9589  4c 00 bc          jmp chkout      ausgabekanal öffnen, fehlertest
3582  958c
3583  958c
3584  958c  ===> system-status schreiben, iec-talk <===
3585  958c
3586  958c  20 94 b9  patch2 jsr oldclr       länge disk.status =0, system-status
                                              schreiben
3587  958f  4c fa bb          jmp chkin       eingabekanal öffnen, fehlertest
3588  9592
3589  9592
3590  9592  ===> alle logischen files schliessen <===
3591  9592
3592  9592  38        patch3 sec
3593  9593  4c e7 ff          jmp kclall      alle logischen files schliessen
3594  9596
3595  9596
3596  9596                 * = *+$2b
3597  95c1
3598  95c1  6c 8c 02  frmevl jmp (ifrmev)     addrl beliebigen ausdruck auswerten
3599  95c4
3600  95c4
3601  95c4  ===> auswertung beliebiger ausdruck <===
3602  95c4
3603  95c4  a6 85     nfrmev ldx txtptr       addrl zeiger auf akt. term
3604  95c6  d0 02            bne frmev1
3605  95c8  c6 86            dec txtptr+1     addrh zeiger auf akt. term
3606  95ca  c6 85     frmev1 dec txtptr       addrl zeiger auf akt. term
3607  95cc  a2 00            ldx #$00
3608  95ce  24               .byte $24
3609  95cf
3610  95cf
3611  95cf  ===> schleifenanfang <===
3612  95cf
3613  95cf  48        lpoper pha
3614  95d0  8a               txa
3615  95d1  48               pha
3616  95d2  a9 01            lda #$01
3617  95d4  20 66 88         jsr getstk       test stapelplatz
3618  95d7  20 ae 96         jsr eval         jmp: folgenden beliebigen ausdruck
                                              auswerten
3619  95da  a9 00            lda #$00
3620  95dc  85 5a            sta opmask       vom akt. operator erzeugte bit-maske
3621  95de  20 29 ba  tstop  jsr chrgot       letztes zeichen erneut nach ac
                                              (indirekter sprung)
3622  95e1  38        loprel sec
3623  95e2  e9 b1            sbc #$b1
3624  95e4  90 17            bcc endrel
3625  95e6  c9 03            cmp #$03
```

```
zeile adr.  obj.-code   source-code

3626  95e8  b0 13           bcs endrel
3627  95ea  c9 01           cmp #$01
3628  95ec  2a              rol a
3629  95ed  49 01           eor #$01
3630  95ef  45 5a           eor opmask      vom akt. operator erzeugte bit-maske
3631  95f1  c5 5a           cmp opmask      vom akt. operator erzeugte bit-maske
3632  95f3  90 61           bcc snerr5      ausgabe '?syntax error', ready
3633  95f5  85 5a           sta opmask      vom akt. operator erzeugte bit-maske
3634  95f7  20 26 ba        jsr chrget      nächstes zeichen nach ac (ind.jmp)
3635  95fa  4c e1 95        jmp loprel
3636  95fd
3637  95fd
3638  95fd  a6 5a     endrel ldx opmask      vom akt. operator erzeugte bit-maske
3639  95ff  d0 2c           bne finrel
3640  9601  b0 7f           bcs qop
3641  9603  69 07           adc #$07
3642  9605  90 7b           bcc qop
3643  9607  65 11           adc valtyp      flag für variablen typ (0=num,
                                            1=string)
3644  9609  d0 03           bne endre1
3645  960b  4c 7a aa        jmp cat         string verketten '+'
3646  960e
3647  960e
3648  960e  69 ff     endre1 adc #$ff
3649  9610  85 22           sta index1      addrl indirekter index #1
3650  9612  0a              asl a
3651  9613  65 22           adc index1      addrl indirekter index #1
3652  9615  a8              tay
3653  9616  68       qprec  pla
3654  9617  d9 d1 80        cmp optab,y     adresse / prioritätsflags der math.
                                            rout.
3655  961a  b0 6b           bcs qchnum
3656  961c  20 04 b5        jsr chknum      prüfen, ob numerische variable
3657  961f  48       doprec pha
3658  9620  20 46 96 negprc jsr dopre1      op-rout.adr. u. operand auf stapel
3659  9623  68              pla
3660  9624  a4 57           ldy opptr       addrl zeiger akt. operator-routine
3661  9626  10 17           bpl qprec1
3662  9628  aa              tax
3663  9629  f0 5a           beq qopgo
3664  962b  d0 63           bne pulstk
3665  962d  46 11     finrel lsr valtyp      flag für variablen typ (0=num,
                                            1=string)
3666  962f  8a              txa
3667  9630  2a              rol a
3668  9631  a6 85           ldx txtptr      addrl zeiger auf akt. term
3669  9633  d0 02           bne finre2
3670  9635  c6 86           dec txtptr+1    addrh zeiger auf akt. term
3671  9637  c6 85     finre2 dec txtptr      addrl zeiger auf akt. term
3672  9639  a0 1b           ldy #$1b
3673  963b  85 5a           sta opmask      vom akt. operator erzeugte bit-maske
3674  963d  d0 d7           bne qprec
3675  963f  d9 d1 80  qprec1 cmp optab,y     adresse / prioritätsflags der math.
                                            rout.
3676  9642  b0 4c           bcs pulstk
3677  9644  90 d9           bcc doprec
3678  9646
3679  9646
```

```
zeile adr.  obj.-code   source-code

3680  9646  ===> op-rout.adr. u. operand auf stapel <===
3681  9646
3682  9646  b9 d3 80  dopre1 lda optab+1+1,y  adresse / prioritätsflags der math.
                                              rout.
3683  9649  48               pha
3684  964a  b9 d2 80         lda optab+1,y    adresse / prioritätsflags der math.
                                              rout.
3685  964d  48               pha
3686  964e  20 59 96         jsr pushf1       operanden auf stapel legen
3687  9651  a5 5a            lda opmask       vom akt. operator erzeugte bit-maske
3688  9653  4c cf 95         jmp lpoper       schleifenanfang
3689  9656
3690  9656
3691  9656  4c 4f 97  snerr5 jmp snerr        ausgabe '?syntax error', ready
3692  9659
3693  9659
3694  9659  ===> operanden auf stapel legen <===
3695  9659
3696  9659  a5 76     pushf1 lda facsgn       fac #1: vorzeichen
3697  965b  be d1 80         ldx optab,y      adresse / prioritätsflags der math.
                                              rout.
3698  965e  a8         pushf  tay
3699  965f  68                pla
3700  9660  85 22              sta index1       addrl indirekter index #1
3701  9662  68                pla
3702  9663  85 23              sta index1+1     addrh indirekter index #1
3703  9665  98                tya
3704  9666  48                pha
3705  9667  e6 22              inc index1       addrl indirekter index #1
3706  9669  d0 02              bne forpsh
3707  966b  e6 23              inc index1+1     addrh indirekter index #1
3708  966d  20 f2 a1  forpsh jsr round         fac #1 rundung
3709  9670  a5 75              lda faclo        fac #1: mantisse
3710  9672  48                pha
3711  9673  a5 74              lda facmo        fac #1: mantisse
3712  9675  48                pha
3713  9676  a5 73              lda facmoh       fac #1: mantisse
3714  9678  48                pha
3715  9679  a5 72              lda facho        fac #1: mantisse
3716  967b  48                pha
3717  967c  a5 71              lda facexp       fac #1: exponent
3718  967e  48                pha
3719  967f  6c 22 00           jmp (index1)     addrl indirekter index #1
3720  9682
3721  9682
3722  9682  a0 ff      qop     ldy #$ff
3723  9684  68                pla
3724  9685  f0 24      qopgo   beq qoprts
3725  9687  c9 64      qchnum  cmp #$64
3726  9689  f0 03              beq unpstk
3727  968b  20 04 b5           jsr chknum       prüfen, ob numerische variable
3728  968e  84 57      unpstk sty opptr         addrl zeiger akt. operator-routine
3729  9690  68        pulstk pla
3730  9691  4a        pulst1 lsr a
3731  9692  8d 59 02         sta tansgn       flag für 'tan' vorzeichen
3732  9695  68                pla
3733  9696  85 79              sta argexp       fac-arg ungepackt: exponent
3734  9698  68                pla
```

```
zeile adr.  obj.-code   source-code

3735  9699  85 7a           sta argho       fac-arg ungepackt: mantisse
3736  969b  68              pla
3737  969c  85 7b           sta argmoh      fac-arg ungepackt: mantisse
3738  969e  68              pla
3739  969f  85 7c           sta argmo       fac-arg ungepackt: mantisse
3740  96a1  68              pla
3741  96a2  85 7d           sta arglo       fac-arg ungepackt: mantisse
3742  96a4  68              pla
3743  96a5  85 7e           sta argsgn      fac-arg ungepackt: vorzeichen
3744  96a7  45 76           eor facsgn      fac #1: vorzeichen
3745  96a9  85 7f           sta arisgn      fac-arg ungepackt:
                                            vorzeichenduplikat
3746  96ab  a5 71     qoprts lda facexp     fac #1: exponent
3747  96ad  60               rts
3748  96ae
3749  96ae
3750  96ae  6c 8a 02  eval   jmp (ieval)    addrl beliebigen ausdruck auswerten
3751  96b1
3752  96b1
3753  96b1  ===> folgenden num. ausdruck auswerten <===
3754  96b1
3755  96b1  a9 00     neval  lda #$00
3756  96b3  85 11           sta valtyp      flag für variablen typ (0=num,
                                            1=string)
3757  96b5
3758  96b5
3759  96b5  ===> folgenden beliebigen ausdruck auswerten <===
3760  96b5
3761  96b5  20 26 ba  eval0  jsr chrget      nächstes zeichen nach ac (ind.jmp)
3762  96b8  b0 16           bcs eval2       folgenden ausdruck (keine ziffer)
                                            auswerten
3763  96ba  4c d8 a2  eval1  jmp fin         jmp: umw. zahlenstring in gk-zahl
3764  96bd
3765  96bd
3766  96bd  ===> prüfen ob konstante 'pi' <===
3767  96bd
3768  96bd  c9 ff     eval22 cmp #$ff
3769  96bf  d0 16           bne qdot        prüfen ob ausdruck mit '.' beginnt
3770  96c1  a9 cb           lda #$cb
3771  96c3  a0 96           ldy #$96
3772  96c5  20 66 a1        jsr movfm       vari adr.lo=ac, hi=yr nach fac #1
3773  96c8  4c 26 ba        jmp chrget      nächstes zeichen nach ac (ind.jmp)
3774  96cb
3775  96cb
3776  96cb  ===> konstante 'pi' <===
3777  96cb
3778  96cb  82        pival  .byte $82, $49, $0f, $da, $a1
3778  96cc  49
3778  96cd  0f
3778  96ce  da
3778  96cf  a1
3779  96d0
3780  96d0
3781  96d0  ===> folgenden ausdruck (keine ziffer) auswerten <===
3782  96d0
3783  96d0  20 bf 99  eval2  jsr isletc      test 'ac'= buchstabe, ja carry=1
3784  96d3  90 e8           bcc eval22      prüfen ob konstante 'pi'
3785  96d5  b0 7d           bcs isvar       vorber. für variablensuche
```

```
zeile adr.  obj.-code   source-code


3786  96d7
3787  96d7
3788  96d7  ===> prüfen ob ausdruck mit '.' beginnt <===
3789  96d7
3790  96d7  c9 2e      qdot   cmp #$2e
3791  96d9  f0 df             beq eval1        jmp: umw. zahlenstring in gk-zahl
3792  96db  c9 ab             cmp #$ab
3793  96dd  f0 3e             beq domin        basic-routine '-'
3794  96df  c9 aa             cmp #$aa
3795  96e1  f0 d2             beq eval0        folgenden beliebigen ausdruck
                                               auswerten
3796  96e3  c9 22             cmp #$22
3797  96e5  d0 09             bne eval3        prüfen ob token 'not'
3798  96e7
3799  96e7
3800  96e7  ===> string in '"" erfassen <===
3801  96e7
3802  96e7  20 6b 90  strtxt jsr sav30         bank akt. text nach xr, wenn carry=1
                                               textzeiger increment
3803  96ea  20 1f a8          jsr strlit       sucht elemente von zeichenkette
3804  96ed  4c f3 ab          jmp st2txt
3805  96f0
3806  96f0
3807  96f0  ===> prüfen ob token 'not' <===
3808  96f0
3809  96f0  c9 a8      eval3  cmp #$a8
3810  96f2  d0 13             bne eval4        prüfen ob token 'fn'
3811  96f4  a0 18             ldy #$18
3812  96f6  d0 27             bne gonprc       'not' o. '-' bearbeiten
3813  96f8
3814  96f8
3815  96f8  ===> basic operations-routine 'not' <===
3816  96f8
3817  96f8  20 13 9b  notop  jsr ayint        umwandlung real nach integer
3818  96fb  a5 75             lda faclo        fac #1: mantisse
3819  96fd  49 ff             eor #$ff
3820  96ff  a8                tay
3821  9700  a5 74             lda facmo        fac #1: mantisse
3822  9702  49 ff             eor #$ff
3823  9704  4c 39 9d          jmp givayf       umw. integer nach real
3824  9707
3825  9707
3826  9707  ===> prüfen ob token 'fn' <===
3827  9707
3828  9707  c9 a5      eval4  cmp #$a5
3829  9709  d0 03             bne eval5        auswerten math. funktion
3830  970b  4c 71 9d          jmp fndoer       umwandlung 'fnx'
3831  970e
3832  970e
3833  970e  ===> auswerten math. funktion <===
3834  970e
3835  970e  c9 b4      eval5  cmp #$b4
3836  9710  90 12             bcc parchk       test '(' ')', sonst fehler, ready
3837  9712  c9 e7             cmp #$e7
3838  9714  b0 04             bcs eval6        sprung nach funktionsart-test
3839  9716  c9 cb             cmp #$cb
3840  9718  b0 35             bcs snerr        ausgabe '?syntax error', ready
3841  971a  4c 21 98  eval6  jmp isfun        sprung nach funktionsart-test
```

```
zeile adr.  obj.-code   source-code


3842  971d
3843  971d
3844  971d  ===> basic-routine '-' <===
3845  971d
3846  971d  a0 15     domin  ldy #$15
3847  971f
3848  971f
3849  971f  ===> 'not' o. '-' bearbeiten <===
3850  971f
3851  971f  68        gonprc pla
3852  9720  68               pla
3853  9721  4c 20 96        jmp negprc
3854  9724
3855  9724
3856  9724  ===> test '(' ')', sonst fehler, ready <===
3857  9724
3858  9724  20 2d 97  parchk jsr chkopn       test '(', sonst fehler + ready
3859  9727  20 c1 95        jsr frmevl       auswertung beliebiger ausdruck
3860  972a
3861  972a
3862  972a  ===> test')', sonst fehler + ready <===
3863  972a
3864  972a  a9 29     chkcls lda #$29
3865  972c  2c               .byte $2c
3866  972d
3867  972d
3868  972d  ===> test '(', sonst fehler + ready <===
3869  972d
3870  972d  a9 28     chkopn lda #$28
3871  972f  2c               .byte $2c
3872  9730
3873  9730
3874  9730  ===> test 'komma' sonst fehler + ready <===
3875  9730
3876  9730  a9 2c     chkcom lda #$2c
3877  9732
3878  9732
3879  9732  ===> test:folgt ascii o. token im akt. text, sonst error <===
3880  9732
3881  9732  8d 5b 02  synchr sta ldaadr       addrl modifiable adresse
3882  9735  a4 01           ldy i6509        6509 indirection register
3883  9737  8c 5c 02        sty ldaadr+1     addrh modifiable adresse
3884  973a  a4 87           ldy txtptr+2     bank zeiger auf akt. term
3885  973c  84 01           sty i6509        6509 indirection register
3886  973e  a0 00           ldy #$00
3887  9740  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
3888  9742  ac 5c 02        ldy ldaadr+1     addrh modifiable adresse
3889  9745  84 01           sty i6509        6509 indirection register
3890  9747  cd 5b 02        cmp ldaadr       addrl modifiable adresse
3891  974a  d0 03           bne snerr        ausgabe '?syntax error', ready
3892  974c  4c 26 ba        jmp chrget       nächstes zeichen nach ac (ind.jmp)
3893  974f
3894  974f
3895  974f  ===> ausgabe '?syntax error', ready <===
3896  974f
3897  974f  a2 2a     snerr  ldx #$2a
3898  9751  4c 52 85        jmp error        ind. jmp zur fehlerroutine
3899  9754
```

```
zeile adr.  obj.-code   source-code


3900  9754
3901  9754  ===> vorber. für variablensuche <===
3902  9754
3903  9754  20 2c 99  isvar  jsr ptrget       variable suchen/anlegen, deren name
                                              folgt
3904  9757  85 74            sta facmo       fac #1: mantisse
3905  9759  84 75            sty faclo       fac #1: mantisse
3906  975b  86 76            stx facsgn      fac #1: vorzeichen
3907  975d  a6 4f            ldx varnam      erstes zeichen akt. variablenname
3908  975f  a4 50            ldy varnam+1    zweites zeichen akt. variablenname
3909  9761  a5 11            lda valtyp      flag für variablen typ (0=num,
                                              1=string)
3910  9763  f0 3c            beq goo         num. variable auswerten
3911  9765  a9 00            lda #$00
3912  9767  85 80            sta facov       fac overflow-byte
3913  9769  20 0c ba         jsr tstrom
3914  976c  b0 2b            bcs isvrts
3915  976e  e0 54            cpx #$54
3916  9770  d0 07            bne isvds       test 'ds$'
3917  9772  c0 c9            cpy #$c9
3918  9774  d0 17            bne strrts      3 byt-vektor top-ram 'memtop' nach
                                              $74
3919  9776  4c 5c ac         jmp gettim      interne uhrzeit nach ti$
3920  9779
3921  9779
3922  9779  ===> test 'ds$' <===
3923  9779
3924  9779  e0 44     isvds  cpx #$44
3925  977b  d0 10            bne strrts      3 byt-vektor top-ram 'memtop' nach
                                              $74
3926  977d  c0 d3            cpy #$d3
3927  977f  d0 0c            bne strrts      3 byt-vektor top-ram 'memtop' nach
                                              $74
3928  9781  20 9a 97         jsr chkds       test/übernahme diskstatus
3929  9784  a5 17            lda dsdesc+1    addrl disc-status string
3930  9786  a4 18            ldy dsdesc+2    addrh disc-status string
3931  9788  a6 19            ldx dsdesc+3    bank disc-status string
3932  978a  4c 1f a8         jmp strlit      sucht elemente von zeichenkette
3933  978d
3934  978d
3935  978d  ===> 3 byt-vektor top-ram 'memtop' nach $74 <===
3936  978d
3937  978d  a5 3f     strrts lda memtop      addrl höchste ram-speicherstelle
3938  978f  85 74            sta facmo       fac #1: mantisse
3939  9791  a5 40            lda memtop+1    addrh höchste ram-speicherstelle
3940  9793  85 75            sta faclo       fac #1: mantisse
3941  9795  a5 41            lda memtop+2    bank höchste ram-speicherstelle
3942  9797  85 76            sta facsgn      fac #1: vorzeichen
3943  9799  60        isvrts rts
3944  979a
3945  979a
3946  979a  ===> test/übernahme diskstatus <===
3947  979a
3948  979a  a5 16     chkds  lda dsdesc      länge disc-status string
3949  979c  d0 fb            bne isvrts
3950  979e  4c 22 b9         jmp errchl      statusstring über sek.adr. 15
                                              einlesen
3951  97a1
```

```
zeile adr.  obj.-code   source-code

3952  97a1
3953  97a1  ===> num. variable auswerten <===
3954  97a1
3955  97a1  24 12     goo     bit intflg       flag für integer-variable
3956  97a3  10 14             bpl gooo         gk-variable bearbeiten
3957  97a5  a5 76             lda facsgn       fac #1: vorzeichen
3958  97a7  85 01             sta i6509        6509 indirection register
3959  97a9  a0 00             ldy #$00
3960  97ab  b1 74             lda (facmo),y    fac #1: mantisse
3961  97ad  aa                tax
3962  97ae  c8                iny
3963  97af  b1 74             lda (facmo),y    fac #1: mantisse
3964  97b1  a8                tay
3965  97b2  20 8c ba          jsr mapusr       umsch. auf bank # 1
3966  97b5  8a                txa
3967  97b6  4c 39 9d          jmp givayf       umw. integer nach real
3968  97b9
3969  97b9
3970  97b9  ===> gk-variable bearbeiten <===
3971  97b9
3972  97b9  20 0c ba  gooo    jsr tstrom
3973  97bc  90 09             bcc qstatv       status-variable 'st' bearbeiten
3974  97be  a5 74             lda facmo        fac #1: mantisse
3975  97c0  a4 75             ldy faclo        fac #1: mantisse
3976  97c2  a6 76             ldx facsgn       fac #1: vorzeichen
3977  97c4  4c 68 a1          jmp movfum
3978  97c7
3979  97c7
3980  97c7  ===> status-variable 'st' bearbeiten <===
3981  97c7
3982  97c7  e0 53     qstatv cpx #$53
3983  97c9  d0 0a             bne qerlin       fehlerzeile-variable 'el' bearbeiten
3984  97cb  c0 54             cpy #$54
3985  97cd  d0 06             bne qerlin       fehlerzeile-variable 'el' bearbeiten
3986  97cf  20 dd bb          jsr readst       system-status in ac lesen
3987  97d2  4c 13 a2          jmp float        status-byte in fac
3988  97d5
3989  97d5
3990  97d5  ===> fehlerzeile-variable 'el' bearbeiten <===
3991  97d5
3992  97d5  e0 45     qerlin cpx #$45
3993  97d7  d0 17             bne qdsav        diskstatus-variable 'ds' bearbeiten
3994  97d9  c0 52             cpy #$52
3995  97db  f0 0d             beq qnumer       fehlernummer-variable 'er'
                                               bearbeiten
3996  97dd  c0 4c             cpy #$4c
3997  97df  d0 0f             bne qdsav        diskstatus-variable 'ds' bearbeiten
3998  97e1  ad 99 02          lda errlin+1     hi-byte zeilennummer akt. fehler
3999  97e4  ac 98 02          ldy errlin       lo-byte zeilennummer akt. fehler
4000  97e7  4c 2c 9d          jmp noslft
4001  97ea
4002  97ea
4003  97ea  ===> fehlernummer-variable 'er' bearbeiten <===
4004  97ea
4005  97ea  a5 8f     qnumer lda errnum       fehlernummer für 'trap'
4006  97ec  4a               lsr a
4007  97ed  4c 13 a2          jmp float        status-byte in fac
4008  97f0
```

```
zeile adr.  obj.-code   source-code

4009  97f0
4010  97f0  ===> diskstatus-variable 'ds' bearbeiten <===
4011  97f0
4012  97f0  e0 44     qdsav  cpx #$44
4013  97f2  d0 26            bne gomovf
4014  97f4  c0 53            cpy #$53
4015  97f6  d0 22            bne gomovf
4016  97f8  20 9a 97         jsr chkds       test/übernahme diskstatus
4017  97fb  20 7d ba         jsr mapstr      umsch. auf bank # 2
4018  97fe  a0 00            ldy #$00
4019  9800  b1 17            lda (dsdesc+1),y addrl disc-status string
4020  9802  29 0f            and #$0f
4021  9804  0a               asl a
4022  9805  85 13            sta dores       flag für token o. ascii/garbage-flag
4023  9807  0a               asl a
4024  9808  0a               asl a
4025  9809  65 13            adc dores       flag für token o. ascii/garbage-flag
4026  980b  85 13            sta dores       flag für token o. ascii/garbage-flag
4027  980d  c8               iny
4028  980e  b1 17            lda (dsdesc+1),y addrl disc-status string
4029  9810  20 8c ba         jsr mapusr      umsch. auf bank # 1
4030  9813  29 0f            and #$0f
4031  9815  65 13            adc dores       flag für token o. ascii/garbage-flag
4032  9817  4c 13 a2         jmp float       status-byte in fac
4033  981a
4034  981a
4035  981a  a5 74     gomovf lda facmo       fac #1: mantisse
4036  981c  a4 75            ldy faclo       fac #1: mantisse
4037  981e  4c 66 a1         jmp movfm       vari adr.lo=ac, hi=yr nach fac #1
4038  9821
4039  9821
4040  9821  ===> funktionsart überprüfen <===
4041  9821
4042  9821  0a         isfun  asl a
4043  9822  48               pha
4044  9823  aa               tax
4045  9824  20 26 ba         jsr chrget      nächstes zeichen nach ac (ind.jmp)
4046  9827  e0 8f            cpx #$8f
4047  9829  90 2b            bcc oknorm
4048  982b  e0 ce            cpx #$ce
4049  982d  f0 76            beq errjmp      sprung basic-routine 'err$ (x)'
4050  982f  20 2d 97         jsr chkopn      test '(', sonst fehler + ready
4051  9832  20 c1 95         jsr frmevl      auswertung beliebiger ausdruck
4052  9835  20 30 97         jsr chkcom      test 'komma' sonst fehler + ready
4053  9838  20 06 b5         jsr chkstr      prüfen, ob stringvariable
4054  983b  68               pla
4055  983c  c9 d0            cmp #$d0
4056  983e  f0 62            beq insjmp      basic-routine 'instring'
4057  9840  aa               tax
4058  9841  a5 76            lda facsgn      fac #1: vorzeichen
4059  9843  48               pha
4060  9844  a5 75            lda faclo       fac #1: mantisse
4061  9846  48               pha
4062  9847  a5 74            lda facmo       fac #1: mantisse
4063  9849  48               pha
4064  984a  8a               txa
4065  984b  48               pha
4066  984c  20 d6 b4         jsr getbyt      zahl < 256 aus akt. text nach xr
```

```
zeile adr.  obj.-code   source-code

4067  984f  68              pla
4068  9850  a8              tay
4069  9851  8a              txa
4070  9852  48              pha
4071  9853  4c 5b 98        jmp fingo
4072  9856
4073  9856
4074  9856  20 24 97  oknorm jsr parchk      test '(' ')', sonst fehler, ready
4075  9859  68              pla
4076  985a  a8              tay
4077  985b  b9 3b 80  fingo  lda stmds1,y
4078  985e  85 62            sta jmper+1     addrl sprung zu functions-rout.
4079  9860  b9 3c 80        lda stmds1+1,y
4080  9863  85 63            sta jmper+2     addrh sprung zu functions-rout.
4081  9865  20 61 00         jsr jmper       $4c wert für 'jmp' (functions-rout.)
4082  9868  4c 04 b5         jmp chknum      prüfen, ob numerische variable
4083  986b
4084  986b
4085  986b  ===> basic operations-routine 'or' <===
4086  986b
4087  986b  a0 ff     orop   ldy #$ff
4088  986d  2c               .byte $2c
4089  986e
4090  986e
4091  986e  ===> basic operations-routine 'and' <===
4092  986e
4093  986e  a0 00     andop  ldy #$00
4094  9870  8c 56 02         sty eormsk      bitmaske für 'wait'
4095  9873  20 13 9b         jsr ayint       umwandlung real nach integer
4096  9876  a5 74            lda facmo       fac #1: mantisse
4097  9878  4d 56 02         eor eormsk      bitmaske für 'wait'
4098  987b  85 0c            sta charac      puffer für trennzeichen
4099  987d  a5 75            lda faclo       fac #1: mantisse
4100  987f  4d 56 02         eor eormsk      bitmaske für 'wait'
4101  9882  85 0d            sta endchr      puffer für trennzeichen
4102  9884  20 d3 a1         jsr movfa       copy arg nach fac #1, yr
4103  9887  20 13 9b         jsr ayint       umwandlung real nach integer
4104  988a  a5 75            lda faclo       fac #1: mantisse
4105  988c  4d 56 02         eor eormsk      bitmaske für 'wait'
4106  988f  25 0d            and endchr      puffer für trennzeichen
4107  9891  4d 56 02         eor eormsk      bitmaske für 'wait'
4108  9894  a8              tay
4109  9895  a5 74            lda facmo       fac #1: mantisse
4110  9897  4d 56 02         eor eormsk      bitmaske für 'wait'
4111  989a  25 0c            and charac      puffer für trennzeichen
4112  989c  4d 56 02         eor eormsk      bitmaske für 'wait'
4113  989f  4c 39 9d         jmp givayf      umw. integer nach real
4114  98a2
4115  98a2
4116  98a2  4c e9 ae  insjmp jmp incop0      basic-routine 'instring'
4117  98a5
4118  98a5
4119  98a5  4c 00 ac  errjmp jmp errd        sprung basic-routine 'err$ (x)'
4120  98a8
4121  98a8
4122  98a8  ===> basic operations-routine '<=>' <===
4123  98a8
4124  98a8  20 07 b5  dorel  jsr chkval      prüft ob richtiger variablentyp
```

```
zeile adr.  obj.-code   source-code

4125  98ab  b0 13           bcs strcmp       routine '<=>' für string
4126  98ad  a5 7e           lda argsgn       fac-arg ungepackt: vorzeichen
4127  98af  09 7f           ora #$7f
4128  98b1  25 7a           and argho        fac-arg ungepackt: mantisse
4129  98b3  85 7a           sta argho        fac-arg ungepackt: mantisse
4130  98b5  a9 79           lda #$79
4131  98b7  a0 00           ldy #$00
4132  98b9  20 32 a2        jsr fcomp        vergleich fac #1 mit vari (ac/yr)
4133  98bc  aa              tax
4134  98bd  4c 05 99        jmp qcomp        bringt wahrheitswert nach fac
4135  98c0
4136  98c0
4137  98c0  ===> routine '<=>' für string <===
4138  98c0
4139  98c0  a9 00     strcmp lda #$00
4140  98c2  85 11           sta valtyp       flag für variablen typ (0=num,
                                             1=string)
4141  98c4  c6 5a           dec opmask       vom akt. operator erzeugte bit-maske
4142  98c6  20 e9 a8        jsr frefac       stringverwaltung, freier string
4143  98c9  85 70           sta dsctmp       zeitweise descriptoren-ablage
4144  98cb  a2 02           ldx #$02
4145  98cd  b5 22     grrk   lda index1,x     addrl indirekter index #1
4146  98cf  95 71           sta facexp,x     fac #1: exponent
4147  98d1  ca              dex
4148  98d2  10 f9           bpl grrk
4149  98d4  a5 7c           lda argmo        fac-arg ungepackt: mantisse
4150  98d6  a4 7d           ldy arglo        fac-arg ungepackt: mantisse
4151  98d8  a6 7e           ldx argsgn       fac-arg ungepackt: vorzeichen
4152  98da  20 ef a8        jsr fretmp
4153  98dd  a6 22           ldx index1       addrl indirekter index #1
4154  98df  a4 23           ldy index1+1     addrh indirekter index #1
4155  98e1  86 7c           stx argmo        fac-arg ungepackt: mantisse
4156  98e3  84 7d           sty arglo        fac-arg ungepackt: mantisse
4157  98e5  a4 24           ldy index1+2     bank indirekter index #1
4158  98e7  84 7e           sty argsgn       fac-arg ungepackt: vorzeichen
4159  98e9  aa              tax
4160  98ea  38              sec
4161  98eb  e5 70           sbc dsctmp       zeitweise descriptoren-ablage
4162  98ed  f0 08           beq stasgn
4163  98ef  a9 01           lda #$01
4164  98f1  90 04           bcc stasgn
4165  98f3  a6 70           ldx dsctmp       zeitweise descriptoren-ablage
4166  98f5  a9 ff           lda #$ff
4167  98f7  85 76     stasgn sta facsgn       fac #1: vorzeichen
4168  98f9  20 69 ba        jsr mapdst       umsch. auf bank (# in $73)
4169  98fc  a0 ff           ldy #$ff
4170  98fe  e8              inx
4171  98ff  c8        nxtcmp iny
4172  9900  ca              dex
4173  9901  d0 07           bne getcmp
4174  9903  a6 76           ldx facsgn       fac #1: vorzeichen
4175  9905
4176  9905
4177  9905  ===> bringt wahrheitswert nach fac <===
4178  9905
4179  9905  30 15     qcomp  bmi docmp
4180  9907  18              clc
4181  9908  90 12           bcc docmp
```

```
zeile adr.  obj.-code   source-code

4182  990a  b1 71     getcmp lda (facexp),y   fac #1: exponent
4183  990c  8d 5b 02         sta ldaadr        addrl modifiable adresse
4184  990f  b1 7c            lda (argmo),y     fac-arg ungepackt: mantisse
4185  9911  cd 5b 02         cmp ldaadr        addrl modifiable adresse
4186  9914  f0 e9            beq nxtcmp
4187  9916  a2 ff            ldx #$ff
4188  9918  b0 02            bcs docmp
4189  991a  a2 01            ldx #$01
4190  991c  20 8c ba  docmp   jsr mapusr        umsch. auf bank # 1
4191  991f  e8                inx
4192  9920  8a                txa
4193  9921  2a                rol a
4194  9922  2d 59 02         and tansgn        flag für 'tan' vorzeichen
4195  9925  f0 02            beq goflot
4196  9927  a9 ff            lda #$ff
4197  9929  4c 13 a2  goflot  jmp float         status-byte in fac
4198  992c
4199  992c
4200  992c  ===> variable suchen/anlegen, deren name folgt <===
4201  992c
4202  992c  a2 00     ptrget  ldx #$00
4203  992e  20 29 ba          jsr chrgot        letztes zeichen erneut nach ac
                                                (indirekter sprung)
4204  9931  86 10     ptrgt1  stx dimflg        dim-flag für zeigersuche auf
                                                variable
4205  9933
4206  9933
4207  9933  ===> sucht variable oder legt sie an <===
4208  9933
4209  9933  85 4f     ptrgt2  sta varnam        erstes zeichen akt. variablenname
4210  9935  20 29 ba          jsr chrgot        letztes zeichen erneut nach ac
                                                (indirekter sprung)
4211  9938  20 bf 99          jsr isletc        test 'ac'= buchstabe, ja carry=1
4212  993b  b0 03            bcs ptrgt3
4213  993d  4c 4f 97  interr  jmp snerr         ausgabe '?syntax error', ready
4214  9940
4215  9940
4216  9940  a2 00     ptrgt3  ldx #$00
4217  9942  86 11            stx valtyp        flag für variablen typ (0=num,
                                                1=string)
4218  9944  86 12            stx intflg        flag für integer-variable
4219  9946  20 26 ba          jsr chrget        nächstes zeichen nach ac (ind.jmp)
4220  9949  90 05            bcc issec
4221  994b  20 bf 99          jsr isletc        test 'ac'= buchstabe, ja carry=1
4222  994e  90 0b            bcc nosec
4223  9950  aa        issec   tax
4224  9951  20 26 ba  eatem   jsr chrget        nächstes zeichen nach ac (ind.jmp)
4225  9954  90 fb            bcc eatem
4226  9956  20 bf 99          jsr isletc        test 'ac'= buchstabe, ja carry=1
4227  9959  b0 f6            bcs eatem
4228  995b  c9 24     nosec   cmp #$24
4229  995d  d0 06            bne notstr
4230  995f  a9 ff            lda #$ff
4231  9961  85 11            sta valtyp        flag für variablen typ (0=num,
                                                1=string)
4232  9963  d0 10            bne turnon
4233  9965  c9 25     notstr  cmp #$25
4234  9967  d0 13            bne strnam
```

```
zeile adr.  obj.-code   source-code

4235  9969  a5 14           lda subflg      flag für indizierte variable
4236  996b  d0 d0           bne interr
4237  996d  a9 80           lda #$80
4238  996f  85 12           sta intflg      flag für integer-variable
4239  9971  05 4f           ora varnam      erstes zeichen akt. variablenname
4240  9973  85 4f           sta varnam      erstes zeichen akt. variablenname
4241  9975  8a       turnon txa
4242  9976  09 80          ora #$80
4243  9978  aa             tax
4244  9979  20 26 ba        jsr chrget      nächstes zeichen nach ac (ind.jmp)
4245  997c  86 50     strnam stx varnam+1   zweites zeichen akt. variablenname
4246  997e  38             sec
4247  997f  05 14          ora subflg      flag für indizierte variable
4248  9981  e9 28          sbc #$28
4249  9983  d0 03          bne strna1      floating variable gefunden,
                                           namenzuweisung
4250  9985  4c 28 9b        jmp isary       suchen/einrichten von feldern
4251  9988
4252  9988
4253  9988  ===> floating variable gefunden, namenzuweisung <===
4254  9988
4255  9988  a0 00     strna1 ldy #$00
4256  998a  84 14          sty subflg      flag für indizierte variable
4257  998c  a5 31          lda vartab      addrl zeiger anfang einfache
                                           variable
4258  998e  a6 32          ldx vartab+1    addrh zeiger anfang einfache
                                           variable
4259  9990  86 6e     stxfnd stx lowtr+1    addrh ursprunganfang (verschieben)
                                           temp.fac#2
4260  9992  85 6d     lopfnd sta lowtr      addrl ursprunganfang (verschieben)/
                                           temp.fac#2
4261  9994  e4 36          cpx arytab+1    addrh zeiger anfang array-tabelle
4262  9996  d0 04          bne lopfn
4263  9998  c5 35          cmp arytab      addrl zeiger anfang array-tabelle
4264  999a  f0 2d          beq notfns
4265  999c  20 87 ba  lopfn  jsr mapvar     umsch. auf bank # 2
4266  999f  b1 6d          lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                           temp.fac#2
4267  99a1  c5 4f          cmp varnam      erstes zeichen akt. variablenname
4268  99a3  d0 0d          bne notit
4269  99a5  c8             iny
4270  99a6  b1 6d          lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                           temp.fac#2
4271  99a8  c5 50          cmp varnam+1    zweites zeichen akt. variablenname
4272  99aa  d0 05          bne nxtptr
4273  99ac  4c df 9a       jmp finptr      zeiger auf gefundene variable
4274  99af
4275  99af
4276  99af  f0 8c     gobadv beq interr
4277  99b1  88       nxtptr dey
4278  99b2  20 8c ba  notit  jsr mapusr     umsch. auf bank # 1
4279  99b5  18             clc
4280  99b6  a5 6d          lda lowtr       addrl ursprunganfang (verschieben)/
                                           temp.fac#2
4281  99b8  69 07          adc #$07
4282  99ba  90 d6          bcc lopfnd
4283  99bc  e8             inx
4284  99bd  d0 d1          bne stxfnd
```

```
zeile adr.  obj.-code   source-code

4285  99bf
4286  99bf
4287  99bf  ===> test 'ac'= buchstabe, ja carry=1 <===
4288  99bf
4289  99bf  c9 41     isletc cmp #$41
4290  99c1  90 05            bcc islrts
4291  99c3  e9 5b            sbc #$5b
4292  99c5  38               sec
4293  99c6  e9 a5            sbc #$a5
4294  99c8  60        islrts rts
4295  99c9
4296  99c9
4297  99c9  68        notfns pla
4298  99ca  48               pha
4299  99cb  c9 56            cmp #$56
4300  99cd  d0 0f            bne notevl
4301  99cf  ba               tsx
4302  99d0  bd 02 01         lda stack+2,x    6509 cpu-stack
4303  99d3  c9 97            cmp #$97
4304  99d5  d0 07            bne notevl
4305  99d7  a9 10     ldzr   lda #$10
4306  99d9  a0 a5            ldy #$a5
4307  99db  a2 0f            ldx #$0f
4308  99dd  60               rts
4309  99de
4310  99de
4311  99de  a5 4f     notevl lda varnam       erstes zeichen akt. variablenname
4312  99e0  a4 50            ldy varnam+1     zweites zeichen akt. variablenname
4313  99e2  c9 54            cmp #$54
4314  99e4  d0 04            bne qstavr
4315  99e6  c0 c9            cpy #$c9
4316  99e8  f0 ed            beq ldzr
4317  99ea  c9 53     qstavr cmp #$53
4318  99ec  d0 04            bne qeravr
4319  99ee  c0 54            cpy #$54
4320  99f0  f0 bd            beq gobadv
4321  99f2  c9 45     qeravr cmp #$45
4322  99f4  d0 08            bne qdsvar
4323  99f6  c0 52            cpy #$52
4324  99f8  f0 b5            beq gobadv
4325  99fa  c0 4c            cpy #$4c
4326  99fc  f0 b1            beq gobadv
4327  99fe  c9 44     qdsvar cmp #$44
4328  9a00  d0 08            bne varok
4329  9a02  c0 53            cpy #$53
4330  9a04  f0 a9            beq gobadv
4331  9a06  c0 d3            cpy #$d3
4332  9a08  f0 a5            beq gobadv
4333  9a0a  a5 35     varok  lda arytab       addrl zeiger anfang array-tabelle
4334  9a0c  a4 36            ldy arytab+1     addrh zeiger anfang array-tabelle
4335  9a0e  85 6d            sta lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4336  9a10  84 6e            sty lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
4337  9a12  a5 39            lda strend       addrl ende benutzter ram-bereich
4338  9a14  a4 3a            ldy strend+1     addrh ende benutzter ram-bereich
4339  9a16  85 67            sta hightr       addrl ursprungende (verschieben)/
                                              temp.fac#1
```

```
zeile adr.  obj.-code   source-code

4340  9a18  84 68           sty hightr+1     addrh ursprungende (verschieben)/
                                             temp.fac#1
4341  9a1a  18               clc
4342  9a1b  69 07            adc #$07
4343  9a1d  90 01            bcc noteve
4344  9a1f  c8               iny
4345  9a20  85 64     noteve sta highds       addrl zielende (verschieben)/
                                             temp.fac#1
4346  9a22  84 65            sty highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4347  9a24  20 1e 88         jsr bltuv        blockverschiebe-rout. für variable
4348  9a27  a5 64            lda highds       addrl zielende (verschieben)/
                                             temp.fac#1
4349  9a29  a4 65            ldy highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4350  9a2b  c8               iny
4351  9a2c  85 35            sta arytab       addrl zeiger anfang array-tabelle
4352  9a2e  84 36            sty arytab+1     addrh zeiger anfang array-tabelle
4353  9a30  85 64            sta highds       addrl zielende (verschieben)/
                                             temp.fac#1
4354  9a32  84 65            sty highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4355  9a34  a5 64     aryva2 lda highds       addrl zielende (verschieben)/
                                             temp.fac#1
4356  9a36  a6 65            ldx highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4357  9a38  e4 3a     aryva3 cpx strend+1     addrh ende benutzter ram-bereich
4358  9a3a  d0 07            bne aryvgo
4359  9a3c  c5 39            cmp strend       addrl ende benutzter ram-bereich
4360  9a3e  d0 03            bne aryvgo
4361  9a40  4c c0 9a         jmp arydon
4362  9a43
4363  9a43
4364  9a43  85 22     aryvgo sta index1       addrl indirekter index #1
4365  9a45  86 23            stx index1+1     addrh indirekter index #1
4366  9a47  20 87 ba         jsr mapvar       umsch. auf bank # 2
4367  9a4a  a0 00            ldy #$00
4368  9a4c  b1 22            lda (index1),y   addrl indirekter index #1
4369  9a4e  aa               tax
4370  9a4f  c8               iny
4371  9a50  b1 22            lda (index1),y   addrl indirekter index #1
4372  9a52  08               php
4373  9a53  c8               iny
4374  9a54  b1 22            lda (index1),y   addrl indirekter index #1
4375  9a56  65 64            adc highds       addrl zielende (verschieben)/
                                             temp.fac#1
4376  9a58  85 64            sta highds       addrl zielende (verschieben)/
                                             temp.fac#1
4377  9a5a  c8               iny
4378  9a5b  b1 22            lda (index1),y   addrl indirekter index #1
4379  9a5d  65 65            adc highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4380  9a5f  85 65            sta highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4381  9a61  28               plp
4382  9a62  10 d0            bpl aryva2
4383  9a64  8a               txa
4384  9a65  30 cd            bmi aryva2
```

```
zeile adr.  obj.-code   source-code

4385  9a67  c8              iny
4386  9a68  b1 22          lda (index1),y   addrl indirekter index #1
4387  9a6a  a0 00          ldy #$00
4388  9a6c  0a             asl a
4389  9a6d  69 05          adc #$05
4390  9a6f  65 22          adc index1       addrl indirekter index #1
4391  9a71  85 22          sta index1       addrl indirekter index #1
4392  9a73  90 02          bcc aryget
4393  9a75  e6 23          inc index1+1     addrh indirekter index #1
4394  9a77  a6 23    aryget ldx index1+1     addrh indirekter index #1
4395  9a79  e4 65          cpx highds+1     addrh zielende (verschieben/
                                            temp.fac#1
4396  9a7b  d0 04          bne gogo
4397  9a7d  c5 64          cmp highds       addrl zielende (verschieben)/
                                            temp.fac#1
4398  9a7f  f0 b7          beq aryva3
4399  9a81  20 87 ba gogo  jsr mapvar       umsch. auf bank # 2
4400  9a84  a0 00          ldy #$00
4401  9a86  b1 22          lda (index1),y   addrl indirekter index #1
4402  9a88  f0 29          beq dvarts
4403  9a8a  8d 5b 02       sta ldaadr       addrl modifiable adresse
4404  9a8d  c8             iny
4405  9a8e  b1 22          lda (index1),y   addrl indirekter index #1
4406  9a90  18             clc
4407  9a91  6d 5b 02       adc ldaadr       addrl modifiable adresse
4408  9a94  85 67          sta hightr       addrl ursprungende (verschieben)/
                                            temp.fac#1
4409  9a96  aa             tax
4410  9a97  c8             iny
4411  9a98  b1 22          lda (index1),y   addrl indirekter index #1
4412  9a9a  69 00          adc #$00
4413  9a9c  85 68          sta hightr+1     addrh ursprungende (verschieben)/
                                            temp.fac#1
4414  9a9e  c8             iny
4415  9a9f  b1 22          lda (index1),y   addrl indirekter index #1
4416  9aa1  85 01          sta i6509        6509 indirection register
4417  9aa3  18             clc
4418  9aa4  a0 00          ldy #$00
4419  9aa6  b1 67          lda (hightr),y   addrl ursprungende (verschieben)/
                                            temp.fac#1
4420  9aa8  69 07          adc #$07
4421  9aaa  91 67          sta (hightr),y   addrl ursprungende (verschieben)/
                                            temp.fac#1
4422  9aac  c8             iny
4423  9aad  b1 67          lda (hightr),y   addrl ursprungende (verschieben)/
                                            temp.fac#1
4424  9aaf  69 00          adc #$00
4425  9ab1  91 67          sta (hightr),y   addrl ursprungende (verschieben)/
                                            temp.fac#1
4426  9ab3  a9 04    dvarts lda #$04
4427  9ab5  18             clc
4428  9ab6  65 22          adc index1       addrl indirekter index #1
4429  9ab8  85 22          sta index1       addrl indirekter index #1
4430  9aba  90 bb          bcc aryget
4431  9abc  e6 23          inc index1+1     addrh indirekter index #1
4432  9abe  d0 b7          bne aryget
4433  9ac0  20 87 ba arydon jsr mapvar       umsch. auf bank # 2
4434  9ac3  a0 00          ldy #$00
```

```
zeile adr.  obj.-code   source-code

4435  9ac5  a5 4f           lda varnam       erstes zeichen akt. variablenname
4436  9ac7  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4437  9ac9  c8              iny
4438  9aca  a5 50           lda varnam+1     zweites zeichen akt. variablenname
4439  9acc  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4440  9ace  a9 00           lda #$00
4441  9ad0  c8              iny
4442  9ad1  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4443  9ad3  c8              iny
4444  9ad4  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4445  9ad6  c8              iny
4446  9ad7  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4447  9ad9  c8              iny
4448  9ada  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4449  9adc  c8              iny
4450  9add  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4451  9adf
4452  9adf
4453  9adf  ===> zeiger auf gefundene variable <===
4454  9adf
4455  9adf  20 8c ba  finptr jsr mapusr      umsch. auf bank # 1
4456  9ae2  a5 6d           lda lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4457  9ae4  18              clc
4458  9ae5  69 02           adc #$02
4459  9ae7  a4 6e           ldy lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
4460  9ae9  90 01           bcc finnow
4461  9aeb  c8              iny
4462  9aec  85 51     finnow sta varpnt      addrl zeiger auf variable im ram
4463  9aee  84 52           sty varpnt+1     addrh zeiger auf variable im ram
4464  9af0  a2 02           ldx #$02
4465  9af2  86 53           stx varpnt+2     bank zeiger auf variable im ram
4466  9af4  60              rts
4467  9af5
4468  9af5                  .end
4469  9af5                  .lib math2
```

```
zeile adr.  obj.-code   source-code

4471  9af5
4472  9af5  ===> setzt zeiger $64 auf 1.feldvari <===
4473  9af5
4474  9af5  a5 0e     fmaptr lda count        allgemeiner zähler
4475  9af7  0a               asl a
4476  9af8  69 05            adc #$05
4477  9afa  65 6d            adc lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4478  9afc  a4 6e            ldy lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
4479  9afe  90 01            bcc jsrgm
4480  9b00  c8               iny
4481  9b01  85 64     jsrgm  sta highds       addrl zielende (verschieben)/
                                              temp.fac#1
4482  9b03  84 65            sty highds+1     addrh zielende (verschieben/
                                              temp.fac#1
4483  9b05  60               rts
4484  9b06
4485  9b06
4486  9b06  ===> umw. real(fac) in integer <===
4487  9b06
4488  9b06  20 26 ba  intidx jsr chrget       nächstes zeichen nach ac (ind.jmp)
4489  9b09  20 c1 95         jsr frmevl       auswertung beliebiger ausdruck
4490  9b0c  20 04 b5  posint jsr chknum       prüfen, ob numerische variable
4491  9b0f  a5 76            lda facsgn       fac #1: vorzeichen
4492  9b11  30 0d            bmi nonono
4493  9b13
4494  9b13
4495  9b13  ===> umwandlung real nach integer <===
4496  9b13
4497  9b13  a5 71     ayint  lda facexp       fac #1: exponent
4498  9b15  c9 90            cmp #$90
4499  9b17  90 09            bcc qintgo
4500  9b19  a9 0d            lda #$0d
4501  9b1b  a0 a5            ldy #$a5
4502  9b1d  20 32 a2         jsr fcomp        vergleich fac #1 mit vari (ac/yr)
4503  9b20  d0 03     nonono bne fcerr8
4504  9b22  4c 80 a2  qintgo jmp qint         umw. gk-zahl in integer
4505  9b25
4506  9b25
4507  9b25  4c a7 9b  fcerr8 jmp fcerr        ausgabe '?illegal quantity error',
                                              ready
4508  9b28
4509  9b28
4510  9b28  ===> suchen/einrichten von feldern <===
4511  9b28
4512  9b28  a5 10     isary  lda dimflg       dim-flag für zeigersuche auf
                                              variable
4513  9b2a  05 12            ora intflg       flag für integer-variable
4514  9b2c  48               pha
4515  9b2d  a5 11            lda valtyp       flag für variablen typ (0=num,
                                              1=string)
4516  9b2f  48               pha
4517  9b30  a9 00            lda #$00
4518  9b32  48               pha
4519  9b33  a5 50     indlop lda varnam+1     zweites zeichen akt. variablenname
4520  9b35  48               pha
4521  9b36  a5 4f            lda varnam       erstes zeichen akt. variablenname
```

```
zeile adr.  obj.-code   source-code

4522  9b38  48              pha
4523  9b39  20 06 9b        jsr intidx      umw. real(fac) in integer
4524  9b3c  68              pla
4525  9b3d  85 4f           sta varnam      erstes zeichen akt. variablenname
4526  9b3f  68              pla
4527  9b40  85 50           sta varnam+1    zweites zeichen akt. variablenname
4528  9b42  68              pla
4529  9b43  a8              tay
4530  9b44  ba              tsx
4531  9b45  bd 02 01        lda stack+2,x   6509 cpu-stack
4532  9b48  48              pha
4533  9b49  bd 01 01        lda stack+1,x   6509 cpu-stack
4534  9b4c  48              pha
4535  9b4d  a5 74           lda facmo       fac #1: mantisse
4536  9b4f  9d 02 01        sta stack+2,x   6509 cpu-stack
4537  9b52  a5 75           lda faclo       fac #1: mantisse
4538  9b54  9d 01 01        sta stack+1,x   6509 cpu-stack
4539  9b57  c8              iny
4540  9b58  98              tya
4541  9b59  48              pha
4542  9b5a  20 29 ba        jsr chrgot      letztes zeichen erneut nach ac
                                            (indirekter sprung)
4543  9b5d  c9 2c           cmp #$2c
4544  9b5f  f0 d2           beq indlop
4545  9b61  68              pla
4546  9b62  85 0e           sta count       allgemeiner zähler
4547  9b64  20 2a 97        jsr chkcls      test')', sonst fehler + ready
4548  9b67  68              pla
4549  9b68  85 11           sta valtyp      flag für variablen typ (0=num,
                                            1=string)
4550  9b6a  68              pla
4551  9b6b  85 12           sta intflg      flag für integer-variable
4552  9b6d  29 7f           and #$7f
4553  9b6f  85 10           sta dimflg      dim-flag für zeigersuche auf
                                            variable
4554  9b71  a6 35           ldx arytab      addrl zeiger anfang array-tabelle
4555  9b73  a5 36           lda arytab+1    addrh zeiger anfang array-tabelle
4556  9b75  86 6d     lopfda stx lowtr       addrl ursprunganfang (verschieben)/
                                            temp.fac#2
4557  9b77  85 6e           sta lowtr+1     addrh ursprunganfang (verschieben)
                                            temp.fac#2
4558  9b79  c5 3a           cmp strend+1    addrh ende benutzter ram-bereich
4559  9b7b  d0 04           bne lopfdv
4560  9b7d  e4 39           cpx strend      addrl ende benutzter ram-bereich
4561  9b7f  f0 4f           beq notfdd
4562  9b81  20 82 ba  lopfdv jsr mapary      umsch. auf bank # 2
4563  9b84  a0 00           ldy #$00
4564  9b86  b1 6d           lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                            temp.fac#2
4565  9b88  c8              iny
4566  9b89  c5 4f           cmp varnam      erstes zeichen akt. variablenname
4567  9b8b  d0 06           bne nmary1
4568  9b8d  b1 6d           lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                            temp.fac#2
4569  9b8f  c5 50           cmp varnam+1    zweites zeichen akt. variablenname
4570  9b91  f0 19           beq gotary
4571  9b93  c8        nmary1 iny
4572  9b94  b1 6d           lda (lowtr),y   addrl ursprunganfang (verschieben)/
                                            temp.fac#2
```

```
zeile adr.  obj.-code   source-code

4573  9b96  18              clc
4574  9b97  65 6d           adc lowtr        addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4575  9b99  aa              tax
4576  9b9a  c8              iny
4577  9b9b  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4578  9b9d  20 8c ba        jsr mapusr       umsch. auf bank # 1
4579  9ba0  65 6e           adc lowtr+1      addrh ursprunganfang (verschieben)
                                             temp.fac#2
4580  9ba2  90 d1           bcc lopfda
4581  9ba4
4582  9ba4
4583  9ba4  ===> ausgabe '?bad subscript error', ready <===
4584  9ba4
4585  9ba4  a2 38     bserr  ldx #$38
4586  9ba6  2c               .byte $2c
4587  9ba7
4588  9ba7
4589  9ba7  ===> ausgabe '?illegal quantity error', ready <===
4590  9ba7
4591  9ba7  a2 30     fcerr  ldx #$30
4592  9ba9  4c 52 85  errgo3 jmp error        ind. jmp zur fehlerroutine
4593  9bac
4594  9bac
4595  9bac  20 8c ba  gotary jsr mapusr       umsch. auf bank # 1
4596  9baf  a2 3a            ldx #$3a
4597  9bb1  a5 10            lda dimflg       dim-flag für zeigersuche auf
                                             variable
4598  9bb3  d0 f4            bne errgo3
4599  9bb5  20 f5 9a         jsr fmaptr       setzt zeiger $64 auf 1.feldvari
4600  9bb8  20 82 ba         jsr mapary       umsch. auf bank # 2
4601  9bbb  a0 04            ldy #$04
4602  9bbd  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4603  9bbf  c5 0e            cmp count        allgemeiner zähler
4604  9bc1  d0 e1            bne bserr        ausgabe '?bad subscript error',
                                             ready
4605  9bc3  4c 5e 9c         jmp getdef       suche nach richtigem matrixelement
4606  9bc6
4607  9bc6
4608  9bc6  10 07     sav44  bpl sav45
4609  9bc8  ca               dex
4610  9bc9  a5 4f            lda varnam       erstes zeichen akt. variablenname
4611  9bcb  10 02            bpl sav45
4612  9bcd  ca               dex
4613  9bce  ca               dex
4614  9bcf  60        sav45  rts
4615  9bd0
4616  9bd0
4617  9bd0  20 f5 9a  notfdd jsr fmaptr       setzt zeiger $64 auf 1.feldvari
4618  9bd3  20 77 88         jsr reason       test speicherplatz für variable
4619  9bd6  20 82 ba         jsr mapary       umsch. auf bank # 2
4620  9bd9  a0 00            ldy #$00
4621  9bdb  84 83            sty fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
4622  9bdd  a5 4f            lda varnam       erstes zeichen akt. variablenname
4623  9bdf  91 6d            sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
```

```
zeile adr.  obj.-code   source-code

4624  9be1  c8              iny
4625  9be2  a2 05           ldx #$05
4626  9be4  a5 50           lda varnam+1     zweites zeichen akt. variablenname
4627  9be6  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4628  9be8  20 c6 9b        jsr sav44
4629  9beb  86 82           stx fbufpt       addrl zeiger fac-puffer (fout-rout.)
4630  9bed  a5 0e           lda count        allgemeiner zähler
4631  9bef  c8              iny
4632  9bf0  c8              iny
4633  9bf1  c8              iny
4634  9bf2  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4635  9bf4  a2 0b     loppta ldx #$0b
4636  9bf6  a9 00           lda #$00
4637  9bf8  24 10           bit dimflg       dim-flag für zeigersuche auf
                                              variable
4638  9bfa  50 08           bvc notdim
4639  9bfc  68              pla
4640  9bfd  18              clc
4641  9bfe  69 01           adc #$01
4642  9c00  aa              tax
4643  9c01  68              pla
4644  9c02  69 00           adc #$00
4645  9c04  c8        notdim iny
4646  9c05  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4647  9c07  c8              iny
4648  9c08  8a              txa
4649  9c09  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
4650  9c0b  20 c4 9c        jsr umult        feldgrösse berechnen
4651  9c0e  86 82           stx fbufpt       addrl zeiger fac-puffer (fout-rout.)
4652  9c10  85 83           sta fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
4653  9c12  a4 22           ldy index1       addrl indirekter index #1
4654  9c14  c6 0e           dec count        allgemeiner zähler
4655  9c16  d0 dc           bne loppta
4656  9c18  65 65           adc highds+1     addrh zielende (verschieben/
                                              temp.fac#1
4657  9c1a  b0 60           bcs omerr1       ausgabe '?out of memory error',
                                              ready
4658  9c1c  85 65           sta highds+1     addrh zielende (verschieben/
                                              temp.fac#1
4659  9c1e  a8              tay
4660  9c1f  8a              txa
4661  9c20  65 64           adc highds       addrl zielende (verschieben)/
                                              temp.fac#1
4662  9c22  90 03           bcc grease
4663  9c24  c8              iny
4664  9c25  f0 55           beq omerr1       ausgabe '?out of memory error',
                                              ready
4665  9c27  20 8c ba  grease jsr mapusr      umsch. auf bank # 1
4666  9c2a  20 77 88        jsr reason       test speicherplatz für variable
4667  9c2d  85 39           sta strend       addrl ende benutzter ram-bereich
4668  9c2f  84 3a           sty strend+1     addrh ende benutzter ram-bereich
4669  9c31  20 82 ba        jsr mapary       umsch. auf bank # 2
4670  9c34  a9 00           lda #$00
4671  9c36  e6 83           inc fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
```

```
zeile adr.  obj.-code   source-code

4672  9c38  a4 82           ldy fbufpt       addrl zeiger fac-puffer (fout-rout.)
4673  9c3a  f0 05           beq deccur
4674  9c3c  88        zerita dey
4675  9c3d  91 64           sta (highds),y   addrl zielende (verschieben)/
                                             temp.fac#1
4676  9c3f  d0 fb           bne zerita
4677  9c41  c6 65     deccur dec highds+1    addrh zielende (verschieben/
                                             temp.fac#1
4678  9c43  c6 83           dec fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
4679  9c45  d0 f5           bne zerita
4680  9c47  e6 65           inc highds+1     addrh zielende (verschieben/
                                             temp.fac#1
4681  9c49  38              sec
4682  9c4a  a5 39           lda strend       addrl ende benutzter ram-bereich
4683  9c4c  e5 6d           sbc lowtr        addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4684  9c4e  a0 02           ldy #$02
4685  9c50  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4686  9c52  a5 3a           lda strend+1     addrh ende benutzter ram-bereich
4687  9c54  c8              iny
4688  9c55  e5 6e           sbc lowtr+1      addrh ursprunganfang (verschieben)
                                             temp.fac#2
4689  9c57  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4690  9c59  a5 10           lda dimflg       dim-flag für zeigersuche auf
                                             variable
4691  9c5b  d0 64           bne dimrts
4692  9c5d  c8              iny
4693  9c5e
4694  9c5e
4695  9c5e  ===> suche nach richtigem matrixelement <===
4696  9c5e
4697  9c5e  b1 6d     getdef lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4698  9c60  85 0e           sta count        allgemeiner zähler
4699  9c62  a9 00           lda #$00
4700  9c64  85 82           sta fbufpt       addrl zeiger fac-puffer (fout-rout.)
4701  9c66  85 83     inlpnm sta fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
4702  9c68  c8              iny
4703  9c69  68              pla
4704  9c6a  aa              tax
4705  9c6b  85 74           sta facmo        fac #1: mantisse
4706  9c6d  68              pla
4707  9c6e  85 75           sta faclo        fac #1: mantisse
4708  9c70  b1 6d           lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                             temp.fac#2
4709  9c72  c8              iny
4710  9c73  c5 75           cmp faclo        fac #1: mantisse
4711  9c75  f0 08           beq inlpn0
4712  9c77  b0 0e           bcs inlpn1
4713  9c79  4c a4 9b  bserr7 jmp bserr       ausgabe '?bad subscript error',
                                             ready
4714  9c7c
4715  9c7c
4716  9c7c  4c 50 85  omerr1 jmp omerr       ausgabe '?out of memory error',
                                             ready
4717  9c7f
```

```
zeile adr.  obj.-code   source-code

4718  9c7f
4719  9c7f  b1 6d     inlpn0 lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                               temp.fac#2
4720  9c81  c5 74            cmp facmo        fac #1: mantisse
4721  9c83  f0 f4            beq bserr7
4722  9c85  90 f2            bcc bserr7
4723  9c87  a5 83     inlpn1 lda fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
4724  9c89  05 82            ora fbufpt       addrl zeiger fac-puffer (fout-rout.)
4725  9c8b  18               clc
4726  9c8c  f0 0a            beq addind
4727  9c8e  20 c4 9c         jsr umult        feldgrösse berechnen
4728  9c91  8a               txa
4729  9c92  65 74            adc facmo        fac #1: mantisse
4730  9c94  aa               tax
4731  9c95  98               tya
4732  9c96  a4 22            ldy index1       addrl indirekter index #1
4733  9c98  65 75     addind adc faclo        fac #1: mantisse
4734  9c9a  86 82            stx fbufpt       addrl zeiger fac-puffer (fout-rout.)
4735  9c9c  c6 0e            dec count        allgemeiner zähler
4736  9c9e  d0 c6            bne inlpnm
4737  9ca0  85 83            sta fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
4738  9ca2  a2 05            ldx #$05
4739  9ca4  a5 50            lda varnam+1     zweites zeichen akt. variablenname
4740  9ca6  20 c6 9b         jsr sav44
4741  9ca9  86 2a            stx resmo        zwischenergebnis multipl./division
4742  9cab  a9 00            lda #$00
4743  9cad  20 cd 9c         jsr umultd
4744  9cb0  8a               txa
4745  9cb1  65 64            adc highds       addrl zielende (verschieben)/
                                               temp.fac#1
4746  9cb3  85 51            sta varpnt       addrl zeiger auf variable im ram
4747  9cb5  98               tya
4748  9cb6  65 65            adc highds+1     addrh zielende (verschieben/
                                               temp.fac#1
4749  9cb8  85 52            sta varpnt+1     addrh zeiger auf variable im ram
4750  9cba  a8               tay
4751  9cbb  a5 51            lda varpnt       addrl zeiger auf variable im ram
4752  9cbd  a2 02            ldx #$02
4753  9cbf  86 53            stx varpnt+2     bank zeiger auf variable im ram
4754  9cc1  4c 8c ba  dimrts jmp mapusr       umsch. auf bank # 1
4755  9cc4
4756  9cc4
4757  9cc4  ===> feldgrösse berechnen <===
4758  9cc4
4759  9cc4  84 22     umult  sty index1       addrl indirekter index #1
4760  9cc6  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                               temp.fac#2
4761  9cc8  85 2a            sta resmo        zwischenergebnis multipl./division
4762  9cca  88               dey
4763  9ccb  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                               temp.fac#2
4764  9ccd  85 2b     umultd sta reslo        zwischenergebnis multipl./division
4765  9ccf  a9 10            lda #$10
4766  9cd1  85 6b            sta lowds+1      addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
4767  9cd3  a2 00            ldx #$00
4768  9cd5  a0 00            ldy #$00
4769  9cd7  8a        umultc txa
```

```
zeile adr.  obj.-code   source-code

4770  9cd8  0a              asl a
4771  9cd9  aa              tax
4772  9cda  98              tya
4773  9cdb  2a              rol a
4774  9cdc  a8              tay
4775  9cdd  b0 9d           bcs omerr1        ausgabe '?out of memory error',
                                              ready
4776  9cdf  06 82           asl fbufpt        addrl zeiger fac-puffer (fout-rout.)
4777  9ce1  26 83           rol fbufpt+1      addrh zeiger fac-puffer (fout-rout.)
4778  9ce3  90 0b           bcc umlcnt
4779  9ce5  18              clc
4780  9ce6  8a              txa
4781  9ce7  65 2a           adc resmo         zwischenergebnis multipl./division
4782  9ce9  aa              tax
4783  9cea  98              tya
4784  9ceb  65 2b           adc reslo         zwischenergebnis multipl./division
4785  9ced  a8              tay
4786  9cee  b0 8c           bcs omerr1        ausgabe '?out of memory error',
                                              ready
4787  9cf0  c6 6b     umlcnt dec lowds+1      addrh zielanfang (versch.)/ bytes
                                              vor dez.punkt
4788  9cf2  d0 e3           bne umultc
4789  9cf4  60              rts
4790  9cf5
4791  9cf5
4792  9cf5  ===> basic-routine 'fre' <===
4793  9cf5
4794  9cf5  a5 11     fre   lda valtyp        flag für variablen typ (0=num,
                                              1=string)
4795  9cf7  f0 08           beq fref10
4796  9cf9  20 e9 a8        jsr frefac        stringverwaltung, freier string
4797  9cfc  a2 02           ldx #$02
4798  9cfe  4c 19 9d        jmp frefst        garbage, freien platz berechnen
4799  9d01
4800  9d01
4801  9d01  20 d9 b4  fref10 jsr conint       zahl < 256 aus akt. text nach xr
4802  9d04  e0 01           cpx #$01
4803  9d06  d0 0d           bne fref80
4804  9d08  38              sec
4805  9d09  a5 88           lda buffpt        addrl zeiger auf eingabe-puffer
4806  9d0b  e5 2f           sbc txtend        addrl zeiger ende  basic-text
4807  9d0d  a8              tay
4808  9d0e  a5 89           lda buffpt+1      addrh zeiger auf eingabe-puffer
4809  9d10  e5 30           sbc txtend+1      addrh zeiger ende  basic-text
4810  9d12  4c 2c 9d        jmp noslft
4811  9d15
4812  9d15
4813  9d15  e0 02     fref80 cpx #$02
4814  9d17  d0 10           bne fref70
4815  9d19
4816  9d19
4817  9d19  ===> garbage, freien platz berechnen <===
4818  9d19
4819  9d19  20 b5 ad  frefst jsr garba2       garbage collect (müll entfernen)
4820  9d1c  38              sec
4821  9d1d  a5 3b           lda fretop        addrl top of string free space
4822  9d1f  e5 39           sbc strend        addrl ende benutzter ram-bereich
4823  9d21  a8              tay
```

```
zeile adr.  obj.-code   source-code

4824  9d22  a5 3c           lda fretop+1     addrh top of string free space
4825  9d24  e5 3a           sbc strend+1     addrh ende benutzter ram-bereich
4826  9d26  4c 2c 9d        jmp noslft
4827  9d29
4828  9d29
4829  9d29  a9 00     fref70 lda #$00
4830  9d2b  a8               tay
4831  9d2c  20 3f 9d  noslft jsr stoint       umwandlung integer nach real
4832  9d2f  38               sec
4833  9d30  4c 20 a2         jmp floatc       ende umw. fliesskomma
4834  9d33
4835  9d33
4836  9d33  ===> basic-routine 'pos' <===
4837  9d33
4838  9d33  38        pos    sec
4839  9d34  20 f0 ff         jsr kplot        x,y-pos. des cursor lesen/schreiben
4840  9d37  a9 00     sngflt lda #$00
4841  9d39
4842  9d39
4843  9d39  ===> umw. integer nach real <===
4844  9d39
4845  9d39  20 3f 9d  givayf jsr stoint       umwandlung integer nach real
4846  9d3c  4c 1b a2         jmp floats
4847  9d3f
4848  9d3f
4849  9d3f  ===> umwandlung integer nach real <===
4850  9d3f
4851  9d3f  a2 00     stoint ldx #$00
4852  9d41  86 11            stx valtyp       flag für variablen typ (0=num,
                                              1=string)
4853  9d43  85 72            sta facho        fac #1: mantisse
4854  9d45  84 73            sty facmoh       fac #1: mantisse
4855  9d47  a2 90            ldx #$90
4856  9d49  60        storts rts
4857  9d4a
4858  9d4a
4859  9d4a  ===> test direkt, ausgabe '?illegal direct error', ready <===
4860  9d4a
4861  9d4a  20 57 9d  errdir jsr tstdir       test direktmodus, hi-byt zeile=$ff
4862  9d4d  d0 fa            bne storts
4863  9d4f  a2 3e            ldx #$3e
4864  9d51  2c               .byte $2c
4865  9d52
4866  9d52
4867  9d52  ===> ausgabe '?undefined function error', ready <===
4868  9d52
4869  9d52  a2 4a     errguf ldx #$4a
4870  9d54  4c 52 85         jmp error        ind. jmp zur fehlerroutine
4871  9d57
4872  9d57
4873  9d57  ===> test direktmodus, hi-byt zeile=$ff <===
4874  9d57
4875  9d57  a5 43     tstdir lda curlin+1     hi-byte akt. zeilennummer
4876  9d59  c9 ff            cmp #$ff
4877  9d5b  60               rts
4878  9d5c
4879  9d5c
4880  9d5c  ===> 'fn' variable suchen / anlegen <===
```

```
zeile adr.  obj.-code   source-code

4881  9d5c
4882  9d5c  a9 a5     getfnm lda #$a5
4883  9d5e  20 32 97         jsr synchr      test:folgt ascii o. token im akt.
                                              text, sonst error
4884  9d61  09 80             ora #$80
4885  9d63  85 14             sta subflg      flag für indizierte variable
4886  9d65  20 33 99         jsr ptrgt2      sucht variable oder legt sie an
4887  9d68  85 5b             sta defpnt      addrl zeiger funkt.-def./ temp.fac#3
4888  9d6a  84 5c             sty defpnt+1    addrh zeiger funkt.-def./ temp.fac#3
4889  9d6c  86 5d             stx defpnt+2    bank zeiger funkt.-def./ temp.fac#3
4890  9d6e  4c 04 b5          jmp chknum      prüfen, ob numerische variable
4891  9d71
4892  9d71
4893  9d71  ===> umwandlung 'fnx' <===
4894  9d71
4895  9d71  20 5c 9d  fndoer jsr getfnm       'fn' variable suchen / anlegen
4896  9d74  a5 5d             lda defpnt+2    bank zeiger funkt.-def./ temp.fac#3
4897  9d76  48               pha
4898  9d77  a5 5c             lda defpnt+1    addrh zeiger funkt.-def./ temp.fac#3
4899  9d79  48               pha
4900  9d7a  a5 5b             lda defpnt      addrl zeiger funkt.-def./ temp.fac#3
4901  9d7c  48               pha
4902  9d7d  20 24 97         jsr parchk      test '(' ')', sonst fehler, ready
4903  9d80  20 04 b5         jsr chknum      prüfen, ob numerische variable
4904  9d83  68               pla
4905  9d84  85 5b             sta defpnt      addrl zeiger funkt.-def./ temp.fac#3
4906  9d86  68               pla
4907  9d87  85 5c             sta defpnt+1    addrh zeiger funkt.-def./ temp.fac#3
4908  9d89  68               pla
4909  9d8a  85 5d             sta defpnt+2    bank zeiger funkt.-def./ temp.fac#3
4910  9d8c  85 01             sta i6509       6509 indirection register
4911  9d8e  a0 02             ldy #$02
4912  9d90  b1 5b             lda (defpnt),y  addrl zeiger funkt.-def./ temp.fac#3
4913  9d92  85 51             sta varpnt      addrl zeiger auf variable im ram
4914  9d94  aa               tax
4915  9d95  c8               iny
4916  9d96  b1 5b             lda (defpnt),y  addrl zeiger funkt.-def./ temp.fac#3
4917  9d98  85 52             sta varpnt+1    addrh zeiger auf variable im ram
4918  9d9a  05 51             ora varpnt      addrl zeiger auf variable im ram
4919  9d9c  f0 b4             beq errguf      ausgabe '?undefined function error',
                                              ready
4920  9d9e  c8               iny
4921  9d9f  b1 5b             lda (defpnt),y  addrl zeiger funkt.-def./ temp.fac#3
4922  9da1  85 53             sta varpnt+2    bank zeiger auf variable im ram
4923  9da3  85 01             sta i6509       6509 indirection register
4924  9da5  b1 51     fndo50 lda (varpnt),y   addrl zeiger auf variable im ram
4925  9da7  48               pha
4926  9da8  88               dey
4927  9da9  10 fa             bpl fndo50
4928  9dab  a4 52             ldy varpnt+1    addrh zeiger auf variable im ram
4929  9dad  a5 53             lda varpnt+2    bank zeiger auf variable im ram
4930  9daf  20 a5 a1          jsr movumf      fac #1 runden,nach xr, yr
4931  9db2  a5 86             lda txtptr+1    addrh zeiger auf akt. term
4932  9db4  48               pha
4933  9db5  a5 85             lda txtptr      addrl zeiger auf akt. term
4934  9db7  48               pha
4935  9db8  a5 5d             lda defpnt+2    bank zeiger funkt.-def./ temp.fac#3
4936  9dba  85 01             sta i6509       6509 indirection register
```

```
zeile adr.  obj.-code   source-code

4937  9dbc  b1 5b           lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4938  9dbe  85 85           sta txtptr       addrl zeiger auf akt. term
4939  9dc0  c8             iny
4940  9dc1  b1 5b           lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4941  9dc3  85 86           sta txtptr+1     addrh zeiger auf akt. term
4942  9dc5  20 8c ba        jsr mapusr       umsch. auf bank # 1
4943  9dc8  a5 53           lda varpnt+2     bank zeiger auf variable im ram
4944  9dca  48             pha
4945  9dcb  a5 52           lda varpnt+1     addrh zeiger auf variable im ram
4946  9dcd  48             pha
4947  9dce  a5 51           lda varpnt       addrl zeiger auf variable im ram
4948  9dd0  48             pha
4949  9dd1  20 01 b5        jsr frmnum       numerischen ausdruck holen
4950  9dd4  68             pla
4951  9dd5  85 5b           sta defpnt       addrl zeiger funkt.-def./ temp.fac#3
4952  9dd7  68             pla
4953  9dd8  85 5c           sta defpnt+1     addrh zeiger funkt.-def./ temp.fac#3
4954  9dda  68             pla
4955  9ddb  85 5d           sta defpnt+2     bank zeiger funkt.-def./ temp.fac#3
4956  9ddd  20 29 ba        jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
4957  9de0  f0 03           beq deffi        prg./funkt.-zeiger vom stack holen
4958  9de2  4c 4f 97        jmp snerr        ausgabe '?syntax error', ready
4959  9de5
4960  9de5
4961  9de5  ===> prg./funkt.-zeiger vom stack holen <===
4962  9de5
4963  9de5  68        deffi  pla
4964  9de6  85 85           sta txtptr       addrl zeiger auf akt. term
4965  9de8  68             pla
4966  9de9  85 86           sta txtptr+1     addrh zeiger auf akt. term
4967  9deb
4968  9deb
4969  9deb  ===> funktionen-zeiger vom stack holen <===
4970  9deb
4971  9deb  a5 5d     deffin lda defpnt+2     bank zeiger funkt.-def./ temp.fac#3
4972  9ded  85 01           sta i6509        6509 indirection register
4973  9def  a0 00           ldy #$00
4974  9df1  68             pla
4975  9df2  91 5b           sta (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4976  9df4  68             pla
4977  9df5  c8             iny
4978  9df6  91 5b           sta (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4979  9df8  68             pla
4980  9df9  c8             iny
4981  9dfa  91 5b           sta (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4982  9dfc  68             pla
4983  9dfd  c8             iny
4984  9dfe  91 5b           sta (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4985  9e00  68             pla
4986  9e01  c8             iny
4987  9e02  91 5b           sta (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
4988  9e04  4c 8c ba        jmp mapusr       umsch. auf bank # 1
4989  9e07
4990  9e07
4991  9e07  ===> basic-routine 'peek' <===
4992  9e07
4993  9e07  a5 1c     peek   lda linnum+1     hi-byte akt. zeilennummer
```

```
zeile adr.  obj.-code   source-code

4994  9e09  48              pha
4995  9e0a  a5 1b           lda linnum       lo-byte akt. zeilennummer
4996  9e0c  48              pha
4997  9e0d  20 e8 b4        jsr getadr       gk-zahl in fac als adresse nach
                                             $11,$12
4998  9e10  a6 01           ldx i6509        6509 indirection register
4999  9e12  ad 57 02        lda dfbank       vorgabe für bank-nummer
5000  9e15  85 01           sta i6509        6509 indirection register
5001  9e17  a0 00           ldy #$00
5002  9e19  b1 1b           lda (linnum),y   lo-byte akt. zeilennummer
5003  9e1b  a8              tay
5004  9e1c  68              pla
5005  9e1d  85 1b           sta linnum       lo-byte akt. zeilennummer
5006  9e1f  68              pla
5007  9e20  85 1c           sta linnum+1     hi-byte akt. zeilennummer
5008  9e22  86 01           stx i6509        6509 indirection register
5009  9e24  4c 37 9d        jmp sngflt
5010  9e27
5011  9e27
5012  9e27  ===> fac #1 + 0.5 <===
5013  9e27
5014  9e27  a9 0e     faddh  lda #$0e
5015  9e29  a0 a5            ldy #$a5
5016  9e2b  d0 1d            bne fadd         konstante in arg, fac = arg+fac
5017  9e2d
5018  9e2d
5019  9e2d  ===> konstante in arg, fac = arg-fac <===
5020  9e2d
5021  9e2d  20 5c a0  fsub   jsr conupk       holt arg lo=ac, hi=yr in fac #2
5022  9e30
5023  9e30
5024  9e30  ===> basic-routine '-' (dez. subtr.) <===
5025  9e30
5026  9e30  20 7e a5  fsubt  jsr sav41        'facsgn' invertieren
5027  9e33  45 7e            eor argsgn       fac-arg ungepackt: vorzeichen
5028  9e35  85 7f            sta arisgn       fac-arg ungepackt:
                                             vorzeichenduplikat
5029  9e37  a5 71            lda facexp       fac #1: exponent
5030  9e39  4c 4d 9e         jmp faddt        basic-routine '+' (dez. add.)
5031  9e3c
5032  9e3c
5033  9e3c  a4 89     sav75  ldy buffpt+1     addrh zeiger auf eingabe-puffer
5034  9e3e  a6 88            ldx buffpt       addrl zeiger auf eingabe-puffer
5035  9e40  d0 01            bne sav76
5036  9e42  88               dey
5037  9e43  ca        sav76  dex
5038  9e44  60        zerrts rts
5039  9e45
5040  9e45
5041  9e45  ===> exp. gleichsetzen, konstante holen, fac = arg+fac <===
5042  9e45
5043  9e45  20 79 9f  fadd5  jsr shiftr       schiebt fac rechts bis exp. = 0
5044  9e48  90 3c            bcc fadd4
5045  9e4a
5046  9e4a
5047  9e4a  ===> konstante in arg, fac = arg+fac <===
5048  9e4a
5049  9e4a  20 5c a0  fadd   jsr conupk       holt arg lo=ac, hi=yr in fac #2
```

```
zeile adr.  obj.-code   source-code

5050  9e4d
5051  9e4d
5052  9e4d  ===> basic-routine '+' (dez. add.) <===
5053  9e4d
5054  9e4d  d0 03     faddt  bne faddb
5055  9e4f  4c d3 a1         jmp movfa        copy arg nach fac #1, yr
5056  9e52
5057  9e52
5058  9e52  a6 80     faddb  ldx facov        fac overflow-byte
5059  9e54  86 63            stx jmper+2      addrh sprung zu functions-rout.
5060  9e56  a2 79            ldx #$79
5061  9e58  a5 79            lda argexp       fac-arg ungepackt: exponent
5062  9e5a  a8       faddc  tay
5063  9e5b  f0 e7            beq zerrts
5064  9e5d  38               sec
5065  9e5e  e5 71            sbc facexp       fac #1: exponent
5066  9e60  f0 24            beq fadd4
5067  9e62  90 12            bcc fadda
5068  9e64  84 71            sty facexp       fac #1: exponent
5069  9e66  a4 7e            ldy argsgn       fac-arg ungepackt: vorzeichen
5070  9e68  84 76            sty facsgn       fac #1: vorzeichen
5071  9e6a  49 ff            eor #$ff
5072  9e6c  69 00            adc #$00
5073  9e6e  a0 00            ldy #$00
5074  9e70  84 63            sty jmper+2      addrh sprung zu functions-rout.
5075  9e72  a2 71            ldx #$71
5076  9e74  d0 04            bne fadd1
5077  9e76  a0 00     fadda  ldy #$00
5078  9e78  84 80            sty facov        fac overflow-byte
5079  9e7a  c9 f9     fadd1  cmp #$f9
5080  9e7c  30 c7            bmi fadd5        exp. gleichsetzen, konstante holen,
                                              fac = arg+fac
5081  9e7e  a8               tay
5082  9e7f  a5 80            lda facov        fac overflow-byte
5083  9e81  56 01            lsr i6509,x      6509 indirection register
5084  9e83  20 90 9f         jsr rolshf       schiebt fac bitweise nach rechts
5085  9e86  24 7f     fadd4  bit arisgn       fac-arg ungepackt:
                                              vorzeichenduplikat
5086  9e88  10 57            bpl fadd2        mantissen-add. bei gleichen vorz.
5087  9e8a  a0 71            ldy #$71
5088  9e8c  e0 79            cpx #$79
5089  9e8e  f0 02            beq subit
5090  9e90  a0 79            ldy #$79
5091  9e92  38       subit  sec
5092  9e93  49 ff            eor #$ff
5093  9e95  65 63            adc jmper+2      addrh sprung zu functions-rout.
5094  9e97  85 80            sta facov        fac overflow-byte
5095  9e99  b9 04 00         lda usrpok+2,y   addrh der 'usr'-routine
5096  9e9c  f5 04            sbc usrpok+2,x   addrh der 'usr'-routine
5097  9e9e  85 75            sta faclo        fac #1: mantisse
5098  9ea0  b9 03 00         lda usrpok+1,y   addrl der 'usr'-routine
5099  9ea3  f5 03            sbc usrpok+1,x   addrl der 'usr'-routine
5100  9ea5  85 74            sta facmo        fac #1: mantisse
5101  9ea7  b9 02 00         lda usrpok,y     jmp code für user-routine
5102  9eaa  f5 02            sbc usrpok,x     jmp code für user-routine
5103  9eac  85 73            sta facmoh       fac #1: mantisse
5104  9eae  b9 01 00         lda i6509,y      6509 indirection register
5105  9eb1  f5 01            sbc i6509,x      6509 indirection register
```

```
zeile adr.  obj.-code   source-code

5106  9eb3  85 72           sta facho        fac #1: mantisse
5107  9eb5
5108  9eb5
5109  9eb5  ===> macht fac linksbündig <===
5110  9eb5
5111  9eb5  b0 03     fadflt bcs normal      fac linksbündig machen
5112  9eb7  20 2a 9f         jsr negfac      mantisse von fac invertieren
5113  9eba
5114  9eba
5115  9eba  ===> fac linksbündig machen <===
5116  9eba
5117  9eba  a0 00     normal ldy #$00
5118  9ebc  98               tya
5119  9ebd  18               clc
5120  9ebe  a6 72     norm3  ldx facho        fac #1: mantisse
5121  9ec0  d0 4a            bne norm1
5122  9ec2  a6 73            ldx facmoh       fac #1: mantisse
5123  9ec4  86 72            stx facho        fac #1: mantisse
5124  9ec6  a6 74            ldx facmo        fac #1: mantisse
5125  9ec8  86 73            stx facmoh       fac #1: mantisse
5126  9eca  a6 75            ldx faclo        fac #1: mantisse
5127  9ecc  86 74            stx facmo        fac #1: mantisse
5128  9ece  a6 80            ldx facov        fac overflow-byte
5129  9ed0  86 75            stx faclo        fac #1: mantisse
5130  9ed2  84 80            sty facov        fac overflow-byte
5131  9ed4  69 08            adc #$08
5132  9ed6  c9 20            cmp #$20
5133  9ed8  d0 e4            bne norm3
5134  9eda
5135  9eda
5136  9eda  ===> fac auf 00 setzen <===
5137  9eda
5138  9eda  a9 00     zerofc lda #$00
5139  9edc  85 71     zerof1 sta facexp       fac #1: exponent
5140  9ede  85 76     zeroml sta facsgn       fac #1: vorzeichen
5141  9ee0  60               rts
5142  9ee1
5143  9ee1
5144  9ee1  ===> mantissen-add. bei gleichen vorz. <===
5145  9ee1
5146  9ee1  65 63     fadd2  adc jmper+2      addrh sprung zu functions-rout.
5147  9ee3  85 80            sta facov        fac overflow-byte
5148  9ee5  a5 75            lda faclo        fac #1: mantisse
5149  9ee7  65 7d            adc arglo        fac-arg ungepackt: mantisse
5150  9ee9  85 75            sta faclo        fac #1: mantisse
5151  9eeb  a5 74            lda facmo        fac #1: mantisse
5152  9eed  65 7c            adc argmo        fac-arg ungepackt: mantisse
5153  9eef  85 74            sta facmo        fac #1: mantisse
5154  9ef1  a5 73            lda facmoh       fac #1: mantisse
5155  9ef3  65 7b            adc argmoh       fac-arg ungepackt: mantisse
5156  9ef5  85 73            sta facmoh       fac #1: mantisse
5157  9ef7  a5 72            lda facho        fac #1: mantisse
5158  9ef9  65 7a            adc argho        fac-arg ungepackt: mantisse
5159  9efb  85 72            sta facho        fac #1: mantisse
5160  9efd  4c 19 9f         jmp squeez       wenn nötig, überlaufstelle einsch.
5161  9f00
5162  9f00
5163  9f00  69 01     norm2  adc #$01
```

```
zeile adr.  obj.-code   source-code

5164  9f02  06 80           asl facov       fac overflow-byte
5165  9f04  26 75           rol faclo       fac #1: mantisse
5166  9f06  26 74           rol facmo       fac #1: mantisse
5167  9f08  26 73           rol facmoh      fac #1: mantisse
5168  9f0a  26 72           rol facho       fac #1: mantisse
5169  9f0c  10 f2     norm1 bpl norm2
5170  9f0e  38              sec
5171  9f0f  e5 71           sbc facexp      fac #1: exponent
5172  9f11  b0 c7           bcs zerofc      fac auf 00 setzen
5173  9f13  49 ff           eor #$ff
5174  9f15  69 01           adc #$01
5175  9f17  85 71           sta facexp      fac #1: exponent
5176  9f19
5177  9f19
5178  9f19  ===> wenn nötig, überlaufstelle einsch. <===
5179  9f19
5180  9f19  90 0e     squeez bcc rndrts
5181  9f1b  e6 71     rndshf inc facexp      fac #1: exponent
5182  9f1d  f0 3f            beq overr       ausgabe '?overflow error', ready
5183  9f1f  66 72            ror facho       fac #1: mantisse
5184  9f21  66 73            ror facmoh      fac #1: mantisse
5185  9f23  66 74            ror facmo       fac #1: mantisse
5186  9f25  66 75            ror faclo       fac #1: mantisse
5187  9f27  66 80            ror facov       fac overflow-byte
5188  9f29  60        rndrts rts
5189  9f2a
5190  9f2a
5191  9f2a  ===> mantisse von fac invertieren <===
5192  9f2a
5193  9f2a  20 7e a5  negfac jsr sav41       'facsgn' invertieren
5194  9f2d
5195  9f2d
5196  9f2d  ===> mantisse von fac invertieren <===
5197  9f2d
5198  9f2d  a5 72     negfch lda facho       fac #1: mantisse
5199  9f2f  49 ff            eor #$ff
5200  9f31  85 72            sta facho       fac #1: mantisse
5201  9f33  a5 73            lda facmoh      fac #1: mantisse
5202  9f35  49 ff            eor #$ff
5203  9f37  85 73            sta facmoh      fac #1: mantisse
5204  9f39  a5 74            lda facmo       fac #1: mantisse
5205  9f3b  49 ff            eor #$ff
5206  9f3d  85 74            sta facmo       fac #1: mantisse
5207  9f3f  a5 75            lda faclo       fac #1: mantisse
5208  9f41  49 ff            eor #$ff
5209  9f43  85 75            sta faclo       fac #1: mantisse
5210  9f45  a5 80            lda facov       fac overflow-byte
5211  9f47  49 ff            eor #$ff
5212  9f49  85 80            sta facov       fac overflow-byte
5213  9f4b  e6 80            inc facov       fac overflow-byte
5214  9f4d  d0 0e            bne incfrt
5215  9f4f  e6 75     incfac inc faclo       fac #1: mantisse
5216  9f51  d0 0a            bne incfrt
5217  9f53  e6 74            inc facmo       fac #1: mantisse
5218  9f55  d0 06            bne incfrt
5219  9f57  e6 73            inc facmoh      fac #1: mantisse
5220  9f59  d0 02            bne incfrt
5221  9f5b  e6 72            inc facho       fac #1: mantisse
```

```
zeile adr.  obj.-code   source-code


5222  9f5d  60        incfrt rts
5223  9f5e
5224  9f5e
5225  9f5e  ===> ausgabe '?overflow error', ready <===
5226  9f5e
5227  9f5e  a2 32     overr  ldx #$32
5228  9f60  4c 52 85         jmp error        ind. jmp zur fehlerroutine
5229  9f63
5230  9f63
5231  9f63  ===> register rechts schieben <===
5232  9f63
5233  9f63  a2 27     mulshf ldx #$27
5234  9f65  b4 04     shftr2 ldy usrpok+2,x   addrh der 'usr'-routine
5235  9f67  84 80            sty facov        fac overflow-byte
5236  9f69  b4 03            ldy usrpok+1,x   addrl der 'usr'-routine
5237  9f6b  94 04            sty usrpok+2,x   addrh der 'usr'-routine
5238  9f6d  b4 02            ldy usrpok,x     jmp code für user-routine
5239  9f6f  94 03            sty usrpok+1,x   addrl der 'usr'-routine
5240  9f71  b4 01            ldy i6509,x      6509 indirection register
5241  9f73  94 02            sty usrpok,x     jmp code für user-routine
5242  9f75  a4 78            ldy bits         shift-zähler
5243  9f77  94 01            sty i6509,x      6509 indirection register
5244  9f79
5245  9f79
5246  9f79  ===> schiebt fac rechts bis exp. = 0 <===
5247  9f79
5248  9f79  69 08     shiftr adc #$08
5249  9f7b  30 e8            bmi shftr2
5250  9f7d  f0 e6            beq shftr2
5251  9f7f  e9 08            sbc #$08
5252  9f81  a8               tay
5253  9f82  a5 80            lda facov        fac overflow-byte
5254  9f84  b0 14            bcs shftrt
5255  9f86  16 01     shftr3 asl i6509,x      6509 indirection register
5256  9f88  90 02            bcc shftr4
5257  9f8a  f6 01            inc i6509,x      6509 indirection register
5258  9f8c  76 01     shftr4 ror i6509,x      6509 indirection register
5259  9f8e  76 01            ror i6509,x      6509 indirection register
5260  9f90
5261  9f90
5262  9f90  ===> schiebt fac bitweise nach rechts <===
5263  9f90
5264  9f90  76 02     roishf ror usrpok,x     jmp code für user-routine
5265  9f92  76 03            ror usrpok+1,x   addrl der 'usr'-routine
5266  9f94  76 04            ror usrpok+2,x   addrh der 'usr'-routine
5267  9f96  6a               ror a
5268  9f97  c8               iny
5269  9f98  d0 ec            bne shftr3
5270  9f9a  18        shftrt clc
5271  9f9b  60               rts
5272  9f9c
5273  9f9c
5274  9f9c  ===> konstante 'eins' <===
5275  9f9c
5276  9f9c  81        fone   .byte $81, $00, $00, $00, $00
5276  9f9d  00
5276  9f9e  00
5276  9f9f  00
```

```
zeile adr.  obj.-code   source-code

5276  9fa0  00
5277  9fa1
5278  9fa1
5279  9fa1  ===> 4 konstanten für funktion 'log' (polynomgrad 3) <===
5280  9fa1
5281  9fa1  03        logcn2 .byte $03
5282  9fa2  7f               .byte $7f, $5e, $56, $cb, $79 konstante 0.434255942
                                für 'log'
5282  9fa3  5e
5282  9fa4  56
5282  9fa5  cb
5282  9fa6  79
5283  9fa7  80               .byte $80, $13, $9b, $0b, $64 konstante 0.576584541
                                für 'log'
5283  9fa8  13
5283  9fa9  9b
5283  9faa  0b
5283  9fab  64
5284  9fac  80               .byte $80, $76, $38, $93, $16 konstante 0.961800759
                                für 'log'
5284  9fad  76
5284  9fae  38
5284  9faf  93
5284  9fb0  16
5285  9fb1  82               .byte $82, $38, $aa, $3b, $20 konstante 2.885390007
                                für 'log'
5285  9fb2  38
5285  9fb3  aa
5285  9fb4  3b
5285  9fb5  20
5286  9fb6
5287  9fb6
5288  9fb6  ===> 0.707106781 = 1/2 sqr 2 <===
5289  9fb6
5290  9fb6  80        sqr05  .byte $80, $35, $04, $f3, $34
5290  9fb7  35
5290  9fb8  04
5290  9fb9  f3
5290  9fba  34
5291  9fbb
5292  9fbb
5293  9fbb  ===> 1.41421356  = sqr 2 <===
5294  9fbb
5295  9fbb  81        sqr20  .byte $81, $35, $04, $f3, $34
5295  9fbc  35
5295  9fbd  04
5295  9fbe  f3
5295  9fbf  34
5296  9fc0
5297  9fc0
5298  9fc0  ===> -0.5 <===
5299  9fc0
5300  9fc0  80        neghlf .byte $80, $80, $00, $00, $00
5300  9fc1  80
5300  9fc2  00
5300  9fc3  00
5300  9fc4  00
5301  9fc5
```

```
zeile adr.  obj.-code   source-code


5302  9fc5
5303  9fc5  ===> 0.693147181 = log(2) <===
5304  9fc5
5305  9fc5  80        log2    .byte $80, $31, $72, $17, $f8
5305  9fc6  31
5305  9fc7  72
5305  9fc8  17
5305  9fc9  f8
5306  9fca
5307  9fca                    .end
5308  9fca                    .lib math3
```

```
zeile adr.  obj.-code   source-code


5310  9fca
5311  9fca  ===> basic-routine 'log' <===
5312  9fca
5313  9fca  20 02 a2  log    jsr sign         vorzeichentest fac #1
5314  9fcd  f0 02            beq logerr
5315  9fcf  10 03            bpl log1
5316  9fd1  4c a7 9b  logerr jmp fcerr        ausgabe '?illegal quantity error',
                                              ready
5317  9fd4
5318  9fd4
5319  9fd4  a5 71     log1   lda facexp       fac #1: exponent
5320  9fd6  e9 7f            sbc #$7f
5321  9fd8  48               pha
5322  9fd9  a9 80            lda #$80
5323  9fdb  85 71            sta facexp       fac #1: exponent
5324  9fdd  a9 b6            lda #$b6
5325  9fdf  a0 9f            ldy #$9f
5326  9fe1  20 4a 9e         jsr fadd         konstante in arg, fac = arg+fac
5327  9fe4  a9 bb            lda #$bb
5328  9fe6  a0 9f            ldy #$9f
5329  9fe8  20 e6 a0         jsr fdiv         holt wert in arg, fac = arg / fac
5330  9feb  a9 9c            lda #$9c
5331  9fed  a0 9f            ldy #$9f
5332  9fef  20 2d 9e         jsr fsub         konstante in arg, fac = arg-fac
5333  9ff2  a9 a1            lda #$a1
5334  9ff4  a0 9f            ldy #$9f
5335  9ff6  20 05 a6         jsr polyx        polynom-auswertung
5336  9ff9  a9 c0            lda #$c0
5337  9ffb  a0 9f            ldy #$9f
5338  9ffd  20 4a 9e         jsr fadd         konstante in arg, fac = arg+fac
5339  a000  68               pla
5340  a001  20 63 a3         jsr finlog       liest ascii (0..9) aus basic
5341  a004  a9 c5            lda #$c5
5342  a006  a0 9f            ldy #$9f
5343  a008
5344  a008
5345  a008  ===> holt wert in arg  fac = arg * fac <===
5346  a008
5347  a008  20 5c a0  fmult  jsr conupk       holt arg lo=ac, hi=yr in fac #2
5348  a00b
5349  a00b
5350  a00b  ===> basic-routine '*' (dez. multipl.) <===
5351  a00b
5352  a00b  f0 4e     fmultt beq multrt
5353  a00d  20 8e a0         jsr muldiv       korrektur von fac u. arg
5354  a010  a9 00            lda #$00
5355  a012  85 28            sta resho        zwischenergebnis multipl./division
5356  a014  85 29            sta resmoh       zwischenergebnis multipl./division
5357  a016  85 2a            sta resmo        zwischenergebnis multipl./division
5358  a018  85 2b            sta reslo        zwischenergebnis multipl./division
5359  a01a  a5 80            lda facov        fac overflow-byte
5360  a01c  20 36 a0         jsr mltply       multiplikation 1 bit
5361  a01f  a5 75            lda faclo        fac #1: mantisse
5362  a021  20 36 a0         jsr mltply       multiplikation 1 bit
5363  a024  a5 74            lda facmo        fac #1: mantisse
5364  a026  20 36 a0         jsr mltply       multiplikation 1 bit
5365  a029  a5 73            lda facmoh       fac #1: mantisse
5366  a02b  20 36 a0         jsr mltply       multiplikation 1 bit
```

```
zeile adr.  obj.-code   source-code

5367  a02e  a5 72           lda facho        fac #1: mantisse
5368  a030  20 3b a0        jsr mltpl1
5369  a033  4c 5a a1        jmp movfr        erg. nach fac, linksbündig machen
5370  a036
5371  a036
5372  a036  ===> multiplikation 1 bit <===
5373  a036
5374  a036  d0 03     mltply bne mltpl1
5375  a038  4c 63 9f        jmp mulshf       register rechts schieben
5376  a03b
5377  a03b
5378  a03b  4a        mltpl1 lsr a
5379  a03c  09 80            ora #$80
5380  a03e  a8        mltpl2 tay
5381  a03f  90 0c            bcc mltpl3
5382  a041  18               clc
5383  a042  a2 03            ldx #$03
5384  a044  b5 28     mltpl4 lda resho,x      zwischenergebnis multipl./division
5385  a046  75 7a            adc argho,x      fac-arg ungepackt: mantisse
5386  a048  95 28            sta resho,x      zwischenergebnis multipl./division
5387  a04a  ca               dex
5388  a04b  10 f7            bpl mltpl4
5389  a04d  66 28     mltpl3 ror resho        zwischenergebnis multipl./division
5390  a04f  66 29            ror resmoh       zwischenergebnis multipl./division
5391  a051  66 2a            ror resmo        zwischenergebnis multipl./division
5392  a053  66 2b            ror reslo        zwischenergebnis multipl./division
5393  a055  66 80            ror facov        fac overflow-byte
5394  a057  98               tya
5395  a058  4a               lsr a
5396  a059  d0 e3            bne mltpl2
5397  a05b  60        multrt rts
5398  a05c
5399  a05c
5400  a05c  ===> holt arg lo=ac, hi=yr in fac #2 <===
5401  a05c
5402  a05c  a2 0f     conupk ldx #$0f
5403  a05e  86 01     ucnupk stx i6509        6509 indirection register
5404  a060  85 22            sta index1       addrl indirekter index #1
5405  a062  84 23            sty index1+1     addrh indirekter index #1
5406  a064  a0 04            ldy #$04
5407  a066  b1 22            lda (index1),y   addrl indirekter index #1
5408  a068  85 7d            sta arglo        fac-arg ungepackt: mantisse
5409  a06a  88               dey
5410  a06b  b1 22            lda (index1),y   addrl indirekter index #1
5411  a06d  85 7c            sta argmo        fac-arg ungepackt: mantisse
5412  a06f  88               dey
5413  a070  b1 22            lda (index1),y   addrl indirekter index #1
5414  a072  85 7b            sta argmoh       fac-arg ungepackt: mantisse
5415  a074  88               dey
5416  a075  b1 22            lda (index1),y   addrl indirekter index #1
5417  a077  85 7e            sta argsgn       fac-arg ungepackt: vorzeichen
5418  a079  45 76            eor facsgn       fac #1: vorzeichen
5419  a07b  85 7f            sta arisgn       fac-arg ungepackt:
                                              vorzeichenduplikat
5420  a07d  a5 7e            lda argsgn       fac-arg ungepackt: vorzeichen
5421  a07f  09 80            ora #$80
5422  a081  85 7a            sta argho        fac-arg ungepackt: mantisse
5423  a083  88               dey
```

```
zeile adr.  obj.-code   source-code

5424  a084  b1 22           lda (index1),y   addrl indirekter index #1
5425  a086  20 8c ba        jsr mapusr       umsch. auf bank # 1
5426  a089  85 79           sta argexp       fac-arg ungepackt: exponent
5427  a08b  a5 71           lda facexp       fac #1: exponent
5428  a08d  60              rts
5429  a08e
5430  a08e
5431  a08e  ===> korrektur von fac u. arg <===
5432  a08e
5433  a08e  a5 79     muldiv lda argexp       fac-arg ungepackt: exponent
5434  a090  f0 1f     mldexp beq zeremv
5435  a092  18              clc
5436  a093  65 71           adc facexp       fac #1: exponent
5437  a095  90 04           bcc tryoff
5438  a097  30 1d           bmi goover
5439  a099  18              clc
5440  a09a  2c              .byte $2c
5441  a09b  10 14     tryoff bpl zeremv
5442  a09d  69 80           adc #$80
5443  a09f  85 71           sta facexp       fac #1: exponent
5444  a0a1  d0 03           bne muld
5445  a0a3  4c de 9e        jmp zeroml
5446  a0a6
5447  a0a6
5448  a0a6  a5 7f     muld   lda arisgn       fac-arg ungepackt:
                                              vorzeichenduplikat
5449  a0a8  85 76           sta facsgn       fac #1: vorzeichen
5450  a0aa  60              rts
5451  a0ab
5452  a0ab
5453  a0ab  a5 76     mldvex lda facsgn       fac #1: vorzeichen
5454  a0ad  49 ff           eor #$ff
5455  a0af  30 05           bmi goover
5456  a0b1  68        zeremv pla
5457  a0b2  68              pla
5458  a0b3  4c da 9e        jmp zerofc       fac auf 00 setzen
5459  a0b6
5460  a0b6
5461  a0b6  4c 5e 9f  goover jmp overr        ausgabe '?overflow error', ready
5462  a0b9
5463  a0b9
5464  a0b9  ===> multiplikation fac #1 = fac * 10 <===
5465  a0b9
5466  a0b9  20 e3 a1  mul10  jsr movaf        copy fac #1 nach arg
5467  a0bc  aa              tax
5468  a0bd  f0 10           beq mul10r
5469  a0bf  18              clc
5470  a0c0  69 02           adc #$02
5471  a0c2  b0 f2           bcs goover
5472  a0c4  a2 00           ldx #$00
5473  a0c6  86 7f           stx arisgn       fac-arg ungepackt:
                                              vorzeichenduplikat
5474  a0c8  20 5a 9e        jsr faddc
5475  a0cb  e6 71           inc facexp       fac #1: exponent
5476  a0cd  f0 e7           beq goover
5477  a0cf  60        mul10r rts
5478  a0d0
5479  a0d0
```

```
zeile adr.  obj.-code   source-code


5480  a0d0  ===> gleitkommakonstante 10 <===
5481  a0d0
5482  a0d0  84        tenc    .byte $84, $20, $00, $00, $00
5482  a0d1  20
5482  a0d2  00
5482  a0d3  00
5482  a0d4  00
5483  a0d5
5484  a0d5
5485  a0d5  ===> division fac #1 = fac / 10 <===
5486  a0d5
5487  a0d5  20 e3 a1  div10   jsr movaf        copy fac #1 nach arg
5488  a0d8  a9 d0             lda #$d0
5489  a0da  a0 a0             ldy #$a0
5490  a0dc  a2 00             ldx #$00
5491  a0de  86 7f     fdivf   stx arisgn       fac-arg ungepackt:
                                               vorzeichenduplikat
5492  a0e0  20 66 a1          jsr movfm        vari adr.lo=ac, hi=yr nach fac #1
5493  a0e3  4c e9 a0          jmp fdivt        basic-routine '/' (dez. division)
5494  a0e6
5495  a0e6
5496  a0e6  ===> holt wert in arg, fac = arg / fac <===
5497  a0e6
5498  a0e6  20 5c a0  fdiv    jsr conupk       holt arg lo=ac, hi=yr in fac #2
5499  a0e9
5500  a0e9
5501  a0e9  ===> basic-routine '/' (dez. division) <===
5502  a0e9
5503  a0e9  f0 5d     fdivt   beq dv0err       ausgabe '?division by zero error',
                                               ready
5504  a0eb  20 f2 a1          jsr round        fac #1 rundung
5505  a0ee  a9 00             lda #$00
5506  a0f0  38                sec
5507  a0f1  e5 71             sbc facexp       fac #1: exponent
5508  a0f3  85 71             sta facexp       fac #1: exponent
5509  a0f5  20 8e a0          jsr muldiv       korrektur von fac u. arg
5510  a0f8  e6 71             inc facexp       fac #1: exponent
5511  a0fa  f0 ba             beq goover
5512  a0fc  a2 fc             ldx #$fc
5513  a0fe  a9 01             lda #$01
5514  a100  a4 7a     divide  ldy argho        fac-arg ungepackt: mantisse
5515  a102  c4 72             cpy facho        fac #1: mantisse
5516  a104  d0 10             bne savawo
5517  a106  a4 7b             ldy argmoh       fac-arg ungepackt: mantisse
5518  a108  c4 73             cpy facmoh       fac #1: mantisse
5519  a10a  d0 0a             bne savawo
5520  a10c  a4 7c             ldy argmo        fac-arg ungepackt: mantisse
5521  a10e  c4 74             cpy facmo        fac #1: mantisse
5522  a110  d0 04             bne savawo
5523  a112  a4 7d             ldy arglo        fac-arg ungepackt: mantisse
5524  a114  c4 75             cpy faclo        fac #1: mantisse
5525  a116  08        savawo  php
5526  a117  2a                rol a
5527  a118  90 09             bcc qshft
5528  a11a  e8                inx
5529  a11b  95 2b             sta reslo,x      zwischenergebnis multipl./division
5530  a11d  f0 2e             beq ld100
5531  a11f  10 30             bpl divnrm
```

```
zeile adr.  obj.-code   source-code

5532  a121  a9 01           lda #$01
5533  a123  28       qshft  plp
5534  a124  b0 0e           bcs divsub
5535  a126  06 7d     shfarg asl arglo        fac-arg ungepackt: mantisse
5536  a128  26 7c           rol argmo        fac-arg ungepackt: mantisse
5537  a12a  26 7b           rol argmoh       fac-arg ungepackt: mantisse
5538  a12c  26 7a           rol argho        fac-arg ungepackt: mantisse
5539  a12e  b0 e6           bcs savawo
5540  a130  30 ce           bmi divide
5541  a132  10 e2           bpl savawo
5542  a134  a8       divsub tay
5543  a135  8a               txa
5544  a136  48               pha
5545  a137  a2 03           ldx #$03
5546  a139  b5 7a     divsb1 lda argho,x      fac-arg ungepackt: mantisse
5547  a13b  f5 72           sbc facho,x      fac #1: mantisse
5548  a13d  95 7a           sta argho,x      fac-arg ungepackt: mantisse
5549  a13f  ca               dex
5550  a140  10 f7           bpl divsb1
5551  a142  68               pla
5552  a143  aa               tax
5553  a144  98               tya
5554  a145  4c 26 a1         jmp shfarg
5555  a148
5556  a148
5557  a148  ===> ausgabe '?division by zero error', ready <===
5558  a148
5559  a148  a2 3c     dv0err ldx #$3c
5560  a14a  4c 52 85         jmp error        ind. jmp zur fehlerroutine
5561  a14d
5562  a14d
5563  a14d  a9 40     ld100  lda #$40
5564  a14f  d0 d2           bne qshft
5565  a151  0a       divnrm asl a
5566  a152  0a               asl a
5567  a153  0a               asl a
5568  a154  0a               asl a
5569  a155  0a               asl a
5570  a156  0a               asl a
5571  a157  85 80           sta facov        fac overflow-byte
5572  a159  28               plp
5573  a15a
5574  a15a
5575  a15a  ===> erg. nach fac, linksbündig machen <===
5576  a15a
5577  a15a  a2 03     movfr  ldx #$03
5578  a15c  b5 28     movfrz lda resho,x      zwischenergebnis multipl./division
5579  a15e  95 72           sta facho,x      fac #1: mantisse
5580  a160  ca               dex
5581  a161  10 f9           bpl movfrz
5582  a163  4c ba 9e         jmp normal       fac linksbündig machen
5583  a166
5584  a166
5585  a166  ===> vari adr.lo=ac, hi=yr nach fac #1 <===
5586  a166
5587  a166  a2 0f     movfm  ldx #$0f
5588  a168  85 22     movfum sta index1       addrl indirekter index #1
5589  a16a  84 23           sty index1+1     addrh indirekter index #1
```

```
zeile adr.  obj.-code   source-code

5590  a16c  86 01           stx i6509        6509 indirection register
5591  a16e  a0 04           ldy #$04
5592  a170  b1 22           lda (index1),y   addrl indirekter index #1
5593  a172  85 75           sta faclo        fac #1: mantisse
5594  a174  88              dey
5595  a175  b1 22           lda (index1),y   addrl indirekter index #1
5596  a177  85 74           sta facmo        fac #1: mantisse
5597  a179  88              dey
5598  a17a  b1 22           lda (index1),y   addrl indirekter index #1
5599  a17c  85 73           sta facmoh       fac #1: mantisse
5600  a17e  88              dey
5601  a17f  b1 22           lda (index1),y   addrl indirekter index #1
5602  a181  85 76           sta facsgn       fac #1: vorzeichen
5603  a183  09 80           ora #$80
5604  a185  85 72           sta facho        fac #1: mantisse
5605  a187  88              dey
5606  a188  b1 22           lda (index1),y   addrl indirekter index #1
5607  a18a  85 71           sta facexp       fac #1: exponent
5608  a18c  84 80           sty facov        fac overflow-byte
5609  a18e  4c 8c ba        jmp mapusr       umsch. auf bank # 1
5610  a191
5611  a191
5612  a191  ===> fac #2 nach $6a-$6e <===
5613  a191
5614  a191  a2 6a     mov2f  ldx #$6a
5615  a193  2c               .byte $2c
5616  a194
5617  a194
5618  a194  ===> fac #1 nach $64-$68 <===
5619  a194
5620  a194  a2 64     mov1f  ldx #$64
5621  a196  a0 00            ldy #$00
5622  a198
5623  a198
5624  a198  ===> fac #1 runden, nach xr yr <===
5625  a198
5626  a198  20 f2 a1  movmf  jsr round        fac #1 rundung
5627  a19b  a9 0f            lda #$0f
5628  a19d  10 0b            bpl mov001
5629  a19f
5630  a19f
5631  a19f  ===> gk-zahl aus fac in variable <===
5632  a19f
5633  a19f  a6 54     movvf  ldx lstpnt       addrl zeiger letzter string
5634  a1a1  a4 55            ldy lstpnt+1     addrh zeiger letzter string
5635  a1a3  a5 56            lda lstpnt+2     bank zeiger letzter string
5636  a1a5
5637  a1a5
5638  a1a5  ===> fac #1 runden,nach xr, yr <===
5639  a1a5
5640  a1a5  48        movumf pha
5641  a1a6  20 f2 a1         jsr round        fac #1 rundung
5642  a1a9  68               pla
5643  a1aa  85 01     mov001 sta i6509        6509 indirection register
5644  a1ac  84 23            sty index1+1     addrh indirekter index #1
5645  a1ae  86 22            stx index1       addrl indirekter index #1
5646  a1b0  a0 04            ldy #$04
5647  a1b2  a5 75            lda faclo        fac #1: mantisse
```

```
zeile adr.  obj.-code   source-code

5648  a1b4  91 22           sta (index1),y   addrl indirekter index #1
5649  a1b6  88              dey
5650  a1b7  a5 74           lda facmo        fac #1: mantisse
5651  a1b9  91 22           sta (index1),y   addrl indirekter index #1
5652  a1bb  88              dey
5653  a1bc  a5 73           lda facmoh       fac #1: mantisse
5654  a1be  91 22           sta (index1),y   addrl indirekter index #1
5655  a1c0  88              dey
5656  a1c1  a5 76           lda facsgn       fac #1: vorzeichen
5657  a1c3  09 7f           ora #$7f
5658  a1c5  25 72           and facho        fac #1: mantisse
5659  a1c7  91 22           sta (index1),y   addrl indirekter index #1
5660  a1c9  88              dey
5661  a1ca  a5 71           lda facexp       fac #1: exponent
5662  a1cc  91 22           sta (index1),y   addrl indirekter index #1
5663  a1ce  84 80           sty facov        fac overflow-byte
5664  a1d0  4c 8c ba        jmp mapusr       umsch. auf bank # 1
5665  a1d3
5666  a1d3
5667  a1d3  ===> copy arg nach fac #1, yr <===
5668  a1d3
5669  a1d3  a5 7e     movfa lda argsgn       fac-arg ungepackt: vorzeichen
5670  a1d5  85 76     movfa1 sta facsgn      fac #1: vorzeichen
5671  a1d7  a2 05           ldx #$05
5672  a1d9  b5 78     movfal lda bits,x      shift-zähler
5673  a1db  95 70           sta dsctmp,x     zeitweise descriptoren-ablage
5674  a1dd  ca              dex
5675  a1de  d0 f9           bne movfal
5676  a1e0  86 80           stx facov        fac overflow-byte
5677  a1e2  60              rts
5678  a1e3
5679  a1e3
5680  a1e3  ===> copy fac #1 nach arg <===
5681  a1e3
5682  a1e3  20 f2 a1  movaf  jsr round       fac #1 rundung
5683  a1e6  a2 06     movef  ldx #$06
5684  a1e8  b5 70     movafl lda dsctmp,x    zeitweise descriptoren-ablage
5685  a1ea  95 78            sta bits,x      shift-zähler
5686  a1ec  ca               dex
5687  a1ed  d0 f9            bne movafl
5688  a1ef  86 80            stx facov       fac overflow-byte
5689  a1f1  60        movrts rts
5690  a1f2
5691  a1f2
5692  a1f2  ===> fac #1 rundung <===
5693  a1f2
5694  a1f2  a5 71     round  lda facexp      fac #1: exponent
5695  a1f4  f0 fb            beq movrts
5696  a1f6  06 80            asl facov       fac overflow-byte
5697  a1f8  90 f7            bcc movrts
5698  a1fa
5699  a1fa
5700  a1fa  ===> mantisse von fac um 1 erhöhen <===
5701  a1fa
5702  a1fa  20 4f 9f  incrnd jsr incfac
5703  a1fd  d0 f2            bne movrts
5704  a1ff  4c 1b 9f         jmp rndshf
5705  a202
```

```
zeile adr.  obj.-code   source-code


5706  a202
5707  a202  ===> vorzeichentest fac #1 <===
5708  a202
5709  a202  a5 71     sign   lda facexp       fac #1: exponent
5710  a204  f0 09            beq signrt
5711  a206  a5 76     fcsign lda facsgn       fac #1: vorzeichen
5712  a208
5713  a208
5714  a208  ===> vergleichsroutine für 0 <===
5715  a208
5716  a208  2a        fcomps rol a
5717  a209  a9 ff            lda #$ff
5718  a20b  b0 02            bcs signrt
5719  a20d  a9 01            lda #$01
5720  a20f  60        signrt rts
5721  a210
5722  a210
5723  a210  ===> basic-routine 'sgn' fac=sgn(fac) <===
5724  a210
5725  a210  20 02 a2  sgn    jsr sign         vorzeichentest fac #1
5726  a213
5727  a213
5728  a213  ===> status-byte in fac <===
5729  a213
5730  a213  85 72     float  sta facho        fac #1: mantisse
5731  a215  a9 00            lda #$00
5732  a217  85 73            sta facmoh       fac #1: mantisse
5733  a219  a2 88            ldx #$88
5734  a21b  a5 72     floats lda facho        fac #1: mantisse
5735  a21d  49 ff            eor #$ff
5736  a21f  2a               rol a
5737  a220
5738  a220
5739  a220  ===> ende umw. fliesskomma <===
5740  a220
5741  a220  a9 00     floatc lda #$00
5742  a222  85 75            sta faclo        fac #1: mantisse
5743  a224  85 74            sta facmo        fac #1: mantisse
5744  a226  86 71            stx facexp       fac #1: exponent
5745  a228  85 80            sta facov        fac overflow-byte
5746  a22a  85 76            sta facsgn       fac #1: vorzeichen
5747  a22c  4c b5 9e         jmp fadflt       macht fac linksbündig
5748  a22f
5749  a22f
5750  a22f  ===> basic-routine 'abs' abs=abs(fac) <===
5751  a22f
5752  a22f  46 76     abs    lsr facsgn       fac #1: vorzeichen
5753  a231  60               rts
5754  a232
5755  a232
5756  a232  ===> vergleich fac #1 mit vari (ac/yr) <===
5757  a232
5758  a232  85 25     fcomp  sta index2       addrl indirekter index #2
5759  a234  84 26     fcompn sty index2+1     addrh indirekter index #2
5760  a236  a0 00            ldy #$00
5761  a238  20 78 a2         jsr fcinx2       lda(index2),y von bank 15
5762  a23b  c8               iny
5763  a23c  aa               tax
```

```
zeile adr.  obj.-code   source-code

5764  a23d  f0 c3           beq sign         vorzeichentest fac #1
5765  a23f  20 78 a2        jsr fcinx2       lda(index2),y von bank 15
5766  a242  45 76           eor facsgn       fac #1: vorzeichen
5767  a244  30 c0           bmi fcsign
5768  a246  e4 71           cpx facexp       fac #1: exponent
5769  a248  d0 25           bne fcompc
5770  a24a  20 78 a2        jsr fcinx2       lda(index2),y von bank 15
5771  a24d  09 80           ora #$80
5772  a24f  c5 72           cmp facho        fac #1: mantisse
5773  a251  d0 1c           bne fcompc
5774  a253  c8              iny
5775  a254  20 78 a2        jsr fcinx2       lda(index2),y von bank 15
5776  a257  c5 73           cmp facmoh       fac #1: mantisse
5777  a259  d0 14           bne fcompc
5778  a25b  c8              iny
5779  a25c  20 78 a2        jsr fcinx2       lda(index2),y von bank 15
5780  a25f  c5 74           cmp facmo        fac #1: mantisse
5781  a261  d0 0c           bne fcompc
5782  a263  c8              iny
5783  a264  a9 7f           lda #$7f
5784  a266  c5 80           cmp facov        fac overflow-byte
5785  a268  20 78 a2        jsr fcinx2       lda(index2),y von bank 15
5786  a26b  e5 75           sbc faclo        fac #1: mantisse
5787  a26d  f0 30           beq qintrt
5788  a26f  a5 76     fcompc lda facsgn      fac #1: vorzeichen
5789  a271  90 02           bcc fcompd
5790  a273  49 ff           eor #$ff
5791  a275  4c 08 a2  fcompd jmp fcomps       vergleichsroutine für 0
5792  a278
5793  a278
5794  a278  ===> lda(index2),y von bank 15 <===
5795  a278
5796  a278  20 78 ba  fcinx2 jsr mapsys       umsch. auf bank # 15
5797  a27b  b1 25           lda (index2),y   addrl indirekter index #2
5798  a27d  4c 8c ba        jmp mapusr       umsch. auf bank # 1
5799  a280
5800  a280                  .end
5801  a280                  .lib math4
```

```
zeile adr.  obj.-code   source-code

5803  a280
5804  a280  ===> umw. gk-zahl in integer <===
5805  a280
5806  a280  a5 71     qint   lda facexp      fac #1: exponent
5807  a282  f0 4a            beq clrfac     fliesskommaaccu löschen
5808  a284  38               sec
5809  a285  e9 a0            sbc #$a0
5810  a287  24 76            bit facsgn     fac #1: vorzeichen
5811  a289  10 09            bpl qishft
5812  a28b  aa               tax
5813  a28c  a9 ff            lda #$ff
5814  a28e  85 78            sta bits       shift-zähler
5815  a290  20 2d 9f         jsr negfch     mantisse von fac invertieren
5816  a293  8a               txa
5817  a294  a2 71     qishft ldx #$71
5818  a296  c9 f9            cmp #$f9
5819  a298  10 06            bpl qint1
5820  a29a  20 79 9f         jsr shiftr     schiebt fac rechts bis exp. = 0
5821  a29d  84 78            sty bits       shift-zähler
5822  a29f  60        qintrt rts
5823  a2a0
5824  a2a0
5825  a2a0  a8        qint1  tay
5826  a2a1  a5 76            lda facsgn     fac #1: vorzeichen
5827  a2a3  29 80            and #$80
5828  a2a5  46 72            lsr facho      fac #1: mantisse
5829  a2a7  05 72            ora facho      fac #1: mantisse
5830  a2a9  85 72            sta facho      fac #1: mantisse
5831  a2ab  20 90 9f         jsr rolshf     schiebt fac bitweise nach rechts
5832  a2ae  84 78            sty bits       shift-zähler
5833  a2b0  60               rts
5834  a2b1
5835  a2b1
5836  a2b1  ===> basic-routine 'int' fac=int(fac) <===
5837  a2b1
5838  a2b1  a5 71     int    lda facexp      fac #1: exponent
5839  a2b3  c9 a0            cmp #$a0
5840  a2b5  b0 20            bcs intrts
5841  a2b7  20 80 a2         jsr qint       umw. gk-zahl in integer
5842  a2ba  84 80            sty facov      fac overflow-byte
5843  a2bc  a5 76            lda facsgn     fac #1: vorzeichen
5844  a2be  84 76            sty facsgn     fac #1: vorzeichen
5845  a2c0  49 80            eor #$80
5846  a2c2  2a               rol a
5847  a2c3  a9 a0            lda #$a0
5848  a2c5  85 71            sta facexp     fac #1: exponent
5849  a2c7  a5 75            lda faclo      fac #1: mantisse
5850  a2c9  85 0c            sta charac     puffer für trennzeichen
5851  a2cb  4c b5 9e         jmp fadflt     macht fac linksbündig
5852  a2ce
5853  a2ce
5854  a2ce  ===> fliesskommaaccu löschen <===
5855  a2ce
5856  a2ce  85 72     clrfac sta facho      fac #1: mantisse
5857  a2d0  85 73            sta facmoh     fac #1: mantisse
5858  a2d2  85 74            sta facmo      fac #1: mantisse
5859  a2d4  85 75            sta faclo      fac #1: mantisse
5860  a2d6  a8               tay
```

```
zeile adr.  obj.-code   source-code

5861  a2d7  60        intrts rts
5862  a2d8
5863  a2d8
5864  a2d8  ===> umw. zahlenstring in gk-zahl <===
5865  a2d8
5866  a2d8  a0 00     fin    ldy #$00
5867  a2da  a2 0c            ldx #$0c
5868  a2dc  94 6b     finzlp sty lowds+1,x    addrh zielanfang (versch.)/ bytes
                                              vor dez.punkt
5869  a2de  ca               dex
5870  a2df  10 fb            bpl finzlp
5871  a2e1  90 0f            bcc findgq
5872  a2e3  c9 2d            cmp #$2d
5873  a2e5  d0 04            bne qplus
5874  a2e7  86 77            stx sgnflg       fac #1: vorzeichenduplikat
5875  a2e9  f0 04            beq finc
5876  a2eb  c9 2b     qplus  cmp #$2b
5877  a2ed  d0 05            bne fin1
5878  a2ef  20 26 ba  finc   jsr chrget       nächstes zeichen nach ac (ind.jmp)
5879  a2f2  90 5b     findgq bcc findig       aufruf durch mantissen-ziffer
5880  a2f4  c9 2e     fin1   cmp #$2e
5881  a2f6  f0 2e            beq findp        aufruf durch dezimalpunkt
5882  a2f8  c9 45            cmp #$45
5883  a2fa  d0 30            bne fine
5884  a2fc  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
5885  a2ff  90 17            bcc fnedg1
5886  a301  c9 ab            cmp #$ab
5887  a303  f0 0e            beq finec1
5888  a305  c9 2d            cmp #$2d
5889  a307  f0 0a            beq finec1
5890  a309  c9 aa            cmp #$aa
5891  a30b  f0 08            beq finec
5892  a30d  c9 2b            cmp #$2b
5893  a30f  f0 04            beq finec
5894  a311  d0 07            bne finec2
5895  a313  66 6f     finec1 ror lowtr+2      bank ursprunganfang (verschieben)/
                                              vorz.exp.
5896  a315  20 26 ba  finec  jsr chrget       nächstes zeichen nach ac (ind.jmp)
5897  a318  90 5c     fnedg1 bcc finedg
5898  a31a  24 6f     finec2 bit lowtr+2      bank ursprunganfang (verschieben)/
                                              vorz.exp.
5899  a31c  10 0e            bpl fine
5900  a31e  a9 00            lda #$00
5901  a320  38               sec
5902  a321  e5 6c            sbc lowds+2      bank zielanfang (versch.)/
                                              exp.basis10
5903  a323  4c 2e a3         jmp fine1
5904  a326
5905  a326
5906  a326  ===> aufruf durch dezimalpunkt <===
5907  a326
5908  a326  66 6e     findp  ror lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
5909  a328  24 6e            bit lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
5910  a32a  50 c3            bvc finc
5911  a32c  a5 6c     fine   lda lowds+2      bank zielanfang (versch.)/
                                              exp.basis10
```

```
zeile adr.  obj.-code   source-code

5912  a32e  38        fine1  sec
5913  a32f  e5 6b            sbc lowds+1      addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
5914  a331  85 6c            sta lowds+2      bank zielanfang (versch.)/
                                               exp.basis10
5915  a333  f0 12            beq finqng
5916  a335  10 09            bpl finmul
5917  a337  20 d5 a0  findiv jsr div10        division fac #1 = fac / 10
5918  a33a  e6 6c            inc lowds+2      bank zielanfang (versch.)/
                                               exp.basis10
5919  a33c  d0 f9            bne findiv
5920  a33e  f0 07            beq finqng
5921  a340  20 b9 a0  finmul jsr mul10        multiplikation fac #1 = fac * 10
5922  a343  c6 6c            dec lowds+2      bank zielanfang (versch.)/
                                               exp.basis10
5923  a345  d0 f9            bne finmul
5924  a347  a5 77      finqng lda sgnflg      fac #1: vorzeichenduplikat
5925  a349  30 01            bmi negxqs       sprung 'negation', vorzeichen neg.
5926  a34b  60                rts
5927  a34c
5928  a34c
5929  a34c  4c 7a a5  negxqs jmp negop        sprung 'negation', vorzeichen neg.
5930  a34f
5931  a34f
5932  a34f  ===> aufruf durch mantissen-ziffer <===
5933  a34f
5934  a34f  48        findig pha
5935  a350  24 6e            bit lowtr+1      addrh ursprunganfang (verschieben)
                                               temp.fac#2
5936  a352  10 02            bpl findg1
5937  a354  e6 6b            inc lowds+1      addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
5938  a356  20 b9 a0  findg1 jsr mul10        multiplikation fac #1 = fac * 10
5939  a359  68                pla
5940  a35a  38                sec
5941  a35b  e9 30             sbc #$30
5942  a35d  20 63 a3          jsr finlog       liest ascii (0..9) aus basic
5943  a360  4c ef a2          jmp finc
5944  a363
5945  a363
5946  a363  ===> liest ascii (0..9) aus basic <===
5947  a363
5948  a363  48        finlog pha
5949  a364  20 e3 a1          jsr movaf        copy fac #1 nach arg
5950  a367  68                pla
5951  a368  20 13 a2          jsr float        status-byte in fac
5952  a36b  a5 7e             lda argsgn       fac-arg ungepackt: vorzeichen
5953  a36d  45 76             eor facsgn       fac #1: vorzeichen
5954  a36f  85 7f             sta arisgn       fac-arg ungepackt:
                                               vorzeichenduplikat
5955  a371  a6 71             ldx facexp       fac #1: exponent
5956  a373  4c 4d 9e          jmp faddt        basic-routine '+' (dez. add.)
5957  a376
5958  a376
5959  a376  a5 6c     finedg lda lowds+2       bank zielanfang (versch.)/
                                               exp.basis10
5960  a378  c9 0a            cmp #$0a
5961  a37a  90 09            bcc mlex10
```

```
zeile adr.  obj.-code   source-code

5962  a37c  a9 64           lda #$64
5963  a37e  24 6f           bit lowtr+2      bank ursprunganfang (verschieben)/
                                             vorz.exp.
5964  a380  30 1e           bmi mlexmi
5965  a382  4c 5e 9f        jmp overr        ausgabe '?overflow error', ready
5966  a385
5967  a385
5968  a385  0a       mlex10 asl a
5969  a386  0a              asl a
5970  a387  18              clc
5971  a388  65 6c           adc lowds+2      bank zielanfang (versch.)/
                                             exp.basis10
5972  a38a  0a              asl a
5973  a38b  18              clc
5974  a38c  a0 00           ldy #$00
5975  a38e  8d 5b 02        sta ldaadr       addrl modifiable adresse
5976  a391  a5 87           lda txtptr+2     bank zeiger auf akt. term
5977  a393  85 01           sta i6509        6509 indirection register
5978  a395  b1 85           lda (txtptr),y   addrl zeiger auf akt. term
5979  a397  20 8c ba        jsr mapusr       umsch. auf bank # 1
5980  a39a  6d 5b 02        adc ldaadr       addrl modifiable adresse
5981  a39d  38              sec
5982  a39e  e9 30           sbc #$30
5983  a3a0  85 6c    mlexmi sta lowds+2      bank zielanfang (versch.)/
                                             exp.basis10
5984  a3a2  4c 15 a3        jmp finec
5985  a3a5
5986  a3a5
5987  a3a5  ===> gleitkommakonst. 999999999.9 <===
5988  a3a5
5989  a3a5  9b       n0999   .byte $9b, $3e, $bc, $1f, $fd
5989  a3a6  3e
5989  a3a7  bc
5989  a3a8  1f
5989  a3a9  fd
5990  a3aa
5991  a3aa
5992  a3aa  ===> gleitkommakonst. 9999999999 <===
5993  a3aa
5994  a3aa  9e       n9999   .byte $9e, $6e, $6b, $27, $fd
5994  a3ab  6e
5994  a3ac  6b
5994  a3ad  27
5994  a3ae  fd
5995  a3af
5996  a3af
5997  a3af  ===> gleitkommakonst. 1e9 <===
5998  a3af
5999  a3af  9e       nmil    .byte $9e, $6e, $6b, $28, $00
5999  a3b0  6e
5999  a3b1  6b
5999  a3b2  28
5999  a3b3  00
6000  a3b4
6001  a3b4
6002  a3b4  ===> umw. hex/dez, lo=xr, hi=ac, druck zeilen # <===
6003  a3b4
6004  a3b4  85 72    linprt sta facho         fac #1: mantisse
```

```
zeile adr.  obj.-code   source-code

6005  a3b6  86 73           stx facmoh       fac #1: mantisse
6006  a3b8  a2 90           ldx #$90
6007  a3ba  38              sec
6008  a3bb  20 20 a2        jsr floatc       ende umw. fliesskomma
6009  a3be  20 e2 a3  outfac jsr fout        umw. gk-zahl in string (ablage ab
                                             $200)
6010  a3c1  a2 22           ldx #$22
6011  a3c3
6012  a3c3
6013  a3c3  ===> textzeiger + xr, textausgabe <===
6014  a3c3
6015  a3c3  bd 8f 82  msg    lda ebase,x      ind. sprungtabelle basic-/system
                                             fehlermeldungen
6016  a3c6  bc 90 82         ldy ebase+1,x    ind. sprungtabelle basic-/system
                                             fehlermeldungen
6017  a3c9  8d 5b 02         sta ldaadr       addrl modifiable adresse
6018  a3cc  8c 5c 02         sty ldaadr+1     addrh modifiable adresse
6019  a3cf
6020  a3cf
6021  a3cf  ===> textausgabe (zeiger $25b) <===
6022  a3cf
6023  a3cf  20 5a 02  msgprt jsr ldaabs       lda abs routine
6024  a3d2  f0 0d            beq msgrts
6025  a3d4  20 3a b5         jsr ochr         ausgabe 1 zeichen (in ac)
6026  a3d7  ee 5b 02         inc ldaadr       addrl modifiable adresse
6027  a3da  d0 f3            bne msgprt       textausgabe (zeiger $25b)
6028  a3dc  ee 5c 02         inc ldaadr+1     addrh modifiable adresse
6029  a3df  d0 ee            bne msgprt       textausgabe (zeiger $25b)
6030  a3e1  60        msgrts rts
6031  a3e2
6032  a3e2
6033  a3e2  ===> umw. gk-zahl in string (ablage ab $200) <===
6034  a3e2
6035  a3e2  a0 01     fout   ldy #$01
6036  a3e4  a9 20            lda #$20
6037  a3e6  24 76            bit facsgn       fac #1: vorzeichen
6038  a3e8  10 02            bpl fout1
6039  a3ea  a9 2d            lda #$2d
6040  a3ec  99 ff 01  fout1  sta buf-1,y      basic input-puffer (-$2ff) -1
6041  a3ef  85 76            sta facsgn       fac #1: vorzeichen
6042  a3f1  84 82            sty fbufpt       addrl zeiger fac-puffer (fout-rout.)
6043  a3f3  c8               iny
6044  a3f4  a9 30            lda #$30
6045  a3f6  a6 71            ldx facexp       fac #1: exponent
6046  a3f8  d0 03            bne fout10
6047  a3fa  4c 04 a5         jmp fout19
6048  a3fd
6049  a3fd
6050  a3fd  a9 00     fout10 lda #$00
6051  a3ff  e0 80            cpx #$80
6052  a401  f0 02            beq fout37
6053  a403  b0 09            bcs fout7
6054  a405  a9 af     fout37 lda #$af
6055  a407  a0 a3            ldy #$a3
6056  a409  20 08 a0         jsr fmult        holt wert in arg  fac = arg * fac
6057  a40c  a9 f7            lda #$f7
6058  a40e  85 6b     fout7  sta lowds+1       addrh zielanfang (versch.)/ bytes
                                             vor dez.punkt
```

```
zeile adr.  obj.-code   source-code

6059  a410  a9 aa      fout4  lda #$aa
6060  a412  a0 a3             ldy #$a3
6061  a414  20 32 a2          jsr fcomp       vergleich fac #1 mit vari (ac/yr)
6062  a417  f0 1e             beq bigges
6063  a419  10 12             bpl fout9
6064  a41b  a9 a5      fout3  lda #$a5
6065  a41d  a0 a3             ldy #$a3
6066  a41f  20 32 a2          jsr fcomp       vergleich fac #1 mit vari (ac/yr)
6067  a422  f0 02             beq fout38
6068  a424  10 0e             bpl fout5
6069  a426  20 b9 a0  fout38  jsr mul10       multiplikation fac #1 = fac * 10
6070  a429  c6 6b             dec lowds+1     addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
6071  a42b  d0 ee             bne fout3
6072  a42d  20 d5 a0  fout9   jsr div10       division fac #1 = fac / 10
6073  a430  e6 6b             inc lowds+1     addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
6074  a432  d0 dc             bne fout4
6075  a434  20 27 9e  fout5   jsr faddh       fac #1 + 0.5
6076  a437  20 80 a2  bigges  jsr qint        umw. gk-zahl in integer
6077  a43a  a2 01             ldx #$01
6078  a43c  a5 6b             lda lowds+1     addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
6079  a43e  18                clc
6080  a43f  69 0a             adc #$0a
6081  a441  30 09             bmi foutpi
6082  a443  c9 0b             cmp #$0b
6083  a445  b0 06             bcs fout6
6084  a447  69 ff             adc #$ff
6085  a449  aa                tax
6086  a44a  a9 02             lda #$02
6087  a44c  38        foutpi  sec
6088  a44d  e9 02     fout6   sbc #$02
6089  a44f  85 6c             sta lowds+2     bank zielanfang (versch.)/
                                               exp.basis10
6090  a451  86 6b             stx lowds+1     addrh zielanfang (versch.)/ bytes
                                               vor dez.punkt
6091  a453  8a                txa
6092  a454  f0 02             beq fout39
6093  a456  10 13             bpl fout8       einz. ziffern des zahlenstrings
                                               nacheinander berechnen
6094  a458  a4 82     fout39  ldy fbufpt       addrl zeiger fac-puffer (fout-rout.)
6095  a45a  a9 2e             lda #$2e
6096  a45c  c8                iny
6097  a45d  99 ff 01          sta buf-1,y     basic input-puffer (-$2ff) -1
6098  a460  8a                txa
6099  a461  f0 06             beq fout16
6100  a463  a9 30             lda #$30
6101  a465  c8                iny
6102  a466  99 ff 01          sta buf-1,y     basic input-puffer (-$2ff) -1
6103  a469  84 82     fout16  sty fbufpt       addrl zeiger fac-puffer (fout-rout.)
6104  a46b
6105  a46b
6106  a46b  ===> einz. ziffern des zahlenstrings nacheinander berechnen <===
6107  a46b
6108  a46b  a0 00     fout8   ldy #$00
6109  a46d  a2 80             ldx #$80
6110  a46f  a5 75     fout2   lda faclo        fac #1: mantisse
```

```
zeile adr.  obj.-code   source-code

6111  a471  18              clc
6112  a472  79 16 a5        adc foutbl+3,y
6113  a475  85 75           sta faclo        fac #1: mantisse
6114  a477  a5 74           lda facmo        fac #1: mantisse
6115  a479  79 15 a5        adc foutbl+2,y
6116  a47c  85 74           sta facmo        fac #1: mantisse
6117  a47e  a5 73           lda facmoh       fac #1: mantisse
6118  a480  79 14 a5        adc foutbl+1,y
6119  a483  85 73           sta facmoh       fac #1: mantisse
6120  a485  a5 72           lda facho        fac #1: mantisse
6121  a487  79 13 a5        adc foutbl,y     konst. für gleitkomma nach ascii:
                                             '-100 000 000'
6122  a48a  85 72           sta facho        fac #1: mantisse
6123  a48c  e8              inx
6124  a48d  b0 03           bcs fout41
6125  a48f  10 de           bpl fout2
6126  a491  2c              .byte $2c
6127  a492  30 db     fout41 bmi fout2
6128  a494  8a              txa
6129  a495  90 04           bcc foutyp
6130  a497  49 ff           eor #$ff
6131  a499  69 0a           adc #$0a
6132  a49b  69 2f     foutyp adc #$2f
6133  a49d  c8              iny
6134  a49e  c8              iny
6135  a49f  c8              iny
6136  a4a0  c8              iny
6137  a4a1  84 51           sty varpnt       addrl zeiger auf variable im ram
6138  a4a3  a4 82           ldy fbufpt       addrl zeiger fac-puffer (fout-rout.)
6139  a4a5  c8              iny
6140  a4a6  aa              tax
6141  a4a7  29 7f           and #$7f
6142  a4a9  99 ff 01        sta buf-1,y      basic input-puffer (-$2ff) -1
6143  a4ac  c6 6b           dec lowds+1      addrh zielanfang (versch.)/ bytes
                                             vor dez.punkt
6144  a4ae  d0 06           bne stxbuf
6145  a4b0  a9 2e           lda #$2e
6146  a4b2  c8              iny
6147  a4b3  99 ff 01        sta buf-1,y      basic input-puffer (-$2ff) -1
6148  a4b6  84 82     stxbuf sty fbufpt       addrl zeiger fac-puffer (fout-rout.)
6149  a4b8  a4 51           ldy varpnt       addrl zeiger auf variable im ram
6150  a4ba  8a              txa
6151  a4bb  49 ff           eor #$ff
6152  a4bd  29 80           and #$80
6153  a4bf  aa              tax
6154  a4c0  c0 24           cpy #$24
6155  a4c2  d0 ab           bne fout2
6156  a4c4  a4 82           ldy fbufpt       addrl zeiger fac-puffer (fout-rout.)
6157  a4c6  b9 ff 01  fout11 lda buf-1,y      basic input-puffer (-$2ff) -1
6158  a4c9  88              dey
6159  a4ca  c9 30           cmp #$30
6160  a4cc  f0 f8           beq fout11
6161  a4ce  c9 2e           cmp #$2e
6162  a4d0  f0 01           beq fout12
6163  a4d2  c8              iny
6164  a4d3  a9 2b     fout12 lda #$2b
6165  a4d5  a6 6c           ldx lowds+2      bank zielanfang (versch.)/
                                             exp.basis10
```

```
zeile adr.  obj.-code   source-code

6166  a4d7  f0 2e            beq fout17
6167  a4d9  10 08            bpl fout14
6168  a4db  a9 00            lda #$00
6169  a4dd  38               sec
6170  a4de  e5 6c            sbc lowds+2      bank zielanfang (versch.)/
                                               exp.basis10
6171  a4e0  aa               tax
6172  a4e1  a9 2d            lda #$2d
6173  a4e3  99 01 02  fout14 sta buf+1,y      basic input-puffer (-$2ff) +1
6174  a4e6  a9 45            lda #$45
6175  a4e8  99 00 02         sta buf,y        basic input-puffer (-$2ff)
6176  a4eb  8a               txa
6177  a4ec  a2 2f            ldx #$2f
6178  a4ee  38               sec
6179  a4ef  e8        fout15 inx
6180  a4f0  e9 0a            sbc #$0a
6181  a4f2  b0 fb            bcs fout15
6182  a4f4  69 3a            adc #$3a
6183  a4f6  99 03 02         sta buf+3,y      basic input-puffer (-$2ff) +3
6184  a4f9  8a               txa
6185  a4fa  99 02 02         sta buf+2,y      basic input-puffer (-$2ff) +2
6186  a4fd  a9 00            lda #$00
6187  a4ff  99 04 02         sta buf+4,y      basic input-puffer (-$2ff) +4
6188  a502  f0 08            beq fout20
6189  a504  99 ff 01  fout19 sta buf-1,y      basic input-puffer (-$2ff) -1
6190  a507  a9 00     fout17 lda #$00
6191  a509  99 00 02         sta buf,y        basic input-puffer (-$2ff)
6192  a50c  60        fout20 rts
6193  a50d
6194  a50d
6195  a50d  ===> fliesskomma-konstante 32768 (32 k) <===
6196  a50d
6197  a50d  90        n32768 .byte $90
6198  a50e
6199  a50e
6200  a50e  ===> konstante 0.5 für funktion 'sqr' <===
6201  a50e
6202  a50e  80        fhalf  .byte $80, $00, $00, $00, $00
6202  a50f  00
6202  a510  00
6202  a511  00
6202  a512  00
6203  a513
6204  a513
6205  a513  ===> konst. für gleitkomma nach ascii: '-100 000 000' <===
6206  a513
6207  a513  fa        foutbl .byte $fa
6208  a514  0a               .byte $0a
6209  a515  1f               .byte $1f
6210  a516  00               .byte $00
6211  a517  00               .byte $00, $98, $96, $80 konst. für gleitkomma nach
                                                      ascii: '10 000 000'
6211  a518  98
6211  a519  96
6211  a51a  80
6212  a51b  ff               .byte $ff, $f0, $bd, $c0 konst. für gleitkomma nach
                                                      ascii: '-1 000 000'
6212  a51c  f0
```

```
zeile adr.  obj.-code   source-code

6212  a51d  bd
6212  a51e  c0
6213  a51f  00              .byte $00, $01, $86, $a0 konst. für gleitkomma nach
                                    ascii: '100 000'
6213  a520  01
6213  a521  86
6213  a522  a0
6214  a523  ff              .byte $ff, $ff, $d8, $f0 konst. für gleitkomma nach
                                    ascii: '-10 000'
6214  a524  ff
6214  a525  d8
6214  a526  f0
6215  a527  00              .byte $00, $00, $03, $e8 konst. für gleitkomma nach
                                    ascii: '1 000'
6215  a528  00
6215  a529  03
6215  a52a  e8
6216  a52b  ff              .byte $ff, $ff, $ff, $9c konst. für gleitkomma nach
                                    ascii: '-100'
6216  a52c  ff
6216  a52d  ff
6216  a52e  9c
6217  a52f  00              .byte $00, $00, $00, $0a konst. für gleitkomma nach
                                    ascii: '10'
6217  a530  00
6217  a531  00
6217  a532  0a
6218  a533  ff              .byte $ff, $ff, $ff, $ff konst. für gleitkomma nach
                                    ascii: '-1'
6218  a534  ff
6218  a535  ff
6218  a536  ff
6219  a537
6220  a537
6221  a537  ===> basic-routine 'sqr' fac=sqr(fac) <===
6222  a537
6223  a537  20 e3 a1  sqr     jsr movaf         copy fac #1 nach arg
6224  a53a  a9 0e             lda #$0e
6225  a53c  a0 a5             ldy #$a5
6226  a53e  20 66 a1          jsr movfm         vari adr.lo=ac, hi=yr nach fac #1
6227  a541
6228  a541
6229  a541  ===> basic-routine 'dez. potenz' <===
6230  a541
6231  a541  f0 70      fpwrt  beq exp           basic-routine 'exp' fac=exp(fac)
6232  a543  a5 79             lda argexp        fac-arg ungepackt: exponent
6233  a545  d0 03             bne fpwrt1
6234  a547  4c dc 9e          jmp zerof1
6235  a54a
6236  a54a
6237  a54a  a2 5b      fpwrt1 ldx #$5b
6238  a54c  a0 00             ldy #$00
6239  a54e  20 98 a1          jsr movmf         fac #1 runden, nach xr yr
6240  a551  a5 7e             lda argsgn        fac-arg ungepackt: vorzeichen
6241  a553  10 0f             bpl fpwr1
6242  a555  20 b1 a2          jsr int           basic-routine 'int' fac=int(fac)
6243  a558  a9 5b             lda #$5b
6244  a55a  a0 00             ldy #$00
```

```
zeile adr.  obj.-code   source-code

6245  a55c  20 32 a2          jsr fcomp        vergleich fac #1 mit vari (ac/yr)
6246  a55f  d0 03             bne fpwr1
6247  a561  98                tya
6248  a562  a4 0c             ldy charac       puffer für trennzeichen
6249  a564  20 d5 a1  fpwr1   jsr movfa1
6250  a567  98                tya
6251  a568  48                pha
6252  a569  20 ca 9f          jsr log          basic-routine 'log'
6253  a56c  a9 5b             lda #$5b
6254  a56e  a0 00             ldy #$00
6255  a570  20 08 a0          jsr fmult        holt wert in arg  fac = arg * fac
6256  a573  20 b3 a5          jsr exp          basic-routine 'exp' fac=exp(fac)
6257  a576  68                pla
6258  a577  4a                lsr a
6259  a578  90 0a             bcc negrts
6260  a57a
6261  a57a
6262  a57a  ===> basic-routine negation <===
6263  a57a
6264  a57a  a5 71      negop  lda facexp       fac #1: exponent
6265  a57c  f0 06             beq negrts
6266  a57e
6267  a57e
6268  a57e  ===> 'facsgn' invertieren <===
6269  a57e
6270  a57e  a5 76      sav41  lda facsgn       fac #1: vorzeichen
6271  a580  49 ff             eor #$ff
6272  a582  85 76             sta facsgn       fac #1: vorzeichen
6273  a584  60        negrts rts
6274  a585
6275  a585
6276  a585  ===> konstante 1.44269504 = 1/log(2) <===
6277  a585
6278  a585  81        logeb2 .byte $81, $38, $aa, $3b, $29
6278  a586  38
6278  a587  aa
6278  a588  3b
6278  a589  29
6279  a58a
6280  a58a
6281  a58a  ===> 8 konstanten für funktion 'exp' (polynomgrad 7) <===
6282  a58a
6283  a58a  07        expcon .byte $07
6284  a58b  71               .byte $71, $34, $58, $3e, $56 konst. 2.14987637 e-5
                                     für 'exp'
6284  a58c  34
6284  a58d  58
6284  a58e  3e
6284  a58f  56
6285  a590  74               .byte $74, $16, $7e, $b3, $1b konst. 1.43523140 e-4
                                     für 'exp'
6285  a591  16
6285  a592  7e
6285  a593  b3
6285  a594  1b
6286  a595  77               .byte $77, $2f, $ee, $e3, $85 konst. 1.34226348 e-3
                                     für 'exp'
6286  a596  2f
```

```
zeile adr.  obj.-code   source-code

6286  a597  ee
6286  a598  e3
6286  a599  85
6287  a59a  7a              .byte $7a, $1d, $84, $1c, $2a konst. 9.61401170 e-3
                                  für 'exp'
6287  a59b  1d
6287  a59c  84
6287  a59d  1c
6287  a59e  2a
6288  a59f  7c              .byte $7c, $63, $59, $58, $0a konst. 0.05550051269
                                  für 'exp'
6288  a5a0  63
6288  a5a1  59
6288  a5a2  58
6288  a5a3  0a
6289  a5a4  7e              .byte $7e, $75, $fd, $e7, $c6 konst. 0.240226385 für
                                  'exp'
6289  a5a5  75
6289  a5a6  fd
6289  a5a7  e7
6289  a5a8  c6
6290  a5a9  80              .byte $80, $31, $72, $18, $10 konst. 0.693147186 für
                                  'exp'
6290  a5aa  31
6290  a5ab  72
6290  a5ac  18
6290  a5ad  10
6291  a5ae  81              .byte $81, $00, $00, $00, $00 konst. 1 für 'exp'
6291  a5af  00
6291  a5b0  00
6291  a5b1  00
6291  a5b2  00
6292  a5b3
6293  a5b3
6294  a5b3  ===> basic-routine 'exp' fac=exp(fac) <===
6295  a5b3
6296  a5b3  a9 85     exp     lda #$85
6297  a5b5  a0 a5             ldy #$a5
6298  a5b7  20 08 a0          jsr fmult       holt wert in arg  fac = arg * fac
6299  a5ba  a5 80             lda facov       fac overflow-byte
6300  a5bc  69 50             adc #$50
6301  a5be  90 03             bcc stold
6302  a5c0  20 fa a1          jsr incrnd      mantisse von fac um 1 erhöhen
6303  a5c3  85 63     stold   sta jmper+2     addrh sprung zu functions-rout.
6304  a5c5  20 e6 a1          jsr movef
6305  a5c8  a5 71             lda facexp      fac #1: exponent
6306  a5ca  c9 88             cmp #$88
6307  a5cc  90 03             bcc exp1
6308  a5ce  20 ab a0  gomldv  jsr mldvex
6309  a5d1  20 b1 a2  exp1    jsr int         basic-routine 'int' fac=int(fac)
6310  a5d4  a5 0c             lda charac      puffer für trennzeichen
6311  a5d6  18                clc
6312  a5d7  69 81             adc #$81
6313  a5d9  f0 f3             beq gomldv
6314  a5db  38                sec
6315  a5dc  e9 01             sbc #$01
6316  a5de  48                pha
6317  a5df  a2 05             ldx #$05
```

```
zeile adr.  obj.-code   source-code

6318  a5e1  b5 79      swalp  lda argexp,x     fac-arg ungepackt: exponent
6319  a5e3  b4 71             ldy facexp,x     fac #1: exponent
6320  a5e5  95 71             sta facexp,x     fac #1: exponent
6321  a5e7  94 79             sty argexp,x     fac-arg ungepackt: exponent
6322  a5e9  ca                dex
6323  a5ea  10 f5             bpl swalp
6324  a5ec  a5 63             lda jmper+2      addrh sprung zu functions-rout.
6325  a5ee  85 80             sta facov        fac overflow-byte
6326  a5f0  20 30 9e          jsr fsubt        basic-routine '-' (dez. subtr.)
6327  a5f3  20 7a a5          jsr negop        basic-routine negation
6328  a5f6  a9 8a             lda #$8a
6329  a5f8  a0 a5             ldy #$a5
6330  a5fa  20 1b a6          jsr poly
6331  a5fd  a9 00             lda #$00
6332  a5ff  85 7f             sta arisgn       fac-arg ungepackt:
                                               vorzeichenduplikat
6333  a601  68                pla
6334  a602  4c 90 a0          jmp mldexp
6335  a605
6336  a605
6337  a605  ===> polynom-auswertung <===
6338  a605
6339  a605  85 82      polyx  sta fbufpt       addrl zeiger fac-puffer (fout-rout.)
6340  a607  84 83             sty fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
6341  a609  20 94 a1          jsr mov1f        fac #1 nach $64-$68
6342  a60c  a9 64             lda #$64
6343  a60e  20 08 a0          jsr fmult        holt wert in arg  fac = arg * fac
6344  a611  20 1f a6          jsr poly1
6345  a614  a9 64             lda #$64
6346  a616  a0 00             ldy #$00
6347  a618  4c 08 a0          jmp fmult        holt wert in arg  fac = arg * fac
6348  a61b
6349  a61b
6350  a61b  85 82      poly   sta fbufpt       addrl zeiger fac-puffer (fout-rout.)
6351  a61d  84 83             sty fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
6352  a61f  20 91 a1  poly1  jsr mov2f        fac #2 nach $6a-$6e
6353  a622  a9 00             lda #$00
6354  a624  11 82             ora (fbufpt),y   addrl zeiger fac-puffer (fout-rout.)
6355  a626  85 77             sta sgnflg       fac #1: vorzeichenduplikat
6356  a628  a4 82             ldy fbufpt       addrl zeiger fac-puffer (fout-rout.)
6357  a62a  c8                iny
6358  a62b  98                tya
6359  a62c  d0 02             bne poly3
6360  a62e  e6 83             inc fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
6361  a630  85 82      poly3  sta fbufpt       addrl zeiger fac-puffer (fout-rout.)
6362  a632  a4 83             ldy fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
6363  a634  20 08 a0  poly2  jsr fmult        holt wert in arg  fac = arg * fac
6364  a637  a5 82             lda fbufpt       addrl zeiger fac-puffer (fout-rout.)
6365  a639  a4 83             ldy fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
6366  a63b  18                clc
6367  a63c  69 05             adc #$05
6368  a63e  90 01             bcc poly4
6369  a640  c8                iny
6370  a641  85 82      poly4  sta fbufpt       addrl zeiger fac-puffer (fout-rout.)
6371  a643  84 83             sty fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
6372  a645  20 4a 9e          jsr fadd         konstante in arg, fac = arg+fac
6373  a648  a9 6a             lda #$6a
6374  a64a  a0 00             ldy #$00
```

```
zeile adr.  obj.-code   source-code


6375  a64c  c6 77           dec sgnflg       fac #1: vorzeichenduplikat
6376  a64e  d0 e4           bne poly2
6377  a650  60              rts
6378  a651
6379  a651
6380  a651  ===> konst. 11879546 für 'rnd' <===
6381  a651
6382  a651  98        rmulc  .byte $98, $35, $44, $7a
6382  a652  35
6382  a653  44
6382  a654  7a
6383  a655
6384  a655
6385  a655  ===> konst. 3.92767774 e-4 für 'rnd' <===
6386  a655
6387  a655  68        raddc  .byte $68, $28, $b1, $46
6387  a656  28
6387  a657  b1
6387  a658  46
6388  a659
6389  a659
6390  a659  ===> basic-routine 'rnd' <===
6391  a659
6392  a659  20 02 a2  rnd    jsr sign         vorzeichentest fac #1
6393  a65c  30 1c            bmi rnd1
6394  a65e  a6 8d            ldx seedpt       addrl zeiger auf akt. zufallszahl
6395  a660  a4 8e            ldy seedpt+1     addrh zeiger auf akt. zufallszahl
6396  a662  e8               inx
6397  a663  d0 01            bne rnd10
6398  a665  c8               iny
6399  a666  8a        rnd10  txa
6400  a667  a2 02            ldx #$02
6401  a669  20 68 a1         jsr movfum
6402  a66c  a9 51            lda #$51
6403  a66e  a0 a6            ldy #$a6
6404  a670  20 08 a0         jsr fmult        holt wert in arg  fac = arg * fac
6405  a673  a9 55            lda #$55
6406  a675  a0 a6            ldy #$a6
6407  a677  20 4a 9e         jsr fadd         konstante in arg, fac = arg+fac
6408  a67a  a6 75     rnd1   ldx faclo        fac #1: mantisse
6409  a67c  a5 72            lda facho        fac #1: mantisse
6410  a67e  85 75            sta faclo        fac #1: mantisse
6411  a680  86 72            stx facho        fac #1: mantisse
6412  a682  a6 73            ldx facmoh       fac #1: mantisse
6413  a684  a5 74            lda facmo        fac #1: mantisse
6414  a686  85 73            sta facmoh       fac #1: mantisse
6415  a688  86 74            stx facmo        fac #1: mantisse
6416  a68a  a9 00            lda #$00
6417  a68c  85 76            sta facsgn       fac #1: vorzeichen
6418  a68e  a5 71            lda facexp       fac #1: exponent
6419  a690  85 80            sta facov        fac overflow-byte
6420  a692  a9 80            lda #$80
6421  a694  85 71            sta facexp       fac #1: exponent
6422  a696  20 ba 9e         jsr normal       fac linksbündig machen
6423  a699  a6 8d            ldx seedpt       addrl zeiger auf akt. zufallszahl
6424  a69b  a4 8e            ldy seedpt+1     addrh zeiger auf akt. zufallszahl
6425  a69d  a9 02            lda #$02
6426  a69f  e8               inx
```

```
zeile adr.  obj.-code   source-code

6427  a6a0  d0 01           bne rnd20
6428  a6a2  c8              iny
6429  a6a3  4c a5 a1  rnd20 jmp movumf      fac #1 runden,nach xr, yr
6430  a6a6
6431  a6a6
6432  a6a6  ===> basic-routine 'cos' fac=cos(fac) <===
6433  a6a6
6434  a6a6  a9 26     cos   lda #$26
6435  a6a8  a0 a7           ldy #$a7
6436  a6aa  20 4a 9e        jsr fadd        konstante in arg, fac = arg+fac
6437  a6ad
6438  a6ad
6439  a6ad  ===> basic-routine 'sin' <===
6440  a6ad
6441  a6ad  20 e3 a1  sin   jsr movaf       copy fac #1 nach arg
6442  a6b0  a9 2b           lda #$2b
6443  a6b2  a0 a7           ldy #$a7
6444  a6b4  a6 7e           ldx argsgn      fac-arg ungepackt: vorzeichen
6445  a6b6  20 de a0        jsr fdivf
6446  a6b9  20 e3 a1        jsr movaf       copy fac #1 nach arg
6447  a6bc  20 b1 a2        jsr int         basic-routine 'int' fac=int(fac)
6448  a6bf  a9 00           lda #$00
6449  a6c1  85 7f           sta arisgn      fac-arg ungepackt:
                                            vorzeichenduplikat
6450  a6c3  20 30 9e        jsr fsubt       basic-routine '-' (dez. subtr.)
6451  a6c6  a9 30           lda #$30
6452  a6c8  a0 a7           ldy #$a7
6453  a6ca  20 2d 9e        jsr fsub        konstante in arg, fac = arg-fac
6454  a6cd  a5 76           lda facsgn      fac #1: vorzeichen
6455  a6cf  48              pha
6456  a6d0  10 0f           bpl sin1
6457  a6d2  20 27 9e        jsr faddh       fac #1 + 0.5
6458  a6d5  a5 76           lda facsgn      fac #1: vorzeichen
6459  a6d7  30 0b           bmi sin2
6460  a6d9  ad 59 02        lda tansgn      flag für 'tan' vorzeichen
6461  a6dc  49 ff           eor #$ff
6462  a6de  8d 59 02        sta tansgn      flag für 'tan' vorzeichen
6463  a6e1  20 7a a5  sin1  jsr negop       basic-routine negation
6464  a6e4  a9 30     sin2  lda #$30
6465  a6e6  a0 a7           ldy #$a7
6466  a6e8  20 4a 9e        jsr fadd        konstante in arg, fac = arg+fac
6467  a6eb  68              pla
6468  a6ec  10 03           bpl sin3
6469  a6ee  20 7a a5        jsr negop       basic-routine negation
6470  a6f1  a9 35     sin3  lda #$35
6471  a6f3  a0 a7           ldy #$a7
6472  a6f5  4c 05 a6        jmp polyx       polynom-auswertung
6473  a6f8
6474  a6f8
6475  a6f8  ===> basic-routine 'tan' fac=tan(fac) <===
6476  a6f8
6477  a6f8  20 94 a1  tan   jsr mov1f       fac #1 nach $64-$68
6478  a6fb  a9 00           lda #$00
6479  a6fd  8d 59 02        sta tansgn      flag für 'tan' vorzeichen
6480  a700  20 ad a6        jsr sin         basic-routine 'sin'
6481  a703  a2 5b           ldx #$5b
6482  a705  a0 00           ldy #$00
6483  a707  20 98 a1        jsr movmf       fac #1 runden, nach xr yr
```

```
zeile adr.  obj.-code   source-code

6484  a70a  a9 64           lda #$64
6485  a70c  a0 00           ldy #$00
6486  a70e  20 66 a1        jsr movfm        vari adr.lo=ac, hi=yr nach fac #1
6487  a711  a9 00           lda #$00
6488  a713  85 76           sta facsgn       fac #1: vorzeichen
6489  a715  ad 59 02        lda tansgn       flag für 'tan' vorzeichen
6490  a718  20 22 a7        jsr cosc
6491  a71b  a9 5b           lda #$5b
6492  a71d  a0 00           ldy #$00
6493  a71f  4c e6 a0        jmp fdiv         holt wert in arg, fac = arg / fac
6494  a722
6495  a722
6496  a722  48        cosc   pha
6497  a723  4c e1 a6        jmp sin1
6498  a726
6499  a726
6500  a726  ===> konstante pi/2 = 1.57079633 <===
6501  a726
6502  a726  81        pi2    .byte $81, $49, $0f, $da, $a2
6502  a727  49
6502  a728  0f
6502  a729  da
6502  a72a  a2
6503  a72b
6504  a72b
6505  a72b  ===> konstante 2*pi = 6.28318531 <===
6506  a72b
6507  a72b  83        twopi  .byte $83, $49, $0f, $da, $a2
6507  a72c  49
6507  a72d  0f
6507  a72e  da
6507  a72f  a2
6508  a730
6509  a730
6510  a730  ===> konstante 0.25 <===
6511  a730
6512  a730  7f        fr4    .byte $7f, $00, $00, $00, $00
6512  a731  00
6512  a732  00
6512  a733  00
6512  a734  00
6513  a735
6514  a735
6515  a735  ===> 6 konstanten für funktionen 'sin + cos' (polynomgrad 5) <===
6516  a735
6517  a735  05        sincon .byte $05
6518  a736  84               .byte $84, $e6, $1a, $2d, $1b konst. -14.3813907 für
                                    'sin + cos'
6518  a737  e6
6518  a738  1a
6518  a739  2d
6518  a73a  1b
6519  a73b  86               .byte $86, $28, $07, $f6, $f8 konst. 42.0077971 für
                                    'sin + cos'
6519  a73c  28
6519  a73d  07
6519  a73e  f6
6519  a73f  f8
```

```
zeile adr.  obj.-code   source-code

6520  a740  87              .byte $87, $99, $68, $89, $01 konst. -76.7041703 für
                                    'sin + cos'
6520  a741  99
6520  a742  68
6520  a743  89
6520  a744  01
6521  a745  87              .byte $87, $23, $35, $df, $e1 konst. 81.6052237 für
                                    'sin + cos'
6521  a746  23
6521  a747  35
6521  a748  df
6521  a749  e1
6522  a74a  86              .byte $86, $a5, $5d, $e7, $28 konst. -41.3147021 für
                                    'sin + cos'
6522  a74b  a5
6522  a74c  5d
6522  a74d  e7
6522  a74e  28
6523  a74f  83              .byte $83, $49, $0f, $da, $a2 konst. 6.28318531 für
                                    'sin + cos'
6523  a750  49
6523  a751  0f
6523  a752  da
6523  a753  a2
6524  a754
6525  a754
6526  a754  ===> 12 konstanten für 'atn' (polynomgrad 11) <===
6527  a754
6528  a754  0b        atncon .byte $0b
6529  a755  76              .byte $76, $b3, $83, $bd, $d3 konst. -6.84793912 e-4
                                    für 'atn'
6529  a756  b3
6529  a757  83
6529  a758  bd
6529  a759  d3
6530  a75a  79              .byte $79, $1e, $f4, $a6, $f5 konst. 4.85094216 e-3
                                    für 'atn'
6530  a75b  1e
6530  a75c  f4
6530  a75d  a6
6530  a75e  f5
6531  a75f  7b              .byte $7b, $83, $fc, $b0, $10 konst. -0.0161117015
                                    für 'atn'
6531  a760  83
6531  a761  fc
6531  a762  b0
6531  a763  10
6532  a764  7c              .byte $7c, $0c, $1f, $67, $ca konst. 0.034209638 für
                                    'atn'
6532  a765  0c
6532  a766  1f
6532  a767  67
6532  a768  ca
6533  a769  7c              .byte $7c, $de, $53, $cb, $c1 konst. -0.054279133 für
                                    'atn'
6533  a76a  de
6533  a76b  53
6533  a76c  cb
```

```
zeile adr.  obj.-code   source-code

6533  a76d  c1
6534  a76e  7d              .byte $7d, $14, $64, $70, $4c konst. 0.0724571965 für
                                'atn'
6534  a76f  14
6534  a770  64
6534  a771  70
6534  a772  4c
6535  a773  7d              .byte $7d, $b7, $ea, $51, $7a konst. -0.0898019185
                                für 'atn'
6535  a774  b7
6535  a775  ea
6535  a776  51
6535  a777  7a
6536  a778  7d              .byte $7d, $63, $30, $88, $7e konst. 0.110932413 für
                                'atn'
6536  a779  63
6536  a77a  30
6536  a77b  88
6536  a77c  7e
6537  a77d  7e              .byte $7e, $92, $44, $99, $3a konst. -0.142839808 für
                                'atn'
6537  a77e  92
6537  a77f  44
6537  a780  99
6537  a781  3a
6538  a782  7e              .byte $7e, $4c, $cc, $91, $c7 konst. 0.19999912 für
                                'atn'
6538  a783  4c
6538  a784  cc
6538  a785  91
6538  a786  c7
6539  a787  7f              .byte $7f, $aa, $aa, $aa, $13 konst. -00.333333316
                                für 'atn'
6539  a788  aa
6539  a789  aa
6539  a78a  aa
6539  a78b  13
6540  a78c  81              .byte $81, $00, $00, $00, $00 konst. 1 für 'atn'
6540  a78d  00
6540  a78e  00
6540  a78f  00
6540  a790  00
6541  a791
6542  a791
6543  a791  ===> basic-routine 'atn' fac=atn(fac) <===
6544  a791
6545  a791  a5 76     atn     lda facsgn      fac #1: vorzeichen
6546  a793  48                pha
6547  a794  10 03              bpl atn1
6548  a796  20 7a a5          jsr negop       basic-routine negation
6549  a799  a5 71     atn1    lda facexp      fac #1: exponent
6550  a79b  48                pha
6551  a79c  c9 81             cmp #$81
6552  a79e  90 07             bcc atn2
6553  a7a0  a9 9c             lda #$9c
6554  a7a2  a0 9f             ldy #$9f
6555  a7a4  20 e6 a0          jsr fdiv        holt wert in arg, fac = arg / fac
6556  a7a7  a9 54     atn2    lda #$54
```

```
zeile adr.  obj.-code   source-code

6557  a7a9  a0 a7           ldy #$a7
6558  a7ab  20 05 a6        jsr polyx        polynom-auswertung
6559  a7ae  68              pla
6560  a7af  c9 81           cmp #$81
6561  a7b1  90 07           bcc atn3
6562  a7b3  a9 26           lda #$26
6563  a7b5  a0 a7           ldy #$a7
6564  a7b7  20 2d 9e        jsr fsub         konstante in arg, fac = arg-fac
6565  a7ba  68        atn3  pla
6566  a7bb  10 61           bpl atn4
6567  a7bd  4c 7a a5        jmp negop        basic-routine negation
6568  a7c0
6569  a7c0                  .end
6570  a7c0                  .lib strng1
```

```
zeile adr.  obj.-code   source-code

6572  a7c0
6573  a7c0  ===> basic-routine 'pudef' <===
6574  a7c0
6575  a7c0  20 b6 91  puctrl jsr sav13
6576  a7c3  20 73 ba         jsr mapinx       umsch. auf bank (# in $24)
6577  a7c6  a8               tay
6578  a7c7  f0 0f            beq pucbye
6579  a7c9  c0 04            cpy #$04
6580  a7cb  90 02            bcc puc50
6581  a7cd  a0 04            ldy #$04
6582  a7cf  88        puc50  dey
6583  a7d0  b1 22     puc60  lda (index1),y   addrl indirekter index #1
6584  a7d2  99 73 02         sta pufill,y     print using: füllzeichen
6585  a7d5  88               dey
6586  a7d6  10 f8            bpl puc60
6587  a7d8  4c 8c ba  pucbye jmp mapusr       umsch. auf bank # 1
6588  a7db
6589  a7db
6590  a7db  ===> basic-routine 'str$' <===
6591  a7db
6592  a7db  20 04 b5  strd   jsr chknum       prüfen, ob numerische variable
6593  a7de  20 e2 a3         jsr fout         umw. gk-zahl in string (ablage ab
                                              $200)
6594  a7e1  68               pla
6595  a7e2  68               pla
6596  a7e3  a0 00     foutst ldy #$00
6597  a7e5  b9 00 02  str010 lda buf,y        basic input-puffer (-$2ff)
6598  a7e8  f0 03            beq str020
6599  a7ea  c8               iny
6600  a7eb  d0 f8            bne str010
6601  a7ed  98        str020 tya
6602  a7ee  20 11 a8         jsr strspa       adr./länge des string ablegen
6603  a7f1  84 01            sty i6509        6509 indirection register
6604  a7f3  a0 00            ldy #$00
6605  a7f5  b9 00 02  str030 lda buf,y        basic input-puffer (-$2ff)
6606  a7f8  f0 05            beq str040
6607  a7fa  91 71            sta (facexp),y   fac #1: exponent
6608  a7fc  c8               iny
6609  a7fd  d0 f6            bne str030
6610  a7ff  20 8c ba  str040 jsr mapusr       umsch. auf bank # 1
6611  a802  4c 6c a8         jmp putnew
6612  a805
6613  a805
6614  a805  ===> berechnet stringzeiger <===
6615  a805
6616  a805  a6 76      strini ldx facsgn       fac #1: vorzeichen
6617  a807  86 60            stx dscpnt+2     bank zeiger string descriptor
6618  a809  a6 74            ldx facmo        fac #1: mantisse
6619  a80b  a4 75            ldy faclo        fac #1: mantisse
6620  a80d  86 5e            stx dscpnt       addrl zeiger string
                                              descr./temp.fac#3
6621  a80f  84 5f            sty dscpnt+1     addrh zeiger string
                                              descr./temp.fac#3
6622  a811
6623  a811
6624  a811  ===> adr./länge des string ablegen <===
6625  a811
6626  a811  20 53 ad  strspa jsr getspa       platz für string reservieren, länge
                                              in ac
```

```
zeile adr.  obj.-code   source-code

6627  a814  85 70           sta dsctmp      zeitweise descriptoren-ablage
6628  a816  86 71           stx facexp      fac #1: exponent
6629  a818  84 72           sty facho       fac #1: mantisse
6630  a81a  a0 02           ldy #$02
6631  a81c  84 73           sty facmoh      fac #1: mantisse
6632  a81e  60       atn4   rts
6633  a81f
6634  a81f
6635  a81f  ===> sucht elemente von zeichenkette <===
6636  a81f
6637  a81f  48       strlit pha
6638  a820  a9 22           lda #$22
6639  a822  85 0c           sta charac      puffer für trennzeichen
6640  a824  85 0d           sta endchr      puffer für trennzeichen
6641  a826  68              pla
6642  a827  85 7f    strlt2 sta arisgn      fac-arg ungepackt:
                                            vorzeichenduplikat
6643  a829  84 80           sty facov       fac overflow-byte
6644  a82b  86 81           stx facov+1     fac overflow-byte
6645  a82d  86 01           stx i6509       6509 indirection register
6646  a82f  85 71           sta facexp      fac #1: exponent
6647  a831  84 72           sty facho       fac #1: mantisse
6648  a833  86 73           stx facmoh      fac #1: mantisse
6649  a835  86 84           stx fbufpt+2    bank zeiger fac-puffer (fout-rout.)
6650  a837  a0 ff           ldy #$ff
6651  a839  c8       strget iny
6652  a83a  b1 7f           lda (arisgn),y  fac-arg ungepackt:
                                            vorzeichenduplikat
6653  a83c  f0 0c           beq strfi1
6654  a83e  c5 0c           cmp charac      puffer für trennzeichen
6655  a840  f0 04           beq strfin
6656  a842  c5 0d           cmp endchr      puffer für trennzeichen
6657  a844  d0 f3           bne strget
6658  a846  c9 22    strfin cmp #$22
6659  a848  f0 01           beq strfi2
6660  a84a  18       strfi1 clc
6661  a84b  20 8c ba strfi2 jsr mapusr      umsch. auf bank # 1
6662  a84e  84 70           sty dsctmp      zeitweise descriptoren-ablage
6663  a850  98              tya
6664  a851  65 7f           adc arisgn      fac-arg ungepackt:
                                            vorzeichenduplikat
6665  a853  85 82           sta fbufpt      addrl zeiger fac-puffer (fout-rout.)
6666  a855  a6 80           ldx facov       fac overflow-byte
6667  a857  90 01           bcc strst2
6668  a859  e8              inx
6669  a85a  86 83    strst2 stx fbufpt+1    addrh zeiger fac-puffer (fout-rout.)
6670  a85c  98              tya
6671  a85d  20 05 a8        jsr strini      berechnet stringzeiger
6672  a860  a2 02           ldx #$02
6673  a862  b4 7f    sotl   ldy arisgn,x    fac-arg ungepackt:
                                            vorzeichenduplikat
6674  a864  94 22           sty index1,x    addrl indirekter index #1
6675  a866  ca              dex
6676  a867  10 f9           bpl sotl
6677  a869  20 c4 a8 glar   jsr movdo
6678  a86c  a4 1d    putnew ldy temmpt      zeiger auf zeitw. stringdescriptor
                                            (rel. offset)
6679  a86e  c0 0c           cpy #$0c
```

```
zeile adr.  obj.-code   source-code

6680  a870  d0 05           bne putnw1
6681  a872  a2 46           ldx #$46
6682  a874  4c 52 85        jmp error        ind. jmp zur fehlerroutine
6683  a877
6684  a877
6685  a877  a9 02     putnw1 lda #$02
6686  a879  85 01           sta i6509        6509 indirection register
6687  a87b  85 76           sta facsgn       fac #1: vorzeichen
6688  a87d  a2 00           ldx #$00
6689  a87f  b5 70     putnwl lda dsctmp,x     zeitweise descriptoren-ablage
6690  a881  91 20           sta (tempst),y   addrl für 3 temp.
                                             string-descriptoren
6691  a883  c8              iny
6692  a884  e8              inx
6693  a885  e0 04           cpx #$04
6694  a887  d0 f6           bne putnwl
6695  a889  20 8c ba        jsr mapusr       umsch. auf bank # 1
6696  a88c  a5 21           lda tempst+1     addrh für 3 temp.
                                             string-descriptoren
6697  a88e  85 75           sta faclo        fac #1: mantisse
6698  a890  85 1f           sta lastpt+1     addrh letzt benutzter string-descr.
6699  a892  18              clc
6700  a893  a5 20           lda tempst       addrl für 3 temp.
                                             string-descriptoren
6701  a895  65 1d           adc temmpt       zeiger auf zeitw. stringdescriptor
                                             (rel. offset)
6702  a897  85 74           sta facmo        fac #1: mantisse
6703  a899  85 1e           sta lastpt       addrl letzt benutzter string-descr.
6704  a89b  90 04           bcc putnw2
6705  a89d  e6 75           inc faclo        fac #1: mantisse
6706  a89f  e6 1f           inc lastpt+1     addrh letzt benutzter string-descr.
6707  a8a1  84 1d     putnw2 sty temmpt       zeiger auf zeitw. stringdescriptor
                                             (rel. offset)
6708  a8a3  a0 00           ldy #$00
6709  a8a5  84 80           sty facov        fac overflow-byte
6710  a8a7  88              dey
6711  a8a8  84 11           sty valtyp       flag für variablen typ (0=num,
                                             1=string)
6712  a8aa  60              rts
6713  a8ab
6714  a8ab
6715  a8ab  ===> string ablegen <===
6716  a8ab
6717  a8ab  a5 81     movins lda facov+1      fac overflow-byte
6718  a8ad  85 01           sta i6509        6509 indirection register
6719  a8af  a0 00           ldy #$00
6720  a8b1  b1 7f           lda (arisgn),y   fac-arg ungepackt:
                                             vorzeichenduplikat
6721  a8b3  aa              tax
6722  a8b4  c8              iny
6723  a8b5  b1 7f           lda (arisgn),y   fac-arg ungepackt:
                                             vorzeichenduplikat
6724  a8b7  85 22           sta index1       addrl indirekter index #1
6725  a8b9  c8              iny
6726  a8ba  b1 7f           lda (arisgn),y   fac-arg ungepackt:
                                             vorzeichenduplikat
6727  a8bc  85 23           sta index1+1     addrh indirekter index #1
6728  a8be  c8              iny
```

```
zeile adr.  obj.-code   source-code

6729  a8bf  b1 7f            lda (arisgn),y   fac-arg ungepackt:
                                               vorzeichenduplikat
6730  a8c1  85 24            sta index1+2     bank indirekter index #1
6731  a8c3  8a               txa
6732  a8c4  a8        movdo  tay
6733  a8c5  f0 12            beq mvdone
6734  a8c7  48               pha
6735  a8c8  a2 02            ldx #$02
6736  a8ca  88        movlp  dey
6737  a8cb  a5 24            lda index1+2     bank indirekter index #1
6738  a8cd  85 01            sta i6509        6509 indirection register
6739  a8cf  b1 22            lda (index1),y   addrl indirekter index #1
6740  a8d1  86 01            stx i6509        6509 indirection register
6741  a8d3  91 3d            sta (frespc),y   addrl zeiger auf neuen string
6742  a8d5  98               tya
6743  a8d6  d0 f2            bne movlp
6744  a8d8  68               pla
6745  a8d9  20 8c ba  mvdone jsr mapusr       umsch. auf bank # 1
6746  a8dc  18               clc
6747  a8dd  65 3d            adc frespc       addrl zeiger auf neuen string
6748  a8df  85 3d            sta frespc       addrl zeiger auf neuen string
6749  a8e1  90 02            bcc mvstrt
6750  a8e3  e6 3e            inc frespc+1     addrh zeiger auf neuen string
6751  a8e5  60        mvstrt rts
6752  a8e6
6753  a8e6
6754  a8e6  ===> alten string entfernen <===
6755  a8e6
6756  a8e6  20 06 b5  frestr jsr chkstr       prüfen, ob stringvariable
6757  a8e9
6758  a8e9
6759  a8e9  ===> stringverwaltung, freier string <===
6760  a8e9
6761  a8e9  a5 74     frefac lda facmo         fac #1: mantisse
6762  a8eb  a4 75            ldy faclo        fac #1: mantisse
6763  a8ed  a6 76            ldx facsgn       fac #1: vorzeichen
6764  a8ef  20 4f a9  fretmp jsr sav10
6765  a8f2  d0 3c            bne fre02
6766  a8f4  20 33 aa         jsr sadi2
6767  a8f7  90 37            bcc fre02
6768  a8f9  20 20 aa         jsr mkgarb       string-trailer eintragen: ungültiger
                                               string
6769  a8fc  48               pha
6770  a8fd  49 ff            eor #$ff
6771  a8ff  38               sec
6772  a900  65 22            adc index1       addrl indirekter index #1
6773  a902  a4 23            ldy index1+1     addrh indirekter index #1
6774  a904  b0 01            bcs res00
6775  a906  88               dey
6776  a907  85 22     res00  sta index1       addrl indirekter index #1
6777  a909  84 23            sty index1+1     addrh indirekter index #1
6778  a90b  aa               tax
6779  a90c  68               pla
6780  a90d  a4 24            ldy index1+2     bank indirekter index #1
6781  a90f  c0 02            cpy #$02
6782  a911  d0 5f            bne frerts
6783  a913  a4 23            ldy index1+1     addrh indirekter index #1
6784  a915  c4 3c            cpy fretop+1     addrh top of string free space
```

```
zeile adr.  obj.-code   source-code

6785  a917  d0 59           bne frerts
6786  a919  e4 3b           cpx fretop       addrl top of string free space
6787  a91b  d0 55           bne frerts
6788  a91d  48              pha
6789  a91e  18              clc
6790  a91f  65 3b           adc fretop       addrl top of string free space
6791  a921  90 02           bcc fre01
6792  a923  e6 3c           inc fretop+1     addrh top of string free space
6793  a925  18        fre01 clc
6794  a926  69 03           adc #$03
6795  a928  85 3b           sta fretop       addrl top of string free space
6796  a92a  90 02           bcc frepla
6797  a92c  e6 3c           inc fretop+1     addrh top of string free space
6798  a92e  68        frepla pla
6799  a92f  60              rts
6800  a930
6801  a930
6802  a930  20 73 ba  fre02  jsr mapinx       umsch. auf bank (# in $24)
6803  a933  a0 00           ldy #$00
6804  a935  b1 22           lda (index1),y   addrl indirekter index #1
6805  a937  48              pha
6806  a938  c8              iny
6807  a939  b1 22           lda (index1),y   addrl indirekter index #1
6808  a93b  48              pha
6809  a93c  c8              iny
6810  a93d  b1 22           lda (index1),y   addrl indirekter index #1
6811  a93f  aa              tax
6812  a940  c8              iny
6813  a941  b1 22           lda (index1),y   addrl indirekter index #1
6814  a943  85 24           sta index1+2     bank indirekter index #1
6815  a945  86 23           stx index1+1     addrh indirekter index #1
6816  a947  20 8c ba        jsr mapusr       umsch. auf bank # 1
6817  a94a  68              pla
6818  a94b  85 22           sta index1       addrl indirekter index #1
6819  a94d  68              pla
6820  a94e  60              rts
6821  a94f
6822  a94f
6823  a94f  85 22     sav10 sta index1       addrl indirekter index #1
6824  a951  84 23           sty index1+1     addrh indirekter index #1
6825  a953  86 24           stx index1+2     bank indirekter index #1
6826  a955  e0 02     fretms cpx #$02
6827  a957  d0 19           bne frerts
6828  a959  c4 1f           cpy lastpt+1     addrh letzt benutzter string-descr.
6829  a95b  d0 15           bne frerts
6830  a95d  c5 1e           cmp lastpt       addrl letzt benutzter string-descr.
6831  a95f  d0 11           bne frerts
6832  a961  e9 04           sbc #$04
6833  a963  85 1e           sta lastpt       addrl letzt benutzter string-descr.
6834  a965  b0 02           bcs fret10
6835  a967  c6 1f           dec lastpt+1     addrh letzt benutzter string-descr.
6836  a969  38        fret10 sec
6837  a96a  a5 1d           lda temmpt       zeiger auf zeitw. stringdescriptor
                                             (rel. offset)
6838  a96c  e9 04           sbc #$04
6839  a96e  85 1d           sta temmpt       zeiger auf zeitw. stringdescriptor
                                             (rel. offset)
6840  a970  a0 00           ldy #$00
```

```
zeile adr.  obj.-code   source-code

6841  a972  60        frerts rts
6842  a973
6843  a973
6844  a973  a4 56     inpcom ldy lstpnt+2     bank zeiger letzter string
6845  a975  c0 0f            cpy #$0f
6846  a977  d0 03            bne getspt
6847  a979  4c d6 ac         jmp mktime       interne uhr mit string stellen
6848  a97c
6849  a97c
6850  a97c  a4 76     getspt ldy facsgn       fac #1: vorzeichen
6851  a97e  c0 02            cpy #$02
6852  a980  d0 0e            bne dntcpy
6853  a982  a4 75            ldy faclo        fac #1: mantisse
6854  a984  c4 3a            cpy strend+1     addrh ende benutzter ram-bereich
6855  a986  90 35            bcc copy         string ins ram, gültig machen
6856  a988  d0 06            bne dntcpy
6857  a98a  a5 74            lda facmo        fac #1: mantisse
6858  a98c  c5 39            cmp strend       addrl ende benutzter ram-bereich
6859  a98e  90 2d            bcc copy         string ins ram, gültig machen
6860  a990  a5 74     dntcpy lda facmo        fac #1: mantisse
6861  a992  a4 75            ldy faclo        fac #1: mantisse
6862  a994  a6 76            ldx facsgn       fac #1: vorzeichen
6863  a996  85 5e            sta dscpnt       addrl zeiger string
                                               descr./temp.fac#3
6864  a998  84 5f            sty dscpnt+1     addrh zeiger string
                                               descr./temp.fac#3
6865  a99a  86 60            stx dscpnt+2     bank zeiger string descriptor
6866  a99c  20 4f a9         jsr sav10
6867  a99f  20 33 aa         jsr sadi2
6868  a9a2  90 03            bcc dcop02
6869  a9a4  20 fc a9         jsr bcklnk       string-trailer eintragen:
                                               rückwärtszeiger
6870  a9a7  20 12 aa  dcop02 jsr fixold
6871  a9aa  a6 56            ldx lstpnt+2     bank zeiger letzter string
6872  a9ac  a0 03            ldy #$03
6873  a9ae  20 6e ba  dcop01 jsr mapdsp       umsch. auf bank (# in $60)
6874  a9b1  b1 5e            lda (dscpnt),y   addrl zeiger string
                                               descr./temp.fac#3
6875  a9b3  86 01            stx i6509        6509 indirection register
6876  a9b5  91 54            sta (lstpnt),y   addrl zeiger letzter string
6877  a9b7  88               dey
6878  a9b8  10 f4            bpl dcop01
6879  a9ba  4c 8c ba         jmp mapusr       umsch. auf bank # 1
6880  a9bd
6881  a9bd
6882  a9bd  ===> string ins ram, gültig machen <===
6883  a9bd
6884  a9bd  a5 76     copy   lda facsgn       fac #1: vorzeichen
6885  a9bf  85 01            sta i6509        6509 indirection register
6886  a9c1  a0 00            ldy #$00
6887  a9c3  b1 74            lda (facmo),y    fac #1: mantisse
6888  a9c5  20 8c ba         jsr mapusr       umsch. auf bank # 1
6889  a9c8  20 05 a8         jsr strini       berechnet stringzeiger
6890  a9cb  a2 02            ldx #$02
6891  a9cd  b4 5e     soth   ldy dscpnt,x     addrl zeiger string
                                               descr./temp.fac#3
6892  a9cf  94 7f            sty arisgn,x     fac-arg ungepackt:
                                               vorzeichenduplikat
```

```
zeile adr.  obj.-code   source-code

6893  a9d1  ca              dex
6894  a9d2  10 f9           bpl soth
6895  a9d4  20 ab a8        jsr movins      string ablegen
6896  a9d7  a5 7f           lda arisgn      fac-arg ungepackt:
                                            vorzeichenduplikat
6897  a9d9  a4 80           ldy facov       fac overflow-byte
6898  a9db  a6 81           ldx facov+1     fac overflow-byte
6899  a9dd  20 55 a9        jsr fretms
6900  a9e0  20 6c aa        jsr stradj      strings über fretop: carry=1,
                                            länge=ac, zeiger > linkbytes
6901  a9e3  90 03           bcc copy02
6902  a9e5  20 fc a9        jsr bcklnk      string-trailer eintragen:
                                            rückwärtszeiger
6903  a9e8  20 12 aa  copy02 jsr fixold
6904  a9eb  a0 03           ldy #$03
6905  a9ed  a5 56           lda lstpnt+2    bank zeiger letzter string
6906  a9ef  85 01           sta i6509       6509 indirection register
6907  a9f1  b9 70 00  copy01 lda dsctmp,y    zeitweise descriptoren-ablage
6908  a9f4  91 54           sta (lstpnt),y  addrl zeiger letzter string
6909  a9f6  88              dey
6910  a9f7  10 f8           bpl copy01
6911  a9f9  4c 8c ba        jmp mapusr      umsch. auf bank # 1
6912  a9fc
6913  a9fc
6914  a9fc  ===> string-trailer eintragen: rückwärtszeiger <===
6915  a9fc
6916  a9fc  20 73 ba  bcklnk jsr mapinx      umsch. auf bank (# in $24)
6917  a9ff  a0 00           ldy #$00
6918  aa01  a5 54           lda lstpnt      addrl zeiger letzter string
6919  aa03  91 22           sta (index1),y  addrl indirekter index #1
6920  aa05  c8              iny
6921  aa06  a5 55           lda lstpnt+1    addrh zeiger letzter string
6922  aa08  91 22           sta (index1),y  addrl indirekter index #1
6923  aa0a  c8              iny
6924  aa0b  a5 56           lda lstpnt+2    bank zeiger letzter string
6925  aa0d  91 22           sta (index1),y  addrl indirekter index #1
6926  aa0f  4c 8c ba        jmp mapusr      umsch. auf bank # 1
6927  aa12
6928  aa12
6929  aa12  a2 02     fixold ldx #$02
6930  aa14  b4 54     sump   ldy lstpnt,x    addrl zeiger letzter string
6931  aa16  94 22           sty index1,x    addrl indirekter index #1
6932  aa18  ca              dex
6933  aa19  10 f9           bpl sump
6934  aa1b  20 33 aa        jsr sadi2
6935  aa1e  90 45           bcc fnk05
6936  aa20
6937  aa20
6938  aa20  ===> string-trailer eintragen: ungültiger string <===
6939  aa20
6940  aa20  20 73 ba  mkgarb jsr mapinx      umsch. auf bank (# in $24)
6941  aa23  a0 02           ldy #$02
6942  aa25  a9 ff           lda #$ff
6943  aa27  91 22           sta (index1),y  addrl indirekter index #1
6944  aa29  88              dey
6945  aa2a  91 22           sta (index1),y  addrl indirekter index #1
6946  aa2c  88              dey
6947  aa2d  8a              txa
```

```
zeile adr.  obj.-code   source-code

6948  aa2e  91 22           sta (index1),y   addrl indirekter index #1
6949  aa30  4c 8c ba        jmp mapusr       umsch. auf bank # 1
6950  aa33
6951  aa33
6952  aa33  20 73 ba  sadi2  jsr mapinx       umsch. auf bank (# in $24)
6953  aa36  a0 00           ldy #$00
6954  aa38  b1 22           lda (index1),y   addrl indirekter index #1
6955  aa3a  48              pha
6956  aa3b  f0 29           beq sadi8
6957  aa3d  c8              iny
6958  aa3e  b1 22           lda (index1),y   addrl indirekter index #1
6959  aa40  48              pha
6960  aa41  c8              iny
6961  aa42  b1 22           lda (index1),y   addrl indirekter index #1
6962  aa44  aa              tax
6963  aa45  c8              iny
6964  aa46  b1 22           lda (index1),y   addrl indirekter index #1
6965  aa48  a8              tay
6966  aa49  20 8c ba        jsr mapusr       umsch. auf bank # 1
6967  aa4c  68              pla
6968  aa4d  c4 19     sadi3 cpy dsdesc+3     bank disc-status string
6969  aa4f  d0 08           bne sadi4
6970  aa51  e4 18           cpx dsdesc+2     addrh disc-status string
6971  aa53  d0 04           bne sadi4
6972  aa55  c5 17           cmp dsdesc+1     addrl disc-status string
6973  aa57  f0 0d           beq sadi8
6974  aa59  85 22     sadi4 sta index1       addrl indirekter index #1
6975  aa5b  86 23           stx index1+1     addrh indirekter index #1
6976  aa5d  84 24           sty index1+2     bank indirekter index #1
6977  aa5f  68              pla
6978  aa60  aa              tax
6979  aa61  20 18 ab        jsr sav15
6980  aa64  38              sec
6981  aa65  60        fnk05 rts
6982  aa66
6983  aa66
6984  aa66  20 8c ba  sadi8 jsr mapusr       umsch. auf bank # 1
6985  aa69  68              pla
6986  aa6a  18              clc
6987  aa6b  60              rts
6988  aa6c
6989  aa6c
6990  aa6c  ===> strings über fretop: carry=1, länge=ac, zeiger > linkbytes <===
6991  aa6c
6992  aa6c  a5 70     stradj lda dsctmp       zeitweise descriptoren-ablage
6993  aa6e  48              pha
6994  aa6f  f0 f5           beq sadi8
6995  aa71  a5 71           lda facexp       fac #1: exponent
6996  aa73  a6 72           ldx facho        fac #1: mantisse
6997  aa75  a4 73           ldy facmoh       fac #1: mantisse
6998  aa77  4c 4d aa        jmp sadi3
6999  aa7a
7000  aa7a                  .end
7001  aa7a                  .lib strng2
```

```
zeile adr.  obj.-code   source-code

7003  aa7a
7004  aa7a  ===> string verketten '+' <===
7005  aa7a
7006  aa7a  a5 76     cat    lda facsgn       fac #1: vorzeichen
7007  aa7c  48               pha
7008  aa7d  a5 75            lda faclo        fac #1: mantisse
7009  aa7f  48               pha
7010  aa80  a5 74            lda facmo        fac #1: mantisse
7011  aa82  48               pha
7012  aa83  20 ae 96         jsr eval         jmp: folgenden beliebigen ausdruck
                                              auswerten
7013  aa86  20 06 b5         jsr chkstr       prüfen, ob stringvariable
7014  aa89  68               pla
7015  aa8a  85 7f            sta arisgn       fac-arg ungepackt:
                                              vorzeichenduplikat
7016  aa8c  68               pla
7017  aa8d  85 80            sta facov        fac overflow-byte
7018  aa8f  68               pla
7019  aa90  85 81            sta facov+1      fac overflow-byte
7020  aa92  85 01            sta i6509        6509 indirection register
7021  aa94  a0 00            ldy #$00
7022  aa96  b1 7f            lda (arisgn),y   fac-arg ungepackt:
                                              vorzeichenduplikat
7023  aa98  8d 5b 02         sta ldaadr       addrl modifiable adresse
7024  aa9b  a5 76            lda facsgn       fac #1: vorzeichen
7025  aa9d  85 01            sta i6509        6509 indirection register
7026  aa9f  b1 74            lda (facmo),y    fac #1: mantisse
7027  aaa1  20 8c ba         jsr mapusr       umsch. auf bank # 1
7028  aaa4  18               clc
7029  aaa5  6d 5b 02         adc ldaadr       addrl modifiable adresse
7030  aaa8  90 03            bcc sizeok
7031  aaaa  4c 85 b7         jmp errlen       ausgabe '?string too long', ready
7032  aaad
7033  aaad
7034  aaad  20 05 a8  sizeok jsr strini       berechnet stringzeiger
7035  aab0  20 ab a8         jsr movins       string ablegen
7036  aab3  20 c8 aa         jsr sav47
7037  aab6  20 c4 a8         jsr movdo
7038  aab9  a5 7f            lda arisgn       fac-arg ungepackt:
                                              vorzeichenduplikat
7039  aabb  a4 80            ldy facov        fac overflow-byte
7040  aabd  a6 81            ldx facov+1      fac overflow-byte
7041  aabf  20 ef a8         jsr fretmp
7042  aac2  20 6c a8         jsr putnew
7043  aac5  4c de 95         jmp tstop
7044  aac8
7045  aac8
7046  aac8  a5 5e     sav47  lda dscpnt       addrl zeiger string
                                              descr./temp.fac#3
7047  aaca  a4 5f            ldy dscpnt+1     addrh zeiger string
                                              descr./temp.fac#3
7048  aacc  a6 60            ldx dscpnt+2     bank zeiger string descriptor
7049  aace  4c ef a8         jmp fretmp
7050  aad1
7051  aad1
7052  aad1  ===> basic-routine 'chr$' <===
7053  aad1
7054  aad1  20 d9 b4  chrd   jsr conint       zahl < 256 aus akt. text nach xr
```

```
zeile adr.  obj.-code   source-code

7055  aad4  8a              txa
7056  aad5  48              pha
7057  aad6  a9 01           lda #$01
7058  aad8  20 11 a8        jsr strspa      adr./länge des string ablegen
7059  aadb  20 69 ba        jsr mapdst      umsch. auf bank (# in $73)
7060  aade  68              pla
7061  aadf  a0 00           ldy #$00
7062  aae1  91 71           sta (facexp),y  fac #1: exponent
7063  aae3  20 8c ba        jsr mapusr      umsch. auf bank # 1
7064  aae6  68              pla
7065  aae7  68              pla
7066  aae8  4c 6c a8        jmp putnew
7067  aaeb
7068  aaeb
7069  aaeb  ===> basic-routine 'left$' <===
7070  aaeb
7071  aaeb  20 70 ab  leftd  jsr pream       stringadresse u. parameter vom
                                             stapel holen
7072  aaee  48              pha
7073  aaef  20 6e ba        jsr mapdsp      umsch. auf bank (# in $60)
7074  aaf2  b1 5e           lda (dscpnt),y  addrl zeiger string
                                             descr./temp.fac#3
7075  aaf4  8d 5b 02        sta ldaadr      addrl modifiable adresse
7076  aaf7  68              pla
7077  aaf8  cd 5b 02        cmp ldaadr      addrl modifiable adresse
7078  aafb  98              tya
7079  aafc  90 04     rleft  bcc rleft1
7080  aafe  b1 5e           lda (dscpnt),y  addrl zeiger string
                                             descr./temp.fac#3
7081  ab00  aa              tax
7082  ab01  98              tya
7083  ab02  20 8c ba  rleft1 jsr mapusr      umsch. auf bank # 1
7084  ab05  48              pha
7085  ab06  8a        rleft2 txa
7086  ab07  48        rleft3 pha
7087  ab08  20 11 a8        jsr strspa      adr./länge des string ablegen
7088  ab0b  20 c8 aa        jsr sav47
7089  ab0e  68              pla
7090  ab0f  a8              tay
7091  ab10  68              pla
7092  ab11  20 18 ab        jsr sav15
7093  ab14  98              tya
7094  ab15  4c 69 a8        jmp glar
7095  ab18
7096  ab18
7097  ab18  18        sav15  clc
7098  ab19  65 22     sav14  adc index1      addrl indirekter index #1
7099  ab1b  85 22            sta index1      addrl indirekter index #1
7100  ab1d  90 02            bcc sav16
7101  ab1f  e6 23            inc index1+1    addrh indirekter index #1
7102  ab21  60        sav16  rts
7103  ab22
7104  ab22
7105  ab22  ===> basic-routine 'right$' <===
7106  ab22
7107  ab22  20 70 ab  rightd jsr pream       stringadresse u. parameter vom
                                             stapel holen
7108  ab25  48              pha
```

```
zeile adr.  obj.-code   source-code

7109  ab26  20 6e ba          jsr mapdsp       umsch. auf bank (# in $60)
7110  ab29  b1 5e             lda (dscpnt),y   addrl zeiger string
                                                descr./temp.fac#3
7111  ab2b  8d 5b 02          sta ldaadr       addrl modifiable adresse
7112  ab2e  68                pla
7113  ab2f  18                clc
7114  ab30  ed 5b 02          sbc ldaadr       addrl modifiable adresse
7115  ab33  49 ff             eor #$ff
7116  ab35  4c fc aa          jmp rleft
7117  ab38
7118  ab38
7119  ab38  20 29 ba  sav17   jsr chrgot       letztes zeichen erneut nach ac
                                                (indirekter sprung)
7120  ab3b  c9 29             cmp #$29
7121  ab3d  f0 4e             beq sav18
7122  ab3f  4c cd b4          jmp combyt       kommatest, holt 1byte in xr
7123  ab42
7124  ab42
7125  ab42  ===> basic-routine 'mid$' <===
7126  ab42
7127  ab42  a9 ff     midd    lda #$ff
7128  ab44  85 75             sta faclo        fac #1: mantisse
7129  ab46  20 38 ab          jsr sav17
7130  ab49  20 70 ab          jsr pream        stringadresse u. parameter vom
                                                stapel holen
7131  ab4c  f0 5d             beq gofuc
7132  ab4e  ca                dex
7133  ab4f  8a                txa
7134  ab50  48                pha
7135  ab51  48                pha
7136  ab52  18                clc
7137  ab53  a2 00             ldx #$00
7138  ab55  20 6e ba          jsr mapdsp       umsch. auf bank (# in $60)
7139  ab58  b1 5e             lda (dscpnt),y   addrl zeiger string
                                                descr./temp.fac#3
7140  ab5a  20 8c ba          jsr mapusr       umsch. auf bank # 1
7141  ab5d  8d 5b 02          sta ldaadr       addrl modifiable adresse
7142  ab60  68                pla
7143  ab61  ed 5b 02          sbc ldaadr       addrl modifiable adresse
7144  ab64  b0 a0             bcs rleft2
7145  ab66  49 ff             eor #$ff
7146  ab68  c5 75             cmp faclo        fac #1: mantisse
7147  ab6a  90 9b             bcc rleft3
7148  ab6c  a5 75             lda faclo        fac #1: mantisse
7149  ab6e  b0 97             bcs rleft3
7150  ab70
7151  ab70
7152  ab70  ===> stringadresse u. parameter vom stapel holen <===
7153  ab70
7154  ab70  20 2a 97  pream   jsr chkcls       test')', sonst fehler + ready
7155  ab73  68                pla
7156  ab74  a8                tay
7157  ab75  68                pla
7158  ab76  85 62             sta jmper+1      addrl sprung zu functions-rout.
7159  ab78  68                pla
7160  ab79  68                pla
7161  ab7a  68                pla
7162  ab7b  aa                tax
```

```
zeile adr.  obj.-code   source-code

7163  ab7c  68              pla
7164  ab7d  85 5e           sta dscpnt      addrl zeiger string
                                            descr./temp.fac#3
7165  ab7f  68              pla
7166  ab80  85 5f           sta dscpnt+1    addrh zeiger string
                                            descr./temp.fac#3
7167  ab82  68              pla
7168  ab83  85 60           sta dscpnt+2    bank zeiger string descriptor
7169  ab85  a5 62           lda jmper+1     addrl sprung zu functions-rout.
7170  ab87  48              pha
7171  ab88  98              tya
7172  ab89  48              pha
7173  ab8a  a0 00           ldy #$00
7174  ab8c  8a              txa
7175  ab8d  60       sav18  rts
7176  ab8e
7177  ab8e
7178  ab8e  ===> basic-routine 'len' <===
7179  ab8e
7180  ab8e  20 94 ab  len    jsr len1
7181  ab91  4c 37 9d         jmp sngflt
7182  ab94
7183  ab94
7184  ab94  20 e6 a8  len1   jsr frestr      alten string entfernen
7185  ab97  a2 00            ldx #$00
7186  ab99  86 11            stx valtyp      flag für variablen typ (0=num,
                                            1=string)
7187  ab9b  a8               tay
7188  ab9c  60               rts
7189  ab9d
7190  ab9d
7191  ab9d  ===> basic-routine 'asc' <===
7192  ab9d
7193  ab9d  20 94 ab  asc    jsr len1
7194  aba0  f0 09            beq gofuc
7195  aba2  a0 00            ldy #$00
7196  aba4  20 df b0         jsr sav12
7197  aba7  a8               tay
7198  aba8  4c 37 9d         jmp sngflt
7199  abab
7200  abab
7201  abab  4c a7 9b  gofuc  jmp fcerr       ausgabe '?illegal quantity error',
                                            ready
7202  abae
7203  abae
7204  abae  ===> basic-routine 'val' <===
7205  abae
7206  abae  20 94 ab  val    jsr len1
7207  abb1  d0 03            bne val1        string auswerten
7208  abb3  4c da 9e         jmp zerofc      fac auf 00 setzen
7209  abb6
7210  abb6
7211  abb6  ===> string auswerten <===
7212  abb6
7213  abb6  a2 02     val1   ldx #$02
7214  abb8  b4 85     tryg   ldy txtptr,x    addrl zeiger auf akt. term
7215  abba  94 82            sty fbufpt,x    addrl zeiger fac-puffer (fout-rout.)
7216  abbc  ca               dex
```

```
zeile adr.  obj.-code   source-code

7217  abbd  10 f9           bpl tryg
7218  abbf  a6 24           ldx index1+2     bank indirekter index #1
7219  abc1  86 87           stx txtptr+2     bank zeiger auf akt. term
7220  abc3  86 01           stx i6509        6509 indirection register
7221  abc5  a6 22           ldx index1       addrl indirekter index #1
7222  abc7  86 85           stx txtptr       addrl zeiger auf akt. term
7223  abc9  18              clc
7224  abca  65 22           adc index1       addrl indirekter index #1
7225  abcc  85 25           sta index2       addrl indirekter index #2
7226  abce  a6 23           ldx index1+1     addrh indirekter index #1
7227  abd0  86 86           stx txtptr+1     addrh zeiger auf akt. term
7228  abd2  90 01           bcc val2
7229  abd4  e8              inx
7230  abd5  86 26     val2  stx index2+1     addrh indirekter index #2
7231  abd7  a0 00           ldy #$00
7232  abd9  b1 25           lda (index2),y   addrl indirekter index #2
7233  abdb  48              pha
7234  abdc  98              tya
7235  abdd  91 25           sta (index2),y   addrl indirekter index #2
7236  abdf  20 8c ba        jsr mapusr       umsch. auf bank # 1
7237  abe2  20 29 ba        jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
7238  abe5  20 d8 a2        jsr fin          umw. zahlenstring in gk-zahl
7239  abe8  20 7d ba        jsr mapstr       umsch. auf bank # 2
7240  abeb  68              pla
7241  abec  a0 00           ldy #$00
7242  abee  91 25           sta (index2),y   addrl indirekter index #2
7243  abf0  20 8c ba        jsr mapusr       umsch. auf bank # 1
7244  abf3  a6 82     st2txt ldx fbufpt      addrl zeiger fac-puffer (fout-rout.)
7245  abf5  a4 83           ldy fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
7246  abf7  a5 84           lda fbufpt+2     bank zeiger fac-puffer (fout-rout.)
7247  abf9  86 85           stx txtptr       addrl zeiger auf akt. term
7248  abfb  84 86           sty txtptr+1     addrh zeiger auf akt. term
7249  abfd  85 87           sta txtptr+2     bank zeiger auf akt. term
7250  abff  60              rts
7251  ac00
7252  ac00
7253  ac00  ===> basic-routine 'err$  (x)' <===
7254  ac00
7255  ac00  68        errd  pla
7256  ac01  20 24 97        jsr parchk       test '(' ')', sonst fehler, ready
7257  ac04  20 04 b5        jsr chknum       prüfen, ob numerische variable
7258  ac07  20 d9 b4        jsr conint       zahl < 256 aus akt. text nach xr
7259  ac0a  8a              txa
7260  ac0b  0a              asl a
7261  ac0c  c9 56           cmp #$56
7262  ac0e  b0 9b           bcs gofuc
7263  ac10  a8              tay
7264  ac11  b9 8f 82        lda ebase,y      ind. sprungtabelle basic-/system
                                             fehlermeldungen
7265  ac14  85 25           sta index2       addrl indirekter index #2
7266  ac16  b9 90 82        lda ebase+1,y    ind. sprungtabelle basic-/system
                                             fehlermeldungen
7267  ac19  85 26           sta index2+1     addrh indirekter index #2
7268  ac1b  20 78 ba        jsr mapsys       umsch. auf bank # 15
7269  ac1e  a0 ff           ldy #$ff
7270  ac20  a2 00           ldx #$00
7271  ac22  c8        erflp1 iny
```

```
zeile adr.  obj.-code   source-code

7272  ac23  b1 25           lda (index2),y   addrl indirekter index #2
7273  ac25  f0 07           beq errd1
7274  ac27  c9 20           cmp #$20
7275  ac29  90 f7           bcc erflp1
7276  ac2b  e8              inx
7277  ac2c  d0 f4           bne erflp1
7278  ac2e  8a        errd1 txa
7279  ac2f  20 11 a8        jsr strspa       adr./länge des string ablegen
7280  ac32  a0 ff           ldy #$ff
7281  ac34  a2 00           ldx #$00
7282  ac36  c8        erflp2 iny
7283  ac37  20 78 ba        jsr mapsys       umsch. auf bank # 15
7284  ac3a  b1 25           lda (index2),y   addrl indirekter index #2
7285  ac3c  f0 18           beq errd2
7286  ac3e  c9 20           cmp #$20
7287  ac40  90 f4           bcc erflp2
7288  ac42  8c 5b 02        sty ldaadr       addrl modifiable adresse
7289  ac45  48              pha
7290  ac46  8a              txa
7291  ac47  a8              tay
7292  ac48  68              pla
7293  ac49  20 69 ba        jsr mapdst       umsch. auf bank (# in $73)
7294  ac4c  91 71           sta (facexp),y   fac #1: exponent
7295  ac4e  98              tya
7296  ac4f  aa              tax
7297  ac50  ac 5b 02        ldy ldaadr       addrl modifiable adresse
7298  ac53  e8              inx
7299  ac54  d0 e0           bne erflp2
7300  ac56  20 8c ba  errd2 jsr mapusr       umsch. auf bank # 1
7301  ac59  4c 6c a8        jmp putnew
7302  ac5c
7303  ac5c
7304  ac5c  ===> interne uhrzeit nach ti$ <===
7305  ac5c
7306  ac5c  20 de ff  gettim jsr krdtim       interne uhr ablesen
7307  ac5f  48              pha
7308  ac60  29 7f           and #$7f
7309  ac62  85 07           sta tmsec        zeit 'sekunden' für ti$-berechnung
7310  ac64  98              tya
7311  ac65  29 9f           and #$9f
7312  ac67  08              php
7313  ac68  29 1f           and #$1f
7314  ac6a  c9 12           cmp #$12
7315  ac6c  d0 02           bne getti1
7316  ac6e  a9 00           lda #$00
7317  ac70  28        getti1 plp
7318  ac71  10 07           bpl lkt50
7319  ac73  78              sei
7320  ac74  f8              sed
7321  ac75  18              clc
7322  ac76  69 12           adc #$12
7323  ac78  d8              cld
7324  ac79  58              cli
7325  ac7a  85 05     lkt50 sta tmhour       zeit 'stunden' für ti$-berechnung
7326  ac7c  a9 00           lda #$00
7327  ac7e  85 08           sta tmten        zeit '1/10-sek.' für ti$-berechnung
7328  ac80  98              tya
7329  ac81  2a              rol a
```

```
zeile adr.  obj.-code   source-code

7330  ac82  2a              rol a
7331  ac83  26 08           rol tmten       zeit '1/10-sek.' für ti$-berechnung
7332  ac85  2a              rol a
7333  ac86  26 08           rol tmten       zeit '1/10-sek.' für ti$-berechnung
7334  ac88  8a              txa
7335  ac89  2a              rol a
7336  ac8a  26 08           rol tmten       zeit '1/10-sek.' für ti$-berechnung
7337  ac8c  4a              lsr a
7338  ac8d  85 06           sta tmmin       zeit 'minuten' für ti$-berechnung
7339  ac8f  68              pla
7340  ac90  2a              rol a
7341  ac91  26 08           rol tmten       zeit '1/10-sek.' für ti$-berechnung
7342  ac93  a9 08           lda #$08
7343  ac95  20 53 ad        jsr getspa      platz für string reservieren, länge
                                            in ac
7344  ac98  86 22           stx index1      addrl indirekter index #1
7345  ac9a  84 23           sty index1+1    addrh indirekter index #1
7346  ac9c  a0 02           ldy #$02
7347  ac9e  84 24           sty index1+2    bank indirekter index #1
7348  aca0  84 01           sty i6509       6509 indirection register
7349  aca2  a8              tay
7350  aca3  88              dey
7351  aca4  a9 00           lda #$00
7352  aca6  91 22           sta (index1),y  addrl indirekter index #1
7353  aca8  88              dey
7354  aca9  a5 08           lda tmten       zeit '1/10-sek.' für ti$-berechnung
7355  acab  18              clc
7356  acac  69 30           adc #$30
7357  acae  91 22           sta (index1),y  addrl indirekter index #1
7358  acb0  88              dey
7359  acb1  a2 02           ldx #$02
7360  acb3  b5 05     gti70 lda tmhour,x    zeit 'stunden' für ti$-berechnung
7361  acb5  48              pha
7362  acb6  29 0f           and #$0f
7363  acb8  18              clc
7364  acb9  69 30           adc #$30
7365  acbb  91 22           sta (index1),y  addrl indirekter index #1
7366  acbd  88              dey
7367  acbe  68              pla
7368  acbf  29 70           and #$70
7369  acc1  4a              lsr a
7370  acc2  4a              lsr a
7371  acc3  4a              lsr a
7372  acc4  4a              lsr a
7373  acc5  69 30           adc #$30
7374  acc7  91 22           sta (index1),y  addrl indirekter index #1
7375  acc9  88              dey
7376  acca  ca              dex
7377  accb  10 e6           bpl gti70
7378  accd  a5 22           lda index1      addrl indirekter index #1
7379  accf  a4 23           ldy index1+1    addrh indirekter index #1
7380  acd1  a6 24           ldx index1+2    bank indirekter index #1
7381  acd3  4c 1f a8        jmp strlit      sucht elemente von zeichenkette
7382  acd6
7383  acd6
7384  acd6  ===> interne uhr mit string stellen <===
7385  acd6
7386  acd6  20 e9 a8  mktime jsr frefac      stringverwaltung, freier string
```

```
zeile adr.  obj.-code   source-code

7387  acd9  48              pha
7388  acda  c9 06           cmp #$06
7389  acdc  f0 04           beq mktmb
7390  acde  c9 07           cmp #$07
7391  ace0  d0 6e           bne fcerr2
7392  ace2  a0 00     mktmb ldy #$00
7393  ace4  84 82           sty fbufpt      addrl zeiger fac-puffer (fout-rout.)
7394  ace6  20 3d ad  mkti10 jsr timnum
7395  ace9  0a              asl a
7396  acea  0a              asl a
7397  aceb  0a              asl a
7398  acec  0a              asl a
7399  aced  99 05 00        sta tmhour,y    zeit 'stunden' für ti$-berechnung
7400  acf0  20 3d ad        jsr timnum
7401  acf3  19 05 00        ora tmhour,y    zeit 'stunden' für ti$-berechnung
7402  acf6  99 05 00        sta tmhour,y    zeit 'stunden' für ti$-berechnung
7403  acf9  c8              iny
7404  acfa  c0 03           cpy #$03
7405  acfc  d0 e8           bne mkti10
7406  acfe  68              pla
7407  acff  c9 06           cmp #$06
7408  ad01  f0 05           beq mktmc
7409  ad03  20 3d ad        jsr timnum
7410  ad06  d0 02           bne mktmd
7411  ad08  a9 00     mktmc lda #$00
7412  ad0a  85 08     mktmd sta tmten       zeit '1/10-sek.' für ti$-berechnung
7413  ad0c  a5 05           lda tmhour      zeit 'stunden' für ti$-berechnung
7414  ad0e  c9 12           cmp #$12
7415  ad10  90 0a           bcc mkti50
7416  ad12  78              sei
7417  ad13  f8              sed
7418  ad14  e9 12           sbc #$12
7419  ad16  d8              cld
7420  ad17  58              cli
7421  ad18  09 80           ora #$80
7422  ad1a  85 05           sta tmhour      zeit 'stunden' für ti$-berechnung
7423  ad1c  a9 00     mkti50 lda #$00
7424  ad1e  66 08           ror tmten       zeit '1/10-sek.' für ti$-berechnung
7425  ad20  6a              ror a
7426  ad21  05 07           ora tmsec       zeit 'sekunden' für ti$-berechnung
7427  ad23  48              pha
7428  ad24  a9 00           lda #$00
7429  ad26  66 08           ror tmten       zeit '1/10-sek.' für ti$-berechnung
7430  ad28  6a              ror a
7431  ad29  05 06           ora tmmin       zeit 'minuten' für ti$-berechnung
7432  ad2b  aa              tax
7433  ad2c  a9 00           lda #$00
7434  ad2e  66 08           ror tmten       zeit '1/10-sek.' für ti$-berechnung
7435  ad30  6a              ror a
7436  ad31  66 08           ror tmten       zeit '1/10-sek.' für ti$-berechnung
7437  ad33  6a              ror a
7438  ad34  4a              lsr a
7439  ad35  05 05           ora tmhour      zeit 'stunden' für ti$-berechnung
7440  ad37  a8              tay
7441  ad38  68              pla
7442  ad39  18              clc
7443  ad3a  4c db ff        jmp ksttim      interne uhr stellen
7444  ad3d
```

```
zeile adr.  obj.-code   source-code

7445  ad3d
7446  ad3d  84 83     timnum sty fbufpt+1     addrh zeiger fac-puffer (fout-rout.)
7447  ad3f  a4 82            ldy fbufpt      addrl zeiger fac-puffer (fout-rout.)
7448  ad41  e6 82            inc fbufpt      addrl zeiger fac-puffer (fout-rout.)
7449  ad43  20 df b0         jsr sav12
7450  ad46  20 50 ba         jsr qnum        test inhalt ac numerisch
7451  ad49  b0 05            bcs fcerr2
7452  ad4b  e9 2f            sbc #$2f
7453  ad4d  a4 83            ldy fbufpt+1    addrh zeiger fac-puffer (fout-rout.)
7454  ad4f  60               rts
7455  ad50
7456  ad50
7457  ad50  4c a7 9b  fcerr2 jmp fcerr       ausgabe '?illegal quantity error',
                                             ready
7458  ad53
7459  ad53
7460  ad53  ===> platz für string reservieren, länge in ac <===
7461  ad53
7462  ad53  46 13     getspa lsr dores       flag für token o. ascii/garbage-flag
7463  ad55  aa        tryag2 tax
7464  ad56  f0 4c            beq getrts
7465  ad58  48               pha
7466  ad59  a5 3b            lda fretop      addrl top of string free space
7467  ad5b  38               sec
7468  ad5c  e9 03            sbc #$03
7469  ad5e  a4 3c            ldy fretop+1    addrh top of string free space
7470  ad60  b0 03            bcs tryag3
7471  ad62  f0 41            beq garbag      garbage collect (ungültige strings
                                             entfernen)
7472  ad64  88               dey
7473  ad65  85 22     tryag3 sta index1      addrl indirekter index #1
7474  ad67  84 23            sty index1+1    addrh indirekter index #1
7475  ad69  8a               txa
7476  ad6a  49 ff            eor #$ff
7477  ad6c  38               sec
7478  ad6d  65 22            adc index1      addrl indirekter index #1
7479  ad6f  b0 0a            bcs tryag4
7480  ad71  8d 5b 02         sta ldaadr      addrl modifiable adresse
7481  ad74  98               tya
7482  ad75  f0 2e            beq garbag      garbage collect (ungültige strings
                                             entfernen)
7483  ad77  ad 5b 02         lda ldaadr      addrl modifiable adresse
7484  ad7a  88               dey
7485  ad7b  c4 3a     tryag4 cpy strend+1    addrh ende benutzter ram-bereich
7486  ad7d  90 26            bcc garbag      garbage collect (ungültige strings
                                             entfernen)
7487  ad7f  d0 04            bne tryag5
7488  ad81  c5 39            cmp strend      addrl ende benutzter ram-bereich
7489  ad83  90 20            bcc garbag      garbage collect (ungültige strings
                                             entfernen)
7490  ad85  85 3d     tryag5 sta frespc      addrl zeiger auf neuen string
7491  ad87  84 3e            sty frespc+1    addrh zeiger auf neuen string
7492  ad89  20 7d ba         jsr mapstr      umsch. auf bank # 2
7493  ad8c  a0 02            ldy #$02
7494  ad8e  a9 ff            lda #$ff
7495  ad90  91 22            sta (index1),y  addrl indirekter index #1
7496  ad92  88               dey
7497  ad93  91 22            sta (index1),y  addrl indirekter index #1
```

```
zeile adr.  obj.-code   source-code

7498  ad95  88              dey
7499  ad96  68              pla
7500  ad97  91 22           sta (index1),y   addrl indirekter index #1
7501  ad99  20 8c ba        jsr mapusr       umsch. auf bank # 1
7502  ad9c  a6 3d           ldx frespc       addrl zeiger auf neuen string
7503  ad9e  a4 3e           ldy frespc+1     addrh zeiger auf neuen string
7504  ada0  86 3b           stx fretop       addrl top of string free space
7505  ada2  84 3c           sty fretop+1     addrh top of string free space
7506  ada4  60       getrts rts
7507  ada5
7508  ada5
7509  ada5  ===> garbage collect (ungültige strings entfernen) <===
7510  ada5
7511  ada5  a5 13    garbag lda dores        flag für token o. ascii/garbage-flag
7512  ada7  30 09           bmi grbg99
7513  ada9  20 b5 ad        jsr garba2       garbage collect (müll entfernen)
7514  adac  38              sec
7515  adad  66 13           ror dores        flag für token o. ascii/garbage-flag
7516  adaf  68              pla
7517  adb0  d0 a3           bne tryag2
7518  adb2  4c 50 85 grbg99 jmp omerr        ausgabe '?out of memory error',
                                             ready
7519  adb5
7520  adb5
7521  adb5  ===> garbage collect (müll entfernen) <===
7522  adb5
7523  adb5  a9 02    garba2 lda #$02
7524  adb7  85 01           sta i6509        6509 indirection register
7525  adb9  a5 1d           lda temmpt       zeiger auf zeitw. stringdescriptor
                                             (rel. offset)
7526  adbb  f0 1c           beq naa
7527  adbd  48       la     pha
7528  adbe  20 af ae        jsr slr1
7529  adc1  98              tya
7530  adc2  18              clc
7531  adc3  65 20           adc tempst       addrl für 3 temp.
                                             string-descriptoren
7532  adc5  a0 00           ldy #$00
7533  adc7  91 6a           sta (lowds),y    addrl zielanfang (verschieben)/
                                             temp.fac#2
7534  adc9  a5 21           lda tempst+1     addrh für 3 temp.
                                             string-descriptoren
7535  adcb  c8              iny
7536  adcc  91 6a           sta (lowds),y    addrl zielanfang (verschieben)/
                                             temp.fac#2
7537  adce  a9 02           lda #$02
7538  add0  c8              iny
7539  add1  91 6a           sta (lowds),y    addrl zielanfang (verschieben)/
                                             temp.fac#2
7540  add3  68              pla
7541  add4  38              sec
7542  add5  e9 04           sbc #$04
7543  add7  d0 e4           bne la
7544  add9  a0 00     naa    ldy #$00
7545  addb  84 64           sty highds       addrl zielende (verschieben)/
                                             temp.fac#1
7546  addd  a5 3f           lda memtop       addrl höchste ram-speicherstelle
7547  addf  a4 40           ldy memtop+1     addrh höchste ram-speicherstelle
```

```
zeile adr.  obj.-code   source-code

7548  ade1  85 6d           sta lowtr        addrl ursprunganfang (verschieben)/
                                            temp.fac#2
7549  ade3  85 5b           sta defpnt       addrl zeiger funkt.-def./ temp.fac#3
7550  ade5  85 3d           sta frespc       addrl zeiger auf neuen string
7551  ade7  84 6e           sty lowtr+1      addrh ursprunganfang (verschieben)
                                            temp.fac#2
7552  ade9  84 5c           sty defpnt+1     addrh zeiger funkt.-def./ temp.fac#3
7553  adeb  84 3e           sty frespc+1     addrh zeiger auf neuen string
7554  aded  20 56 ae  gloop  jsr chkgrb       trailer-test, ob string ungültig
7555  adf0  d0 0f           bne col01
7556  adf2  a0 00     col00a ldy #$00
7557  adf4  20 7d ba        jsr mapstr       umsch. auf bank # 2
7558  adf7  b1 5b           lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
7559  adf9  20 c8 ae        jsr movpnt       zeiger für garb. coll. stellen
7560  adfc  38              sec
7561  adfd  66 64           ror highds       addrl zielende (verschieben)/
                                            temp.fac#1
7562  adff  d0 ec           bne gloop
7563  ae01  24 64     col01  bit highds       addrl zielende (verschieben)/
                                            temp.fac#1
7564  ae03  10 38           bpl col03        gültige strings überspringen
7565  ae05  a2 00           ldx #$00
7566  ae07  86 64           stx highds       addrl zielende (verschieben)/
                                            temp.fac#1
7567  ae09
7568  ae09
7569  ae09  ===> gültigen string über ungültigen kopieren <===
7570  ae09
7571  ae09  a0 02     col02  ldy #$02
7572  ae0b  20 7d ba        jsr mapstr       umsch. auf bank # 2
7573  ae0e  b1 5b     col02a lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
7574  ae10  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                            temp.fac#2
7575  ae12  88              dey
7576  ae13  10 f9           bpl col02a
7577  ae15  20 43 ae        jsr sav7
7578  ae18  8a              txa
7579  ae19  a8              tay
7580  ae1a  20 7d ba        jsr mapstr       umsch. auf bank # 2
7581  ae1d  88        glop1  dey
7582  ae1e  b1 5b           lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
7583  ae20  91 6d           sta (lowtr),y    addrl ursprunganfang (verschieben)/
                                            temp.fac#2
7584  ae22  ca              dex
7585  ae23  d0 f8           bne glop1
7586  ae25  20 73 ba        jsr mapinx       umsch. auf bank (# in $24)
7587  ae28  a0 02           ldy #$02
7588  ae2a  b9 6c 00  col02b lda lowds+2,y    bank zielanfang (versch.)/
                                            exp.basis10
7589  ae2d  91 22           sta (index1),y   addrl indirekter index #1
7590  ae2f  88              dey
7591  ae30  d0 f8           bne col02b
7592  ae32  a5 5b           lda defpnt       addrl zeiger funkt.-def./ temp.fac#3
7593  ae34  a4 5c           ldy defpnt+1     addrh zeiger funkt.-def./ temp.fac#3
7594  ae36  20 56 ae        jsr chkgrb       trailer-test, ob string ungültig
7595  ae39  f0 b7           beq col00a
7596  ae3b  d0 cc           bne col02        gültigen string über ungültigen
                                            kopieren
```

```
zeile adr.  obj.-code   source-code


7597  ae3d
7598  ae3d
7599  ae3d  ===> gültige strings überspringen <===
7600  ae3d
7601  ae3d  20 43 ae  col03  jsr sav7
7602  ae40  4c ed ad         jmp gloop
7603  ae43
7604  ae43
7605  ae43  a0 00     sav7   ldy #$00
7606  ae45  20 73 ba         jsr mapinx       umsch. auf bank (# in $24)
7607  ae48  b1 22            lda (index1),y   addrl indirekter index #1
7608  ae4a  aa               tax
7609  ae4b  20 d7 ae         jsr movtop
7610  ae4e  85 3d            sta frespc       addrl zeiger auf neuen string
7611  ae50  84 3e            sty frespc+1     addrh zeiger auf neuen string
7612  ae52  8a               txa
7613  ae53  4c c8 ae         jmp movpnt       zeiger für garb. coll. stellen
7614  ae56
7615  ae56
7616  ae56  ===> trailer-test, ob string ungültig <===
7617  ae56
7618  ae56  c4 3c     chkgrb cpy fretop+1     addrh top of string free space
7619  ae58  90 2b            bcc cfre4
7620  ae5a  d0 06            bne cfre1
7621  ae5c  c5 3b            cmp fretop       addrl top of string free space
7622  ae5e  f0 25            beq cfre4
7623  ae60  90 23            bcc cfre4
7624  ae62  24 64     cfre1  bit highds       addrl zielende (verschieben)/
                                              temp.fac#1
7625  ae64  30 05            bmi cfre2
7626  ae66  a9 03            lda #$03
7627  ae68  20 d7 ae         jsr movtop
7628  ae6b  a9 03     cfre2  lda #$03
7629  ae6d  20 c8 ae         jsr movpnt       zeiger für garb. coll. stellen
7630  ae70  a0 02            ldy #$02
7631  ae72  20 7d ba         jsr mapstr       umsch. auf bank # 2
7632  ae75  b1 5b            lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
7633  ae77  c9 ff            cmp #$ff
7634  ae79  d0 01            bne cfre3
7635  ae7b  60               rts
7636  ae7c
7637  ae7c
7638  ae7c  b1 5b     cfre3  lda (defpnt),y   addrl zeiger funkt.-def./ temp.fac#3
7639  ae7e  99 22 00         sta index1,y     addrl indirekter index #1
7640  ae81  88               dey
7641  ae82  10 f8            bpl cfre3
7642  ae84  60               rts
7643  ae85
7644  ae85
7645  ae85  a5 1d     cfre4  lda temmpt       zeiger auf zeitw. stringdescriptor
                                              (rel. offset)
7646  ae87  f0 18            beq naa2         müll beseitigt, fretop eintragen
7647  ae89  48        fa     pha
7648  ae8a  20 af ae         jsr slr1
7649  ae8d  b1 20            lda (tempst),y   addrl für 3 temp.
                                              string-descriptoren
7650  ae8f  a0 00            ldy #$00
7651  ae91  91 6a            sta (lowds),y    addrl zielanfang (verschieben)/
                                              temp.fac#2
```

```
zeile adr.  obj.-code   source-code

7652  ae93  c8              iny
7653  ae94  a9 ff          lda #$ff
7654  ae96  91 6a          sta (lowds),y     addrl zielanfang (verschieben)/
                                              temp.fac#2
7655  ae98  c8              iny
7656  ae99  91 6a          sta (lowds),y     addrl zielanfang (verschieben)/
                                              temp.fac#2
7657  ae9b  68              pla
7658  ae9c  38              sec
7659  ae9d  e9 04           sbc #$04
7660  ae9f  d0 e8           bne fa
7661  aea1
7662  aea1
7663  aea1  ===> müll beseitigt, fretop eintragen <===
7664  aea1
7665  aea1  20 8c ba  naa2  jsr mapusr        umsch. auf bank # 1
7666  aea4  68              pla
7667  aea5  68              pla
7668  aea6  a5 3d           lda frespc        addrl zeiger auf neuen string
7669  aea8  a4 3e           ldy frespc+1      addrh zeiger auf neuen string
7670  aeaa  85 3b           sta fretop        addrl top of string free space
7671  aeac  84 3c           sty fretop+1      addrh top of string free space
7672  aeae  60              rts
7673  aeaf
7674  aeaf
7675  aeaf  a8        slr1  tay
7676  aeb0  88              dey
7677  aeb1  88              dey
7678  aeb2  b1 20           lda (tempst),y    addrl für 3 temp.
                                              string-descriptoren
7679  aeb4  85 6b           sta lowds+1       addrh zielanfang (versch.)/ bytes
                                              vor dez.punkt
7680  aeb6  88              dey
7681  aeb7  b1 20           lda (tempst),y    addrl für 3 temp.
                                              string-descriptoren
7682  aeb9  85 6a           sta lowds         addrl zielanfang (verschieben)/
                                              temp.fac#2
7683  aebb  88              dey
7684  aebc  b1 20           lda (tempst),y    addrl für 3 temp.
                                              string-descriptoren
7685  aebe  18              clc
7686  aebf  65 6a           adc lowds         addrl zielanfang (verschieben)/
                                              temp.fac#2
7687  aec1  85 6a           sta lowds         addrl zielanfang (verschieben)/
                                              temp.fac#2
7688  aec3  90 02           bcc slr2
7689  aec5  e6 6b           inc lowds+1       addrh zielanfang (versch.)/ bytes
                                              vor dez.punkt
7690  aec7  60        slr2  rts
7691  aec8
7692  aec8
7693  aec8  ===> zeiger für garb. coll. stellen <===
7694  aec8
7695  aec8  49 ff     movpnt eor #$ff
7696  aeca  38              sec
7697  aecb  65 5b           adc defpnt        addrl zeiger funkt.-def./ temp.fac#3
7698  aecd  a4 5c           ldy defpnt+1      addrh zeiger funkt.-def./ temp.fac#3
7699  aecf  b0 01           bcs mov00
```

```
zeile adr.  obj.-code   source-code

7700  aed1  88              dey
7701  aed2  85 5b     mov00  sta defpnt       addrl zeiger funkt.-def./ temp.fac#3
7702  aed4  84 5c            sty defpnt+1     addrh zeiger funkt.-def./ temp.fac#3
7703  aed6  60               rts
7704  aed7
7705  aed7
7706  aed7  49 ff     movtop eor #$ff
7707  aed9  38               sec
7708  aeda  65 6d            adc lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7709  aedc  a4 6e            ldy lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7710  aede  b0 01            bcs mov01
7711  aee0  88               dey
7712  aee1  85 6d     mov01  sta lowtr         addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7713  aee3  84 6e            sty lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7714  aee5  60               rts
7715  aee6
7716  aee6
7717  aee6  4c a7 9b  infcer jmp fcerr        ausgabe '?illegal quantity error',
                                              ready
7718  aee9
7719  aee9
7720  aee9  ===> basic-routine instring <===
7721  aee9
7722  aee9  a2 02     incop0 ldx #$02
7723  aeeb  b5 74     incop1 lda facmo,x       fac #1: mantisse
7724  aeed  9d a0 02         sta tmpdes,x     temp. speicher 'instr$'
7725  aef0  ca               dex
7726  aef1  10 f8            bpl incop1
7727  aef3  20 c1 95         jsr frmevl       auswertung beliebiger ausdruck
7728  aef6  20 06 b5         jsr chkstr       prüfen, ob stringvariable
7729  aef9  a2 02            ldx #$02
7730  aefb  b5 74     incop2 lda facmo,x      fac #1: mantisse
7731  aefd  9d a3 02         sta tmpdes+3,x   temp. speicher 'instr$'
7732  af00  ca               dex
7733  af01  10 f8            bpl incop2
7734  af03  a2 01            ldx #$01
7735  af05  86 75            stx faclo        fac #1: mantisse
7736  af07  20 38 ab         jsr sav17
7737  af0a  20 2a 97         jsr chkcls       test')', sonst fehler + ready
7738  af0d  a6 75            ldx faclo        fac #1: mantisse
7739  af0f  f0 d5            beq infcer
7740  af11  ca               dex
7741  af12  86 73            stx facmoh       fac #1: mantisse
7742  af14  a2 05            ldx #$05
7743  af16  bd a0 02  inst2  lda tmpdes,x     temp. speicher 'instr$'
7744  af19  95 64            sta highds,x     addrl zielende (verschieben)/
                                              temp.fac#1
7745  af1b  ca               dex
7746  af1c  10 f8            bpl inst2
7747  af1e  a5 66            lda highds+2     bank zielende (verschieben/
                                              temp.fac#1
7748  af20  85 01            sta i6509        6509 indirection register
7749  af22  a0 03            ldy #$03
7750  af24  b1 64     inst3  lda (highds),y   addrl zielende (verschieben)/
                                              temp.fac#1
```

```
zeile adr.  obj.-code   source-code

7751  af26  99 6a 00        sta lowds,y      addrl zielanfang (verschieben)/
                                              temp.fac#2
7752  af29  88              dey
7753  af2a  10 f8           bpl inst3
7754  af2c  a5 69           lda hightr+2     bank ursprungende (verschieben)
7755  af2e  85 01           sta i6509        6509 indirection register
7756  af30  a0 03           ldy #$03
7757  af32  b1 67     inst4 lda (hightr),y   addrl ursprungende (verschieben)/
                                              temp.fac#1
7758  af34  99 6e 00        sta lowtr+1,y    addrh ursprunganfang (verschieben)
                                              temp.fac#2
7759  af37  88              dey
7760  af38  10 f8           bpl inst4
7761  af3a  a5 6e           lda lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7762  af3c  f0 39           beq instnf
7763  af3e  a5 6d           lda lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7764  af40  85 01           sta i6509        6509 indirection register
7765  af42  a9 00     inst5 lda #$00
7766  af44  85 74           sta facmo        fac #1: mantisse
7767  af46  18              clc
7768  af47  a5 6e           lda lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7769  af49  65 73           adc facmoh       fac #1: mantisse
7770  af4b  b0 2a           bcs instnf
7771  af4d  c5 6a           cmp lowds        addrl zielanfang (verschieben)/
                                              temp.fac#2
7772  af4f  90 02           bcc inst6
7773  af51  d0 24           bne instnf
7774  af53  a4 74     inst6 ldy facmo        fac #1: mantisse
7775  af55  c4 6e           cpy lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7776  af57  f0 19           beq instfd
7777  af59  98              tya
7778  af5a  18              clc
7779  af5b  65 73           adc facmoh       fac #1: mantisse
7780  af5d  a8              tay
7781  af5e  b1 6b           lda (lowds+1),y  addrh zielanfang (versch.)/ bytes
                                              vor dez.punkt
7782  af60  85 72           sta facho        fac #1: mantisse
7783  af62  a4 74           ldy facmo        fac #1: mantisse
7784  af64  b1 6f           lda (lowtr+2),y  bank ursprunganfang (verschieben)/
                                              vorz.exp.
7785  af66  c5 72           cmp facho        fac #1: mantisse
7786  af68  f0 04           beq inst7
7787  af6a  e6 73           inc facmoh       fac #1: mantisse
7788  af6c  d0 d4           bne inst5
7789  af6e  e6 74     inst7 inc facmo        fac #1: mantisse
7790  af70  d0 e1           bne inst6
7791  af72  e6 73     instfd inc facmoh      fac #1: mantisse
7792  af74  a5 73           lda facmoh       fac #1: mantisse
7793  af76  2c              .byte $2c
7794  af77  a9 00     instnf lda #$00
7795  af79  48              pha
7796  af7a  ad a3 02        lda tmpdes+3     temp. speicher 'instr$'
7797  af7d  ac a4 02        ldy tmpdes+4     temp. speicher 'instr$'
7798  af80  ae a5 02        ldx tmpdes+5     temp. speicher 'instr$'
```

```
zeile adr.  obj.-code   source-code


7799  af83  20 ef a8        jsr fretmp
7800  af86  ad a0 02        lda tmpdes        temp. speicher 'instr$'
7801  af89  ac a1 02        ldy tmpdes+1      temp. speicher 'instr$'
7802  af8c  ae a2 02        ldx tmpdes+2      temp. speicher 'instr$'
7803  af8f  20 ef a8        jsr fretmp
7804  af92  20 8c ba        jsr mapusr        umsch. auf bank # 1
7805  af95  68              pla
7806  af96  a8              tay
7807  af97  4c 37 9d        jmp sngflt
7808  af9a
7809  af9a                  .end
7810  af9a                  .lib delete
```

```
zeile adr.  obj.-code   source-code

7812  af9a
7813  af9a  ===> ausgabe '?syntax error', ready <===
7814  af9a
7815  af9a  4c 4f 97  nrange jmp snerr        ausgabe '?syntax error', ready
7816  af9d
7817  af9d
7818  af9d  ===> basic-routine 'delete' <===
7819  af9d
7820  af9d  f0 fb     delete beq nrange       ausgabe '?syntax error', ready
7821  af9f  20 f4 af         jsr range        zeilenbereich einlesen, test gültig
7822  afa2  f0 f6            beq nrange       ausgabe '?syntax error', ready
7823  afa4  a5 6d            lda lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7824  afa6  a6 6e            ldx lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7825  afa8  85 25            sta index2       addrl indirekter index #2
7826  afaa  86 26            stx index2+1     addrh indirekter index #2
7827  afac  20 1f 87         jsr fndlin       startadr. von prg.zeile berechnen
7828  afaf  90 12            bcc del300
7829  afb1  a0 01            ldy #$01
7830  afb3  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7831  afb5  88               dey
7832  afb6  aa               tax
7833  afb7  d0 04            bne noteos
7834  afb9  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7835  afbb  f0 06            beq del300
7836  afbd  b1 6d     noteos lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7837  afbf  85 6d            sta lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7838  afc1  86 6e            stx lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7839  afc3  a5 25     del300 lda index2       addrl indirekter index #2
7840  afc5  38               sec
7841  afc6  e5 6d            sbc lowtr        addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7842  afc8  aa               tax
7843  afc9  a5 26            lda index2+1     addrh indirekter index #2
7844  afcb  e5 6e            sbc lowtr+1      addrh ursprunganfang (verschieben)
                                              temp.fac#2
7845  afcd  a8               tay
7846  afce  b0 21            bcs notdel
7847  afd0  8a               txa
7848  afd1  18               clc
7849  afd2  65 2f            adc txtend       addrl zeiger ende  basic-text
7850  afd4  85 2f            sta txtend       addrl zeiger ende  basic-text
7851  afd6  98               tya
7852  afd7  65 30            adc txtend+1     addrh zeiger ende  basic-text
7853  afd9  85 30            sta txtend+1     addrh zeiger ende  basic-text
7854  afdb  a0 00            ldy #$00
7855  afdd  20 8c ba  del500 jsr mapusr       umsch. auf bank # 1
7856  afe0  b1 6d            lda (lowtr),y    addrl ursprunganfang (verschieben)/
                                              temp.fac#2
7857  afe2  91 25            sta (index2),y   addrl indirekter index #2
7858  afe4  c8               iny
7859  afe5  d0 f6            bne del500
```

```
zeile adr.  obj.-code   source-code

7860  afe7  e6 6e            inc lowtr+1      addrh ursprunganfang (verschieben)
                                               temp.fac#2
7861  afe9  e6 26            inc index2+1     addrh indirekter index #2
7862  afeb  a5 30            lda txtend+1     addrh zeiger ende  basic-text
7863  afed  c5 26            cmp index2+1     addrh indirekter index #2
7864  afef  b0 ec            bcs del500
7865  aff1  4c 9b 86  notdel  jmp fini         linkadr. berechnen, ready
7866  aff4
7867  aff4
7868  aff4  ===> zeilenbereich einlesen, test gültig <===
7869  aff4
7870  aff4  90 09     range   bcc rng100
7871  aff6  f0 07             beq rng100
7872  aff8  c9 ab             cmp #$ab
7873  affa  f0 03             beq rng100
7874  affc  a9 00     rngerr  lda #$00
7875  affe  60                rts
7876  afff
7877  afff
7878  afff  20 8c ba  rng100  jsr mapusr       umsch. auf bank # 1
7879  b002  20 4e 8d          jsr linget       zeilennummer lesen, in adressformat
                                               umwandeln
7880  b005  20 1f 87          jsr fndlin       startadr. von prg.zeile berechnen
7881  b008  20 29 ba          jsr chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
7882  b00b  f0 0c             beq rng200
7883  b00d  c9 ab             cmp #$ab
7884  b00f  d0 eb             bne rngerr
7885  b011  20 26 ba          jsr chrget       nächstes zeichen nach ac (ind.jmp)
7886  b014  20 4e 8d          jsr linget       zeilennummer lesen, in adressformat
                                               umwandeln
7887  b017  d0 e3             bne rngerr
7888  b019  a5 1b     rng200  lda linnum       lo-byte akt. zeilennummer
7889  b01b  05 1c             ora linnum+1     hi-byte akt. zeilennummer
7890  b01d  d0 06             bne rngrts
7891  b01f  a9 ff             lda #$ff
7892  b021  85 1b             sta linnum       lo-byte akt. zeilennummer
7893  b023  85 1c             sta linnum+1     hi-byte akt. zeilennummer
7894  b025  60        rngrts  rts
7895  b026
7896  b026                    .end
7897  b026                    .lib using
```

```
zeile adr.  obj.-code   source-code

7899  b026
7900  b026  ===> basic-routine 'printusing' <===
7901  b026
7902  b026  a2 ff     using  ldx #$ff
7903  b028  8e 72 02         stx endfd        print using: zeiger auf ende feld
7904  b02b  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
7905  b02e  20 c1 95         jsr frmevl       auswertung beliebiger ausdruck
7906  b031  20 06 b5         jsr chkstr       prüfen, ob stringvariable
7907  b034  a5 74            lda facmo        fac #1: mantisse
7908  b036  48               pha
7909  b037  a5 75            lda faclo        fac #1: mantisse
7910  b039  48               pha
7911  b03a  a5 76            lda facsgn       fac #1: vorzeichen
7912  b03c  48               pha
7913  b03d  85 01            sta i6509        6509 indirection register
7914  b03f  a0 03            ldy #$03
7915  b041  b1 74            lda (facmo),y    fac #1: mantisse
7916  b043  aa               tax
7917  b044  88               dey
7918  b045
7919  b045
7920  b045  ===> pu-stringzeiger in formatzeiger <===
7921  b045
7922  b045  99 09 00  ldform sta form,y       addrl format-zeiger
7923  b048  b1 74            lda (facmo),y    fac #1: mantisse
7924  b04a  88               dey
7925  b04b  10 f8            bpl ldform       pu-stringzeiger in formatzeiger
7926  b04d  8d 71 02         sta lfor         print using: länge formatstrings
7927  b050  86 01            stx i6509        6509 indirection register
7928  b052  a8               tay
7929  b053  f0 0a            beq ser
7930  b055  88        cncj   dey
7931  b056  b1 09            lda (form),y     addrl format-zeiger
7932  b058  c9 23            cmp #$23
7933  b05a  f0 06            beq cscol        test ';' im printusing-string
7934  b05c  98               tya
7935  b05d  d0 f6            bne cncj
7936  b05f  4c 4f 97  ser    jmp snerr        ausgabe '?syntax error', ready
7937  b062
7938  b062
7939  b062  ===> test ';' im printusing-string <===
7940  b062
7941  b062  a9 3b     cscol  lda #$3b
7942  b064  20 32 97  eex2   jsr synchr       test:folgt ascii o. token im akt.
                                               text, sonst error
7943  b067  84 8b            sty parsts       dos analyse-byte 1
7944  b069  8c 5f 02         sty bnr          print using: zeiger auf beginn
7945  b06c  20 c1 95         jsr frmevl       auswertung beliebiger ausdruck
7946  b06f  24 11            bit valtyp       flag für variablen typ (0=num,
                                               1=string)
7947  b071  10 3b            bpl conv
7948  b073  20 a5 b2         jsr ini
7949  b076  20 ea b3         jsr anaf
7950  b079  ae 67 02         ldx chsn         print using: justify flag
7951  b07c  f0 16            beq prcha
7952  b07e  a2 00            ldx #$00
7953  b080  38               sec
7954  b081  ad 6d 02         lda cform        print using: zeichen zähler (feld)
```

```
zeile adr.  obj.-code   source-code

7955  b084  ed 5e 02        sbc hulp        print using: zähler
7956  b087  90 0b           bcc prcha
7957  b089  a2 3d           ldx #$3d
7958  b08b  ec 67 02        cpx chsn        print using: justify flag
7959  b08e  d0 03           bne schs1
7960  b090  4a              lsr a
7961  b091  69 00           adc #$00
7962  b093  aa        schs1 tax
7963  b094  a0 00     prcha ldy #$00
7964  b096  8a        chx   txa
7965  b097  f0 05           beq cpef
7966  b099  ca              dex
7967  b09a  a9 20     oblk  lda #$20
7968  b09c  d0 09           bne outc
7969  b09e  cc 5e 02  cpef  cpy hulp        print using: zähler
7970  b0a1  b0 f7           bcs oblk
7971  b0a3  20 df b0        jsr sav12
7972  b0a6  c8              iny
7973  b0a7  20 e3 b3  outc  jsr cdout       ausg. 1 pu-char, decrement zähler
7974  b0aa  d0 ea           bne chx
7975  b0ac  f0 0c           beq reay
7976  b0ae  20 e2 a3  conv  jsr fout        umw. gk-zahl in string (ablage ab
                                            $200)
7977  b0b1  20 e3 a7        jsr foutst
7978  b0b4  20 a5 b2        jsr ini
7979  b0b7  20 e7 b0        jsr fform
7980  b0ba  20 29 ba  reay  jsr chrgot      letztes zeichen erneut nach ac
                                            (indirekter sprung)
7981  b0bd  c9 2c           cmp #$2c
7982  b0bf  f0 a3           beq eex2
7983  b0c1  38              sec
7984  b0c2  66 8b           ror parsts      dos analyse-byte 1
7985  b0c4  20 ea b3        jsr anaf
7986  b0c7  68              pla
7987  b0c8  aa              tax
7988  b0c9  68              pla
7989  b0ca  a8              tay
7990  b0cb  68              pla
7991  b0cc  20 8c ba        jsr mapusr      umsch. auf bank # 1
7992  b0cf  20 ef a8        jsr fretmp
7993  b0d2  20 29 ba        jsr chrgot      letztes zeichen erneut nach ac
                                            (indirekter sprung)
7994  b0d5  c9 3b           cmp #$3b
7995  b0d7  f0 03           beq nocr
7996  b0d9  4c c8 8e        jmp ocrlf       ausgabe cr (+lf) auf akt. kanal
7997  b0dc
7998  b0dc
7999  b0dc  4c 26 ba  nocr  jmp chrget      nächstes zeichen nach ac (ind.jmp)
8000  b0df
8001  b0df
8002  b0df  20 73 ba  sav12 jsr mapinx      umsch. auf bank (# in $24)
8003  b0e2  b1 22           lda (index1),y  addrl indirekter index #1
8004  b0e4  4c 8c ba        jmp mapusr      umsch. auf bank # 1
8005  b0e7
8006  b0e7
8007  b0e7  ad 73 02  fform lda pufill      print using: füllzeichen
8008  b0ea  8d 6f 02        sta blfd        print using: blank/stern flag
8009  b0ed  a9 ff           lda #$ff
```

```
zeile adr.  obj.-code   source-code

8010  b0ef  8d 6e 02  ana    sta sno          print using: vorzeichen nummer
8011  b0f2  2c                .byte $2c
8012  b0f3  86 8c      stp    stx parstx       dos analyse-byte 2
8013  b0f5  cc 5e 02  insy   cpy hulp         print using: zähler
8014  b0f8  f0 33             beq eoa
8015  b0fa  b9 00 02          lda buf,y        basic input-puffer (-$2ff)
8016  b0fd  c8                iny
8017  b0fe  c9 20             cmp #$20
8018  b100  f0 f3             beq insy
8019  b102  c9 2d             cmp #$2d
8020  b104  f0 e9             beq ana
8021  b106  c9 2e             cmp #$2e
8022  b108  f0 e9             beq stp
8023  b10a  c9 45             cmp #$45
8024  b10c  f0 11             beq lsg
8025  b10e  9d 00 02          sta buf,x        basic input-puffer (-$2ff)
8026  b111  8e 60 02          stx enr          print using: zeiger auf ende
8027  b114  e8                inx
8028  b115  24 8c             bit parstx       dos analyse-byte 2
8029  b117  10 dc             bpl insy
8030  b119  ee 66 02          inc vn           print using: anzahl stellen vor
                                                dezimalpunkt
8031  b11c  4c f5 b0          jmp insy
8032  b11f
8033  b11f
8034  b11f  b9 00 02  lsg     lda buf,y        basic input-puffer (-$2ff)
8035  b122  c9 2d             cmp #$2d
8036  b124  d0 03             bne nomn
8037  b126  6e 64 02          ror usgn         print using: vorzeichen exponent
8038  b129  c8        nomn    iny
8039  b12a  8c 65 02          sty uexp         print using: zeiger auf exponent
8040  b12d  a5 8c      eoa    lda parstx       dos analyse-byte 2
8041  b12f  10 02             bpl rtts
8042  b131  86 8c             stx parstx       dos analyse-byte 2
8043  b133  20 ea b3  rtts    jsr anaf
8044  b136  ad 68 02          lda vf           print using: # position vor dez.pkt.
8045  b139  c9 ff             cmp #$ff
8046  b13b  f0 29             beq erst         ungültiger wert, feldausg. mit'*'
8047  b13d  ad 6b 02          lda fesp         print using: exponent flag (feld)
8048  b140  f0 3f             beq cff
8049  b142  ad 65 02          lda uexp         print using: zeiger auf exponent
8050  b145  d0 12             bne ete
8051  b147  ae 60 02          ldx enr          print using: zeiger auf ende
8052  b14a  20 71 b2          jsr et2
8053  b14d  de 02 02          dec buf+2,x      basic input-puffer (-$2ff) +2
8054  b150  e8                inx
8055  b151  8e 65 02          stx uexp         print using: zeiger auf exponent
8056  b154  20 02 b3          jsr alg
8057  b157  f0 25             beq hup
8058  b159  ac 6a 02  ete     ldy posp         print using: +/- flag (feld)
8059  b15c  d0 17             bne sswe
8060  b15e  ac 6e 02          ldy sno          print using: vorzeichen nummer
8061  b161  30 12             bmi sswe
8062  b163  ad 68 02          lda vf           print using: # position vor dez.pkt.
8063  b166
8064  b166
8065  b166  ===> ungültiger wert, feldausg. mit'*' <===
8066  b166
```

```
zeile adr.  obj.-code   source-code

8067  b166  f0 69      erst   beq errf          ausgabefeld mit '*' füllen
8068  b168  ce 68 02          dec vf            print using: # position vor dez.pkt.
8069  b16b  d0 05              bne rspa
8070  b16d  ad 69 02          lda nf            print using: # position nach
                                                dez.pkt.
8071  b170  f0 5f              beq errf          ausgabefeld mit '*' füllen
8072  b172  ee 63 02  rspa     inc swe           print using: zähler
8073  b175  20 e9 b1  sswe     jsr shpn
8074  b178  20 c0 b2          jsr uround        ausgabezahl runden
8075  b17b  20 e9 b1          jsr shpn
8076  b17e  4c 38 b3  hup      jmp chout
8077  b181
8078  b181
8079  b181  ac 65 02  cff      ldy uexp          print using: zeiger auf exponent
8080  b184  f0 16              beq ftf
8081  b186  8d 5e 02          sta hulp          print using: zähler
8082  b189  38                sec
8083  b18a  6e 6c 02          ror etof          print using: switch
8084  b18d  a4 8c              ldy parstx        dos analyse-byte 2
8085  b18f  ad 64 02          lda usgn          print using: vorzeichen exponent
8086  b192  10 05              bpl no4
8087  b194  20 23 b2          jsr nos3
8088  b197  f0 11              beq rndd
8089  b199  20 04 b2  no4      jsr nos4
8090  b19c  a4 8c      ftf     ldy parstx        dos analyse-byte 2
8091  b19e  f0 0a              beq rndd
8092  b1a0  20 06 b3          jsr cho
8093  b1a3  d0 05              bne rndd
8094  b1a5  ce 66 02          dec vn            print using: anzahl stellen vor
                                                dezimalpunkt
8095  b1a8  b0 03              bcs devn2
8096  b1aa  20 c0 b2  rndd     jsr uround        ausgabezahl runden
8097  b1ad  38        devn2    sec
8098  b1ae  ad 68 02          lda vf            print using: # position vor dez.pkt.
8099  b1b1  ed 66 02          sbc vn            print using: anzahl stellen vor
                                                dezimalpunkt
8100  b1b4  90 1b              bcc errf          ausgabefeld mit '*' füllen
8101  b1b6  8d 63 02          sta swe           print using: zähler
8102  b1b9  ac 6a 02          ldy posp          print using: +/- flag (feld)
8103  b1bc  d0 1b              bne ahp
8104  b1be  ac 6e 02          ldy sno           print using: vorzeichen nummer
8105  b1c1  30 16              bmi ahp
8106  b1c3  a8                tay
8107  b1c4  f0 0b              beq errf          ausgabefeld mit '*' füllen
8108  b1c6  88                dey
8109  b1c7  d0 13              bne ldvn
8110  b1c9  ad 69 02          lda nf            print using: # position nach
                                                dez.pkt.
8111  b1cc  0d 66 02          ora vn            print using: anzahl stellen vor
                                                dezimalpunkt
8112  b1cf  d0 ad              bne hup
8113  b1d1
8114  b1d1
8115  b1d1  ===> ausgabefeld mit '*' füllen <===
8116  b1d1
8117  b1d1  a9 2a      errf    lda #$2a
8118  b1d3  20 e3 b3   stout   jsr cdout         ausg. 1 pu-char, decrement zähler
8119  b1d6  d0 fb              bne stout
```

```
zeile adr.  obj.-code   source-code

8120  b1d8  60              rts
8121  b1d9
8122  b1d9
8123  b1d9  a8        ahp   tay
8124  b1da  f0 a2           beq hup
8125  b1dc  ad 66 02  ldvn  lda vn          print using: anzahl stellen vor
                                            dezimalpunkt
8126  b1df  d0 9d           bne hup
8127  b1e1  ce 63 02        dec swe         print using: zähler
8128  b1e4  e6 8b           inc parsts      dos analyse-byte 1
8129  b1e6  4c 7e b1        jmp hup
8130  b1e9
8131  b1e9
8132  b1e9  38        shpn  sec
8133  b1ea  ad 68 02        lda vf          print using: # position vor dez.pkt.
8134  b1ed  ed 66 02        sbc vn          print using: anzahl stellen vor
                                            dezimalpunkt
8135  b1f0  f0 3b           beq rdy
8136  b1f2  a4 8c           ldy parstx      dos analyse-byte 2
8137  b1f4  90 18           bcc pntl
8138  b1f6  8d 5e 02        sta hulp        print using: zähler
8139  b1f9  cc 60 02  incy  cpy enr         print using: zeiger auf ende
8140  b1fc  f0 02           beq ity
8141  b1fe  b0 01           bcs nos1
8142  b200  c8        ity   iny
8143  b201  ee 66 02  nos1  inc vn          print using: anzahl stellen vor
                                            dezimalpunkt
8144  b204  20 3a b2  nos4  jsr eado
8145  b207  ce 5e 02        dec hulp        print using: zähler
8146  b20a  d0 ed           bne incy
8147  b20c  f0 1d           beq poit
8148  b20e  49 ff     pntl  eor #$ff
8149  b210  69 01           adc #$01
8150  b212  8d 5e 02        sta hulp        print using: zähler
8151  b215  cc 5f 02  decy  cpy bnr         print using: zeiger auf beginn
8152  b218  f0 05           beq inz
8153  b21a  88              dey
8154  b21b  ce 66 02        dec vn          print using: anzahl stellen vor
                                            dezimalpunkt
8155  b21e  2c              .byte $2c
8156  b21f  e6 8b     inz   inc parsts      dos analyse-byte 1
8157  b221  a9 80           lda #$80
8158  b223  20 3c b2  nos3  jsr eadj
8159  b226  ce 5e 02        dec hulp        print using: zähler
8160  b229  d0 ea           bne decy
8161  b22b  84 8c     poit  sty parstx      dos analyse-byte 2
8162  b22d  60        rdy   rts
8163  b22e
8164  b22e
8165  b22e  d0 39     sexp  bne retrn
8166  b230  49 09           eor #$09
8167  b232  9d 00 02        sta buf,x       basic input-puffer (-$2ff)
8168  b235  ca              dex
8169  b236  ec 65 02        cpx uexp        print using: zeiger auf exponent
8170  b239  60              rts
8171  b23a
8172  b23a
8173  b23a  a9 00     eado  lda #$00
```

```
zeile adr.  obj.-code   source-code

8174  b23c  ae 65 02  eadj   ldx uexp         print using: zeiger auf exponent
8175  b23f  e8               inx
8176  b240  2c 6c 02         bit etof         print using: switch
8177  b243  30 10             bmi tag2
8178  b245  4d 64 02         eor usgn         print using: vorzeichen exponent
8179  b248  f0 0b             beq tag2
8180  b24a  20 7f b2  tag1   jsr tag3
8181  b24d  20 2e b2         jsr sexp
8182  b250  b0 f8             bcs tag1
8183  b252  4c 5e 9f         jmp overr        ausgabe '?overflow error', ready
8184  b255
8185  b255
8186  b255  bd 00 02  tag2   lda buf,x        basic input-puffer (-$2ff)
8187  b258  de 00 02         dec buf,x        basic input-puffer (-$2ff)
8188  b25b  c9 30             cmp #$30
8189  b25d  20 2e b2         jsr sexp
8190  b260  b0 f3             bcs tag2
8191  b262  2c 6c 02         bit etof         print using: switch
8192  b265  10 05             bpl et3
8193  b267  84 8c             sty parstx       dos analyse-byte 2
8194  b269  68        retrn  pla
8195  b26a  68               pla
8196  b26b  60               rts
8197  b26c
8198  b26c
8199  b26c  ad 64 02  et3    lda usgn         print using: vorzeichen exponent
8200  b26f  49 80             eor #$80
8201  b271  8d 64 02  et2    sta usgn         print using: vorzeichen exponent
8202  b274  a9 30             lda #$30
8203  b276  9d 01 02         sta buf+1,x      basic input-puffer (-$2ff) +1
8204  b279  a9 31             lda #$31
8205  b27b  9d 02 02         sta buf+2,x      basic input-puffer (-$2ff) +2
8206  b27e  60               rts
8207  b27f
8208  b27f
8209  b27f  bd 00 02  tag3   lda buf,x        basic input-puffer (-$2ff)
8210  b282  fe 00 02         inc buf,x        basic input-puffer (-$2ff)
8211  b285  c9 39             cmp #$39
8212  b287  60               rts
8213  b288
8214  b288
8215  b288  18        ansub  clc
8216  b289  c8               iny
8217  b28a  f0 05             beq ans010
8218  b28c  cc 71 02         cpy lfor         print using: länge formatstrings
8219  b28f  90 04             bcc ans020
8220  b291  a4 8b     ans010 ldy parsts       dos analyse-byte 1
8221  b293  d0 d4             bne retrn
8222  b295  20 9c b2  ans020 jsr sav1         bank formatstring nach i6509
8223  b298  ee 6d 02         inc cform        print using: zeichen zähler (feld)
8224  b29b  60               rts
8225  b29c
8226  b29c
8227  b29c  ===> bank formatstring nach i6509 <===
8228  b29c
8229  b29c  a5 0b     sav1   lda form+2       bank format-zeiger
8230  b29e  85 01             sta i6509        6509 indirection register
8231  b2a0  b1 09             lda (form),y     addr1 format-zeiger
```

```
zeile adr.  obj.-code   source-code

8232  b2a2  4c 8c ba          jmp mapusr       umsch. auf bank # 1
8233  b2a5
8234  b2a5
8235  b2a5  20 e9 a8  ini     jsr frefac       stringverwaltung, freier string
8236  b2a8  8d 5e 02          sta hulp         print using: zähler
8237  b2ab  a2 0a             ldx #$0a
8238  b2ad  a9 00             lda #$00
8239  b2af  9d 63 02  stz     sta swe,x        print using: zähler
8240  b2b2  ca                dex
8241  b2b3  10 fa             bpl stz
8242  b2b5  8e 62 02          stx flag         print using: komma flag
8243  b2b8  86 8c             stx parstx       dos analyse-byte 2
8244  b2ba  8e 61 02          stx dolr         print using: dollar flag
8245  b2bd  aa                tax
8246  b2be  a8                tay
8247  b2bf  60                rts
8248  b2c0
8249  b2c0
8250  b2c0  ===> ausgabezahl runden <===
8251  b2c0
8252  b2c0  18        uround clc
8253  b2c1  a5 8c             lda parstx       dos analyse-byte 2
8254  b2c3  6d 69 02          adc nf           print using: # position nach
                                               dez.pkt.
8255  b2c6  b0 39             bcs rrts
8256  b2c8  38                sec
8257  b2c9  e5 8b             sbc parsts       dos analyse-byte 1
8258  b2cb  90 34             bcc rrts
8259  b2cd  cd 60 02          cmp enr          print using: zeiger auf ende
8260  b2d0  f0 02             beq cbn
8261  b2d2  b0 2d             bcs rrts
8262  b2d4  cd 5f 02  cbn     cmp bnr          print using: zeiger auf beginn
8263  b2d7  90 28             bcc rrts
8264  b2d9  aa                tax
8265  b2da  bd 00 02          lda buf,x        basic input-puffer (-$2ff)
8266  b2dd  c9 35             cmp #$35
8267  b2df  90 20             bcc rrts
8268  b2e1  ec 5f 02  con1    cpx bnr          print using: zeiger auf beginn
8269  b2e4  f0 0a             beq add1         zahl <1 aufrunden auf '1'
8270  b2e6  ca                dex
8271  b2e7  20 7f b2          jsr tag3
8272  b2ea  8e 60 02          stx enr          print using: zeiger auf ende
8273  b2ed  f0 f2             beq con1
8274  b2ef  60                rts
8275  b2f0
8276  b2f0
8277  b2f0  ===> zahl <1 aufrunden auf '1' <===
8278  b2f0
8279  b2f0  a9 31     add1    lda #$31
8280  b2f2  9d 00 02          sta buf,x        basic input-puffer (-$2ff)
8281  b2f5  e8                inx
8282  b2f6  86 8c             stx parstx       dos analyse-byte 2
8283  b2f8  c6 8b             dec parsts       dos analyse-byte 1
8284  b2fa  10 05             bpl rrts
8285  b2fc  e6 8b             inc parsts       dos analyse-byte 1
8286  b2fe  ee 66 02          inc vn           print using: anzahl stellen vor
                                               dezimalpunkt
8287  b301  60        rrts    rts
```

```
zeile adr.  obj.-code   source-code

8288  b302
8289  b302
8290  b302  a4 8c     alg    ldy parstx       dos analyse-byte 2
8291  b304  f0 17            beq szer         '0' in zahl suchen
8292  b306  ac 5f 02  cho    ldy bnr          print using: zeiger auf beginn
8293  b309
8294  b309
8295  b309  ===> buffer mit '0' vergleichen <===
8296  b309
8297  b309  b9 00 02  cmo    lda buf,y        basic input-puffer (-$2ff)
8298  b30c  c9 30             cmp #$30
8299  b30e  60                rts
8300  b30f
8301  b30f
8302  b30f  e6 8c     nbr    inc parstx       dos analyse-byte 2
8303  b311  20 3a b2          jsr eado
8304  b314  ee 5f 02          inc bnr          print using: zeiger auf beginn
8305  b317  cc 60 02          cpy enr          print using: zeiger auf ende
8306  b31a  f0 e5             beq rrts
8307  b31c  c8                iny
8308  b31d
8309  b31d
8310  b31d  ===> '0' in zahl suchen <===
8311  b31d
8312  b31d  20 09 b3  szer   jsr cmo          buffer mit '0' vergleichen
8313  b320  f0 ed            beq nbr
8314  b322  60                rts
8315  b323
8316  b323
8317  b323  ec 60 02  lzer   cpx enr          print using: zeiger auf ende
8318  b326  f0 05             beq zout
8319  b328  e8        telx   inx
8320  b329  bd 00 02          lda buf,x        basic input-puffer (-$2ff)
8321  b32c  2c                .byte $2c
8322  b32d  a9 30     zout   lda #$30
8323  b32f  4e 62 02  outs   lsr flag         print using: komma flag
8324  b332  20 e3 b3  out    jsr cdout        ausg. 1 pu-char, decrement zähler
8325  b335  d0 0f             bne afrm
8326  b337  60                rts
8327  b338
8328  b338
8329  b338  ad 61 02  chout  lda dolr         print using: dollar flag
8330  b33b  30 02             bmi choutz
8331  b33d  e6 8b             inc parsts       dos analyse-byte 1
8332  b33f  ae 5f 02  choutz ldx bnr          print using: zeiger auf beginn
8333  b342  ca                dex
8334  b343  ac 70 02          ldy begfd        print using: zeiger auf anfang feld
8335  b346  20 9c b2  afrm   jsr sav1         bank formatstring nach i6509
8336  b349  c8                iny
8337  b34a  c9 2c             cmp #$2c
8338  b34c  d0 11             bne punt
8339  b34e  2c 62 02          bit flag         print using: komma flag
8340  b351  30 06             bmi bout
8341  b353  ad 74 02          lda pucoma       print using: kommazeichen
8342  b356  4c 32 b3          jmp out
8343  b359
8344  b359
8345  b359  ad 6f 02  bout   lda blfd         print using: blank/stern flag
```

```
zeile adr.  obj.-code   source-code

8346  b35c  4c 32 b3        jmp out
8347  b35f
8348  b35f
8349  b35f  c9 2e     punt   cmp #$2e
8350  b361  d0 1b            bne afplus
8351  b363  ad 75 02         lda pudot       print using: zeichen dez.punkt
8352  b366  4c 32 b3        jmp out
8353  b369
8354  b369
8355  b369  a5 8b     zerot  lda parsts      dos analyse-byte 1
8356  b36b  f0 b6            beq lzer
8357  b36d  c6 8b            dec parsts      dos analyse-byte 1
8358  b36f  ad 61 02         lda dolr        print using: dollar flag
8359  b372  30 b9            bmi zout
8360  b374  38               sec
8361  b375  6e 61 02         ror dolr        print using: dollar flag
8362  b378  ad 76 02         lda pumony      print using: währungszeichen
8363  b37b  4c 2f b3        jmp outs
8364  b37e
8365  b37e
8366  b37e  c9 2b     afplus cmp #$2b
8367  b380  f0 3b            beq ispl
8368  b382  c9 2d            cmp #$2d
8369  b384  f0 32            beq ispl1
8370  b386  c9 5e            cmp #$5e
8371  b388  d0 39            bne pndd
8372  b38a  a9 45            lda #$45
8373  b38c  20 e3 b3         jsr cdout       ausg. 1 pu-char, decrement zähler
8374  b38f  ac 65 02         ldy uexp        print using: zeiger auf exponent
8375  b392  20 09 b3         jsr cmo         buffer mit '0' vergleichen
8376  b395  d0 06            bne mint
8377  b397  c8               iny
8378  b398  20 09 b3         jsr cmo         buffer mit '0' vergleichen
8379  b39b  f0 07            beq post
8380  b39d  a9 2d     mint   lda #$2d
8381  b39f  2c 64 02         bit usgn        print using: vorzeichen exponent
8382  b3a2  30 02            bmi mout
8383  b3a4  a9 2b     post   lda #$2b
8384  b3a6  20 e3 b3  mout   jsr cdout       ausg. 1 pu-char, decrement zähler
8385  b3a9  ae 65 02         ldx uexp        print using: zeiger auf exponent
8386  b3ac  bd 00 02         lda buf,x       basic input-puffer (-$2ff)
8387  b3af  20 e3 b3         jsr cdout       ausg. 1 pu-char, decrement zähler
8388  b3b2  ac 72 02         ldy endfd       print using: zeiger auf ende feld
8389  b3b5  4c 28 b3        jmp telx
8390  b3b8
8391  b3b8
8392  b3b8  ad 6e 02  ispl1  lda sno         print using: vorzeichen nummer
8393  b3bb  30 9c            bmi bout
8394  b3bd  ad 6e 02  ispl   lda sno         print using: vorzeichen nummer
8395  b3c0  4c 32 b3        jmp out
8396  b3c3
8397  b3c3
8398  b3c3  ad 63 02  pndd   lda swe         print using: zähler
8399  b3c6  f0 a1            beq zerot
8400  b3c8  ce 63 02         dec swe         print using: zähler
8401  b3cb  d0 8c     pndx   bne bout
8402  b3cd  ad 6a 02         lda posp        print using: +/- flag (feld)
8403  b3d0  30 f9            bmi pndx
```

```
zeile adr.  obj.-code   source-code

8404  b3d2  20 9c b2  tat    jsr sav1         bank formatstring nach i6509
8405  b3d5  c9 2c             cmp #$2c
8406  b3d7  d0 df             bne ispl1
8407  b3d9  ad 6f 02          lda blfd         print using: blank/stern flag
8408  b3dc  20 e3 b3          jsr cdout        ausg. 1 pu-char, decrement zähler
8409  b3df  c8                iny
8410  b3e0  4c d2 b3          jmp tat
8411  b3e3
8412  b3e3
8413  b3e3  ===> ausg. 1 pu-char, decrement zähler <===
8414  b3e3
8415  b3e3  20 3a b5  cdout   jsr ochr         ausgabe 1 zeichen (in ac)
8416  b3e6  ce 6d 02          dec cform        print using: zeichen zähler (feld)
8417  b3e9  60        gooop   rts
8418  b3ea
8419  b3ea
8420  b3ea  ac 72 02  anaf    ldy endfd        print using: zeiger auf ende feld
8421  b3ed  20 88 b2  gfor    jsr ansub
8422  b3f0  20 9e b4          jsr comp
8423  b3f3  d0 14             bne pchar        ausgabe 1 pu-char aus ac
8424  b3f5  8c 70 02          sty begfd        print using: zeiger auf anfang feld
8425  b3f8  90 19             bcc ffoun
8426  b3fa  aa                tax
8427  b3fb  20 88 b2  sfar    jsr ansub
8428  b3fe  b0 05             bcs nono
8429  b400  20 a6 b4          jsr com1
8430  b403  f0 09             beq foun1
8431  b405  ac 70 02  nono    ldy begfd        print using: zeiger auf anfang feld
8432  b408  8a                txa
8433  b409
8434  b409
8435  b409  ===> ausgabe 1 pu-char aus ac <===
8436  b409
8437  b409  20 3a b5  pchar   jsr ochr         ausgabe 1 zeichen (in ac)
8438  b40c  90 df             bcc gfor
8439  b40e  b0 eb     foun1   bcs sfar
8440  b410  ac 70 02          ldy begfd        print using: zeiger auf anfang feld
8441  b413  a6 8b     ffoun   ldx parsts       dos analyse-byte 1
8442  b415  d0 d2             bne gooop
8443  b417  8e 6d 02          stx cform        print using: zeichen zähler (feld)
8444  b41a  88                dey
8445  b41b  ce 6d 02  hyo2    dec cform        print using: zeichen zähler (feld)
8446  b41e  20 88 b2  hyo     jsr ansub
8447  b421  b0 74             bcs efo
8448  b423  c9 2c             cmp #$2c
8449  b425  f0 f7             beq hyo
8450  b427  20 75 b4          jsr isp          test/ausführen '+','-'
8451  b42a  90 ef             bcc hyo2
8452  b42c  c9 2e             cmp #$2e
8453  b42e  d0 08             bne avf1
8454  b430  e8                inx
8455  b431  e0 02             cpx #$02
8456  b433  90 e9             bcc hyo
8457  b435  4c 4f 97  ero     jmp snerr        ausgabe '?syntax error', ready
8458  b438
8459  b438
8460  b438  20 aa b4  avf1    jsr com2
8461  b43b  d0 0b             bne llar         test/ausführen '$','exp','+','-'
```

```
zeile adr.  obj.-code   source-code

8462  b43d  90 03           bcc hyo1
8463  b43f  8d 67 02        sta chsn          print using: justify flag
8464  b442  fe 68 02  hyo1  inc vf,x          print using: # position vor dez.pkt.
8465  b445  4c 1e b4        jmp hyo
8466  b448
8467  b448
8468  b448  ===> test/ausführen '$','exp','+','-' <===
8469  b448
8470  b448  c9 24     llar  cmp #$24
8471  b44a  d0 0f           bne expo          test/ausführen 'exp','+','-'
8472  b44c  2c 61 02        bit dolr          print using: dollar flag
8473  b44f  10 f1           bpl hyo1
8474  b451  18              clc
8475  b452  6e 61 02        ror dolr          print using: dollar flag
8476  b455  ce 68 02        dec vf            print using: # position vor dez.pkt.
8477  b458  4c 42 b4        jmp hyo1
8478  b45b
8479  b45b
8480  b45b  ===> test/ausführen 'exp','+','-' <===
8481  b45b
8482  b45b  c9 5e     expo  cmp #$5e
8483  b45d  d0 16           bne isp           test/ausführen '+','-'
8484  b45f  a2 02           ldx #$02
8485  b461  20 88 b2  chkk  jsr ansub
8486  b464  b0 cf           bcs ero           sprung 'syntax error', ready
8487  b466  c9 5e           cmp #$5e
8488  b468  d0 cb           bne ero           sprung 'syntax error', ready
8489  b46a  ca              dex
8490  b46b  10 f4           bpl chkk
8491  b46d  ee 6b 02        inc fesp          print using: exponent flag (feld)
8492  b470  20 88 b2        jsr ansub
8493  b473  b0 22           bcs efo
8494  b475
8495  b475
8496  b475  ===> test/ausführen '+','-' <===
8497  b475
8498  b475  c9 2b     isp   cmp #$2b
8499  b477  d0 19           bne chom          test/ausführen '-'
8500  b479  ad 6e 02        lda sno           print using: vorzeichen nummer
8501  b47c  10 05           bpl spos
8502  b47e  a9 2b           lda #$2b
8503  b480  8d 6e 02        sta sno           print using: vorzeichen nummer
8504  b483  ad 6a 02  spos  lda posp          print using: +/- flag (feld)
8505  b486  d0 ad           bne ero           sprung 'syntax error', ready
8506  b488  6e 6a 02        ror posp          print using: +/- flag (feld)
8507  b48b  8c 72 02        sty endfd         print using: zeiger auf ende feld
8508  b48e  ee 6d 02        inc cform         print using: zeichen zähler (feld)
8509  b491  60              rts
8510  b492
8511  b492
8512  b492  ===> test/ausführen '-' <===
8513  b492
8514  b492  c9 2d     chom  cmp #$2d
8515  b494  f0 ed           beq spos
8516  b496  38              sec
8517  b497  8c 72 02  efo   sty endfd         print using: zeiger auf ende feld
8518  b49a  ce 72 02        dec endfd         print using: zeiger auf ende feld
8519  b49d  60              rts
```

```
zeile adr.  obj.-code   source-code


8520  b49e
8521  b49e
8522  b49e  c9 2b      comp    cmp #$2b
8523  b4a0  f0 15              beq rt
8524  b4a2  c9 2d              cmp #$2d
8525  b4a4  f0 11              beq rt
8526  b4a6  c9 2e      com1    cmp #$2e
8527  b4a8  f0 0d              beq rt
8528  b4aa  c9 3d      com2    cmp #$3d
8529  b4ac  f0 09              beq rt
8530  b4ae  c9 3e              cmp #$3e
8531  b4b0  f0 05              beq rt
8532  b4b2  c9 23              cmp #$23
8533  b4b4  d0 01              bne rt
8534  b4b6  18                 clc
8535  b4b7  60         rt      rts
8536  b4b8
8537  b4b8                     .end
8538  b4b8                     .lib butes1
```

```
zeile adr.  obj.-code   source-code

8540  b4b8
8541  b4b8  ===> programmzeiger auf start <===
8542  b4b8
8543  b4b8  a9 01     stxtpt lda #$01
8544  b4ba  85 87            sta txtptr+2     bank zeiger auf akt. term
8545  b4bc  18               clc
8546  b4bd  a5 2d            lda txttab       addrl zeiger anfang  basic-text
8547  b4bf  69 ff            adc #$ff
8548  b4c1  85 85            sta txtptr       addrl zeiger auf akt. term
8549  b4c3  a5 2e            lda txttab+1     addrh zeiger anfang  basic-text
8550  b4c5  69 ff            adc #$ff
8551  b4c7  85 86            sta txtptr+1     addrh zeiger auf akt. term
8552  b4c9  60               rts
8553  b4ca
8554  b4ca
8555  b4ca  ===> 2 byte-zahl in 'linnum', kommatest, 1 byte-zahl in xr <===
8556  b4ca
8557  b4ca  20 e5 b4  getnum jsr getpin       dezimal-zahl in klammer nach linnum
                                              einlesen
8558  b4cd
8559  b4cd
8560  b4cd  ===> kommatest, holt 1byte in xr <===
8561  b4cd
8562  b4cd  20 30 97  combyt jsr chkcom       test 'komma' sonst fehler + ready
8563  b4d0  4c d6 b4          jmp getbyt       zahl < 256 aus akt. text nach xr
8564  b4d3
8565  b4d3
8566  b4d3  ===> charget, 1 byte-zahl in xr <===
8567  b4d3
8568  b4d3  20 26 ba  gtbytc jsr chrget        nächstes zeichen nach ac (ind.jmp)
8569  b4d6
8570  b4d6
8571  b4d6  ===> zahl < 256 aus akt. text nach xr <===
8572  b4d6
8573  b4d6  20 01 b5  getbyt jsr frmnum        numerischen ausdruck holen
8574  b4d9
8575  b4d9
8576  b4d9  ===> zahl < 256 aus akt. text nach xr <===
8577  b4d9
8578  b4d9  20 0c 9b  conint jsr posint
8579  b4dc  a6 14             ldx facmo        fac #1: mantisse
8580  b4de  d0 1e             bne fcer1        ausgabe 'illegal quantity error',
                                               ready
8581  b4e0  a6 15             ldx faclo        fac #1: mantisse
8582  b4e2  4c 29 ba          jmp chrgot       letztes zeichen erneut nach ac
                                               (indirekter sprung)
8583  b4e5
8584  b4e5
8585  b4e5  ===> dezimal-zahl in klammer nach linnum einlesen <===
8586  b4e5
8587  b4e5  20 01 b5  getpin jsr frmnum        numerischen ausdruck holen
8588  b4e8
8589  b4e8
8590  b4e8  ===> gk-zahl in fac als adresse nach $11,$12 <===
8591  b4e8
8592  b4e8  a5 76     getadr lda facsgn        fac #1: vorzeichen
8593  b4ea  30 12            bmi fcer1        ausgabe 'illegal quantity error',
                                               ready
```

```
zeile adr.  obj.-code   source-code

8594  b4ec  a5 71           lda facexp       fac #1: exponent
8595  b4ee  c9 91           cmp #$91
8596  b4f0  b0 0c           bcs fcer1        ausgabe 'illegal quantity error',
                                             ready
8597  b4f2  20 80 a2        jsr qint         umw. gk-zahl in integer
8598  b4f5  a5 74           lda facmo        fac #1: mantisse
8599  b4f7  a4 75           ldy faclo        fac #1: mantisse
8600  b4f9  84 1b           sty linnum       lo-byte akt. zeilennummer
8601  b4fb  85 1c           sta linnum+1     hi-byte akt. zeilennummer
8602  b4fd  60              rts
8603  b4fe
8604  b4fe
8605  b4fe  4c a7 9b  fcer1 jmp fcerr        ausgabe '?illegal quantity error',
                                             ready
8606  b501
8607  b501
8608  b501  ===> numerischen ausdruck holen <===
8609  b501
8610  b501  20 c1 95  frmnum jsr frmevl      auswertung beliebiger ausdruck
8611  b504
8612  b504
8613  b504  ===> prüfen, ob numerische variable <===
8614  b504
8615  b504  18        chknum clc
8616  b505  24               .byte $24
8617  b506
8618  b506
8619  b506  ===> prüfen, ob stringvariable <===
8620  b506
8621  b506  38        chkstr sec
8622  b507
8623  b507
8624  b507  ===> prüft ob richtiger variablentyp <===
8625  b507
8626  b507  24 11     chkval bit valtyp      flag für variablen typ (0=num,
                                             1=string)
8627  b509  30 03           bmi docstr
8628  b50b  b0 03           bcs chkerr       ausgabe '?type mismatch error',
                                             ready
8629  b50d  60        chkok  rts
8630  b50e
8631  b50e
8632  b50e  b0 fd     docstr bcs chkok
8633  b510
8634  b510
8635  b510  ===> ausgabe '?type mismatch error', ready <===
8636  b510
8637  b510  a2 40     chkerr ldx #$40
8638  b512  4c 52 85         jmp error        ind. jmp zur fehlerroutine
8639  b515
8640  b515
8641  b515  ===> ausgabe string -$00 (addr. in zeropage) <===
8642  b515
8643  b515  20 e9 a8  strprt jsr frefac       stringverwaltung, freier string
8644  b518  aa               tax
8645  b519  a0 00            ldy #$00
8646  b51b  e8               inx
8647  b51c  20 73 ba         jsr mapinx       umsch. auf bank (# in $24)
```

```
zeile adr.  obj.-code   source-code

8648  b51f  ca        strp2  dex
8649  b520  f0 09            beq strp3
8650  b522  b1 22            lda (index1),y   addrl indirekter index #1
8651  b524  20 3a b5         jsr ochr         ausgabe 1 zeichen (in ac)
8652  b527  c8               iny
8653  b528  4c 1f b5         jmp strp2
8654  b52b
8655  b52b
8656  b52b  4c 8c ba  strp3  jmp mapusr       umsch. auf bank # 1
8657  b52e
8658  b52e
8659  b52e  ===> bei bs cursor rechts, sonst space <===
8660  b52e
8661  b52e  a5 1a     ospc   lda channl       speicherstelle für aktiven kanal
8662  b530  f0 03            beq crtskp       ausgabe 'cursor rechts'
8663  b532  a9 20            lda #$20
8664  b534  2c               .byte $2c
8665  b535
8666  b535
8667  b535  ===> ausgabe 'cursor rechts' <===
8668  b535
8669  b535  a9 1d     crtskp lda #$1d
8670  b537  2c               .byte $2c
8671  b538
8672  b538
8673  b538  ===> ausgabe '?' <===
8674  b538
8675  b538  a9 3f     outqst lda #$3f
8676  b53a
8677  b53a
8678  b53a  ===> ausgabe 1 zeichen (in ac) <===
8679  b53a
8680  b53a  20 f4 bb  ochr   jsr bsout        ausgabe 1 char auf akt. kanal
8681  b53d  29 ff            and #$ff
8682  b53f  60               rts
8683  b540
8684  b540
8685  b540  ===> ersatz segment start adresse <===
8686  b540
8687  b540  ff        dostbl .byte $ff, $ff
8687  b541  ff
8688  b542  ff               .byte $ff, $ff   ersatz segment end adresse
8688  b543  ff
8689  b544  0e               .byte $0e        ersatz logische filenummer für
                                              floppy (14)
8690  b545  08               .byte $08        ersatz geräteadr. für floppy (8)
8691  b546  6f               .byte $6f        ersatz sek.adr. für floppy (15)
8692  b547
8693  b547
8694  b547  ===> ersatzwerte vor disk-befehl init. <===
8695  b547
8696  b547  a9 00     dospar lda #$00
8697  b549  a2 ff     dosprs ldx #$ff
8698  b54b  48        dosprx pha
8699  b54c  8a               txa
8700  b54d  48               pha
8701  b54e  a9 00            lda #$00
8702  b550  85 8b            sta parsts       dos analyse-byte 1
```

```
zeile adr.  obj.-code   source-code

8703  b552  85 8c           sta parstx      dos analyse-byte 2
8704  b554  a2 26           ldx #$26
8705  b556  9d 00 02  dos01 sta buf,x       basic input-puffer (-$2ff)
8706  b559  ca               dex
8707  b55a  d0 fa           bne dos01
8708  b55c  a2 06           ldx #$06
8709  b55e  bd 40 b5  dos02 lda dostbl,x    ersatz segment start adresse
8710  b561  9d 1b 02         sta dosofl,x    addrl dos offset untergrenze
                                            (bsave,bload)
8711  b564  ca               dex
8712  b565  10 f7           bpl dos02
8713  b567  ae 57 02         ldx dfbank      vorgabe für bank-nummer
8714  b56a  8e 1a 02         stx dosbnk      dos bank nummer
8715  b56d  20 29 ba         jsr chrgot      letztes zeichen erneut nach ac
                                            (indirekter sprung)
8716  b570  d0 0e           bne parse1      disk-parameter 1 einlesen
8717  b572  68       done   pla
8718  b573  25 8c           and parstx      dos analyse-byte 2
8719  b575  d0 5c           bne dn20
8720  b577  68               pla
8721  b578  20 b5 b7         jsr prmrpt      syntaxtest: ac = syntax-byte 1
8722  b57b  a5 8b           lda parsts      dos analyse-byte 1
8723  b57d  a6 8c           ldx parstx      dos analyse-byte 2
8724  b57f  60               rts
8725  b580
8726  b580
8727  b580  ===> disk-parameter 1 einlesen <===
8728  b580
8729  b580  c9 23     parse1 cmp #$23
8730  b582  f0 3c           beq logadr      log. file# einlesen (disk-b.)
8731  b584  c9 57           cmp #$57
8732  b586  f0 4e           beq reclen      record-länge eintragen (disk-b.)
8733  b588  c9 4c           cmp #$4c
8734  b58a  f0 4a           beq reclen      record-länge eintragen (disk-b.)
8735  b58c  c9 52           cmp #$52
8736  b58e  d0 16           bne dos5
8737  b590  20 26 ba         jsr chrget      nächstes zeichen nach ac (ind.jmp)
8738  b593  4c 92 b6         jmp delim1
8739  b596
8740  b596
8741  b596  ===> geräte# 1,bank# 1 einlesen (disk-b.) 'on' <===
8742  b596
8743  b596  20 1e b7  on1    jsr on          abfrage auf folgende unit o. bank,
                                            sonst fehler
8744  b599  4c 8e b6  sav60  jmp del1        bit im dos-syntax-byte 1 setzen,
                                            kommatest, parameter einlesen
8745  b59c
8746  b59c
8747  b59c  ===> geräte# 1 einlesen (disk-b.) 'u' <===
8748  b59c
8749  b59c  20 2b b7  unit1  jsr unit        prüfen der geräteadr.(disk-basic),
                                            dann eintragen
8750  b59f  d0 f8           bne sav60
8751  b5a1  20 3c b7  bank1  jsr bank        prüfen der bank # (disk-basic), dann
                                            eintragen
8752  b5a4  f0 f3           beq sav60
8753  b5a6  c9 44     dos5   cmp #$44
8754  b5a8  f0 58           beq drv1        drivenummer 1 einlesen (disk-b.)
```

```
zeile adr.  obj.-code   source-code

8755  b5aa  c9 91           cmp #$91
8756  b5ac  f0 e8           beq on1          geräte# 1,bank# 1 einlesen (disk-b.)
                                              'on'
8757  b5ae  c9 42           cmp #$42
8758  b5b0  f0 ef           beq bank1
8759  b5b2  c9 55           cmp #$55
8760  b5b4  f0 e6           beq unit1        geräte# 1 einlesen (disk-b.) 'u'
8761  b5b6  c9 50           cmp #$50
8762  b5b8  f0 68           beq doff1        offset untergrenze einlesen
                                              (disk-b.)
8763  b5ba  c9 49           cmp #$49
8764  b5bc  d0 3a           bne dos10        auf filenamen überprüfen
8765  b5be  f0 5b           beq ident        test:disk-id eingelesen (disk-b.)
8766  b5c0
8767  b5c0
8768  b5c0  ===> log. file# einlesen (disk-b.) <===
8769  b5c0
8770  b5c0  a9 04     logadr lda #$04
8771  b5c2  20 b5 b7         jsr prmrpt       syntaxtest: ac = syntax-byte 1
8772  b5c5  20 8a b7         jsr getval       1 byte-zahl in xr (auch
                                              eingeklammert)
8773  b5c8  e0 00            cpx #$00
8774  b5ca  f0 4c            beq qtyer2
8775  b5cc  8e 1f 02         stx dosla        dos logische adresse
8776  b5cf  a9 04            lda #$04
8777  b5d1  d0 c6            bne sav60
8778  b5d3  4c 4f 97  dn20   jmp snerr        ausgabe '?syntax error', ready
8779  b5d6
8780  b5d6
8781  b5d6  ===> record-länge eintragen (disk-b.) <===
8782  b5d6
8783  b5d6  aa        reclen tax
8784  b5d7  a9 40            lda #$40
8785  b5d9  20 b5 b7         jsr prmrpt       syntaxtest: ac = syntax-byte 1
8786  b5dc  e0 57            cpx #$57
8787  b5de  d0 06            bne recoo
8788  b5e0  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
8789  b5e3  4c f4 b5         jmp recon
8790  b5e6
8791  b5e6
8792  b5e6  20 8a b7  recoo  jsr getval       1 byte-zahl in xr (auch
                                              eingeklammert)
8793  b5e9  e0 00            cpx #$00
8794  b5eb  f0 2b            beq qtyer2
8795  b5ed  e0 ff            cpx #$ff
8796  b5ef  f0 27            beq qtyer2
8797  b5f1  8e 22 02         stx dosrcl       dos record-länge
8798  b5f4  a9 40     recon  lda #$40
8799  b5f6  d0 1e            bne tacky1
8800  b5f8
8801  b5f8
8802  b5f8  ===> auf filenamen überprüfen <===
8803  b5f8
8804  b5f8  c9 22     dos10  cmp #$22
8805  b5fa  f0 65            beq name1        filenamen 1 einlesen (disk-b.)
8806  b5fc  c9 28            cmp #$28
8807  b5fe  f0 61            beq name1        filenamen 1 einlesen (disk-b.)
8808  b600  d0 d1            bne dn20
```

```
zeile adr.  obj.-code   source-code


8809  b602
8810  b602
8811  b602  ===> drivenummer 1 einlesen (disk-b.) <===
8812  b602
8813  b602  a9 10     drv1   lda #$10
8814  b604  20 b5 b7         jsr prmrpt       syntaxtest: ac = syntax-byte 1
8815  b607  20 8a b7         jsr getval       1 byte-zahl in xr (auch
                                               eingeklammert)
8816  b60a  e0 02            cpx #$02
8817  b60c  b0 0a            bcs qtyer2
8818  b60e  8e 11 02         stx dosds1       dos disk drive 1
8819  b611  8e 16 02         stx dosds2       dos disk drive 2
8820  b614  a9 10            lda #$10
8821  b616  d0 76     tacky1 bne del1         bit im dos-syntax-byte 1 setzen,
                                               kommatest, parameter einlesen
8822  b618  4c 1b b7  qtyer2 jmp qtyerr       ausgabe 'illegal quantity error',
                                               ready
8823  b61b
8824  b61b
8825  b61b  ===> test:disk-id eingelesen (disk-b.) <===
8826  b61b
8827  b61b  ad 25 02  ident  lda didchk       dos did flag
8828  b61e  f0 2a            beq idcon        disk-id einlesen (disk-b.)
8829  b620  d0 b1            bne dn20
8830  b622
8831  b622
8832  b622  ===> offset untergrenze einlesen (disk-b.) <===
8833  b622
8834  b622  a9 02     doffl  lda #$02
8835  b624  20 ba b7         jsr prxrpt       syntaxtest: ac = syntax-byte 2
8836  b627  20 9d b7         jsr getoff       dezimalzahl (auch in klammer) nach
                                               linnum
8837  b62a  8c 1b 02         sty dosofl       addrl dos offset untergrenze
                                               (bsave,bload)
8838  b62d  8d 1c 02         sta dosofl+1     addrh dos offset untergrenze
                                               (bsave,bload)
8839  b630  a9 02            lda #$02
8840  b632
8841  b632
8842  b632  ===> bit im dos-syntax-byte 2 setzen <===
8843  b632
8844  b632  05 8c     dlimx1 ora parstx       dos analyse-byte 2
8845  b634  85 8c            sta parstx       dos analyse-byte 2
8846  b636  d0 5a            bne delim1
8847  b638
8848  b638
8849  b638  ===> offset obergrenze einlesen (disk-b.) <===
8850  b638
8851  b638  a9 04     doffh  lda #$04
8852  b63a  20 ba b7         jsr prxrpt       syntaxtest: ac = syntax-byte 2
8853  b63d  20 9d b7         jsr getoff       dezimalzahl (auch in klammer) nach
                                               linnum
8854  b640  8c 1d 02         sty dosofh       addrl dos offset obergrenze
                                               (bsave,bload)
8855  b643  8d 1e 02         sta dosofh+1     addrh dos offset obergrenze
                                               (bsave,bload)
8856  b646  a9 04            lda #$04
8857  b648  d0 e8            bne dlimx1       bit im dos-syntax-byte 2 setzen
```

```
zeile adr.  obj.-code   source-code

8858  b64a
8859  b64a
8860  b64a  ===> disk-id einlesen (disk-b.) <===
8861  b64a
8862  b64a  20 26 ba  idcon  jsr chrget       nächstes zeichen nach ac (ind.jmp)
8863  b64d  8d 23 02         sta dosdid       dos erstes zeichen 'id'
8864  b650  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
8865  b653  8d 24 02         sta dosdid+1     dos zweites zeichen 'id'
8866  b656  a9 ff            lda #$ff
8867  b658  8d 25 02         sta didchk       dos did flag
8868  b65b  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
8869  b65e  4c 92 b6         jmp delim1
8870  b661
8871  b661
8872  b661  ===> filenamen 1 einlesen (disk-b.) <===
8873  b661
8874  b661  a9 01     name1  lda #$01
8875  b663  20 57 b7         jsr newnam       filenamen einlesen + prüfen
8876  b666  8d 10 02         sta dosf1l       dos filename 1 länge
8877  b669  85 0f            sta xcnt         dos loop-zähler
8878  b66b  a9 00            lda #$00
8879  b66d  8d 12 02         sta dosf1a       addrl dos filename 1
8880  b670  a9 02            lda #$02
8881  b672  8d 13 02         sta dosf1a+1     addrh dos filename 1
8882  b675  a9 0f            lda #$0f
8883  b677  8d 14 02         sta dosfb1       bank dos filename 1
8884  b67a  a0 00            ldy #$00
8885  b67c  20 73 ba         jsr mapinx       umsch. auf bank (# in $24)
8886  b67f  b1 22     loop6  lda (index1),y   addrl indirekter index #1
8887  b681  99 00 02         sta buf,y        basic input-puffer (-$2ff)
8888  b684  c8               iny
8889  b685  c4 0f            cpy xcnt         dos loop-zähler
8890  b687  90 f6            bcc loop6
8891  b689  20 8c ba         jsr mapusr       umsch. auf bank # 1
8892  b68c  a9 01            lda #$01
8893  b68e
8894  b68e
8895  b68e  ===> bit im dos-syntax-byte 1 setzen, kommatest, parameter einlesen
                                             <===
8896  b68e
8897  b68e  05 8b     del1   ora parsts       dos analyse-byte 1
8898  b690  85 8b            sta parsts       dos analyse-byte 1
8899  b692  20 29 ba  delim1 jsr chrgot       letztes zeichen erneut nach ac
                                             (indirekter sprung)
8900  b695  d0 19            bne nxxx         kommatest, nächsten parameter
                                             einlesen (disk-b.)
8901  b697  4c 72 b5  done1  jmp done
8902  b69a
8903  b69a
8904  b69a  ===> prüfen ob 'on' folgt (disk-b.) <===
8905  b69a
8906  b69a  c9 91     next6  cmp #$91
8907  b69c  d0 03            bne next6a       prüfen ob 'to' folgt (disk-b.)
8908  b69e  4c 96 b5         jmp on1          geräte# 1,bank# 1 einlesen (disk-b.)
                                             'on'
8909  b6a1
8910  b6a1
8911  b6a1  ===> prüfen ob 'to' folgt (disk-b.) <===
```

```
zeile adr.  obj.-code   source-code

8912  b6a1
8913  b6a1  c9 a4     next6a cmp #$a4
8914  b6a3  f0 02            beq next6b       prüfen ob 'p' folgt (disk-b.)
8915  b6a5  d0 72            bne sav61
8916  b6a7
8917  b6a7
8918  b6a7  ===> prüfen ob 'p' folgt (disk-b.) <===
8919  b6a7
8920  b6a7  20 26 ba  next6b jsr chrget       nächstes zeichen nach ac (ind.jmp)
8921  b6aa  c9 50            cmp #$50
8922  b6ac  d0 0f            bne pars22
8923  b6ae  f0 88            beq doffh        offset obergrenze einlesen (disk-b.)
8924  b6b0
8925  b6b0
8926  b6b0  ===> kommatest, nächsten parameter einlesen (disk-b.) <===
8927  b6b0
8928  b6b0  c9 2c     nxxx   cmp #$2c
8929  b6b2  d0 e6            bne next6        prüfen ob 'on' folgt (disk-b.)
8930  b6b4  20 26 ba         jsr chrget       nächstes zeichen nach ac (ind.jmp)
8931  b6b7  4c 80 b5         jmp parse1       disk-parameter 1 einlesen
8932  b6ba
8933  b6ba
8934  b6ba  ===> disk-parameter 2 einlesen <===
8935  b6ba
8936  b6ba  20 26 ba  parse2 jsr chrget       nächstes zeichen nach ac (ind.jmp)
8937  b6bd  c9 44     pars22 cmp #$44
8938  b6bf  f0 10            beq drv2         drivenummer 2 einlesen (disk-b.)
8939  b6c1  c9 91            cmp #$91
8940  b6c3  f0 1f            beq on2          geräte# 2, bank# 2 einlesen
                                              (disk-b.) 'on'
8941  b6c5  c9 55            cmp #$55
8942  b6c7  f0 21            beq unit2        geräte# 2 einlesen (disk-b.) 'u'
8943  b6c9  c9 22            cmp #$22
8944  b6cb  f0 22            beq name2
8945  b6cd  c9 28            cmp #$28
8946  b6cf  f0 1e            beq name2
8947  b6d1
8948  b6d1
8949  b6d1  ===> drivenummer 2 einlesen (disk-b.) <===
8950  b6d1
8951  b6d1  a9 20     drv2   lda #$20
8952  b6d3  20 b5 b7         jsr prmrpt       syntaxtest: ac = syntax-byte 1
8953  b6d6  20 8a b7         jsr getval       1 byte-zahl in xr (auch
                                              eingeklammert)
8954  b6d9  e0 02            cpx #$02
8955  b6db  b0 3e            bcs qtyerr       ausgabe 'illegal quantity error',
                                              ready
8956  b6dd  8e 16 02         stx dosds2       dos disk drive 2
8957  b6e0  a9 20            lda #$20
8958  b6e2  d0 20            bne del2         bit im dos-syntax-byte 1 setzen,
                                              kommatest, parameter einlesen
8959  b6e4
8960  b6e4
8961  b6e4  ===> geräte# 2, bank# 2 einlesen (disk-b.) 'on' <===
8962  b6e4
8963  b6e4  20 1e b7  on2    jsr on           abfrage auf folgende unit o. bank,
                                              sonst fehler
8964  b6e7  4c 04 b7         jmp del2         bit im dos-syntax-byte 1 setzen,
                                              kommatest, parameter einlesen
```

```
zeile adr.  obj.-code   source-code


8965  b6ea
8966  b6ea
8967  b6ea  ===> geräte# 2 einlesen (disk-b.) 'u' <===
8968  b6ea
8969  b6ea  20 2b b7  unit2  jsr unit         prüfen der geräteadr.(disk-basic),
                                              dann eintragen
8970  b6ed  d0 15             bne del2        bit im dos-syntax-byte 1 setzen,
                                              kommatest, parameter einlesen
8971  b6ef  a9 02     name2   lda #$02
8972  b6f1  20 57 b7          jsr newnam      filenamen einlesen + prüfen
8973  b6f4  8d 15 02          sta dosf2l      dos filename 2 länge
8974  b6f7  8e 17 02          stx dosf2a      addrl dos filename 2
8975  b6fa  8c 18 02          sty dosf2a+1    addrh dos filename 2
8976  b6fd  a5 24             lda index1+2    bank indirekter index #1
8977  b6ff  8d 19 02          sta dosfb2      bank dos filename 2
8978  b702  a9 02             lda #$02
8979  b704
8980  b704
8981  b704  ===> bit im dos-syntax-byte 1 setzen, kommatest, parameter einlesen
                                              <===
8982  b704
8983  b704  05 8b     del2    ora parsts      dos analyse-byte 1
8984  b706  85 8b             sta parsts      dos analyse-byte 1
8985  b708  20 29 ba          jsr chrgot      letztes zeichen erneut nach ac
                                              (indirekter sprung)
8986  b70b  f0 8a             beq done1
8987  b70d  c9 2c             cmp #$2c
8988  b70f  f0 a9             beq parse2      disk-parameter 2 einlesen
8989  b711  c9 91             cmp #$91
8990  b713  f0 cf             beq on2         geräte# 2, bank# 2 einlesen
                                              (disk-b.) 'on'
8991  b715  c9 55             cmp #$55
8992  b717  f0 d1             beq unit2       geräte# 2 einlesen (disk-b.) 'u'
8993  b719  d0 39     sav61   bne sner
8994  b71b  4c a7 9b  qtyerr  jmp fcerr       ausgabe '?illegal quantity error',
                                              ready
8995  b71e
8996  b71e
8997  b71e  ===> abfrage auf folgende unit o. bank, sonst fehler <===
8998  b71e
8999  b71e  20 26 ba  on      jsr chrget      nächstes zeichen nach ac (ind.jmp)
9000  b721  c9 55             cmp #$55
9001  b723  f0 06             beq unit        prüfen der geräteadr.(disk-basic),
                                              dann eintragen
9002  b725  c9 42             cmp #$42
9003  b727  f0 13             beq bank        prüfen der bank # (disk-basic), dann
                                              eintragen
9004  b729  d0 29             bne sner
9005  b72b
9006  b72b
9007  b72b  ===> prüfen der geräteadr.(disk-basic), dann eintragen <===
9008  b72b
9009  b72b  20 8a b7  unit    jsr getval      1 byte-zahl in xr (auch
                                              eingeklammert)
9010  b72e  e0 20             cpx #$20
9011  b730  b0 e9             bcs qtyerr      ausgabe 'illegal quantity error',
                                              ready
9012  b732  e0 03             cpx #$03
```

```
zeile adr.  obj.-code   source-code

9013  b734  90 e5           bcc qtyerr       ausgabe 'illegal quantity error',
                                              ready
9014  b736  8e 20 02        stx dosfa        dos primär-adresse
9015  b739  a9 08           lda #$08
9016  b73b  60              rts
9017  b73c
9018  b73c
9019  b73c  ===> prüfen der bank # (disk-basic), dann eintragen <===
9020  b73c
9021  b73c  a9 01     bank   lda #$01
9022  b73e  20 ba b7         jsr prxrpt       syntaxtest: ac = syntax-byte 2
9023  b741  20 8a b7         jsr getval       1 byte-zahl in xr (auch
                                              eingeklammert)
9024  b744  e0 10            cpx #$10
9025  b746  b0 d3            bcs qtyerr       ausgabe 'illegal quantity error',
                                              ready
9026  b748  8e 1a 02         stx dosbnk       dos bank nummer
9027  b74b  a9 01            lda #$01
9028  b74d  05 8c            ora parstx       dos analyse-byte 2
9029  b74f  85 8c            sta parstx       dos analyse-byte 2
9030  b751  a9 00            lda #$00
9031  b753  60               rts
9032  b754
9033  b754
9034  b754  4c 4f 97  sner   jmp snerr        ausgabe '?syntax error', ready
9035  b757
9036  b757
9037  b757  ===> filenamen einlesen + prüfen <===
9038  b757
9039  b757  20 b5 b7  newnam jsr prmrpt       syntaxtest: ac = syntax-byte 1
9040  b75a  20 b6 91         jsr sav13
9041  b75d  aa               tax
9042  b75e  f0 bb            beq qtyerr       ausgabe 'illegal quantity error',
                                              ready
9043  b760  a0 00            ldy #$00
9044  b762  20 df b0         jsr sav12
9045  b765  c9 40            cmp #$40
9046  b767  d0 12            bne lenchk       länge des filenamen prüfen
9047  b769  a9 80            lda #$80
9048  b76b  20 b5 b7         jsr prmrpt       syntaxtest: ac = syntax-byte 1
9049  b76e  a5 8b            lda parsts       dos analyse-byte 1
9050  b770  09 80            ora #$80
9051  b772  85 8b            sta parsts       dos analyse-byte 1
9052  b774  ca               dex
9053  b775  e6 22            inc index1       addrl indirekter index #1
9054  b777  d0 02            bne lenchk       länge des filenamen prüfen
9055  b779  e6 23            inc index1+1     addrh indirekter index #1
9056  b77b
9057  b77b
9058  b77b  ===> länge des filenamen prüfen <===
9059  b77b
9060  b77b  8a        lenchk txa
9061  b77c  c9 11            cmp #$11
9062  b77e  b0 05            bcs errlen       ausgabe '?string too long', ready
9063  b780  a6 22            ldx index1       addrl indirekter index #1
9064  b782  a4 23            ldy index1+1     addrh indirekter index #1
9065  b784  60               rts
9066  b785
```

```
zeile adr.  obj.-code   source-code


9067  b785
9068  b785  ===> ausgabe '?string too long', ready <===
9069  b785
9070  b785  a2 42     errlen ldx #$42
9071  b787  4c 52 85         jmp error        ind. jmp zur fehlerroutine
9072  b78a
9073  b78a
9074  b78a  ===> 1 byte-zahl in xr (auch eingeklammert) <===
9075  b78a
9076  b78a  20 26 ba  getval jsr chrget       nächstes zeichen nach ac (ind.jmp)
9077  b78d  f0 c5     gtvl2  beq sner
9078  b78f  90 09            bcc gtvl5
9079  b791  20 2d 97         jsr chkopn       test '(', sonst fehler + ready
9080  b794  20 d6 b4         jsr getbyt       zahl < 256 aus akt. text nach xr
9081  b797  4c 2a 97         jmp chkcls       test')', sonst fehler + ready
9082  b79a
9083  b79a
9084  b79a  4c d6 b4  gtvl5  jmp getbyt       zahl < 256 aus akt. text nach xr
9085  b79d
9086  b79d
9087  b79d  ===> dezimalzahl (auch in klammer) nach linnum <===
9088  b79d
9089  b79d  20 26 ba  getoff jsr chrget       nächstes zeichen nach ac (ind.jmp)
9090  b7a0  f0 b2            beq sner
9091  b7a2  90 0e            bcc gtff5
9092  b7a4  20 2d 97         jsr chkopn       test '(', sonst fehler + ready
9093  b7a7  20 e5 b4         jsr getpin       dezimal-zahl in klammer nach linnum
                                              einlesen
9094  b7aa  20 2a 97         jsr chkcls       test')', sonst fehler + ready
9095  b7ad  a4 1b            ldy linnum       lo-byte akt. zeilennummer
9096  b7af  a5 1c            lda linnum+1     hi-byte akt. zeilennummer
9097  b7b1  60               rts
9098  b7b2
9099  b7b2
9100  b7b2  4c e5 b4  gtff5  jmp getpin       dezimal-zahl in klammer nach linnum
                                              einlesen
9101  b7b5
9102  b7b5
9103  b7b5  ===> syntaxtest: ac = syntax-byte 1 <===
9104  b7b5
9105  b7b5  25 8b     prmrpt and parsts       dos analyse-byte 1
9106  b7b7  d0 9b            bne sner
9107  b7b9  60               rts
9108  b7ba
9109  b7ba
9110  b7ba  ===> syntaxtest: ac = syntax-byte 2 <===
9111  b7ba
9112  b7ba  25 8c     prxrpt and parstx       dos analyse-byte 2
9113  b7bc  d0 96            bne sner
9114  b7be  60               rts
9115  b7bf
9116  b7bf                   .end
9117  b7bf                   .lib butes2
```

```
zeile adr.  obj.-code   source-code


9119  b7bf
9120  b7bf  ===> tab. mit zeigern in diskbefehle <===
9121  b7bf
9122  b7bf  ff        tabfcb .byte $ff, $01, $05, $16, $1b, $21
9122  b7c0  01
9122  b7c1  05
9122  b7c2  16
9122  b7c3  1b
9122  b7c4  21
9123  b7c5  23               .byte $23, $27, $0d, $2f, $37, $3b
9123  b7c6  27
9123  b7c7  0d
9123  b7c8  2f
9123  b7c9  37
9123  b7ca  3b
9124  b7cb
9125  b7cb
9126  b7cb  ===> tabelle der disk-befehlsstrings <===
9127  b7cb
9128  b7cb  49        tabld  .byte $49, $d1
9128  b7cc  d1
9129  b7cd  24               .byte $24, $d1, $3a, $f1, $f0 'cat/dir' '$0:name=typ'
9129  b7ce  d1
9129  b7cf  3a
9129  b7d0  f1
9129  b7d1  f0
9130  b7d2  d1               .byte $d1, $3a, $f1, $2c, $e1, $2c 'dopen'
                                     '0:name,s,r'
9130  b7d3  3a
9130  b7d4  f1
9130  b7d5  2c
9130  b7d6  e1
9130  b7d7  2c
9131  b7d8  e0             .byte $e0
9132  b7d9  43             .byte $43, $d2, $3a, $f2, $3d, $d2 'concat'
                                     'c1:neu=1:alt1,0:alt2'
9132  b7da  d2
9132  b7db  3a
9132  b7dc  f2
9132  b7dd  3d
9132  b7de  d2
9133  b7df  3a             .byte $3a, $f2, $2c
9133  b7e0  f2
9133  b7e1  2c
9134  b7e2  d1             .byte $d1, $3a, $f1, $2c, $41 'append' '0:datnam,a'
9134  b7e3  3a
9134  b7e4  f1
9134  b7e5  2c
9134  b7e6  41
9135  b7e7  4e             .byte $4e, $d1, $3a, $f1, $2c, $d0 'header'
                                     'n0:dsknam,id'
9135  b7e8  d1
9135  b7e9  3a
9135  b7ea  f1
9135  b7eb  2c
9135  b7ec  d0
9136  b7ed  56             .byte $56, $d1   'collect' 'v0'
9136  b7ee  d1
```

```
zeile adr.  obj.-code   source-code

9137  b7ef  44              .byte $44, $d2, $3d, $d1 'backup' 'd0=1'
9137  b7f0  d2
9137  b7f1  3d
9137  b7f2  d1
9138  b7f3  43              .byte $43, $d2, $3a, $f2, $3d, $d1 'copy'
                                     'c0:neu=1:alt'
9138  b7f4  d2
9138  b7f5  3a
9138  b7f6  f2
9138  b7f7  3d
9138  b7f8  d1
9139  b7f9  3a              .byte $3a, $f1
9139  b7fa  f1
9140  b7fb  52              .byte $52, $d1, $3a, $f2, $3d, $d1 'rename'
                                     'r0:neu=0:alt'
9140  b7fc  d1
9140  b7fd  3a
9140  b7fe  f2
9140  b7ff  3d
9140  b800  d1
9141  b801  3a              .byte $3a, $f1
9141  b802  f1
9142  b803  53              .byte $53, $d1, $3a, $f1 'scratch' 's0:name'
9142  b804  d1
9142  b805  3a
9142  b806  f1
9143  b807  50              .byte $50, $c2, $e2, $e0
9143  b808  c2
9143  b809  e2
9143  b80a  e0
9144  b80b  20 fa b8  sav20  jsr chk2         test:eingabe filename 1 (disk-b.)
9145  b80e  a0 02             ldy #$02
9146  b810  a9 04             lda #$04
9147  b812
9148  b812
9149  b812  ===> dos befehlsstring zusammensetzen u. ausgeben <===
9150  b812
9151  b812  85 0f     sendp   sta xcnt         dos loop-zähler
9152  b814  b9 bf b7          lda tabfcb,y     tab. mit zeigern in diskbefehle
9153  b817  48                pha
9154  b818  20 94 b9          jsr oldclr       länge disk.status =0, system-status
                                               schreiben
9155  b81b  a2 00             ldx #$00
9156  b81d  68        sdp1    pla
9157  b81e  c6 0f             dec xcnt         dos loop-zähler
9158  b820  30 48             bmi trnar        disk-befehlstringadr in zero-page
9159  b822  a8                tay
9160  b823  c8                iny
9161  b824  98                tya
9162  b825  48                pha
9163  b826  b9 cb b7          lda tabld,y      tabelle der disk-befehlsstrings
9164  b829  10 37             bpl sdp5         ac, xr in disk-befehlsstring
9165  b82b  c9 c2             cmp #$c2
9166  b82d  f0 4e             beq rsca         sekundäradr. in disk-befehlsstring
9167  b82f  c9 d0             cmp #$d0
9168  b831  f0 59             beq rid          disk 'id' in disk-befehlsstring
9169  b833  c9 e2             cmp #$e2
9170  b835  f0 73             beq rdcn
```

```
zeile adr.  obj.-code   source-code

9171  b837  c9 e1           cmp #$e1
9172  b839  f0 5d           beq rwrt        recordlänge in disk-befehlsstring
9173  b83b  c9 f0           cmp #$f0
9174  b83d  f0 43           beq rfat        filetyp in disk-befehlsstring
9175  b83f  c9 f1           cmp #$f1
9176  b841  f0 71           beq rsfn        zeiger auf filename 1 setzen
9177  b843  c9 f2           cmp #$f2
9178  b845  f0 21           beq gordfn
9179  b847  c9 e0           cmp #$e0
9180  b849  d0 05           bne sdp2        abfrage drive 1
9181  b84b  ad 22 02        lda dosrcl      dos record-länge
9182  b84e  d0 12           bne sdp5        ac, xr in disk-befehlsstring
9183  b850
9184  b850
9185  b850  ===> abfrage drive 1 <===
9186  b850
9187  b850  c9 d1     sdp2  cmp #$d1
9188  b852  d0 05           bne sdp3        abfrage drive 2
9189  b854  ad 11 02        lda dosds1      dos disk drive 1
9190  b857  10 07           bpl sdp4        drive# in ascii umwandeln
9191  b859
9192  b859
9193  b859  ===> abfrage drive 2 <===
9194  b859
9195  b859  c9 d2     sdp3  cmp #$d2
9196  b85b  d0 c0           bne sdp1
9197  b85d  ad 16 02        lda dosds2      dos disk drive 2
9198  b860
9199  b860
9200  b860  ===> drive# in ascii umwandeln <===
9201  b860
9202  b860  09 30     sdp4  ora #$30
9203  b862
9204  b862
9205  b862  ===> ac, xr in disk-befehlsstring <===
9206  b862
9207  b862  9d 26 02  sdp5  sta dosstr,x    dos ausgabe string puffer
9208  b865  e8              inx
9209  b866  d0 b5           bne sdp1
9210  b868  f0 5e     gordfn beq rdfn       zeiger auf filename 2 setzen
9211  b86a
9212  b86a
9213  b86a  ===> disk-befehlstringadr in zero-page <===
9214  b86a
9215  b86a  8a        trnar txa
9216  b86b  48              pha
9217  b86c  a2 26           ldx #$26
9218  b86e  a0 02           ldy #$02
9219  b870  86 64           stx highds      addrl zielende (verschieben)/
                                            temp.fac#1
9220  b872  84 65           sty highds+1    addrh zielende (verschieben/
                                            temp.fac#1
9221  b874  a0 0f           ldy #$0f
9222  b876  84 66           sty highds+2    bank zielende (verschieben/
                                            temp.fac#1
9223  b878  20 e8 b9        jsr sav3        la,fa,sa,len,adr. befehlsstring
                                            eintragen
9224  b87b  68              pla
```

```
zeile adr.  obj.-code   source-code

9225  b87c  60              rts
9226  b87d
9227  b87d
9228  b87d  ===> sekundäradr. in disk-befehlsstring <===
9229  b87d
9230  b87d  ad 21 02  rsca   lda dossa        dos sekundär-adresse
9231  b880  d0 e0            bne sdp5         ac, xr in disk-befehlsstring
9232  b882
9233  b882
9234  b882  ===> filetyp in disk-befehlsstring <===
9235  b882
9236  b882  24 8b     rfat   bit parsts       dos analyse-byte 1
9237  b884  30 02            bmi rfata
9238  b886  10 95            bpl sdp1
9239  b888  a9 40     rfata  lda #$40
9240  b88a  d0 d6            bne sdp5         ac, xr in disk-befehlsstring
9241  b88c
9242  b88c
9243  b88c  ===> disk 'id' in disk-befehlsstring <===
9244  b88c
9245  b88c  ad 23 02  rid    lda dosdid       dos erstes zeichen 'id'
9246  b88f  9d 26 02         sta dosstr,x     dos ausgabe string puffer
9247  b892  e8               inx
9248  b893  ad 24 02         lda dosdid+1     dos zweites zeichen 'id'
9249  b896  d0 ca            bne sdp5         ac, xr in disk-befehlsstring
9250  b898
9251  b898
9252  b898  ===> recordlänge in disk-befehlsstring <===
9253  b898
9254  b898  ad 22 02  rwrt   lda dosrcl       dos record-länge
9255  b89b  f0 04            beq rwrt1
9256  b89d  a9 4c            lda #$4c
9257  b89f  d0 c1            bne sdp5         ac, xr in disk-befehlsstring
9258  b8a1  a9 53     rwrt1  lda #$53
9259  b8a3  8d 22 02         sta dosrcl       dos record-länge
9260  b8a6  a9 57            lda #$57
9261  b8a8  d0 b8            bne sdp5         ac, xr in disk-befehlsstring
9262  b8aa  a5 1b     rdcn   lda linnum       lo-byte akt. zeilennummer
9263  b8ac  9d 26 02         sta dosstr,x     dos ausgabe string puffer
9264  b8af  a5 1c            lda linnum+1     hi-byte akt. zeilennummer
9265  b8b1  e8               inx
9266  b8b2  d0 ae            bne sdp5         ac, xr in disk-befehlsstring
9267  b8b4
9268  b8b4
9269  b8b4  ===> zeiger auf filename 1 setzen <===
9270  b8b4
9271  b8b4  ad 12 02  rsfn   lda dosf1a       addrl dos filename 1
9272  b8b7  85 22            sta index1       addrl indirekter index #1
9273  b8b9  ad 13 02         lda dosf1a+1     addrh dos filename 1
9274  b8bc  85 23            sta index1+1     addrh indirekter index #1
9275  b8be  ad 14 02         lda dosfb1       bank dos filename 1
9276  b8c1  ac 10 02         ldy dosf1l       dos filename 1 länge
9277  b8c4  f0 26            beq rdrt0
9278  b8c6  d0 12            bne xrfn         bank# des filenamen nach i6509
9279  b8c8
9280  b8c8
9281  b8c8  ===> zeiger auf filename 2 setzen <===
9282  b8c8
```

```
zeile adr.  obj.-code   source-code

9283  b8c8  ad 17 02  rdfn   lda dosf2a       addrl dos filename 2
9284  b8cb  85 22             sta index1      addrl indirekter index #1
9285  b8cd  ad 18 02          lda dosf2a+1    addrh dos filename 2
9286  b8d0  85 23             sta index1+1    addrh indirekter index #1
9287  b8d2  ad 19 02          lda dosfb2      bank dos filename 2
9288  b8d5  ac 15 02          ldy dosf2l      dos filename 2 länge
9289  b8d8  f0 12             beq rdrt0
9290  b8da
9291  b8da
9292  b8da  ===> bank# des filenamen nach i6509 <===
9293  b8da
9294  b8da  85 01      xrfn   sta i6509        6509 indirection register
9295  b8dc  84 0e             sty count       allgemeiner zähler
9296  b8de  a0 00             ldy #$00
9297  b8e0
9298  b8e0
9299  b8e0  ===> filename in disk-befehlsstring <===
9300  b8e0
9301  b8e0  b1 22      rdmov  lda (index1),y  addrl indirekter index #1
9302  b8e2  9d 26 02          sta dosstr,x    dos ausgabe string puffer
9303  b8e5  e8                inx
9304  b8e6  c8                iny
9305  b8e7  c4 0e             cpy count       allgemeiner zähler
9306  b8e9  d0 f5             bne rdmov       filename in disk-befehlsstring
9307  b8eb  24                .byte $24
9308  b8ec  ca         rdrt0  dex
9309  b8ed  20 8c ba          jsr mapusr      umsch. auf bank # 1
9310  b8f0  4c 1d b8          jmp sdp1
9311  b8f3
9312  b8f3
9313  b8f3  ===> syntaxprüfung disk-basic <===
9314  b8f3
9315  b8f3  29 e6      chk1   and #$e6
9316  b8f5  f0 03             beq chk2        test:eingabe filename 1 (disk-b.)
9317  b8f7  4c 4f 97   chker1 jmp snerr       ausgabe '?syntax error', ready
9318  b8fa
9319  b8fa
9320  b8fa  ===> test:eingabe filename 1 (disk-b.) <===
9321  b8fa
9322  b8fa  a5 8b      chk2   lda parsts       dos analyse-byte 1
9323  b8fc  29 01             and #$01
9324  b8fe  c9 01             cmp #$01
9325  b900  d0 f5             bne chker1      ausgabe '?syntax error', ready
9326  b902  a5 8b             lda parsts      dos analyse-byte 1
9327  b904  60                rts
9328  b905
9329  b905
9330  b905  29 e7      chk3   and #$e7
9331  b907  d0 ee             bne chker1      ausgabe '?syntax error', ready
9332  b909  60                rts
9333  b90a
9334  b90a
9335  b90a  29 c4      chk4   and #$c4
9336  b90c  d0 e9             bne chker1      ausgabe '?syntax error', ready
9337  b90e  a5 8b             lda parsts      dos analyse-byte 1
9338  b910  29 03      chk5   and #$03
9339  b912  c9 03             cmp #$03
9340  b914  d0 e1             bne chker1      ausgabe '?syntax error', ready
```

```
zeile adr.  obj.-code   source-code

9341  b916  a5 8b           lda parsts       dos analyse-byte 1
9342  b918  60              rts
9343  b919
9344  b919
9345  b919  29 05     chk6   and #$05
9346  b91b  c9 05            cmp #$05
9347  b91d  d0 d8            bne chker1       ausgabe '?syntax error', ready
9348  b91f  a5 8b            lda parsts       dos analyse-byte 1
9349  b921  60              rts
9350  b922
9351  b922
9352  b922  ===> statusstring über sek.adr. 15 einlesen <===
9353  b922
9354  b922  a9 00     errchl lda #$00
9355  b924  20 bd ff         jsr kstnam       länge und adresse des filenamens
                                               eintragen
9356  b927  a0 6f            ldy #$6f
9357  b929  20 5b 93         jsr ochanl
9358  b92c  a2 0e            ldx #$0e
9359  b92e  20 fa bb         jsr chkin        eingabekanal öffnen, fehlertest
9360  b931  a0 ff            ldy #$ff
9361  b933  20 7d ba         jsr mapstr       umsch. auf bank # 2
9362  b936  c8        loop1  iny
9363  b937  20 ee bb         jsr basin        eingabe 1 char vom akt. kanal
                                               (chn-vektor)
9364  b93a  c9 0d            cmp #$0d
9365  b93c  f0 06            beq errend
9366  b93e  91 17            sta (dsdesc+1),y addrl disc-status string
9367  b940  c0 27            cpy #$27
9368  b942  d0 f2            bne loop1
9369  b944  a9 00     errend lda #$00
9370  b946  91 17            sta (dsdesc+1),y addrl disc-status string
9371  b948  a9 28            lda #$28
9372  b94a  85 16            sta dsdesc       länge disc-status string
9373  b94c  20 8c ba         jsr mapusr       umsch. auf bank # 1
9374  b94f  20 cc ff         jsr kclrch       ein-/ausgabekanal schließen
9375  b952  a9 0e            lda #$0e
9376  b954  18               clc
9377  b955  20 13 bc         jsr tclose
9378  b958  4c 50 93         jmp dcat0
9379  b95b
9380  b95b
9381  b95b  ===> bearbeitung 'are you sure ?' <===
9382  b95b
9383  b95b  20 57 9d  rusure jsr tstdir       test direktmodus, hi-byt zeile=$ff
9384  b95e  d0 32            bne ansyes
9385  b960  a2 14            ldx #$14
9386  b962  20 c3 a3         jsr msg          textzeiger + xr, textausgabe
9387  b965  20 cc ff         jsr kclrch       ein-/ausgabekanal schließen
9388  b968  20 ee bb         jsr basin        eingabe 1 char vom akt. kanal
                                               (chn-vektor)
9389  b96b  c9 59            cmp #$59
9390  b96d  d0 19            bne ansno
9391  b96f  20 ee bb         jsr basin        eingabe 1 char vom akt. kanal
                                               (chn-vektor)
9392  b972  c9 0d            cmp #$0d
9393  b974  f0 1c            beq ansyes
9394  b976  c9 45            cmp #$45
```

```
zeile adr.  obj.-code   source-code

9395  b978  d0 0e           bne ansno
9396  b97a  20 ee bb        jsr basin       eingabe 1 char vom akt. kanal
                                            (chn-vektor)
9397  b97d  c9 53           cmp #$53
9398  b97f  d0 07           bne ansno
9399  b981  20 ee bb        jsr basin       eingabe 1 char vom akt. kanal
                                            (chn-vektor)
9400  b984  c9 0d           cmp #$0d
9401  b986  f0 0a           beq ansyes
9402  b988  c9 0d     ansno cmp #$0d
9403  b98a  38              sec
9404  b98b  f0 06           beq ansbye
9405  b98d  20 ee bb        jsr basin       eingabe 1 char vom akt. kanal
                                            (chn-vektor)
9406  b990  d0 f6           bne ansno
9407  b992  18        ansyes clc
9408  b993  60        ansbye rts
9409  b994
9410  b994
9411  b994  ===> länge disk.status =0, system-status schreiben <===
9412  b994
9413  b994  a9 00     oldclr lda #$00
9414  b996  85 16           sta dsdesc      länge disc-status string
9415  b998  18              clc
9416  b999  4c de bb        jmp storst      ac in system-status schreiben
9417  b99c
9418  b99c
9419  b99c  20 b6 91  sav77  jsr sav13
9420  b99f  8d 10 02        sta dosf1l      dos filename 1 länge
9421  b9a2  a5 22           lda index1      addrl indirekter index #1
9422  b9a4  a4 23           ldy index1+1    addrh indirekter index #1
9423  b9a6  8d 12 02        sta dosf1a      addrl dos filename 1
9424  b9a9  8c 13 02        sty dosf1a+1    addrh dos filename 1
9425  b9ac  a4 24           ldy index1+2    bank indirekter index #1
9426  b9ae  8c 14 02        sty dosfb1      bank dos filename 1
9427  b9b1  60              rts
9428  b9b2
9429  b9b2
9430  b9b2  a2 00     plsv   ldx #$00
9431  b9b4  8e 10 02        stx dosf1l      dos filename 1 länge
9432  b9b7  8e 21 02        stx dossa       dos sekundär-adresse
9433  b9ba  8e 1f 02        stx dosla       dos logische adresse
9434  b9bd  a2 01           ldx #$01
9435  b9bf  8e 20 02        stx dosfa       dos primär-adresse
9436  b9c2  20 29 ba        jsr chrgot      letztes zeichen erneut nach ac
                                            (indirekter sprung)
9437  b9c5  f0 0f           beq plsvx
9438  b9c7  20 9c b9        jsr sav77
9439  b9ca  20 f9 b9        jsr plsv27
9440  b9cd  8e 20 02        stx dosfa       dos primär-adresse
9441  b9d0  20 f9 b9        jsr plsv27
9442  b9d3  8e 21 02        stx dossa       dos sekundär-adresse
9443  b9d6  ad 12 02  plsvx  lda dosf1a      addrl dos filename 1
9444  b9d9  ae 13 02        ldx dosf1a+1    addrh dos filename 1
9445  b9dc  ac 14 02        ldy dosfb1      bank dos filename 1
9446  b9df  85 64           sta highds      addrl zielende (verschieben)/
                                            temp.fac#1
9447  b9e1  86 65           stx highds+1    addrh zielende (verschieben/
                                            temp.fac#1
```

```
zeile adr.  obj.-code   source-code

9448  b9e3  84 66           sty highds+2     bank zielende (verschieben/
                                             temp.fac#1
9449  b9e5  ad 10 02        lda dosfil       dos filename 1 länge
9450  b9e8
9451  b9e8
9452  b9e8  ===> la,fa,sa,len,adr. befehlsstring eintragen <===
9453  b9e8
9454  b9e8  a2 64     sav3  ldx #$64
9455  b9ea  20 bd ff        jsr kstnam       länge und adresse des filenamens
                                             eintragen
9456  b9ed  ad 1f 02        lda dosla        dos logische adresse
9457  b9f0  ae 20 02        ldx dosfa        dos primär-adresse
9458  b9f3  ac 21 02        ldy dossa        dos sekundär-adresse
9459  b9f6  4c ba ff        jmp kstlfs       log. filenummer, geräte- und
                                             sekundäradr. eintragen
9460  b9f9
9461  b9f9
9462  b9f9  20 29 ba  plsv27 jsr chrgot      letztes zeichen erneut nach ac
                                             (indirekter sprung)
9463  b9fc  f0 d8           beq plsvx
9464  b9fe  4c 91 92        jmp copg
9465  ba01
9466  ba01
9467  ba01  20 30 97  plsv30 jsr chkcom      test 'komma' sonst fehler + ready
9468  ba04  20 29 ba  plsv32 jsr chrgot      letztes zeichen erneut nach ac
                                             (indirekter sprung)
9469  ba07  d0 12           bne plsrts
9470  ba09  4c 4f 97        jmp snerr        ausgabe '?syntax error', ready
9471  ba0c
9472  ba0c
9473  ba0c  a5 76     tstrom lda facsgn      fac #1: vorzeichen
9474  ba0e  c9 0f           cmp #$0f
9475  ba10  d0 0a           bne xit
9476  ba12  18              clc
9477  ba13  a9 00           lda #$00
9478  ba15  e5 74           sbc facmo        fac #1: mantisse
9479  ba17  a9 80           lda #$80
9480  ba19  e5 75           sbc faclo        fac #1: mantisse
9481  ba1b  60        plsrts rts
9482  ba1c
9483  ba1c
9484  ba1c  38        xit   sec
9485  ba1d  60              rts
9486  ba1e
9487  ba1e
9488  ba1e  ===> umwandlung integer nach gleitkomma <===
9489  ba1e
9490  ba1e  20 13 9b  flpint jsr ayint       umwandlung real nach integer
9491  ba21  a5 74           lda facmo        fac #1: mantisse
9492  ba23  a4 75           ldy faclo        fac #1: mantisse
9493  ba25  60              rts
9494  ba26
9495  ba26
9496  ba26  6c 90 02  chrget jmp (ichrge)    addrl chrget routine
9497  ba29
9498  ba29
9499  ba29  6c 8e 02  chrgot jmp (ichrgo)    addrl chrgot routine
9500  ba2c
```

```
zeile adr.  obj.-code   source-code

9501  ba2c
9502  ba2c  ===> letztes zeichen erneut nach ac <===
9503  ba2c
9504  ba2c  20 5a ba  nchrgo jsr chrmap       i6509 nach 'ldaadr', charget-bank
                                               nach i6509
9505  ba2f  4c 3b ba          jmp chrg20
9506  ba32
9507  ba32
9508  ba32  ===> nächstes zeichen nach ac <===
9509  ba32
9510  ba32  20 5a ba  nchrge jsr chrmap       i6509 nach 'ldaadr', charget-bank
                                               nach i6509
9511  ba35  e6 85     chrg10 inc txtptr       addrl zeiger auf akt. term
9512  ba37  d0 02            bne chrg20
9513  ba39  e6 86            inc txtptr+1     addrh zeiger auf akt. term
9514  ba3b  a0 00     chrg20 ldy #$00
9515  ba3d  b1 85            lda (txtptr),y   addrl zeiger auf akt. term
9516  ba3f  c9 20            cmp #$20
9517  ba41  f0 f2            beq chrg10
9518  ba43  20 50 ba         jsr qnum         test inhalt ac numerisch
9519  ba46  08               php
9520  ba47  48               pha
9521  ba48  ad 5b 02         lda ldaadr       addrl modifiable adresse
9522  ba4b  85 01            sta i6509        6509 indirection register
9523  ba4d  68               pla
9524  ba4e  28               plp
9525  ba4f  60               rts
9526  ba50
9527  ba50
9528  ba50  ===> test inhalt ac numerisch <===
9529  ba50
9530  ba50  c9 3a     qnum   cmp #$3a
9531  ba52  b0 05            bcs qnrts
9532  ba54  e9 2f            sbc #$2f
9533  ba56  38               sec
9534  ba57  e9 d0            sbc #$d0
9535  ba59  60        qnrts  rts
9536  ba5a
9537  ba5a
9538  ba5a  ===> i6509 nach 'ldaadr', charget-bank nach i6509 <===
9539  ba5a
9540  ba5a  a5 01     chrmap lda i6509         6509 indirection register
9541  ba5c  8d 5b 02         sta ldaadr       addrl modifiable adresse
9542  ba5f  a5 87            lda txtptr+2     bank zeiger auf akt. term
9543  ba61  85 01            sta i6509        6509 indirection register
9544  ba63  60               rts
9545  ba64
9546  ba64
9547  ba64  ===> lade (index1),y in ac <===
9548  ba64
9549  ba64  a9 00     ldi1y  lda #$00
9550  ba66  11 22            ora (index1),y   addrl indirekter index #1
9551  ba68  60               rts
9552  ba69
9553  ba69
9554  ba69  ===> umsch. auf bank (# in $73) <===
9555  ba69
9556  ba69  48        mapdst pha
```

```
zeile adr.  obj.-code   source-code

9557  ba6a  a5 73           lda facmoh       fac #1: mantisse
9558  ba6c  10 21           bpl map001       umsch. auf bank (# in ac)
9559  ba6e
9560  ba6e
9561  ba6e  ===> umsch. auf bank (# in $60) <===
9562  ba6e
9563  ba6e  48       mapdsp pha
9564  ba6f  a5 60           lda dscpnt+2     bank zeiger string descriptor
9565  ba71  10 1c           bpl map001       umsch. auf bank (# in ac)
9566  ba73
9567  ba73
9568  ba73  ===> umsch. auf bank (# in $24) <===
9569  ba73
9570  ba73  48       mapinx pha
9571  ba74  a5 24           lda index1+2     bank indirekter index #1
9572  ba76  10 17           bpl map001       umsch. auf bank (# in ac)
9573  ba78
9574  ba78
9575  ba78  ===> umsch. auf bank # 15 <===
9576  ba78
9577  ba78  48       mapsys pha
9578  ba79  a9 0f           lda #$0f
9579  ba7b  10 12           bpl map001       umsch. auf bank (# in ac)
9580  ba7d
9581  ba7d
9582  ba7d  ===> umsch. auf bank # 2 <===
9583  ba7d
9584  ba7d  48       mapstr pha
9585  ba7e  a9 02           lda #$02
9586  ba80  10 0d           bpl map001       umsch. auf bank (# in ac)
9587  ba82
9588  ba82
9589  ba82  ===> umsch. auf bank # 2 <===
9590  ba82
9591  ba82  48       mapary pha
9592  ba83  a9 02           lda #$02
9593  ba85  10 08           bpl map001       umsch. auf bank (# in ac)
9594  ba87
9595  ba87
9596  ba87  ===> umsch. auf bank # 2 <===
9597  ba87
9598  ba87  48       mapvar pha
9599  ba88  a9 02           lda #$02
9600  ba8a  10 03           bpl map001       umsch. auf bank (# in ac)
9601  ba8c
9602  ba8c
9603  ba8c  ===> umsch. auf bank # 1 <===
9604  ba8c
9605  ba8c  48       mapusr pha
9606  ba8d  a9 01           lda #$01
9607  ba8f
9608  ba8f
9609  ba8f  ===> umsch. auf bank (# in ac) <===
9610  ba8f
9611  ba8f  85 01    map001 sta i6509        6509 indirection register
9612  ba91  68              pla
9613  ba92  60              rts
9614  ba93
```

```
zeile adr.  obj.-code   source-code


9615  ba93                  .end
9616  ba93                  .lib init
```

```
zeile adr.  obj.-code   source-code

9618  ba93
9619  ba93  ===> initialisierung <===
9620  ba93
9621  ba93  20 16 bc  initop jsr clall        alle logischen files schliessen
9622  ba96  38                sec
9623  ba97  20 99 ff          jsr kmemtp      höchste speichergrenze
                                               lesen/schreiben
9624  ba9a  8e a6 02          stx highst      lo-byte max. offset user bank
9625  ba9d  8c a7 02          sty highst+1    hi-byte max. offset user bank
9626  baa0  38                sec
9627  baa1  8a                txa
9628  baa2  e9 a1             sbc #$a1
9629  baa4  85 88             sta buffpt      addrl zeiger auf eingabe-puffer
9630  baa6  98                tya
9631  baa7  e9 00             sbc #$00
9632  baa9  85 89             sta buffpt+1    addrh zeiger auf eingabe-puffer
9633  baab  38                sec
9634  baac  8a                txa
9635  baad  e9 2b             sbc #$2b
9636  baaf  85 17             sta dsdesc+1    addrl disc-status string
9637  bab1  aa                tax
9638  bab2  b0 01             bcs itp04
9639  bab4  88                dey
9640  bab5  84 18      itp04  sty dsdesc+2    addrh disc-status string
9641  bab7  38                sec
9642  bab8  8a                txa
9643  bab9  e9 0c             sbc #$0c
9644  babb  85 20             sta tempst      addrl für 3 temp.
                                               string-descriptoren
9645  babd  aa                tax
9646  babe  b0 01             bcs itp03
9647  bac0  88                dey
9648  bac1  84 21      itp03  sty tempst+1    addrh für 3 temp.
                                               string-descriptoren
9649  bac3  38                sec
9650  bac4  8a                txa
9651  bac5  e9 07             sbc #$07
9652  bac7  85 8d             sta seedpt      addrl zeiger auf akt. zufallszahl
9653  bac9  aa                tax
9654  baca  b0 01             bcs itp02
9655  bacc  88                dey
9656  bacd  84 8e      itp02  sty seedpt+1    addrh zeiger auf akt. zufallszahl
9657  bacf  38                sec
9658  bad0  8a                txa
9659  bad1  e9 04             sbc #$04
9660  bad3  aa                tax
9661  bad4  b0 01             bcs itp06
9662  bad6  88                dey
9663  bad7  86 3f      itp06  stx memtop      addrl höchste ram-speicherstelle
9664  bad9  84 40             sty memtop+1    addrh höchste ram-speicherstelle
9665  badb  a4 89             ldy buffpt+1    addrh zeiger auf eingabe-puffer
9666  badd  a6 88             ldx buffpt      addrl zeiger auf eingabe-puffer
9667  badf  d0 01             bne itp07
9668  bae1  88                dey
9669  bae2  ca         itp07  dex
9670  bae3  86 22             stx index1      addrl indirekter index #1
9671  bae5  84 23             sty index1+1    addrh indirekter index #1
9672  bae7  a9 83             lda #$83
```

```
zeile adr.  obj.-code   source-code

9673  bae9  a0 00           ldy #$00
9674  baeb  20 8c ba        jsr mapusr       umsch. auf bank # 1
9675  baee  91 22           sta (index1),y   addrl indirekter index #1
9676  baf0  a9 02           lda #$02
9677  baf2  85 01           sta i6509        6509 indirection register
9678  baf4  85 41           sta memtop+2     bank höchste ram-speicherstelle
9679  baf6  85 19           sta dsdesc+3     bank disc-status string
9680  baf8  a0 06           ldy #$06
9681  bafa  b9 20 bb  itp01 lda rseed-1,y    zufallszahl nach initialisierung
9682  bafd  91 8d           sta (seedpt),y   addrl zeiger auf akt. zufallszahl
9683  baff  88              dey
9684  bb00  d0 f8           bne itp01
9685  bb02  98              tya
9686  bb03  a2 04           ldx #$04
9687  bb05  91 3f     itp05 sta (memtop),y   addrl höchste ram-speicherstelle
9688  bb07  c8              iny
9689  bb08  ca              dex
9690  bb09  d0 fa           bne itp05
9691  bb0b  a9 28           lda #$28
9692  bb0d  a8              tay
9693  bb0e  91 17           sta (dsdesc+1),y addrl disc-status string
9694  bb10  c8              iny
9695  bb11  a9 ff           lda #$ff
9696  bb13  91 17           sta (dsdesc+1),y addrl disc-status string
9697  bb15  c8              iny
9698  bb16  91 17           sta (dsdesc+1),y addrl disc-status string
9699  bb18  a5 3f     settop lda memtop      addrl höchste ram-speicherstelle
9700  bb1a  a4 40           ldy memtop+1     addrh höchste ram-speicherstelle
9701  bb1c  85 3b           sta fretop       addrl top of string free space
9702  bb1e  84 3c           sty fretop+1     addrh top of string free space
9703  bb20  60              rts              zufallszahl nach initialisierung
9704  bb21
9705  bb21
9706  bb21  ===> zufallszahl nach initialisierung <===
9707  bb21
9708  bb21  80        rseed  .byte $80, $4f, $c7, $52, $58, $2c
9708  bb22  4f
9708  bb23  c7
9708  bb24  52
9708  bb25  58
9708  bb26  2c
9709  bb27
9710  bb27
9711  bb27  ===> basic kalt-start <===
9712  bb27
9713  bb27  20 bc bb  init   jsr initv        standard-vektoren ins ram kopieren
9714  bb2a  a2 02     init0  ldx #$02
9715  bb2c  bd fe b4  init05 lda fcer1,x      ausgabe 'illegal quantity error',
                                              ready
9716  bb2f  95 02           sta usrpok,x     jmp code für user-routine
9717  bb31  ca              dex
9718  bb32  10 f8           bpl init05
9719  bb34  85 61           sta jmper        $4c wert für 'jmp' (functions-rout.)
9720  bb36  a2 04           ldx #$04
9721  bb38  bd c7 bb  init10 lda initat-1,x
9722  bb3b  9d 59 02        sta tansgn,x     flag für 'tan' vorzeichen
9723  bb3e  ca              dex
9724  bb3f  d0 f7           bne init10
```

```
zeile adr.  obj.-code   source-code

9725  bb41  86 78            stx bits          shift-zähler
9726  bb43  86 1a            stx channl        speicherstelle für aktiven kanal
9727  bb45  86 16            stx dsdesc        länge disc-status string
9728  bb47  8e 58 02         stx dolu          vorgabe für ausgabe-adresse
9729  bb4a  86 1d            stx temmpt        zeiger auf zeitw. stringdescriptor
                                                (rel. offset)
9730  bb4c  ca               dex
9731  bb4d  86 1e            stx lastpt        addrl letzt benutzter string-descr.
9732  bb4f  86 1f            stx lastpt+1      addrh letzt benutzter string-descr.
9733  bb51  38               sec
9734  bb52  20 9c ff         jsr kmembt        unterste speichergrenze
                                                lesen/schreiben
9735  bb55  86 31            stx vartab        addrl zeiger anfang einfache
                                                variable
9736  bb57  84 32            sty vartab+1      addrh zeiger anfang einfache
                                                variable
9737  bb59  86 2d            stx txttab        addrl zeiger anfang  basic-text
9738  bb5b  84 2e            sty txttab+1      addrh zeiger anfang  basic-text
9739  bb5d  20 93 ba         jsr initop        initialisierung
9740  bb60  20 8c ba         jsr mapusr        umsch. auf bank # 1
9741  bb63  a0 00            ldy #$00
9742  bb65  98               tya
9743  bb66  91 2d            sta (txttab),y    addrl zeiger anfang  basic-text
9744  bb68  e6 2d            inc txttab        addrl zeiger anfang  basic-text
9745  bb6a  d0 02            bne init20
9746  bb6c  e6 2e            inc txttab+1      addrh zeiger anfang  basic-text
9747  bb6e  a2 26     init20 ldx #$26
9748  bb70  20 c3 a3         jsr msg           textzeiger + xr, textausgabe
9749  bb73  20 2b 8a         jsr scrtch
9750  bb76  a2 cc            ldx #$cc
9751  bb78  a0 bb            ldy #$bb
9752  bb7a  20 6f ff         jsr kvrese        netz ein/aus reset-vector
9753  bb7d  58               cli
9754  bb7e  4c c0 85         jmp ready         ausgabe 'ready', ready-status
9755  bb81
9756  bb81
9757  bb81  ===> text: '*** cbm basic 128,v4.0 ***' <===
9758  bb81
9759  bb81  93        signon .byte $93
9760  bb82  2a 2a 2a         .byte '*** commodore basic 128, v4.0 **'
9760  bb85  20 43 4f
9760  bb88  4d 4d 4f
9760  bb8b  44 4f 52
9760  bb8e  45 20 42
9760  bb91  41 53 49
9760  bb94  43 20 31
9760  bb97  32 38 2c
9760  bb9a  20 56 34
9760  bb9d  2e 30 20
9760  bba0  2a 2a
9761  bba2  2a               .byte '*', $0d, $0d, $00
9761  bba3  0d
9761  bba4  0d
9761  bba5  00
9762  bba6
9763  bba6
9764  bba6  ===> tabelle der standard-vektoren <===
9765  bba6
```

```
zeile adr.  obj.-code   source-code


9766  bba6  55 85     bvtrs  .word nerror     error-routine
9767  bba8  cd 85            .word nmain      hauptroutine basic-interpreter
9768  bbaa  c2 88            .word ncnnch     umwandlung text in basic-tokens
9769  bbac  f4 89            .word nqglop     umwandlung basic-tokens in text
9770  bbae  54 87            .word ngone      dispatcher-routine
9771  bbb0  b1 96            .word neval      folgenden num. ausdruck auswerten
9772  bbb2  c4 95            .word nfrmev     auswertung beliebiger ausdruck
9773  bbb4  2c ba            .word nchrgo     letztes zeichen erneut nach ac
9774  bbb6  32 ba            .word nchrge     nächstes zeichen nach ac
9775  bbb8  1e ba            .word flpint     umwandlung integer nach gleitkomma
9776  bbba  39 9d            .word givayf     umw. integer nach real
9777  bbbc
9778  bbbc
9779  bbbc  ===> standard-vektoren ins ram kopieren <===
9780  bbbc
9781  bbbc  a2 15     initv  ldx #$15
9782  bbbe  bd a6 bb  initv5 lda bvtrs,x      tabelle der standard-vektoren
9783  bbc1  9d 80 02         sta ierror,x     addrl fehler routine
9784  bbc4  ca               dex
9785  bbc5  10 f7            bpl initv5
9786  bbc7  60               rts
9787  bbc8
9788  bbc8
9789  bbc8  ad ff ff  initat lda hairq+1      6509 hardware-irq-vektor
9790  bbcb  60               rts
9791  bbcc
9792  bbcc
9793  bbcc  ===> basic-warm-start <===
9794  bbcc
9795  bbcc  78        warm   sei
9796  bbcd  20 7b ff         jsr kioini       initialisierung ein-/ausgabe
9797  bbd0  20 16 bc         jsr clall        alle logischen files schliessen
9798  bbd3  20 7e ff         jsr kcint        initialisierung bildschirm
9799  bbd6  20 8c ba         jsr mapusr       umsch. auf bank # 1
9800  bbd9  58               cli
9801  bbda  4c c0 85         jmp ready        ausgabe 'ready', ready-status
9802  bbdd
9803  bbdd                   .end
9804  bbdd                   .lib syscal
```

```
zeile adr.  obj.-code   source-code


9806  bbdd
9807  bbdd  ===> system-status in ac lesen <===
9808  bbdd
9809  bbdd  38        readst sec
9810  bbde  4c b7 ff  storst jmp kreast       ac in system-status schreiben
9811  bbe1
9812  bbe1
9813  bbe1  ===> log. file öffnen/befehl ausgeben <===
9814  bbe1
9815  bbe1  18        open   clc
9816  bbe2  20 c0 ff  opn10  jsr kopen        log. file öffnen/befehl ausgeben
9817  bbe5  b0 33            bcs ioerr        i/o fehlerbehandlung
9818  bbe7  60               rts
9819  bbe8
9820  bbe8
9821  bbe8  ===> eingabe 1 char vom akt. kanal (queue-vektor) <===
9822  bbe8
9823  bbe8  20 e4 ff  getin  jsr kgetin       eingabe 1 char von tastaturpuffer
9824  bbeb  b0 2d            bcs ioerr        i/o fehlerbehandlung
9825  bbed  60               rts
9826  bbee
9827  bbee
9828  bbee  ===> eingabe 1 char vom akt. kanal (chn-vektor) <===
9829  bbee
9830  bbee  20 cf ff  basin  jsr kbasin       eingabe 1 char vom akt. kanal
9831  bbf1  b0 27            bcs ioerr        i/o fehlerbehandlung
9832  bbf3  60               rts
9833  bbf4
9834  bbf4
9835  bbf4  ===> ausgabe 1 char auf akt. kanal <===
9836  bbf4
9837  bbf4  20 d2 ff  bsout  jsr kbsout       ausgabe 1 char auf akt. kanal
9838  bbf7  b0 21            bcs ioerr        i/o fehlerbehandlung
9839  bbf9  60               rts
9840  bbfa
9841  bbfa
9842  bbfa  ===> eingabekanal öffnen, fehlertest <===
9843  bbfa
9844  bbfa  20 c6 ff  chkin  jsr kchkin       eingabekanal öffnen
9845  bbfd  b0 1b            bcs ioerr        i/o fehlerbehandlung
9846  bbff  60               rts
9847  bc00
9848  bc00
9849  bc00  ===> ausgabekanal öffnen, fehlertest <===
9850  bc00
9851  bc00  20 c9 ff  chkout jsr kckout       ausgabekanal öffnen
9852  bc03  b0 15            bcs ioerr        i/o fehlerbehandlung
9853  bc05  60               rts
9854  bc06
9855  bc06
9856  bc06  ===> einlesen vom logischen file <===
9857  bc06
9858  bc06  20 d5 ff  load   jsr kload        einlesen vom logischen file
9859  bc09  b0 0f            bcs ioerr        i/o fehlerbehandlung
9860  bc0b  60               rts
9861  bc0c
9862  bc0c
9863  bc0c  ===> abspeichern auf logisches file <===
```

```
zeile adr.  obj.-code   source-code


9864  bc0c
9865  bc0c  20 d8 ff  save   jsr ksave        abspeichern auf logisches file
9866  bc0f  b0 09             bcs ioerr       i/o fehlerbehandlung
9867  bc11  60                rts
9868  bc12
9869  bc12
9870  bc12  ===> carry=1, log. file schliessen <===
9871  bc12
9872  bc12  38        close   sec
9873  bc13  4c c3 ff  tclose  jmp kclose      logisches file schliessen
9874  bc16
9875  bc16
9876  bc16  ===> alle logischen files schliessen <===
9877  bc16
9878  bc16  18        clall   clc
9879  bc17  4c e7 ff          jmp kclall      alle logischen files schliessen
9880  bc1a
9881  bc1a
9882  bc1a  ===> i/o fehlerbehandlung <===
9883  bc1a
9884  bc1a  18        ioerr   clc
9885  bc1b  2a                rol a
9886  bc1c  48                pha
9887  bc1d  a9 0e             lda #$0e
9888  bc1f  18                clc
9889  bc20  20 13 bc          jsr tclose
9890  bc23  68                pla
9891  bc24  aa                tax
9892  bc25  4c 52 85          jmp error       ind. jmp zur fehlerroutine
9893  bc28
9894  bc28
9895  bc28  ===> prüfbyte rom $a000-$bfff <===
9896  bc28
9897  bc28  b5        chkbt2  .byte $b5
9898  bc29
9899  bc29
9900  bc29                    * = *+$23d7
9901  e000
9902  e000                    .end
9903  e000
9904  e000                    .end


fehler in pass 1 = 0000
fehler in pass 2 = 0000

assemblierung ende
```

4. TEIL

CROSSREFERENZLISTE 1

CBM 610/710

BASIC 4.0

```
dosds2 $0216   126  8819  8956  9197
dosdsk $0008   299
dosf1a $0212   123  8879  8881  9271  9273  9423  9424  9443  9444
dosf1l $0210   121  3112  8876  9276  9420  9431  9449
dosf2a $0217   127  8974  8975  9283  9285
dosf2l $0215   125  8973  9288
dosfa  $0220   133  3115  3120  3249  3311  3481  9014  9435  9440  9457
dosfb1 $0214   124  8883  9275  9426  9445
dosfb2 $0219   128  8977  9287
dosfnl $0012   298   302   302   302
dosla  $021f   132  3118  3305  3447  3477  8775  9433  9456
doslfn $000e   300
doslst $0046   302
dosofh $021d   131  3357  3358  8854  8855
dosofl $021b   130  3351  3352  3370  3371  8710  8837  8838
dospar $b547  3162  3378  3409  3491  3499  3514  3531  8696
dosprs $b549  3281  3300  3318  3326  3541  3551  8697
dosprx $b54b  3349  3366  8698
dosrcl $0222   135  3440  3474  8797  9181  9254  9259
dossa  $0221   134  3113  3125  3127  3289  3480  9230  9432  9442  9458
dosstr $0226   138  9207  9246  9263  9302
dostbl $b540  8687  8709
drv1   $b602  8754  8813
drv2   $b6d1  8938  8951
dsave  $93c3   411  3317
dscpnt $005e    60  6617  6620  6621  6863  6864  6865  6874  6891  7046
              7047  7048  7074  7080  7110  7139  7164  7166  7168  9564
dsctmp $0070    66  4143  4161  4165  5673  5684  6627  6662  6689  6907
              6992
dsdesc $0016    27  1763  3397  3423  3929  3930  3931  3948  4019  4028
              6968  6970  6972  9366  9370  9372  9414  9636  9640  9679
              9693  9696  9698  9727
dsptmp $029e   185  2459  2464  2466
dtlp0  $8fce  2706  2714
dtlp1  $8fcf  2709  2715
dv0err $a148  5503  5559
dvarts $9ab3  4402  4426
e6509  $0000     9
eadj   $b23c  8158  8174
eado   $b23a  8144  8173  8303
eatem  $9951  4224  4225  4227
ebase  $828f   588  6015  6016  7264  7266
eex2   $b064  7942  7982
efo    $b497  8447  8493  8517
end    $8baa   365  1984
endchr $000d    18  2247  2248  2251  2256  2789  4101  4106  6640  6656
endfd  $0272   166  7903  8388  8420  8507  8517  8518
endre1 $960e  3644  3648
endrel $95fd  3624  3626  3638
endvar $90cb  2824  2858
enr    $0260   148  8026  8051  8139  8259  8272  8305  8317
eoa    $b12d  8014  8040
eoi    $0020   261   268
eoist  $0040   270   278
eom    $0080   234
eormsk $0256   140  2962  2965  4094  4097  4100  4105  4107  4110  4112
erflp1 $ac22  7271  7275  7277
erflp2 $ac36  7282  7287  7299
ero    $b435  8457  8486  8488  8505
errchl $b922  3392  3416  3950  9354
errd   $ac00  4119  7255
errd1  $ac2e  7273  7278
```

5. TEIL

ASSEMBLERLISTING 2

CBM 610/710

OPERATINGSYSTEM

```
b710os.lib......seite 0001

zeile adr.  obj.-code   source-code


 pass 1


0004  0000                    .lib b710os.lab
0191  0000                    .lib b710os.io
0468  0000                    .lib b710os.con
0563  0000                    .lib editor1
1374  e48d                    .lib editor2
2183  e985                    .lib editor3
2631  ed0c                    .lib monitor
3270  f1c3                    .lib kernal1
3974  f67f                    .lib kernal2
4694  fb34                    .lib kernal3

 pass 2


0001  0000  610/710 operatingsystem  library-file
0002  0000
0003  0000
0004  0000                    .lib b710os.lab
```

```
zeile adr.  obj.-code   source-code


0006  0000
0007  0000  ===> externe zero-page-label <===
0008  0000
0009  0000            e6509  = $00          6509 execution register
0010  0000            i6509  = $01          6509 indirection register
0011  0000            usrpok = $02          jmp code für 'usr'-routine
0012  0000            fnadr  = $90          addrl akt. filename
0013  0000            sal    = $93          addrl akt. load/store adresse
0014  0000            sah    = $94          addrh akt. load/store adresse
0015  0000            sas    = $95          bank akt. load/store adresse
0016  0000            eal    = $96          addrl ende akt. load/store
0017  0000            eah    = $97          addrh ende akt. load/store
0018  0000            eas    = $98          bank ende akt. load/store
0019  0000            stal   = $99          addrl anfang akt. load/store
0020  0000            stah   = $9a          addrh anfang akt. load/store
0021  0000            stas   = $9b          bank anfang akt. load/store
0022  0000            status = $9c          i/o operation status
0023  0000            fnlen  = $9d          länge akt. filename
0024  0000            la     = $9e          akt. logische filenummer
0025  0000            fa     = $9f          akt. primäradresse (geräte-nummer)
0026  0000            sa     = $a0          akt. sekundäradresse
0027  0000            dftln  = $a1          vorgabe input-grätenummer
0028  0000            dflto  = $a2          vorgabe output-gerätenummer
0029  0000            tape1  = $a3          addrl kassettenpuffer
0030  0000            ribuf  = $a6          addrl rs232 eingabe-puffer
0031  0000            stkey  = $a9          stop tasten flag
0032  0000            c3po   = $aa          ieee puffer flag
0033  0000            bsour  = $ab          ieee zeichen puffer
0034  0000            ipoint = $ac          addrl ram indirekt-pointer
0035  0000            pch    = $ae          addrh programmzähler pc
0036  0000            pcl    = $af          addrl programmzähler pc
0037  0000            sp     = $b4          stack-pointer sp
0038  0000            xi6509 = $b5          alte indirection bank
0039  0000            invh   = $b7          addrh user interrupt-vektor
0040  0000            invl   = $b8          addrl user interrupt-vektor
0041  0000            tmp1   = $b9          addrl monitor zwei-byte zeiger 1
0042  0000            tmp2   = $bb          addrl monitor zwei-byte zeiger 2
0043  0000            tmpc   = $bd          ablage letzter monitor-befehl
0044  0000            t6509  = $be          laufende 6509 indirection bank
0045  0000            ddisk  = $bf          lfd. disk-primäradr. für monitor
0046  0000            pkybuf = $c0          addrl start programmierbare tasten
0047  0000            keypnt = $c2          addrl akt. pgm tastenpuffer
0048  0000            sedsal = $c4          addrl scroll-zeiger start
0049  0000            sedeal = $c6          addrl scroll-zeiger ende
0050  0000            pnt    = $c8          addrl zeiger auf akt. zeichen im
                                            video-ram
0051  0000            tblx   = $ca          cursor zeile
0052  0000            pntr   = $cb          cursor spalte
0053  0000            grmode = $cc          flag für grafik/text-modus
0054  0000            lstx   = $cd          index letztes zeichen
0055  0000            lstp   = $ce          screen edit startposition
0056  0000            lsxp   = $cf          letzte position (zeile)
0057  0000            crsw   = $d0          flag 'cr' bei bs-eingabe
0058  0000            ndx    = $d1          anzahl bytes im tastaturpuffer
0059  0000            qtsw   = $d2          flag für anführungszeichen-modus
0060  0000            insrt  = $d3          insert flag
0061  0000            config = $d4          cursor type / char. vor blinken
0062  0000            indx   = $d5          position letztes zeichen in zeile
```

```
zeile adr.  obj.-code   source-code

0063  0000            kyndx  = $d6          zeichenzähler für pgm tastenbefehl
0064  0000            rptcnt = $d7          zeit zwischen zeichen bei repeat
0065  0000            delay  = $d8          zeit bis zum einsetzen des repeat
0066  0000            sedt1  = $d9          mehrfach benutzte editorvariable
0067  0000            sedt2  = $da          mehrfach benutzte editorvariable
0068  0000            adata  = $db          akt. ausgabezeichen
0069  0000            sctop  = $dc          oberste zeile bildschirm (0-25)
0070  0000            scbot  = $dd          unterste zeile bildschirm (0-25)
0071  0000            sclf   = $de          linker rand bildschirm
0072  0000            scrt   = $df          rechter rand bildschirm
0073  0000            modkey = $e0          flag shift/control-taste
0074  0000            norkey = $e1          flag normale taste
0075  0000            bitabl = $e2          tabelle basic-doppelzeilen (4 byte)
0076  0000
0077  0000
0078  0000  ===> externe ram-label <===
0079  0000
0080  0000            stack  = $0100        6509 cpu-stack
0081  0000            buf    = $0200        basic input-puffer -$2ff
0082  0000            cinv   = $0300        addrl irq vector
0083  0000            cbinv  = $0302        addrl brk-vector
0084  0000            nminv  = $0304        addrl nmi-vector
0085  0000            iopen  = $0306        addrl open file-vector
0086  0000            iclose = $0308        addrl close-vector
0087  0000            ichkin = $030a        addrl open eingabekanal-vector
0088  0000            ickout = $030c        addrl open ausgabekanal-vector
0089  0000            iclrch = $030e        addrl close kanal-vector
0090  0000            ibasin = $0310        addrl eingabe akt. kanal-vector
0091  0000            ibsout = $0312        addrl ausgabe akt. kanal-vector
0092  0000            istop  = $0314        addrl prüfe stoptaste-vector
0093  0000            igetin = $0316        addrl char aus tast.-puffer vector
0094  0000            iclall = $0318        addrl alle files schliessen-vector
0095  0000            iload  = $031a        addrl lade vom file-vector
0096  0000            isave  = $031c        addrl save auf file-vector
0097  0000            usrcmd = $031e        addrl monitor erweiterung-vector
0098  0000            escvec = $0320        addrl user esc-tasten-vector
0099  0000            ctlvec = $0322        addrl unbenuzte 'ctrl' tasten-vector
0100  0000            isecnd = $0324        addrl ausgabe sek.adr. nach listen
0101  0000            itksa  = $0326        addrl akt. sekundäradr. auf iec-bus
0102  0000            iacptr = $0328        addrl zeichen von iec-bus nach ac
0103  0000            iciout = $032a        addrl ac auf iec-bus ausgeben
0104  0000            iuntlk = $032c        addrl untalk auf iec-bus ausgeben
0105  0000            iunlsn = $032e        addrl unlisten auf iec-bus ausgeben
0106  0000            ilistn = $0330        addrl listen auf iec-bus ausgeben
0107  0000            italk  = $0332        addrl talk auf iec-bus ausgeben
0108  0000            lat    = $0334        tabelle akt. log. file # (10 byte)
0109  0000            fat    = $033e        tabelle akt. primäradr.
0110  0000            sat    = $0348        tabelle akt. sekundäradr.
0111  0000            lowadr = $0352        addrl anfang des system speichers
0112  0000            hiadr  = $0355        addrl ende des system speichers
0113  0000            memstr = $0358        addrl anfang des benutzer speichers
0114  0000            memsiz = $035b        addrl ende des benutzer speichers
0115  0000            timout = $035e        ieee timeout enable flag
0116  0000            verck  = $035f        load/verify flag
0117  0000            ldtnd  = $0360        zeiger in file-tabellen
0118  0000            msgflg = $0361        message flag
0119  0000            t1     = $0363        kernalvariable temporär
0120  0000            t2     = $0364        kernalvariable temporär
```

```
zeile adr.  obj.-code   source-code

0121  0000            xsav   = $0365          kernalvariable temporär
0122  0000            savx   = $0366          kernalvariable temporär
0123  0000            alarm  = $0369          flag für irq von 6526
0124  0000            itape  = $036a          addrl indir. cassette
0125  0000            relsal = $036f          addrl variable start-load-adr.
0126  0000            relsah = $0370          addrh variable start-load-adr.
0127  0000            relsag = $0371          bank variable start-load-adr.
0128  0000            cas1   = $0375          cassette switchflag
0129  0000            m51ctr = $0376          rs232: kopie 6551 control-reg.
0130  0000            m51cdr = $0377          rs232: kopie 6551 command-reg.
0131  0000            rsstat = $037a          rs232: akt. rs232-status
0132  0000            dcdsr  = $037b          rs232: letzter dcd/dsr wert
0133  0000            ridbs  = $037c          rs232: eingabe start zeiger
0134  0000            ridbe  = $037d          rs232: eingabe end zeiger
0135  0000            pkyend = $0380          addrl endadr. funkt.tastenpuffer
0136  0000            keyseg = $0382          ram-segment # für funktionstasten
0137  0000            keysiz = $0383          funktionstasten (20bytes)
0138  0000            rvs    = $0397          revers flag
0139  0000            lintmp = $0398          puffer für bs-zeilen #
0140  0000            lstchr = $0399          letztes gedrucktes zeichen
0141  0000            insflg = $039a          auto insert flag
0142  0000            scrdis = $039b          scroll disable flag
0143  0000            bitmsk = $039c          bit mask/funkt.tasten temporär
0144  0000            keyidx = $039d          index programmierbare befehle
0145  0000            logscr = $039e          logical/physical scroll flag
0146  0000            bellmd = $039f          flag für glocke am zeilenende
0147  0000            pagsav = $03a0          zeitweise ram page
0148  0000            tab    = $03a1          tabstop flags (max.10byt/80bit)
0149  0000            keyd   = $03ab          tastaturpuffer (10 bytes)
0150  0000            funvec = $03b5          addrl ind. jmp für funktionstasten
0151  0000            sedt3  = $03b7          temporär list funktionstasten
0152  0000            evect  = $03f8          addrl warmstart-vector
0153  0000            ipjtab = $0810          coprozessor-jump tabelle
0154  0000            ipptab = $0910          coprozessor-parameter tabelle
0155  0000
0156  0000
0157  0000  ===> externe rom-label <===
0158  0000
0159  0000            pulst1 = $9691
0160  0000
0161  0000
0162  0000  ===> bildschirm-label <===
0163  0000
0164  0000            linz0  = $d000          bildschirm-adr. zeile  1
0165  0000            linz1  = $d050          bildschirm-adr. zeile  2
0166  0000            linz2  = $d0a0          bildschirm-adr. zeile  3
0167  0000            linz3  = $d0f0          bildschirm-adr. zeile  4
0168  0000            linz4  = $d140          bildschirm-adr. zeile  5
0169  0000            linz5  = $d190          bildschirm-adr. zeile  6
0170  0000            linz6  = $d1e0          bildschirm-adr. zeile  7
0171  0000            linz7  = $d230          bildschirm-adr. zeile  8
0172  0000            linz8  = $d280          bildschirm-adr. zeile  9
0173  0000            linz9  = $d2d0          bildschirm-adr. zeile 10
0174  0000            linz10 = $d320          bildschirm-adr. zeile 11
0175  0000            linz11 = $d370          bildschirm-adr. zeile 12
0176  0000            linz12 = $d3c0          bildschirm-adr. zeile 13
0177  0000            linz13 = $d410          bildschirm-adr. zeile 14
0178  0000            linz14 = $d460          bildschirm-adr. zeile 15
```

```
zeile adr.  obj.-code   source-code


0179  0000            linz15 = $d4b0          bildschirm-adr. zeile 16
0180  0000            linz16 = $d500          bildschirm-adr. zeile 17
0181  0000            linz17 = $d550          bildschirm-adr. zeile 18
0182  0000            linz18 = $d5a0          bildschirm-adr. zeile 19
0183  0000            linz19 = $d5f0          bildschirm-adr. zeile 20
0184  0000            linz20 = $d640          bildschirm-adr. zeile 21
0185  0000            linz21 = $d690          bildschirm-adr. zeile 22
0186  0000            linz22 = $d6e0          bildschirm-adr. zeile 23
0187  0000            linz23 = $d730          bildschirm-adr. zeile 24
0188  0000            linz24 = $d780          bildschirm-adr. zeile 25
0189  0000
0190  0000                   .end
0191  0000                   .lib b710os.io
```

```
zeile adr.  obj.-code   source-code


0193  0000
0194  0000  ===>  6509  microprozessor  <===
0195  0000
0196  0000  dieser mikroprozessor unterscheidet sich von seinem bekannten bruder
0197  0000  6502 nur durch die möglichkeit, 1 megabyte zu adressieren. der
0198  0000  befehlssatz ist identisch. es werden beim 6509 lediglich bei zwei
0199  0000  befehlen alle 20 adressleitungen ausgewertet (zero-page byte 0/1),
0200  0000  die verwendete taktfrequenz ist verdoppelt (2 mhz).
0201  0000
0202  0000
0203  0000  ===>  6845 video display controller 'vdc' bzw. <===
0204  0000  ===>  6545 cathode ray tube controller  'crtc' <===
0205  0000
0206  0000  mit dem 'vdc' wird die bildschirmausgabe kontrolliert und verwaltet.
0207  0000  er ist das bindeglied zwischen dem bildschirmspeicher und bildschirm.
0208  0000
0209  0000
0210  0000           vdc    = $d800          basis-adresse des 6845-vdc/6545-crtc
0211  0000
0212  0000           adreg  = 0              adress-register
0213  0000           dareg  = 1              daten-register
0214  0000
0215  0000  -- mögliche werte des adress-registers --
0216  0000
0217  0000           r00    = 0              horizontal total
0218  0000           r01    = 1              horizontal displayed
0219  0000           r02    = 2              horizontal synch position
0220  0000           r03    = 3              horizontal + vertical synch widths
0221  0000           r04    = 4              vertical total
0222  0000           r05    = 5              vertical total adjust
0223  0000           r06    = 6              vertical displayed
0224  0000           r07    = 7              vertical synch position
0225  0000           r08    = 8              mode control
0226  0000           r09    = 9              scan line
0227  0000           r10    = 10             cursor start
0228  0000           r11    = 11             cursor end
0229  0000           r12    = 12             addrh display start address
0230  0000           r13    = 13             addrl display start address
0231  0000           r14    = 14             addrh cursor position
0232  0000           r15    = 15             addrl cursor position
0233  0000           r16    = 16             addrh light-pen
0234  0000           r17    = 17             addrl light-pen
0235  0000
0236  0000
0237  0000  ===>  6581 sound interface device 'sid' <===
0238  0000
0239  0000  dieser dreistimmige synthesizer bietet umfangreiche möglichkeiten
0240  0000  zur erzeugung von tönen und musik. es kann für jeden der drei
0241  0000  oszillatoren die wellenform, frequenz, lautstärke, anstiegs- und
0242  0000  abfallzeit, sowie ein filter programmiert werden.
0243  0000
0244  0000
0245  0000           sid    = $da00          basis-adresse des 6581-sid
0246  0000
0247  0000  -- offsetadressen der oszillatoren  --
0248  0000
0249  0000           osc1   = 0              offset-adresse 6581-oszillator 1
0250  0000           osc2   = 7              offset-adresse 6581-oszillator 2
```

```
zeile adr.  obj.-code   source-code


0251  0000            osc3   = 14             offset-adresse 6581-oszillator 3
0252  0000
0253  0000  -- register der oszillatoren  --
0254  0000
0255  0000            freqlo = 0              addrl frequenz oszillator
0256  0000            freqhi = 1              addrh frequenz oszillator
0257  0000            pulsef = 2              addrl pulsbreite oszillator
0258  0000            pulsec = 3              addrh pulsbreite oszillator
0259  0000            oscctl = 4              steuerregister oszillator
0260  0000            atkdcy = 5              attack/decay-time oszillator
0261  0000            susrel = 6              sustain/release time oszillator
0262  0000
0263  0000  --  offsetadressen filter  --
0264  0000
0265  0000            fclow  = 21             addrl filterfrequenz
0266  0000            fchi   = 22             addrh filterfrequenz
0267  0000            resnce = 23             filter-wahlschalter und
                                              resonanzeinstellung
0268  0000            volume = 24             lautstärkeregler + wahlschalter
                                              tief-, band-, hochpaß
0269  0000
0270  0000  -- restliche offsetadressen  --
0271  0000
0272  0000            potx   = 25             a/d-wandler 1
0273  0000            poty   = 26             a/d-wandler 2
0274  0000            random = 27             rauschgenerator
0275  0000            cenv3  = 28             hüllkurvengenerator
0276  0000
0277  0000
0278  0000  ===>  6526 complex interface adapter 'cia' <===
0279  0000
0280  0000  die zwei eingebauten 'complex interface adapter' steuern über ihre
0281  0000  ports die angeschlossenen einheiten (z.b. user-port, ieee-daten,
0282  0000  coprozessor 'z80 zilog' und '8088 intel'). in den '6526' stehen
0283  0000  auch noch zwei 16-bit-zähler und eine 24-stunden-uhr zur verfügung.
0284  0000
0285  0000
0286  0000  --  offsetadressen der register --
0287  0000
0288  0000            pra    = 0              6526 cia periph. data reg. a (pra)
0289  0000            prb    = 1              6526 cia periph. data reg. b (prb)
0290  0000            ddra   = 2              6526 cia data direct. reg. a (ddra)
0291  0000            ddrb   = 3              6526 cia data direct. reg. b (ddrb)
0292  0000            talo   = 4              6526 cia timer a lo reg. (ta lo)
0293  0000            tahi   = 5              6526 cia timer a hi reg. (ta hi)
0294  0000            tblo   = 6              6526 cia timer b lo reg. (tb lo)
0295  0000            tbhi   = 7              6526 cia timer b hi reg. (tb hi)
0296  0000            tod10  = 8              6526 cia uhrzeit: 1/10 sek. reg.
0297  0000            todsec = 9              6526 cia uhrzeit: sekunden reg.
0298  0000            todmin = 10             6526 cia uhrzeit: minuten reg.
0299  0000            todhr  = 11             6526 cia uhrzeit: std. am/pm reg.
0300  0000            sdr    = 12             6526 cia serial data register (sdr)
0301  0000            icr    = 13             6526 cia irq control reg. (icr)
0302  0000            cra    = 14             6526 cia control register a (cra)
0303  0000            crb    = 15             6526 cia control register b (crb)
0304  0000
0305  0000
0306  0000  ===>  cia 1 für coprozessor-communication (z80/8088)  <===
```

```
zeile adr.  obj.-code   source-code


0307  0000
0308  0000            ipcia  = $db00          basis-adresse cia 1
0309  0000
0310  0000  -- belegung der einzelnen bits --
0311  0000
0312  0000  pra   = data port
0313  0000
0314  0000  prb0: busy1 (1 => 6509 off dbus)
0315  0000  prb1: busy2 (1 => 8088/z80 off dbus)
0316  0000  prb2: signalflag 8088/z80
0317  0000  prb3: signalflag 6509
0318  0000  prb4: unbenutzt
0319  0000  prb5: unbenutzt
0320  0000  prb6: irq to 8088/z80 (lo)
0321  0000  prb7: unbenutzt
0322  0000
0323  0000  -- bit-offset port b --
0324  0000
0325  0000            sem88  = 4              prb bit2
0326  0000            sem65  = 8              prb bit3
0327  0000
0328  0000
0329  0000  ===>  cia 2 für ieee-daten, user-port, spiele usw.  <===
0330  0000
0331  0000            cia    = $dc00          basis-adresse cia 2
0332  0000
0333  0000  -- belegung der einzelnen bits --
0334  0000
0335  0000  timer a: ieee local / cass. local / music / game
0336  0000  timer b: ieee deadm / cass. deadm / music / game
0337  0000
0338  0000  pra0: ieee data1 / user / paddle game 1
0339  0000  pra1: ieee data2 / user / paddle game 2
0340  0000  pra2: ieee data3 / user
0341  0000  pra3: ieee data4 / user
0342  0000  pra4: ieee data5 / user
0343  0000  pra5: ieee data6 / user
0344  0000  pra6: ieee data7 / user / game trigger 14
0345  0000  pra7: ieee data8 / user / game trigger 24
0346  0000
0347  0000  prb0: user / game 10
0348  0000  prb1: user / game 11
0349  0000  prb2: user / game 12
0350  0000  prb3: user / game 13
0351  0000  prb4: user / game 20
0352  0000  prb5: user / game 21
0353  0000  prb6: user / game 22
0354  0000  prb7: user / game 23
0355  0000
0356  0000  flag: user / cassette read
0357  0000  pc  : user
0358  0000  ct  : user
0359  0000  sp  : user
0360  0000
0361  0000
0362  0000  ===>  6551 asynchronius communications interface adapter  <===
0363  0000  ===>       'acia' rs 232-c schnittstellen-controller    <===
0364  0000
```

```
zeile adr.  obj.-code   source-code


0365  0000  dieser baustein übernimmt die steuerung der 'rs 232-c' schnitt-
0366  0000  stelle. es sind übertragungsraten von 50 bis 19200 baud einstellbar.
0367  0000  er überwacht den datentransfer und wandelt die zu übertragende
0368  0000  information in das gewünschte format (bit, parität usw.) um.
0369  0000
0370  0000
0371  0000            acia    = $dd00          basis-adresse 6551 acia
0372  0000
0373  0000  --  offset-adressen der register  -
0374  0000
0375  0000            drsn    = 0              data register (write/read)
0376  0000            srsn    = 1              reset bei write, status bei read
0377  0000            cdr     = 2              command-register
0378  0000            ctr     = 3              control-register
0379  0000
0380  0000
0381  0000  ===>  6525 triport interface device 'tpi' <===
0382  0000
0383  0000  dieser baustein verfügt über 24 eingabe/ausgabe- und 2 handshake-
0384  0000  leitungen, sowie 5 unterbrechungseingängen. es werden damit die
0385  0000  tastatur und der ieee-bus kontrolliert.
0386  0000
0387  0000
0388  0000  -- offset-adressen der register  --
0389  0000
0390  0000            pa      = 0              port register a
0391  0000            pb      = 1              port register b
0392  0000            pc      = 2              port register c
0393  0000            ddpa    = 3              data direction register a
0394  0000            ddpb    = 4              data direction register b
0395  0000            ddpc    = 5              data direction register c
0396  0000            creg    = 6              control register
0397  0000            air     = 7              active interrupt register
0398  0000
0399  0000
0400  0000  ===>  6525 triport 1  für ieee-control  <===
0401  0000             network / vic / irq
0402  0000
0403  0000            tri1    = $de00          basis-adresse triport 1
0404  0000
0405  0000  --  belegung der einzelnen port-bits  --
0406  0000
0407  0000  pa0: ieee dc control (ti parts)
0408  0000  pa1: ieee te control (ti parts) (t/r)
0409  0000  pa2: ieee ren
0410  0000  pa3: ieee atn
0411  0000  pa4: ieee dav
0412  0000  pa5: ieee eoi
0413  0000  pa6: ieee ndac
0414  0000  pa7: ieee nrfd
0415  0000
0416  0000  pb0: ieee ifc
0417  0000  pb1: ieee srq
0418  0000  pb2: network transmitter enable
0419  0000  pb3: network receiver enable
0420  0000  pb4: arbitration logic switch
0421  0000  pb5: cassette write
0422  0000  pb6: cassette motor
```

```
zeile adr.  obj.-code   source-code


0423  0000  pb7: cassette switch
0424  0000
0425  0000  irq0: 50/60 hz irq
0426  0000  irq1: ieee srq
0427  0000  irq2: 6526 irq
0428  0000  irq3: (opt) 6526 inter-processor
0429  0000  irq4: 6551
0430  0000
0431  0000
0432  0000  ===>  6525 triport 2  für keyboard  <===
0433  0000             und vic 16k control
0434  0000
0435  0000           tri2   = $df00          basis-adresse triport 2
0436  0000
0437  0000  --  belegung der einzelnen bits  --
0438  0000
0439  0000  pa0: keyboard out 8
0440  0000  pa1: keyboard out 9
0441  0000  pa2: keyboard out 10
0442  0000  pa3: keyboard out 11
0443  0000  pa4: keyboard out 12
0444  0000  pa5: keyboard out 13
0445  0000  pa6: keyboard out 14
0446  0000  pa7: keyboard out 15
0447  0000
0448  0000  pb0: keyboard out 0
0449  0000  pb1: keyboard out 1
0450  0000  pb2: keyboard out 2
0451  0000  pb3: keyboard out 3
0452  0000  pb4: keyboard out 4
0453  0000  pb5: keyboard out 5
0454  0000  pb6: keyboard out 6
0455  0000  pb7: keyboard out 7
0456  0000
0457  0000  pc0: keyboard in 0
0458  0000  pc1: keyboard in 1
0459  0000  pc2: keyboard in 2
0460  0000  pc3: keyboard in 3
0461  0000  pc4: keyboard in 4
0462  0000  pc5: keyboard in 5
0463  0000
0464  0000  pc6: vic 16k bank select low
0465  0000  pc7: vic 16k bank select hi
0466  0000
0467  0000                  .end
0468  0000                  .lib b710os.con
```

```
zeile adr.  obj.-code   source-code


0470  0000
0471  0000   ===>  ascii- und asc-codes  <===
0472  0000
0473  0000            lf      = $0a             ascii für 'line feed'
0474  0000            cr      = $0d             ascii für 'carriage return'
0475  0000            clrscr  = 147             asc für 'clear screen'
0476  0000            home    = 197             asc für 'cursor home'
0477  0000
0478  0000            eom     = $80             end-of-message-kennung
0479  0000
0480  0000
0481  0000   ===>  system-konstanten und parameter  <===
0482  0000
0483  0000            forsiz = 19
0484  0000            numlev = 26
0485  0000            bufsiz = 161
0486  0000            addprc = 1
0487  0000            addpr2 = addprc+addprc
0488  0000            addpr4 = addpr2+addpr2
0489  0000            addpr8 = addpr4+addpr4
0490  0000            stkend = $01fb            erstes byte für stack
0491  0000            clmwid = 10               spaltenbreite für tabulation mit ','
0492  0000            pi     = $ff              basic-token für 'pi'
0493  0000            strsiz = 4                anzahl speicherstellen für
                                                string-descriptor
0494  0000            ptrsiz = 3                anzahl speicherstellen für
                                                speicher-zeiger
0495  0000            numtmp = 3
0496  0000
0497  0000
0498  0000   ===>  ieee-konstanten und parameter  <===
0499  0000
0500  0000            dc      = $01             75160/75161 control line
0501  0000            te      = $02             75160/75161 control line
0502  0000            ren     = $04             remote enable
0503  0000            attn    = $08             attention
0504  0000            dav     = $10             data available
0505  0000            eoi     = $20             end or identify
0506  0000            ndac    = $40             not data accepted
0507  0000            nrfd    = $80             not ready for data
0508  0000            ifc     = $01             interface clear
0509  0000            srq     = $02             service request
0510  0000
0511  0000            rddb    = nrfd+ndac+te+dc+ren  direction for receiver
0512  0000            tddb    = eoi+dav+attn+te+dc+ren direction for transmit
0513  0000
0514  0000            eoist   = $40             eoi status test
0515  0000            tlkr    = $40             device is talker
0516  0000            lstnr   = $20             device is listener
0517  0000            utlkr   = $5f             device untalk
0518  0000            ulstn   = $3f             device unlisten
0519  0000
0520  0000            toout   = $01             timeout status on output
0521  0000            toin    = $02             timeout status on input
0522  0000            eoist   = $40             eoi on input
0523  0000            nodev   = $80             no device on bus
0524  0000            sperr   = $10             verify error
0525  0000
```

```
zeile adr.  obj.-code   source-code

0526  0000            slock  = $40            screen editor lock-out
0527  0000            dibf   = $80            data in output buffer
0528  0000
0529  0000
0530  0000  ===>  vorbelegte speicher-segmente  <===
0531  0000
0532  0000            sysbnk = $0f            nummer der system-bank
0533  0000            mxbank = sysbnk+1       erste nicht erreichbare bank
0534  0000            txtbnk = 1              bank für basic-programme
0535  0000            varbnk = 3              bank für einfache variablen
                                              (256k-version)
0536  0000            arybnk = 2              bank für variablenfelder
                                              (256k-version)
0537  0000            strbnk = 4              bank für strings (256k-version)
0538  0000
0539  0000
0540  0000  ===>  dos-interface-konstanten  <===
0541  0000
0542  0000            dosfnl = 16+2           länge von filenamen
0543  0000            dosdsk = 8              vorgabewert für disc-gerätenummer
0544  0000            doslfn = 14             dos internal logical file nummer
0545  0000            dosctl = 21
0546  0000            doslst = dosfnl+dosfnl+dosfnl+16
0547  0000
0548  0000
0549  0000  ===>  kernal coprozessor communications-variablen  <===
0550  0000
0551  0000            ipb    = $0800          basisadresse ipc-puffer
0552  0000            ipbsiz = 13             kernal inter process communication
                                              puffer-länge
0553  0000
0554  0000  --->  offsets im ipc-puffer  <---
0555  0000
0556  0000            ipccmd = 0              ipc befehl
0557  0000            ipcjmp = 1              ipc sprung-adresse
0558  0000            ipcin  = 3              anzahl ipc eingabe-bytes
0559  0000            ipcout = 4              anzahl ipc ausgabe-bytes
0560  0000            ipcdat = 5              ipc data-puffer (8 bytes)
0561  0000
0562  0000                   .end
0563  0000                   .lib editor1
```

```
zeile adr.  obj.-code   source-code


0565  0000
0566  0000                    * = $e000
0567  e000
0568  e000  ===> 'jmp' sprung-tabelle <===
0569  e000
0570  e000  4c 09 ee  jmonon jmp monoff       monitor-kaltstart (basic aus)
0571  e003
0572  e003  ea                nop
0573  e004
0574  e004  4c 44 e0  jcint  jmp cint         bildschirm-, funkt.tasten-reset
0575  e007
0576  e007  4c fe e0  jlp2   jmp lp2          zeichen aus tastenpuffer in ac
0577  e00a
0578  e00a  4c 79 e1  jloop5 jmp loop5        char von bs holen (get o. input)
0579  e00d
0580  e00d  4c 99 e2  jprt   jmp prt          zeichen aus ac auf bildschirm
0581  e010
0582  e010  4c 3f e0  jscror jmp scrorg       max.anzahl: spalten=xr, zeilen=yr
0583  e013
0584  e013  4c 65 e8  jkey   jmp key          tastatur lesen/in puffer schreiben
0585  e016
0586  e016  4c da e0  jmvcur jmp movcur       cursor-pos. in reg. 14/15 vdc 6851
0587  e019
0588  e019  4c 25 e0  jplot  jmp plot         akt. x,y-pos. des cursors lesen /
                                              schreiben
0589  e01c
0590  e01c  4c 3a e0  jiobas jmp iobase       basisadr. i/o-gerât nach xr/yr
0591  e01f
0592  e01f  4c 70 e9  jescrt jmp escape       bearbeitung von escape-funktionen
0593  e022
0594  e022  4c f8 e6  jfunky jmp keyfun       vector für funktionstasten
0595  e025
0596  e025
0597  e025  ===> akt. x,y-pos. des cursors lesen / schreiben <===
0598  e025
0599  e025  b0 0e     plot   bcs rdplt        cursorzeile in xr, spalte in yr
0600  e027  86 ca            stx tblx         cursor zeile
0601  e029  86 cf            stx lsxp         letzte position (zeile)
0602  e02b  84 cb            sty pntr         cursor spalte
0603  e02d  84 ce            sty lstp         screen edit startposition
0604  e02f  20 cd e0         jsr stupt        bs-zeiger auf cursor-zeile
0605  e032  20 da e0         jsr movcur       cursor-pos. in reg. 14/15 vdc 6851
0606  e035
0607  e035
0608  e035  ===> cursorzeile in xr, spalte in yr <===
0609  e035
0610  e035  a6 ca     rdplt  ldx tblx         cursor zeile
0611  e037  a4 cb            ldy pntr         cursor spalte
0612  e039  60               rts
0613  e03a
0614  e03a
0615  e03a  ===> basisadr. i/o-gerât nach xr/yr <===
0616  e03a
0617  e03a  a2 00     iobase ldx #$00
0618  e03c  a0 dc            ldy #$dc
0619  e03e  60               rts
0620  e03f
0621  e03f
```

```
zeile adr.  obj.-code   source-code

0622  e03f  ===> max. anzahl spalten nach xr, zeilen nach yr <===
0623  e03f
0624  e03f  a2 50     scrorg ldx #$50
0625  e041  a0 19            ldy #$19
0626  e043  60                rts
0627  e044
0628  e044
0629  e044  ===> bildschirm-, funkt.tasten-reset <===
0630  e044
0631  e044  a9 00     cint   lda #$00
0632  e046  a2 23            ldx #$23
0633  e048  95 c2     cloop1 sta keypnt,x     addrl akt. pgm tastenpuffer
0634  e04a  ca               dex
0635  e04b  10 fb            bpl cloop1
0636  e04d  a2 20            ldx #$20
0637  e04f  9d 97 03  cloop2 sta rvs,x        revers flag
0638  e052  ca               dex
0639  e053  10 fa            bpl cloop2
0640  e055  a9 0c            lda #$0c
0641  e057  85 d8            sta delay        zeit bis zum einsetzen des repeat
0642  e059  a9 60            lda #$60
0643  e05b  85 d4            sta config       cursor type / char. vor blinken
0644  e05d  a9 2a            lda #$2a
0645  e05f  8d b5 03         sta funvec       addrl ind. jmp für funktionstasten
0646  e062  a9 e9            lda #$e9
0647  e064  8d b6 03         sta funvec+1     addrh ind. jmp für funktionstasten
0648  e067  a5 c0            lda pkybuf       addrl start programmierbare tasten
0649  e069  05 c1            ora pkybuf+1     addrh start programmierbare tasten
0650  e06b  d0 3d            bne noroom
0651  e06d  ad 55 03         lda hiadr        addrl ende des system speichers
0652  e070  8d 80 03         sta pkyend       addrl endadr. funkt.tastenpuffer
0653  e073  ad 56 03         lda hiadr+1      addrh ende des system speichers
0654  e076  8d 81 03         sta pkyend+1     addrh endadr. funkt.tastenpuffer
0655  e079  a9 40            lda #$40
0656  e07b  a2 00            ldx #$00
0657  e07d  a0 02            ldy #$02
0658  e07f  20 81 ff         jsr kalloc       allocation-routine
0659  e082  b0 26            bcs noroom
0660  e084  8d 82 03         sta keyseg       ram-segment # für funktionstasten
0661  e087  e8               inx
0662  e088  86 c0            stx pkybuf       addrl start programmierbare tasten
0663  e08a  d0 01            bne room10
0664  e08c  c8               iny
0665  e08d  84 c1     room10 sty pkybuf+1     addrh start programmierbare tasten
0666  e08f  a0 39            ldy #$39
0667  e091  20 82 e2         jsr pagst2       segment funktionstasten setzen
0668  e094
0669  e094
0670  e094  ===> 10 funktionstasten vorbesetzen <===
0671  e094
0672  e094  b9 2d ec  kyset1 lda keydef-1,y   tab.1 funktionstastenbefehle -1
0673  e097  88               dey
0674  e098  91 c0            sta (pkybuf),y   addrl start programmierbare tasten
0675  e09a  d0 f8            bne kyset1       10 funktionstasten vorbesetzen
0676  e09c  20 91 e2         jsr pagres       segment zurücksetzen
0677  e09f  a0 0a            ldy #$0a
0678  e0a1  b9 23 ec  kyset2 lda keylen-2+1,y
0679  e0a4  99 82 03         sta keyseg,y     ram-segment # für funktionstasten
```

```
zeile adr.  obj.-code   source-code

0680  e0a7  88                dey
0681  e0a8  d0 f7             bne kyset2
0682  e0aa  20 c7 e9  noroom   jsr sreset       bs-fenster auf volle grösse
                                                (home-home)
0683  e0ad  20 51 e2           jsr txcrt        textmodus setzen
0684  e0b0  20 60 e2           jsr crtint       crt-controller programmieren
0685  e0b3
0686  e0b3
0687  e0b3  ===> bildschirm löschen <===
0688  e0b3
0689  e0b3  20 c1 e0  clsr     jsr nxtd         home-position setzen
0690  e0b6  20 cf e0  cls10    jsr scrset       bs-zeiger setzen,zeilenoffset=xr
0691  e0b9  20 27 e2           jsr clrln        bs-zeile löschen
0692  e0bc  e4 dd              cpx scbot        unterste zeile bildschirm (0-25)
0693  e0be  e8                 inx
0694  e0bf  90 f5              bcc cls10
0695  e0c1
0696  e0c1
0697  e0c1  ===> home-position setzen <===
0698  e0c1
0699  e0c1  a6 dc      nxtd    ldx sctop        oberste zeile bildschirm (0-25)
0700  e0c3  86 ca              stx tblx         cursor zeile
0701  e0c5  86 cf              stx lsxp         letzte position (zeile)
0702  e0c7  a4 de      stu10   ldy sclf         linker rand bildschirm
0703  e0c9  84 cb              sty pntr         cursor spalte
0704  e0cb  84 ce              sty lstp         screen edit startposition
0705  e0cd
0706  e0cd
0707  e0cd  ===> bs-zeiger auf cursor-zeile <===
0708  e0cd
0709  e0cd  a6 ca      stupt   ldx tblx         cursor zeile
0710  e0cf
0711  e0cf
0712  e0cf  ===> bs-zeiger setzen,zeilenoffset=xr <===
0713  e0cf
0714  e0cf  bd b2 eb  scrset   lda ldtb2,x      tab. anfangs-addrl bs-zeilen
0715  e0d2  85 c8              sta pnt          addrl zeiger auf akt. zeichen im
                                                video-ram
0716  e0d4  bd cb eb           lda ldtb1,x      tab. anfangs-addrh der bs-zeilen
0717  e0d7  85 c9              sta pnt+1        addrh zeiger auf akt. zeichen im
                                                video-ram
0718  e0d9  60                 rts
0719  e0da
0720  e0da
0721  e0da  ===> cursor-pos. in reg. 14/15 vdc 6851 <===
0722  e0da
0723  e0da  a0 0f      movcur  ldy #$0f
0724  e0dc  18                 clc
0725  e0dd  a5 c8              lda pnt          addrl zeiger auf akt. zeichen im
                                                video-ram
0726  e0df  65 cb              adc pntr         cursor spalte
0727  e0e1  8c 00 d8           sty vdc+adreg    6845 vdc: adress-reg.
0728  e0e4  8d 01 d8           sta vdc+dareg    6845 vdc: daten-reg.
0729  e0e7  88                 dey
0730  e0e8  8c 00 d8           sty vdc+adreg    6845 vdc: adress-reg.
0731  e0eb  ad 01 d8           lda vdc+dareg    6845 vdc: daten-reg.
0732  e0ee  29 f8              and #$f8
0733  e0f0  85 d9              sta sedt1        mehrfach benutzte editorvariable
```

```
zeile adr.  obj.-code   source-code

0734  e0f2  a5 c9           lda pnt+1        addrh zeiger auf akt. zeichen im
                                             video-ram
0735  e0f4  69 00           adc #$00
0736  e0f6  29 07           and #$07
0737  e0f8  05 d9           ora sedt1        mehrfach benutzte editorvariable
0738  e0fa  8d 01 d8        sta vdc+dareg    6845 vdc: daten-reg.
0739  e0fd  60              rts
0740  e0fe
0741  e0fe
0742  e0fe  ===> zeichen aus tastenpuffer in ac <===
0743  e0fe
0744  e0fe  a6 d6     lp2   ldx kyndx        zeichenzähler für pgm tastenbefehl
0745  e100  f0 12           beq lp3          tastatur-puffer in ac, pufferinhalt
                                             verschieben
0746  e102  ac 9d 03        ldy keyidx       index programmierbare befehle
0747  e105  20 82 e2        jsr pagst2       segment funktionstasten setzen
0748  e108  b1 c2           lda (keypnt),y   addrl akt. pgm tastenpuffer
0749  e10a  20 91 e2        jsr pagres       segment zurücksetzen
0750  e10d  c6 d6           dec kyndx        zeichenzähler für pgm tastenbefehl
0751  e10f  ee 9d 03        inc keyidx       index programmierbare befehle
0752  e112  58              cli
0753  e113  60              rts
0754  e114
0755  e114
0756  e114  ===> tastatur-puffer in ac, pufferinhalt verschieben <===
0757  e114
0758  e114  ac ab 03  lp3   ldy keyd         tastaturpuffer (10 bytes)
0759  e117  a2 00           ldx #$00
0760  e119  bd ac 03  lp1   lda keyd+1,x     tastaturpuffer +1 (10 bytes)
0761  e11c  9d ab 03        sta keyd,x       tastaturpuffer (10 bytes)
0762  e11f  e8              inx
0763  e120  e4 d1           cpx ndx          anzahl bytes im tastaturpuffer
0764  e122  d0 f5           bne lp1
0765  e124  c6 d1           dec ndx          anzahl bytes im tastaturpuffer
0766  e126  98              tya
0767  e127  58              cli
0768  e128  60              rts
0769  e129
0770  e129
0771  e129  ===> input <===
0772  e129
0773  e129  20 99 e2  loop4 jsr prt          zeichen aus ac auf bildschirm
0774  e12c  a0 0a     loop3 ldy #$0a
0775  e12e  a5 d4           lda config       cursor type / char. vor blinken
0776  e130  8c 00 d8        sty vdc+adreg    6845 vdc: adress-reg.
0777  e133  8d 01 d8        sta vdc+dareg    6845 vdc: daten-reg.
0778  e136  a5 d1     loop3a lda ndx         anzahl bytes im tastaturpuffer
0779  e138  05 d6           ora kyndx        zeichenzähler für pgm tastenbefehl
0780  e13a  f0 fa           beq loop3a
0781  e13c  78              sei
0782  e13d  a9 20           lda #$20
0783  e13f  8c 00 d8        sty vdc+adreg    6845 vdc: adress-reg.
0784  e142  8d 01 d8        sta vdc+dareg    6845 vdc: daten-reg.
0785  e145  20 fe e0        jsr lp2          zeichen aus tastenpuffer in ac
0786  e148  c9 0d           cmp #$0d
0787  e14a  d0 dd           bne loop4        input
0788  e14c  85 d0           sta crsw         flag 'cr' bei bs-eingabe
0789  e14e  20 44 e5        jsr fndend       cursor an zeilenende (esc, k)
```

```
zeile adr.  obj.-code   source-code

0790  e151  8e 98 03          stx lintmp       puffer für bs-zeilen #
0791  e154  20 36 e5          jsr fistrt       cursor auf zeilenanfang
0792  e157  a9 00             lda #$00
0793  e159  85 d2             sta qtsw         flag für anführungszeichen-modus
0794  e15b  a4 de             ldy sclf         linker rand bildschirm
0795  e15d  a5 cf             lda lsxp         letzte position (zeile)
0796  e15f  30 13             bmi lp80
0797  e161  c5 ca             cmp tblx         cursor zeile
0798  e163  90 0f             bcc lp80
0799  e165  a4 ce             ldy lstp         screen edit startposition
0800  e167  cd 98 03          cmp lintmp       puffer für bs-zeilen #
0801  e16a  d0 04             bne lp70
0802  e16c  c4 d5             cpy indx         position letztes zeichen in zeile
0803  e16e  f0 02             beq lp75
0804  e170  b0 11       lp70  bcs clp2
0805  e172  85 ca       lp75  sta tblx         cursor zeile
0806  e174  84 cb       lp80  sty pntr         cursor spalte
0807  e176  4c 8b e1          jmp lop5         char von bs holen (get)
0808  e179
0809  e179
0810  e179  ===> char von bs holen (get o. input) <===
0811  e179
0812  e179  98           loop5 tya
0813  e17a  48                 pha
0814  e17b  8a                 txa
0815  e17c  48                 pha
0816  e17d  a5 d0              lda crsw         flag 'cr' bei bs-eingabe
0817  e17f  f0 ab              beq loop3
0818  e181  10 08              bpl lop5         char von bs holen (get)
0819  e183  a9 00       clp2  lda #$00
0820  e185  85 d0              sta crsw         flag 'cr' bei bs-eingabe
0821  e187  a9 0d              lda #$0d
0822  e189  d0 39              bne clp7
0823  e18b
0824  e18b
0825  e18b  ===> char von bs holen (get) <===
0826  e18b
0827  e18b  20 cd e0     lop5  jsr stupt        bs-zeiger auf cursor-zeile
0828  e18e  20 42 e2           jsr get1ch       zeichen von cursorpos. in ac
0829  e191  85 db              sta adata        akt. ausgabezeichen
0830  e193  29 3f              and #$3f
0831  e195  06 db              asl adata        akt. ausgabezeichen
0832  e197  24 db              bit adata        akt. ausgabezeichen
0833  e199  10 02              bpl lop54
0834  e19b  09 80              ora #$80
0835  e19d  90 04       lop54 bcc lop52
0836  e19f  a6 d2              ldx qtsw         flag für anführungszeichen-modus
0837  e1a1  d0 04              bne lop53
0838  e1a3  70 02       lop52 bvs lop53
0839  e1a5  09 40              ora #$40
0840  e1a7  20 cd e1     lop53 jsr qtswc        quote-mode-flag'"' invertieren
0841  e1aa  a4 ca              ldy tblx         cursor zeile
0842  e1ac  cc 98 03           cpy lintmp       puffer für bs-zeilen #
0843  e1af  90 0a              bcc clp00
0844  e1b1  a4 cb              ldy pntr         cursor spalte
0845  e1b3  c4 d5              cpy indx         position letztes zeichen in zeile
0846  e1b5  90 04              bcc clp00
0847  e1b7  66 d0              ror crsw         flag 'cr' bei bs-eingabe
```

```
zeile adr.  obj.-code   source-code


0848  e1b9  30 03           bmi clp1
0849  e1bb  20 74 e5  clp00  jsr nxtchr       cursor 1 pos. rechts
0850  e1be  c9 de     clp1   cmp #$de
0851  e1c0  d0 02           bne clp7
0852  e1c2  a9 ff           lda #$ff
0853  e1c4  85 db     clp7   sta adata        akt. ausgabezeichen
0854  e1c6  68               pla
0855  e1c7  aa               tax
0856  e1c8  68               pla
0857  e1c9  a8               tay
0858  e1ca  a5 db            lda adata        akt. ausgabezeichen
0859  e1cc  60               rts
0860  e1cd
0861  e1cd
0862  e1cd  ===> quote-mode-flag'"' invertieren <===
0863  e1cd
0864  e1cd  c9 22     qtswc  cmp #$22
0865  e1cf  d0 0c            bne qtswl
0866  e1d1  a5 d3            lda insrt        insert flag
0867  e1d3  d0 06            bne qtswi        '"' in ac
0868  e1d5  a5 d2            lda qtsw         flag für anführungszeichen-modus
0869  e1d7  49 01            eor #$01
0870  e1d9  85 d2            sta qtsw         flag für anführungszeichen-modus
0871  e1db
0872  e1db
0873  e1db  ===> '"' in ac <===
0874  e1db
0875  e1db  a9 22     qtswi  lda #$22
0876  e1dd  60        qtswl  rts
0877  e1de
0878  e1de
0879  e1de  ===> zeichen auf bs ausgeben <===
0880  e1de
0881  e1de  2c 97 03  nxt3   bit rvs          revers flag
0882  e1e1  10 02            bpl nvs
0883  e1e3  09 80            ora #$80
0884  e1e5  a6 d3     nvs    ldx insrt        insert flag
0885  e1e7  f0 02            beq nvsa
0886  e1e9  c6 d3            dec insrt        insert flag
0887  e1eb  2c 9a 03  nvsa   bit insflg       auto insert flag
0888  e1ee  10 09            bpl nvs1
0889  e1f0  48               pha
0890  e1f1  20 e4 e5         jsr insert       zeichen einfügen ($94)
0891  e1f4  a2 00            ldx #$00
0892  e1f6  86 d3            stx insrt        insert flag
0893  e1f8  68               pla
0894  e1f9  20 1c e2  nvs1   jsr dspp         char in ac an cursorposition
0895  e1fc  c0 45            cpy #$45
0896  e1fe  d0 03            bne nvs2
0897  e200  20 8d e4         jsr bell         glocke anschlagen ($07 oder $87)
0898  e203  20 2e e6  nvs2   jsr movchr       test zeilenende, zeile einfügen
0899  e206
0900  e206
0901  e206  ===> ende zeichenausgabe <===
0902  e206
0903  e206  a5 db     loop2  lda adata        akt. ausgabezeichen
0904  e208  8d 99 03         sta lstchr       letztes gedrucktes zeichen
0905  e20b  20 da e0         jsr movcur       cursor-pos. in reg. 14/15 vdc 6851
```

```
zeile adr.  obj.-code   source-code


0906  e20e  68              pla
0907  e20f  a8              tay
0908  e210  a5 d3           lda insrt        insert flag
0909  e212  f0 02           beq lop2
0910  e214  46 d2           lsr qtsw         flag für anführungszeichen-modus
0911  e216  68       lop2   pla
0912  e217  aa              tax
0913  e218  68              pla
0914  e219  60              rts
0915  e21a
0916  e21a
0917  e21a  ===> space an cursorposition <===
0918  e21a
0919  e21a  a9 20     doblnk lda #$20
0920  e21c
0921  e21c
0922  e21c  ===> char in ac an cursorposition <===
0923  e21c
0924  e21c  a4 cb     dspp   ldy pntr          cursor spalte
0925  e21e  20 7d e2         jsr pagset       segment 15 setzen
0926  e221  91 c8            sta (pnt),y      addrl zeiger auf akt. zeichen im
                                               video-ram
0927  e223  20 91 e2         jsr pagres       segment zurücksetzen
0928  e226  60               rts
0929  e227
0930  e227
0931  e227  ===> bs-zeile löschen <===
0932  e227
0933  e227  a4 de     clrln  ldy sclf         linker rand bildschirm
0934  e229  20 05 e5         jsr clrbit       akt. zeile als 1.doppelzeile
                                               eintragen
0935  e22c  8a        clrprt txa
0936  e22d  48               pha
0937  e22e  a5 cb            lda pntr         cursor spalte
0938  e230  48               pha
0939  e231  88               dey
0940  e232  c8        clr10  iny
0941  e233  84 cb            sty pntr         cursor spalte
0942  e235  20 1a e2         jsr doblnk       space an cursorposition
0943  e238  c4 df            cpy scrt         rechter rand bildschirm
0944  e23a  d0 f6            bne clr10
0945  e23c  68               pla
0946  e23d  85 cb            sta pntr         cursor spalte
0947  e23f  68               pla
0948  e240  aa               tax
0949  e241  60               rts
0950  e242
0951  e242
0952  e242  ===> zeichen von cursorpos. in ac <===
0953  e242
0954  e242  a4 cb     get1ch ldy pntr         cursor spalte
0955  e244
0956  e244
0957  e244  ===> zeichen von zeile in ac, spalte=yr <===
0958  e244
0959  e244  20 7d e2  getych jsr pagset       segment 15 setzen
0960  e247  b1 c8            lda (pnt),y      addrl zeiger auf akt. zeichen im
                                               video-ram
```

```
zeile adr.  obj.-code   source-code

0961  e249  20 91 e2          jsr pagres       segment zurücksetzen
0962  e24c  60                rts
0963  e24d
0964  e24d
0965  e24d  ===> umsch. text ($0e) - graphik ($8e) <===
0966  e24d
0967  e24d  a0 10      ctext  ldy #$10
0968  e24f  b0 02             bcs crtset       crt-umschaltung, yr = text/grafik
0969  e251
0970  e251
0971  e251  ===> textmodus setzen <===
0972  e251
0973  e251  a0 00      txcrt  ldy #$00
0974  e253
0975  e253
0976  e253  ===> crt-umschaltung, yr = text/grafik <===
0977  e253
0978  e253  84 cc      crtset sty grmode        flag für grafik/text-modus
0979  e255  ad 06 de          lda tri1+creg    6525 triport 1: control reg.
0980  e258  29 ef             and #$ef
0981  e25a  05 cc             ora grmode       flag für grafik/text-modus
0982  e25c  8d 06 de          sta tri1+creg    6525 triport 1: control reg.
0983  e25f  60                rts
0984  e260
0985  e260
0986  e260  ===> crt-controller programmieren <===
0987  e260
0988  e260  a0 11      crtint ldy #$11
0989  e262  2c 02 df          bit tri2+pc      6525 triport 2: port reg. c
0990  e265  30 06             bmi crt10
0991  e267  a0 23             ldy #$23
0992  e269  70 02             bvs crt10
0993  e26b  a0 35             ldy #$35
0994  e26d  a2 11      crt10  ldx #$11
0995  e26f  b9 6f ec   crt20  lda atext,y      codetab. 1 für bs-crt 6845
0996  e272  8e 00 d8          stx vdc+adreg    6845 vdc: adress-reg.
0997  e275  8d 01 d8          sta vdc+dareg    6845 vdc: daten-reg.
0998  e278  88                dey
0999  e279  ca                dex
1000  e27a  10 f3             bpl crt20
1001  e27c  60                rts
1002  e27d
1003  e27d
1004  e27d  ===> segment 15 setzen <===
1005  e27d
1006  e27d  48         pagset pha
1007  e27e  a9 3f             lda #$3f
1008  e280  d0 04             bne pagsub       segment (# in ac) setzen
1009  e282
1010  e282
1011  e282  ===> segment funktionstasten setzen <===
1012  e282
1013  e282  48         pagst2 pha
1014  e283  ad 82 03          lda keyseg       ram-segment # für funktionstasten
1015  e286
1016  e286
1017  e286  ===> segment (# in ac) setzen <===
1018  e286
```

```
zeile adr.  obj.-code   source-code

1019  e286  48        pagsub pha
1020  e287  a5 01            lda i6509        6509 indirection register
1021  e289  8d a0 03        sta pagsav       zeitweise ram page
1022  e28c  68               pla
1023  e28d  85 01            sta i6509        6509 indirection register
1024  e28f  68               pla
1025  e290  60               rts
1026  e291
1027  e291
1028  e291  ===> segment zurücksetzen <===
1029  e291
1030  e291  48        pagres pha
1031  e292  ad a0 03         lda pagsav       zeitweise ram page
1032  e295  85 01            sta i6509        6509 indirection register
1033  e297  68               pla
1034  e298  60               rts
1035  e299
1036  e299
1037  e299  ===> zeichen aus ac auf bildschirm <===
1038  e299
1039  e299  48        prt    pha
1040  e29a  c9 ff            cmp #$ff
1041  e29c  d0 02            bne prt10
1042  e29e  a9 de            lda #$de
1043  e2a0  85 db     prt10  sta adata        akt. ausgabezeichen
1044  e2a2  8a               txa
1045  e2a3  48               pha
1046  e2a4  98               tya
1047  e2a5  48               pha
1048  e2a6  a9 00            lda #$00
1049  e2a8  85 d0            sta crsw         flag 'cr' bei bs-eingabe
1050  e2aa  a4 cb            ldy pntr         cursor spalte
1051  e2ac  a5 db            lda adata        akt. ausgabezeichen
1052  e2ae  29 7f            and #$7f
1053  e2b0  c9 20            cmp #$20
1054  e2b2  90 1b            bcc ntcn         steuerzeichenbehandlung
1055  e2b4  ae 99 03         ldx lstchr       letztes gedrucktes zeichen
1056  e2b7  e0 1b            cpx #$1b
1057  e2b9  d0 06            bne njt10
1058  e2bb  20 55 e6         jsr sequen       aufruf user esc-funktionen
                                              (ind.-vektor)
1059  e2be  4c 03 e3         jmp getout       ende zeichenausgabe
1060  e2c1
1061  e2c1
1062  e2c1  29 3f     njt10  and #$3f
1063  e2c3  24 db     njt20  bit adata        akt. ausgabezeichen
1064  e2c5  10 02            bpl njt30
1065  e2c7  09 40            ora #$40
1066  e2c9  20 cd e1  njt30  jsr qtswc        quote-mode-flag'"' invertieren
1067  e2cc  4c de e1         jmp nxt3         zeichen auf bs ausgeben
1068  e2cf
1069  e2cf
1070  e2cf  ===> steuerzeichenbehandlung <===
1071  e2cf
1072  e2cf  c9 0d     ntcn   cmp #$0d
1073  e2d1  f0 29            beq ntcn20       ctr-adr. auf stapel, zeichenausg.
1074  e2d3  c9 14            cmp #$14
1075  e2d5  f0 25            beq ntcn20       ctr-adr. auf stapel, zeichenausg.
```

```
zeile adr.  obj.-code   source-code

1076  e2d7  c9 1b           cmp #$1b
1077  e2d9  d0 11           bne ntcn1
1078  e2db  24 db           bit adata        akt. ausgabezeichen
1079  e2dd  30 0d           bmi ntcn1
1080  e2df  a5 d2           lda qtsw         flag für anführungszeichen-modus
1081  e2e1  05 d3           ora insrt        insert flag
1082  e2e3  f0 17           beq ntcn20       ctr-adr. auf stapel, zeichenausg.
1083  e2e5  20 a2 e3        jsr toqm         'inst-','quote-','rvs'-mode aus
1084  e2e8  85 db           sta adata        akt. ausgabezeichen
1085  e2ea  f0 10           beq ntcn20       ctr-adr. auf stapel, zeichenausg.
1086  e2ec  c9 03    ntcn1  cmp #$03
1087  e2ee  f0 0c           beq ntcn20       ctr-adr. auf stapel, zeichenausg.
1088  e2f0  a4 d3           ldy insrt        insert flag
1089  e2f2  d0 04           bne ntcn10
1090  e2f4  a4 d2           ldy qtsw         flag für anführungszeichen-modus
1091  e2f6  f0 04           beq ntcn20       ctr-adr. auf stapel, zeichenausg.
1092  e2f8  09 80    ntcn10 ora #$80
1093  e2fa  d0 c7           bne njt20
1094  e2fc
1095  e2fc
1096  e2fc  ===> ctr-adr. auf stapel, zeichenausg. <===
1097  e2fc
1098  e2fc  a5 db    ntcn20 lda adata        akt. ausgabezeichen
1099  e2fe  0a              asl a
1100  e2ff  aa              tax
1101  e300  20 06 e3        jsr ctdsp        ctr-adr. auf stapel, ac =ausgabez.
1102  e303  4c 06 e2  getout jmp loop2       ende zeichenausgabe
1103  e306
1104  e306
1105  e306  ===> ctr-adr. auf stapel, ac =ausgabez. <===
1106  e306
1107  e306  bd e5 eb  ctdsp  lda ctable+1,x   adress-tab. der control-routinen
1108  e309  48              pha
1109  e30a  bd e4 eb        lda ctable,x     adress-tab. der control-routinen
1110  e30d  48              pha
1111  e30e  a5 db           lda adata        akt. ausgabezeichen
1112  e310  60              rts
1113  e311
1114  e311
1115  e311  6c 22 03  cuser  jmp (ctlvec)     addrl unbenuzte 'ctrl' tasten-vector
1116  e314
1117  e314
1118  e314  ===> cursor unten ($11) - oben ($91) <===
1119  e314
1120  e314  b0 0d     cdnup  bcs cup          cursor nach oben
1121  e316  20 7a e3         jsr nxln         test bs-ende, cursor nächste zeile
1122  e319  20 f5 e4  cdn10  jsr getbit       test auf doppelzeile (akt. zeile)
1123  e31c  b0 03           bcs cdrts
1124  e31e  38              sec
1125  e31f  66 cf           ror lsxp         letzte position (zeile)
1126  e321  18       cdrts  clc
1127  e322  60              rts
1128  e323
1129  e323
1130  e323  ===> cursor nach oben <===
1131  e323
1132  e323  a6 dc     cup    ldx sctop        oberste zeile bildschirm (0-25)
1133  e325  e4 ca           cpx tblx         cursor zeile
```

```
zeile adr.  obj.-code   source-code

1134  e327  b0 0f           bcs critgo
1135  e329  20 19 e3  cup10  jsr cdn10
1136  e32c  c6 ca            dec tblx        cursor zeile
1137  e32e  4c cd e0         jmp stupt       bs-zeiger auf cursor-zeile
1138  e331
1139  e331
1140  e331  ===> cursor rechts ($1d) - links ($9d) <===
1141  e331
1142  e331  b0 06     crtlf  bcs cleft        cursor nach links
1143  e333  20 74 e5         jsr nxtchr      cursor 1 pos. rechts
1144  e336  b0 e1            bcs cdn10
1145  e338  60        critgo rts
1146  e339
1147  e339
1148  e339  ===> cursor nach links <===
1149  e339
1150  e339  20 87 e5  cleft  jsr bakchr       cursor 1 position nach links
1151  e33c  b0 fa            bcs critgo
1152  e33e  d0 e1            bne cdrts
1153  e340  e6 ca            inc tblx        cursor zeile
1154  e342  d0 e5            bne cup10
1155  e344
1156  e344
1157  e344  ===> revers on ($12) - off ($92) <===
1158  e344
1159  e344  49 80     rvsf   eor #$80
1160  e346  8d 97 03         sta rvs         revers flag
1161  e349  60               rts
1162  e34a
1163  e34a
1164  e34a  ===> cursor home ($13) - bs clear ($93) <===
1165  e34a
1166  e34a  90 03     homclr bcc homes        cursor home
1167  e34c  4c b3 e0         jmp clsr        bildschirm löschen
1168  e34f
1169  e34f
1170  e34f  ===> cursor home <===
1171  e34f
1172  e34f  cd 99 03  homes  cmp lstchr       letztes gedrucktes zeichen
1173  e352  d0 03            bne hm110
1174  e354  20 c7 e9         jsr sreset      bs-fenster auf volle grösse
                                             (home-home)
1175  e357  4c c1 e0  hm110  jmp nxtd         home-position setzen
1176  e35a
1177  e35a
1178  e35a  ===> tab ($09) - setzen/löschen ($89) <===
1179  e35a
1180  e35a  a4 cb     tabit  ldy pntr         cursor spalte
1181  e35c  b0 12            bcs tabtog      tabulator setzen/löschen
1182  e35e
1183  e35e
1184  e35e  ===> tabulator löschen <===
1185  e35e
1186  e35e  c4 df     tab1   cpy scrt        rechter rand bildschirm
1187  e360  90 05            bcc tab2
1188  e362  a5 df            lda scrt        rechter rand bildschirm
1189  e364  85 cb            sta pntr        cursor spalte
1190  e366  60               rts
```

```
zeile adr.  obj.-code   source-code


1191  e367
1192  e367
1193  e367  c8        tab2   iny
1194  e368  20 5a e9         jsr gettab       tabulator holen
1195  e36b  f0 f1            beq tab1         tabulator löschen
1196  e36d  84 cb            sty pntr         cursor spalte
1197  e36f  60               rts
1198  e370
1199  e370
1200  e370  ===> tabulator setzen/löschen <===
1201  e370
1202  e370  20 5a e9  tabtog jsr gettab       tabulator holen
1203  e373  4d 9c 03         eor bitmsk       bit mask/funkt.tasten temporär
1204  e376  9d a1 03         sta tab,x        tabstop flags (max.10byt/80bit)
1205  e379  60               rts
1206  e37a
1207  e37a
1208  e37a  ===> test bs-ende, cursor nächste zeile <===
1209  e37a
1210  e37a  a6 ca     nxln   ldx tblx         cursor zeile
1211  e37c  e4 dd            cpx scbot        unterste zeile bildschirm (0-25)
1212  e37e  90 0f            bcc nxln1
1213  e380  2c 9b 03         bit scrdis       scroll disable flag
1214  e383  10 06            bpl doscrl
1215  e385  a5 dc            lda sctop        oberste zeile bildschirm (0-25)
1216  e387  85 ca            sta tblx         cursor zeile
1217  e389  b0 06            bcs nowhop
1218  e38b  20 f6 e3  doscrl jsr scrup        nach oben scrollen
1219  e38e  18               clc
1220  e38f  e6 ca     nxln1  inc tblx         cursor zeile
1221  e391  4c cd e0  nowhop jmp stupt        bs-zeiger auf cursor-zeile
1222  e394
1223  e394
1224  e394  ===> 'carriage return' ($0d oder $8d) <===
1225  e394
1226  e394  20 44 e5  nxt1   jsr fndend       cursor an zeilenende (esc, k)
1227  e397  e8               inx
1228  e398  20 05 e5         jsr clrbit       akt. zeile als 1.doppelzeile
                                              eintragen
1229  e39b  a4 de            ldy sclf         linker rand bildschirm
1230  e39d  84 cb            sty pntr         cursor spalte
1231  e39f  20 7a e3         jsr nxln         test bs-ende, cursor nächste zeile
1232  e3a2
1233  e3a2
1234  e3a2  ===> 'inst-','quote-','rvs'-mode aus <===
1235  e3a2
1236  e3a2  a9 00     toqm   lda #$00
1237  e3a4  85 d3            sta insrt        insert flag
1238  e3a6  8d 97 03         sta rvs          revers flag
1239  e3a9  85 d2            sta qtsw         flag für anführungszeichen-modus
1240  e3ab  cd 9f 03         cmp bellmd       flag für glocke am zeilenende
1241  e3ae  d0 03            bne toqmx
1242  e3b0  8d 18 da         sta sid+volume   lautstärke / schalter tief-, band-,
                                              hochpass
1243  e3b3  60        toqmx  rts
1244  e3b4
1245  e3b4
1246  e3b4  ===> zeilen verschieben <===
```

```
zeile adr.  obj.-code   source-code


1247  e3b4
1248  e3b4  bd b2 eb  movlin lda ldtb2,x      tab. anfangs-addrl bs-zeilen
1249  e3b7  85 c4             sta sedsal      addrl scroll-zeiger start
1250  e3b9  bd cb eb          lda ldtb1,x     tab. anfangs-addrh der bs-zeilen
1251  e3bc  85 c5             sta sedsal+1    addrh scroll-zeiger start
1252  e3be  20 7d e2          jsr pagset      segment 15 setzen
1253  e3c1  b1 c4     movl10  lda (sedsal),y  addrl scroll-zeiger start
1254  e3c3  91 c8             sta (pnt),y     addrl zeiger auf akt. zeichen im
                                              video-ram
1255  e3c5  c4 df             cpy scrt        rechter rand bildschirm
1256  e3c7  c8                iny
1257  e3c8  90 f7             bcc movl10
1258  e3ca  4c 91 e2          jmp pagres      segment zurücksetzen
1259  e3cd
1260  e3cd
1261  e3cd  ===> nach unten scrollen <===
1262  e3cd
1263  e3cd  a6 cf     scrdwn  ldx lsxp         letzte position (zeile)
1264  e3cf  30 06             bmi scd30
1265  e3d1  e4 ca             cpx tblx        cursor zeile
1266  e3d3  90 02             bcc scd30
1267  e3d5  e6 cf             inc lsxp        letzte position (zeile)
1268  e3d7  a6 dd     scd30   ldx scbot       unterste zeile bildschirm (0-25)
1269  e3d9  20 cf e0  scd10   jsr scrset       bs-zeiger setzen,zeilenoffset=xr
1270  e3dc  a4 de             ldy sclf        linker rand bildschirm
1271  e3de  e4 ca             cpx tblx        cursor zeile
1272  e3e0  f0 0e             beq scd20
1273  e3e2  ca                dex
1274  e3e3  20 f7 e4          jsr getbt1      test auf doppelzeile (xr = zeile)
1275  e3e6  e8                inx
1276  e3e7  20 03 e5          jsr putbt1
1277  e3ea  ca                dex
1278  e3eb  20 b4 e3          jsr movlin      zeilen verschieben
1279  e3ee  b0 e9             bcs scd10
1280  e3f0  20 27 e2  scd20   jsr clrln       bs-zeile löschen
1281  e3f3  4c 12 e5          jmp setbit
1282  e3f6
1283  e3f6
1284  e3f6  ===> nach oben scrollen <===
1285  e3f6
1286  e3f6  a6 dc     scrup   ldx sctop       oberste zeile bildschirm (0-25)
1287  e3f8  e8        scru00  inx
1288  e3f9  20 f7 e4          jsr getbt1      test auf doppelzeile (xr = zeile)
1289  e3fc  90 0a             bcc scru15
1290  e3fe  e4 dd             cpx scbot       unterste zeile bildschirm (0-25)
1291  e400  90 f6             bcc scru00
1292  e402  a6 dc             ldx sctop       oberste zeile bildschirm (0-25)
1293  e404  e8                inx
1294  e405  20 05 e5          jsr clrbit      akt. zeile als 1.doppelzeile
                                              eintragen
1295  e408  c6 ca     scru15  dec tblx        cursor zeile
1296  e40a  24 cf             bit lsxp        letzte position (zeile)
1297  e40c  30 02             bmi scru20
1298  e40e  c6 cf             dec lsxp        letzte position (zeile)
1299  e410  a6 dc     scru20  ldx sctop       oberste zeile bildschirm (0-25)
1300  e412  e4 da             cpx sedt2       mehrfach benutzte editorvariable
1301  e414  b0 02             bcs scru30
1302  e416  c6 da             dec sedt2       mehrfach benutzte editorvariable
```

```
zeile adr.  obj.-code   source-code

1303  e418  20 2d e4  scru30 jsr scr10
1304  e41b  a6 dc             ldx sctop       oberste zeile bildschirm (0-25)
1305  e41d  20 f7 e4          jsr getbt1      test auf doppelzeile (xr = zeile)
1306  e420  08                php
1307  e421  20 05 e5          jsr clrbit      akt. zeile als 1.doppelzeile
                                               eintragen
1308  e424  28                plp
1309  e425  90 05             bcc scru10
1310  e427  2c 9e 03          bit logscr      logical/physical scroll flag
1311  e42a  30 dc             bmi scru15
1312  e42c  60         scru10 rts
1313  e42d
1314  e42d
1315  e42d  20 cf e0  scr10   jsr scrset       bs-zeiger setzen,zeilenoffset=xr
1316  e430  a4 de             ldy sclf        linker rand bildschirm
1317  e432  e4 dd             cpx scbot       unterste zeile bildschirm (0-25)
1318  e434  b0 0e             bcs scr40
1319  e436  e8                inx
1320  e437  20 f7 e4          jsr getbt1      test auf doppelzeile (xr = zeile)
1321  e43a  ca                dex
1322  e43b  20 03 e5          jsr putbt1
1323  e43e  e8                inx
1324  e43f  20 b4 e3          jsr movlin      zeilen verschieben
1325  e442  b0 e9             bcs scr10
1326  e444  20 27 e2  scr40   jsr clrln       bs-zeile löschen
1327  e447  a2 ff             ldx #$ff
1328  e449  a0 fe             ldy #$fe
1329  e44b  20 80 e4          jsr getlin      triport 2 setzen, tastenabfrage
1330  e44e  29 20             and #$20
1331  e450  d0 0d             bne scr80
1332  e452
1333  e452
1334  e452  ===> langsam scrollen <===
1335  e452
1336  e452  ea         scr60  nop
1337  e453  ea                nop
1338  e454  ca                dex
1339  e455  d0 fb             bne scr60       langsam scrollen
1340  e457  88                dey
1341  e458  d0 f8             bne scr60       langsam scrollen
1342  e45a  84 d1      scr70  sty ndx         anzahl bytes im tastaturpuffer
1343  e45c  4c f7 e8   scr75  jmp keyxt2
1344  e45f
1345  e45f
1346  e45f  a2 f7      scr80  ldx #$f7
1347  e461  a0 ff             ldy #$ff
1348  e463  20 80 e4          jsr getlin      triport 2 setzen, tastenabfrage
1349  e466  29 10             and #$10
1350  e468  d0 f2             bne scr75
1351  e46a  20 80 e4  scr90   jsr getlin      triport 2 setzen, tastenabfrage
1352  e46d  29 10             and #$10
1353  e46f  f0 f9             beq scr90
1354  e471  a0 00      scr95  ldy #$00
1355  e473  a2 00             ldx #$00
1356  e475  20 80 e4          jsr getlin      triport 2 setzen, tastenabfrage
1357  e478  29 3f             and #$3f
1358  e47a  49 3f             eor #$3f
1359  e47c  f0 f3             beq scr95
```

```
zeile adr.  obj.-code   source-code


1360  e47e  d0 da            bne scr70
1361  e480
1362  e480
1363  e480  ===> triport 2 setzen, tastenabfrage <===
1364  e480
1365  e480  08        getlin php
1366  e481  78              sei
1367  e482  8e 00 df        stx tri2+pa      6525 triport 2: port reg. a
1368  e485  8c 01 df        sty tri2+pb      6525 triport 2: port reg. b
1369  e488  20 1e e9        jsr getkey       tastaturabfrage über triport 2
1370  e48b  28              plp
1371  e48c  60              rts
1372  e48d
1373  e48d                  .end
1374  e48d                  .lib editor2
```

```
zeile adr.  obj.-code   source-code

1376  e48d
1377  e48d  ===> glocke anschlagen ($07 oder $87) <===
1378  e48d
1379  e48d  ad 9f 03  bell   lda bellmd       flag für glocke am zeilenende
1380  e490  d0 27            bne bellgo
1381  e492  a9 0f            lda #$0f
1382  e494  8d 18 da         sta sid+volume   lautstärke / schalter tief-, band-,
                                               hochpass
1383  e497  a0 00            ldy #$00
1384  e499  8c 05 da         sty sid+osc1+atkdcy    6581 sid: attack/decay-time
                                               oszi 1
1385  e49c  a9 0a            lda #$0a
1386  e49e  8d 06 da         sta sid+osc1+susrel    6581 sid: sustain/release time
                                               oszi 1
1387  e4a1  a9 30            lda #$30
1388  e4a3  8d 01 da         sta sid+osc1+freqhi    6581 sid: addrh frequenz oszi
                                               1
1389  e4a6  a9 60            lda #$60
1390  e4a8  8d 0f da         sta sid+osc3+freqhi    6581 sid: addrh frequenz oszi
                                               3
1391  e4ab  a2 15            ldx #$15
1392  e4ad  8e 04 da         stx sid+osc1+oscctl    6581 sid: steuerreg. oszi 1
1393  e4b0  ea        bell10 nop
1394  e4b1  ea               nop
1395  e4b2  c8               iny
1396  e4b3  d0 fb            bne bell10
1397  e4b5  ca               dex
1398  e4b6  8e 04 da         stx sid+osc1+oscctl    6581 sid: steuerreg. oszi 1
1399  e4b9  60        bellgo rts
1400  e4ba
1401  e4ba
1402  e4ba  ===> löscht letzte eingegebene zahl 'ce' <===
1403  e4ba
1404  e4ba  a5 cb     ce     lda pntr         cursor spalte
1405  e4bc  48               pha
1406  e4bd  a4 cb     ce0    ldy pntr         cursor spalte
1407  e4bf  88               dey
1408  e4c0  20 44 e2         jsr getych       zeichen von zeile in ac, spalte=yr
1409  e4c3  c9 2b            cmp #$2b
1410  e4c5  f0 04            beq ce1
1411  e4c7  c9 2d            cmp #$2d
1412  e4c9  d0 08            bne ce2
1413  e4cb  88        ce1    dey
1414  e4cc  20 44 e2         jsr getych       zeichen von zeile in ac, spalte=yr
1415  e4cf  c9 05            cmp #$05
1416  e4d1  d0 1a            bne ce4
1417  e4d3  c9 05     ce2    cmp #$05
1418  e4d5  d0 04            bne ce3
1419  e4d7  88               dey
1420  e4d8  20 44 e2         jsr getych       zeichen von zeile in ac, spalte=yr
1421  e4db  c9 2e     ce3    cmp #$2e
1422  e4dd  90 0e            bcc ce4
1423  e4df  c9 2f            cmp #$2f
1424  e4e1  f0 0a            beq ce4
1425  e4e3  c9 3a            cmp #$3a
1426  e4e5  b0 06            bcs ce4
1427  e4e7  20 b0 e5         jsr deleet       zeichen löschen ($14)
1428  e4ea  4c bd e4         jmp ce0
```

```
zeile adr.  obj.-code   source-code


1429  e4ed
1430  e4ed
1431  e4ed  68         cet4   pla
1432  e4ee  c5 cb             cmp pntr          cursor spalte
1433  e4f0  d0 c7             bne bellgo
1434  e4f2  4c b0 e5          jmp deleet        zeichen löschen ($14)
1435  e4f5
1436  e4f5
1437  e4f5  ===> test auf doppelzeile (akt. zeile) <===
1438  e4f5
1439  e4f5  a6 ca      getbit ldx tblx          cursor zeile
1440  e4f7
1441  e4f7
1442  e4f7  ===> test auf doppelzeile (xr = zeile) <===
1443  e4f7
1444  e4f7  20 1e e5   getbt1 jsr bitpos        flag für doppelzeile holen, zeiger
                                                für bit-tab. setzen
1445  e4fa  35 e2             and bitabl,x      tabelle basic-doppelzeilen (4 byte)
1446  e4fc  c9 01             cmp #$01
1447  e4fe  4c 0e e5          jmp bitout
1448  e501
1449  e501
1450  e501  ===> akt. zeile als doppelzeile eintragen <===
1451  e501
1452  e501  a6 ca      putbit ldx tblx          cursor zeile
1453  e503  b0 0d      putbt1 bcs setbit
1454  e505
1455  e505
1456  e505  ===> akt. zeile als 1.doppelzeile eintragen <===
1457  e505
1458  e505  20 1e e5   clrbit jsr bitpos        flag für doppelzeile holen, zeiger
                                                für bit-tab. setzen
1459  e508  49 ff             eor #$ff
1460  e50a  35 e2             and bitabl,x      tabelle basic-doppelzeilen (4 byte)
1461  e50c  95 e2      bitsav sta bitabl,x      tabelle basic-doppelzeilen (4 byte)
1462  e50e  ae 9c 03   bitout ldx bitmsk        bit mask/funkt.tasten temporär
1463  e511  60                rts
1464  e512
1465  e512
1466  e512  2c 9b 03   setbit bit scrdis        scroll disable flag
1467  e515  70 e0             bvs getbt1        test auf doppelzeile (xr = zeile)
1468  e517  20 1e e5          jsr bitpos        flag für doppelzeile holen, zeiger
                                                für bit-tab. setzen
1469  e51a  15 e2             ora bitabl,x      tabelle basic-doppelzeilen (4 byte)
1470  e51c  d0 ee             bne bitsav
1471  e51e
1472  e51e
1473  e51e  ===> flag für doppelzeile holen, zeiger für bit-tab. setzen <===
1474  e51e
1475  e51e  8e 9c 03   bitpos stx bitmsk        bit mask/funkt.tasten temporär
1476  e521  8a                txa
1477  e522  29 07             and #$07
1478  e524  aa                tax
1479  e525  bd 67 ec          lda keyend,x      bit-tabelle für zeilen-link
1480  e528  48                pha
1481  e529  ad 9c 03          lda bitmsk        bit mask/funkt.tasten temporär
1482  e52c  4a                lsr a
1483  e52d  4a                lsr a
```

```
zeile adr.  obj.-code   source-code


1484  e52e  4a              lsr a
1485  e52f  aa              tax
1486  e530  68              pla
1487  e531  60              rts
1488  e532
1489  e532
1490  e532  ===> cursor an zeilenanfang (esc, j) <===
1491  e532
1492  e532  a4 de     fndfst ldy sclf        linker rand bildschirm
1493  e534  84 cb            sty pntr        cursor spalte
1494  e536
1495  e536
1496  e536  ===> cursor auf zeilenanfang <===
1497  e536
1498  e536  20 f5 e4  fistrt jsr getbit      test auf doppelzeile (akt. zeile)
1499  e539  90 06            bcc fnd0
1500  e53b  c6 ca            dec tblx        cursor zeile
1501  e53d  10 f7            bpl fistrt      cursor auf zeilenanfang
1502  e53f  e6 ca            inc tblx        cursor zeile
1503  e541  4c cd e0  fnd0   jmp stupt       bs-zeiger auf cursor-zeile
1504  e544
1505  e544
1506  e544  ===> cursor an zeilenende (esc, k) <===
1507  e544
1508  e544  a5 ca     fndend lda tblx        cursor zeile
1509  e546  c5 dd            cmp scbot       unterste zeile bildschirm (0-25)
1510  e548  b0 09            bcs eloupp
1511  e54a  e6 ca            inc tblx        cursor zeile
1512  e54c  20 f5 e4         jsr getbit      test auf doppelzeile (akt. zeile)
1513  e54f  b0 f3            bcs fndend      cursor an zeilenende (esc, k)
1514  e551  c6 ca            dec tblx        cursor zeile
1515  e553  20 cd e0  eloupp jsr stupt       bs-zeiger auf cursor-zeile
1516  e556  a4 df            ldy scrt        rechter rand bildschirm
1517  e558  84 cb            sty pntr        cursor spalte
1518  e55a  10 05            bpl eloup2
1519  e55c  20 87 e5  eloup1 jsr bakchr      cursor 1 position nach links
1520  e55f  b0 10            bcs endbye
1521  e561  20 42 e2  eloup2 jsr get1ch      zeichen von cursorpos. in ac
1522  e564  c9 20            cmp #$20
1523  e566  d0 09            bne endbye
1524  e568  c4 de            cpy sclf        linker rand bildschirm
1525  e56a  d0 f0            bne eloup1
1526  e56c  20 f5 e4         jsr getbit      test auf doppelzeile (akt. zeile)
1527  e56f  b0 eb            bcs eloup1
1528  e571  84 d5     endbye sty indx        position letztes zeichen in zeile
1529  e573  60               rts
1530  e574
1531  e574
1532  e574  ===> cursor 1 pos. rechts <===
1533  e574
1534  e574  48        nxtchr pha
1535  e575  a4 cb            ldy pntr        cursor spalte
1536  e577  c4 df            cpy scrt        rechter rand bildschirm
1537  e579  90 07            bcc bumpnt
1538  e57b  20 7a e3         jsr nxln        test bs-ende, cursor nächste zeile
1539  e57e  a4 de            ldy sclf        linker rand bildschirm
1540  e580  88               dey
1541  e581  38               sec
```

```
zeile adr.  obj.-code   source-code


1542  e582  c8        bumpnt iny
1543  e583  84 cb             sty pntr         cursor spalte
1544  e585  68               pla
1545  e586  60               rts
1546  e587
1547  e587
1548  e587  ===> cursor 1 position nach links <===
1549  e587
1550  e587  a4 cb     bakchr ldy pntr         cursor spalte
1551  e589  88               dey
1552  e58a  30 04            bmi bakot1
1553  e58c  c4 de            cpy sclf         linker rand bildschirm
1554  e58e  b0 0f            bcs bakout
1555  e590  a4 dc     bakot1 ldy sctop        oberste zeile bildschirm (0-25)
1556  e592  c4 ca            cpy tblx         cursor zeile
1557  e594  b0 0e            bcs bakot2
1558  e596  c6 ca            dec tblx         cursor zeile
1559  e598  48               pha
1560  e599  20 cd e0         jsr stupt        bs-zeiger auf cursor-zeile
1561  e59c  68               pla
1562  e59d  a4 df            ldy scrt         rechter rand bildschirm
1563  e59f  84 cb     bakout sty pntr         cursor spalte
1564  e5a1  c4 df            cpy scrt         rechter rand bildschirm
1565  e5a3  18               clc
1566  e5a4  60        bakot2 rts
1567  e5a5
1568  e5a5
1569  e5a5  ===> cursor-zeile und -spalte retten <===
1570  e5a5
1571  e5a5  a4 cb     savpos ldy pntr         cursor spalte
1572  e5a7  84 d9            sty sedt1        mehrfach benutzte editorvariable
1573  e5a9  a6 ca            ldx tblx         cursor zeile
1574  e5ab  86 da            stx sedt2        mehrfach benutzte editorvariable
1575  e5ad  60               rts
1576  e5ae
1577  e5ae
1578  e5ae  ===> zeichen löschen ($14), einfügen ($94) <===
1579  e5ae
1580  e5ae  b0 34     delins bcs insert       zeichen einfügen ($94)
1581  e5b0
1582  e5b0
1583  e5b0  ===> zeichen löschen ($14) <===
1584  e5b0
1585  e5b0  20 39 e3  deleet jsr cleft        cursor nach links
1586  e5b3  20 a5 e5         jsr savpos       cursor-zeile und -spalte retten
1587  e5b6  b0 0f            bcs delout
1588  e5b8  c4 df     deloop cpy scrt         rechter rand bildschirm
1589  e5ba  90 16            bcc delop1
1590  e5bc  a6 ca            ldx tblx         cursor zeile
1591  e5be  e8               inx
1592  e5bf  20 f7 e4         jsr getbt1       test auf doppelzeile (xr = zeile)
1593  e5c2  b0 0e            bcs delop1
1594  e5c4  20 1a e2         jsr doblnk       space an cursorposition
1595  e5c7  a5 d9     delout lda sedt1        mehrfach benutzte editorvariable
1596  e5c9  85 cb            sta pntr         cursor spalte
1597  e5cb  a5 da            lda sedt2        mehrfach benutzte editorvariable
1598  e5cd  85 ca            sta tblx         cursor zeile
1599  e5cf  4c cd e0         jmp stupt        bs-zeiger auf cursor-zeile
```

```
zeile adr.  obj.-code   source-code

1600  e5d2
1601  e5d2
1602  e5d2  20 74 e5  delop1 jsr nxtchr       cursor 1 pos. rechts
1603  e5d5  20 42 e2         jsr get1ch       zeichen von cursorpos. in ac
1604  e5d8  20 87 e5         jsr bakchr       cursor 1 position nach links
1605  e5db  20 1c e2         jsr dspp         char in ac an cursorposition
1606  e5de  20 74 e5         jsr nxtchr       cursor 1 pos. rechts
1607  e5e1  4c b8 e5         jmp deloop
1608  e5e4
1609  e5e4
1610  e5e4  ===> zeichen einfügen ($94) <===
1611  e5e4
1612  e5e4  20 a5 e5  insert jsr savpos       cursor-zeile und -spalte retten
1613  e5e7  20 44 e5         jsr fndend       cursor an zeilenende (esc, k)
1614  e5ea  e4 da            cpx sedt2        mehrfach benutzte editorvariable
1615  e5ec  d0 02            bne ins10
1616  e5ee  c4 d9            cpy sedt1        mehrfach benutzte editorvariable
1617  e5f0  90 21     ins10  bcc ins50
1618  e5f2  20 2e e6         jsr movchr       test zeilenende, zeile einfügen
1619  e5f5  b0 22            bcs insout
1620  e5f7  20 87 e5  ins30  jsr bakchr       cursor 1 position nach links
1621  e5fa  20 42 e2         jsr get1ch       zeichen von cursorpos. in ac
1622  e5fd  20 74 e5         jsr nxtchr       cursor 1 pos. rechts
1623  e600  20 1c e2         jsr dspp         char in ac an cursorposition
1624  e603  20 87 e5         jsr bakchr       cursor 1 position nach links
1625  e606  a6 ca            ldx tblx         cursor zeile
1626  e608  e4 da            cpx sedt2        mehrfach benutzte editorvariable
1627  e60a  d0 eb            bne ins30
1628  e60c  c4 d9            cpy sedt1        mehrfach benutzte editorvariable
1629  e60e  d0 e7            bne ins30
1630  e610  20 1a e2         jsr doblnk       space an cursorposition
1631  e613  e6 d3     ins50  inc insrt        insert flag
1632  e615  d0 02            bne insout
1633  e617  c6 d3            dec insrt        insert flag
1634  e619  4c c7 e5  insout jmp delout
1635  e61c
1636  e61c
1637  e61c  ===> dload"*" 'cr' 'run' 'cr' ($83) <===
1638  e61c
1639  e61c  90 0f     stprun bcc runrts
1640  e61e  78               sei
1641  e61f  a2 09            ldx #$09
1642  e621  86 d1            stx ndx          anzahl bytes im tastaturpuffer
1643  e623  bd a8 eb  runlop lda runtb-1,x
1644  e626  9d aa 03         sta keyd-1,x     tastaturpuffer -1 (10 bytes)
1645  e629  ca               dex
1646  e62a  d0 f7            bne runlop
1647  e62c  58               cli
1648  e62d  60        runrts rts
1649  e62e
1650  e62e
1651  e62e  ===> test zeilenende, zeile einfügen <===
1652  e62e
1653  e62e  c4 df     movchr cpy scrt         rechter rand bildschirm
1654  e630  90 0b            bcc movc10
1655  e632  a6 ca            ldx tblx         cursor zeile
1656  e634  e4 dd            cpx scbot        unterste zeile bildschirm (0-25)
1657  e636  90 05            bcc movc10
```

```
zeile adr.  obj.-code   source-code

1658  e638  2c 9b 03         bit scrdis       scroll disable flag
1659  e63b  30 17            bmi movc30
1660  e63d  20 cd e0  movc10 jsr stupt        bs-zeiger auf cursor-zeile
1661  e640  20 74 e5         jsr nxtchr       cursor 1 pos. rechts
1662  e643  90 0f            bcc movc30
1663  e645  20 f5 e4         jsr getbit       test auf doppelzeile (akt. zeile)
1664  e648  b0 09            bcs movc20
1665  e64a  20 00 ed         jsr bszei1       anzahl zeilen bs=1, dann pla, pla
1666  e64d  38               sec
1667  e64e  70 04            bvs movc30
1668  e650  20 cd e3         jsr scrdwn       nach unten scrollen
1669  e653  18        movc20 clc
1670  e654  60        movc30 rts
1671  e655
1672  e655
1673  e655  6c 20 03  sequen jmp (escvec)     addrl user esc-tasten-vector
1674  e658
1675  e658
1676  e658  ===> zeile unter aktueller zeile einfügen (esc, i) <===
1677  e658
1678  e658  20 cd e3  iline  jsr scrdwn       nach unten scrollen
1679  e65b  20 c7 e0         jsr stu10
1680  e65e  e8               inx
1681  e65f  20 f7 e4         jsr getbt1       test auf doppelzeile (xr = zeile)
1682  e662  08               php
1683  e663  20 01 e5         jsr putbit       akt. zeile als doppelzeile eintragen
1684  e666  28               plp
1685  e667  b0 03            bcs linrts
1686  e669  38               sec
1687  e66a  66 cf            ror lsxp         letzte position (zeile)
1688  e66c  60        linrts rts
1689  e66d
1690  e66d
1691  e66d  ===> aktuelle zeile löschen (esc, d) <===
1692  e66d
1693  e66d  20 36 e5  dline  jsr fistrt       cursor auf zeilenanfang
1694  e670  a5 dc            lda sctop        oberste zeile bildschirm (0-25)
1695  e672  48               pha
1696  e673  a5 ca            lda tblx         cursor zeile
1697  e675  85 dc            sta sctop        oberste zeile bildschirm (0-25)
1698  e677  ad 9e 03         lda logscr       logical/physical scroll flag
1699  e67a  48               pha
1700  e67b  a9 80            lda #$80
1701  e67d  8d 9e 03         sta logscr       logical/physical scroll flag
1702  e680  20 08 e4         jsr scru15
1703  e683  68               pla
1704  e684  8d 9e 03         sta logscr       logical/physical scroll flag
1705  e687  a5 dc            lda sctop        oberste zeile bildschirm (0-25)
1706  e689  85 ca            sta tblx         cursor zeile
1707  e68b  68               pla
1708  e68c  85 dc            sta sctop        oberste zeile bildschirm (0-25)
1709  e68e  38               sec
1710  e68f  66 cf            ror lsxp         letzte position (zeile)
1711  e691  4c c7 e0         jmp stu10
1712  e694
1713  e694
1714  e694  ===> zeile von cursor bis ende löschen (esc, q) <===
1715  e694
```

```
zeile adr.  obj.-code   source-code

1716  e694  20 a5 e5  etoeol jsr savpos       cursor-zeile und -spalte retten
1717  e697  20 2c e2  etol   jsr clrprt
1718  e69a  e6 ca            inc tblx         cursor zeile
1719  e69c  20 cd e0         jsr stupt        bs-zeiger auf cursor-zeile
1720  e69f  a4 de            ldy sclf         linker rand bildschirm
1721  e6a1  20 f5 e4         jsr getbit       test auf doppelzeile (akt. zeile)
1722  e6a4  b0 f1            bcs etol
1723  e6a6  4c c7 e5  etout  jmp delout
1724  e6a9
1725  e6a9
1726  e6a9  ===> zeile von cursor bis anfang löschen (esc, p) <===
1727  e6a9
1728  e6a9  20 a5 e5  etosol jsr savpos       cursor-zeile und -spalte retten
1729  e6ac  20 1a e2  etstol jsr doblnk       space an cursorposition
1730  e6af  c4 de            cpy sclf         linker rand bildschirm
1731  e6b1  d0 05            bne ets100
1732  e6b3  20 f5 e4         jsr getbit       test auf doppelzeile (akt. zeile)
1733  e6b6  90 ee            bcc etout
1734  e6b8  20 87 e5  ets100 jsr bakchr       cursor 1 position nach links
1735  e6bb  90 ef            bcc etstol
1736  e6bd
1737  e6bd
1738  e6bd  ===> bildschirm aufwärts rollen (esc, v) <===
1739  e6bd
1740  e6bd  20 a5 e5  suup   jsr savpos       cursor-zeile und -spalte retten
1741  e6c0  8a               txa
1742  e6c1  48               pha
1743  e6c2  20 f6 e3         jsr scrup        nach oben scrollen
1744  e6c5  68               pla
1745  e6c6  85 da            sta sedt2        mehrfach benutzte editorvariable
1746  e6c8  4c a6 e6         jmp etout
1747  e6cb
1748  e6cb
1749  e6cb  ===> bildschirm abwärts rollen (esc, w) <===
1750  e6cb
1751  e6cb  20 a5 e5  sddn   jsr savpos       cursor-zeile und -spalte retten
1752  e6ce  20 f5 e4         jsr getbit       test auf doppelzeile (akt. zeile)
1753  e6d1  b0 03            bcs sddn2
1754  e6d3  38               sec
1755  e6d4  66 cf            ror lsxp         letzte position (zeile)
1756  e6d6  a5 dc     sddn2  lda sctop        oberste zeile bildschirm (0-25)
1757  e6d8  85 ca            sta tblx         cursor zeile
1758  e6da  20 cd e3         jsr scrdwn       nach unten scrollen
1759  e6dd  20 05 e5         jsr clrbit       akt. zeile als 1.doppelzeile
                                              eintragen
1760  e6e0  4c a6 e6         jmp etout
1761  e6e3
1762  e6e3
1763  e6e3  ===> bildschirm-scroll ein (esc, l) <===
1764  e6e3
1765  e6e3  18        scrsw0 clc
1766  e6e4  24               .byte $24
1767  e6e5
1768  e6e5
1769  e6e5  ===> bildschirm-scroll aus (esc, m) <===
1770  e6e5
1771  e6e5  38        scrsw1 sec
1772  e6e6  a9 00            lda #$00
```

```
zeile adr.  obj.-code   source-code

1773  e6e8  6a              ror a
1774  e6e9  8d 9b 03        sta scrdis       scroll disable flag
1775  e6ec  60              rts
1776  e6ed
1777  e6ed
1778  e6ed  18              clc
1779  e6ee  90 01           bcc logsw
1780  e6f0  38              sec
1781  e6f1  a9 00     logsw lda #$00
1782  e6f3  6a              ror a
1783  e6f4  8d 9e 03        sta logscr       logical/physical scroll flag
1784  e6f7  60              rts
1785  e6f8
1786  e6f8
1787  e6f8  ===> funktionstasten setzen, ändern, listen <===
1788  e6f8
1789  e6f8  78        keyfun sei
1790  e6f9  88               dey
1791  e6fa  30 03            bmi listky       def. funktionstasten-liste
1792  e6fc  4c be e7         jmp addkey       zusätzliche funktions-tasten
1793  e6ff
1794  e6ff
1795  e6ff  ===> def. funktionstasten-liste <===
1796  e6ff
1797  e6ff  a0 00     listky ldy #$00
1798  e701  c8        listlp iny
1799  e702  8c b7 03         sty sedt3        temporär list funktionstasten
1800  e705  88               dey
1801  e706  b9 83 03         lda keysiz,y     funktionstasten (20bytes)
1802  e709  f0 6c            beq nodefn
1803  e70b  8d 9d 03         sta keyidx       index programmierbare befehle
1804  e70e  20 49 e9         jsr findky       funktiontasten-adr. in ac,xr
1805  e711  85 c2            sta keypnt       addrl akt. pgm tastenpuffer
1806  e713  86 c3            stx keypnt+1     addrh akt. pgm tastenpuffer
1807  e715  a2 03            ldx #$03
1808  e717  bd a6 e7  preamb lda keword,x     text für funktions-tasten
1809  e71a  20 d2 ff         jsr kbsout       ausgabe 1 char auf akt. kanal
1810  e71d  ca               dex
1811  e71e  10 f7            bpl preamb
1812  e720  a2 2f            ldx #$2f
1813  e722  ad b7 03         lda sedt3        temporär list funktionstasten
1814  e725  38               sec
1815  e726  e8        ky2asc inx
1816  e727  e9 0a            sbc #$0a
1817  e729  b0 fb            bcs ky2asc
1818  e72b  69 3a            adc #$3a
1819  e72d  e0 30            cpx #$30
1820  e72f  f0 06            beq nosec
1821  e731  48               pha
1822  e732  8a               txa
1823  e733  20 d2 ff         jsr kbsout       ausgabe 1 char auf akt. kanal
1824  e736  68               pla
1825  e737  20 d2 ff  nosec  jsr kbsout       ausgabe 1 char auf akt. kanal
1826  e73a  a0 00            ldy #$00
1827  e73c  a9 2c            lda #$2c
1828  e73e  20 d2 ff  lstk20 jsr kbsout       ausgabe 1 char auf akt. kanal
1829  e741  a2 07            ldx #$07
1830  e743  20 82 e2  txtprt jsr pagst2       segment funktionstasten setzen
```

```
zeile adr.  obj.-code   source-code

1831  e746  b1 c2           lda (keypnt),y   addrl akt. pgm tastenpuffer
1832  e748  20 91 e2        jsr pagres       segment zurücksetzen
1833  e74b  c9 0d           cmp #$0d
1834  e74d  f0 32           beq lstkcr
1835  e74f  c9 8d           cmp #$8d
1836  e751  f0 31           beq lstksc
1837  e753  c9 22           cmp #$22
1838  e755  f0 30           beq lstkat
1839  e757  e0 09           cpx #$09
1840  e759  f0 07           beq lstk10
1841  e75b  48              pha
1842  e75c  a9 22           lda #$22
1843  e75e  20 d2 ff        jsr kbsout       ausgabe 1 char auf akt. kanal
1844  e761  68              pla
1845  e762  20 d2 ff  lstk10 jsr kbsout       ausgabe 1 char auf akt. kanal
1846  e765  a2 09           ldx #$09
1847  e767  c8              iny
1848  e768  cc 9d 03        cpy keyidx       index programmierbare befehle
1849  e76b  d0 d6           bne txtprt
1850  e76d  a9 22           lda #$22
1851  e76f  20 d2 ff        jsr kbsout       ausgabe 1 char auf akt. kanal
1852  e772  a9 0d     lstk30 lda #$0d
1853  e774  20 d2 ff        jsr kbsout       ausgabe 1 char auf akt. kanal
1854  e777  ac b7 03  nodefn ldy sedt3        temporär list funktionstasten
1855  e77a  c0 14           cpy #$14
1856  e77c  d0 83           bne listlp
1857  e77e  58              cli
1858  e77f  18              clc
1859  e780  60              rts
1860  e781
1861  e781
1862  e781  a2 0a     lstkcr ldx #$0a
1863  e783  2c               .byte $2c
1864  e784  a2 13     lstksc ldx #$13
1865  e786  2c               .byte $2c
1866  e787  a2 0e     lstkat ldx #$0e
1867  e789  8a               txa
1868  e78a  48               pha
1869  e78b  a2 06            ldx #$06
1870  e78d  bd aa e7  lstklp lda cdword,x
1871  e790  f0 0a            beq lstk40
1872  e792  20 d2 ff         jsr kbsout       ausgabe 1 char auf akt. kanal
1873  e795  ca               dex
1874  e796  10 f5            bpl lstklp
1875  e798  68               pla
1876  e799  aa               tax
1877  e79a  d0 f1            bne lstklp
1878  e79c  c8        lstk40 iny
1879  e79d  cc 9d 03         cpy keyidx       index programmierbare befehle
1880  e7a0  f0 d0            beq lstk30
1881  e7a2  a9 2b            lda #$2b
1882  e7a4  d0 98            bne lstk20
1883  e7a6
1884  e7a6
1885  e7a6  ===> text für funktions-tasten <===
1886  e7a6
1887  e7a6  20 59 45  keword .byte ' yek'
1887  e7a9  4b
```

```
zeile adr.  obj.-code   source-code

1888  e7aa  28 24 52  cdword .byte '($rhc+"', $00
1888  e7ad  48 43 2b
1888  e7b0  22
1888  e7b1  00
1889  e7b2  29 33 31         .byte ')31', $00
1889  e7b5  00
1890  e7b6  29 34 33         .byte ')43', $00
1890  e7b9  00
1891  e7ba  29 31 34         .byte ')141'
1891  e7bd  31
1892  e7be
1893  e7be
1894  e7be  ===> zusätzliche funktions-tasten <===
1895  e7be
1896  e7be  48        addkey pha
1897  e7bf  aa               tax
1898  e7c0  84 d9            sty sedt1        mehrfach benutzte editorvariable
1899  e7c2  b5 00            lda e6509,x      6509 execution register
1900  e7c4  38               sec
1901  e7c5  f9 83 03         sbc keysiz,y     funktionstasten (20bytes)
1902  e7c8  85 da            sta sedt2        mehrfach benutzte editorvariable
1903  e7ca  6e 9c 03         ror bitmsk       bit mask/funkt.tasten temporär
1904  e7cd  c8               iny
1905  e7ce  20 49 e9         jsr findky       funktiontasten-adr. in ac,xr
1906  e7d1  85 c4            sta sedsal       addrl scroll-zeiger start
1907  e7d3  86 c5            stx sedsal+1     addrh scroll-zeiger start
1908  e7d5  a0 14            ldy #$14
1909  e7d7  20 49 e9         jsr findky       funktiontasten-adr. in ac,xr
1910  e7da  85 c6            sta sedeal       addrl scroll-zeiger ende
1911  e7dc  86 c7            stx sedeal+1     addrh scroll-zeiger ende
1912  e7de  ac 9c 03         ldy bitmsk       bit mask/funkt.tasten temporär
1913  e7e1  10 13            bpl keysho
1914  e7e3  18               clc
1915  e7e4  ed 80 03         sbc pkyend       addrl endadr. funkt.tastenpuffer
1916  e7e7  a8               tay
1917  e7e8  8a               txa
1918  e7e9  ed 81 03         sbc pkyend+1     addrh endadr. funkt.tastenpuffer
1919  e7ec  aa               tax
1920  e7ed  98               tya
1921  e7ee  18               clc
1922  e7ef  65 da            adc sedt2        mehrfach benutzte editorvariable
1923  e7f1  8a               txa
1924  e7f2  69 00            adc #$00
1925  e7f4  b0 6c            bcs kyxit
1926  e7f6  20 82 e2  keysho jsr pagst2       segment funktionstasten setzen
1927  e7f9  a5 c6     kymove lda sedeal       addrl scroll-zeiger ende
1928  e7fb  18               clc
1929  e7fc  e5 c4            sbc sedsal       addrl scroll-zeiger start
1930  e7fe  a5 c7            lda sedeal+1     addrh scroll-zeiger ende
1931  e800  e5 c5            sbc sedsal+1     addrh scroll-zeiger start
1932  e802  90 2a            bcc keyins
1933  e804  a0 00            ldy #$00
1934  e806  ad 9c 03         lda bitmsk       bit mask/funkt.tasten temporär
1935  e809  10 11            bpl kshort
1936  e80b  a5 c6            lda sedeal       addrl scroll-zeiger ende
1937  e80d  d0 02            bne newky4
1938  e80f  c6 c7            dec sedeal+1     addrh scroll-zeiger ende
1939  e811  c6 c6     newky4 dec sedeal       addrl scroll-zeiger ende
```

```
zeile adr.  obj.-code   source-code

1940  e813  b1 c6          lda (sedeal),y   addrl scroll-zeiger ende
1941  e815  a4 da          ldy sedt2        mehrfach benutzte editorvariable
1942  e817  91 c6          sta (sedeal),y   addrl scroll-zeiger ende
1943  e819  4c f9 e7       jmp kymove
1944  e81c
1945  e81c
1946  e81c  b1 c4     kshort lda (sedsal),y  addrl scroll-zeiger start
1947  e81e  a4 da          ldy sedt2        mehrfach benutzte editorvariable
1948  e820  c6 c5          dec sedsal+1     addrh scroll-zeiger start
1949  e822  91 c4          sta (sedsal),y   addrl scroll-zeiger start
1950  e824  e6 c5          inc sedsal+1     addrh scroll-zeiger start
1951  e826  e6 c4          inc sedsal       addrl scroll-zeiger start
1952  e828  d0 cf          bne kymove
1953  e82a  e6 c5          inc sedsal+1     addrh scroll-zeiger start
1954  e82c  d0 cb          bne kymove
1955  e82e  a4 d9     keyins ldy sedt1       mehrfach benutzte editorvariable
1956  e830  20 49 e9       jsr findky       funktiontasten-adr. in ac,xr
1957  e833  85 c4          sta sedsal       addrl scroll-zeiger start
1958  e835  86 c5          stx sedsal+1     addrh scroll-zeiger start
1959  e837  a4 d9          ldy sedt1        mehrfach benutzte editorvariable
1960  e839  68             pla
1961  e83a  48             pha
1962  e83b  aa             tax
1963  e83c  b5 00          lda e6509,x      6509 execution register
1964  e83e  99 83 03       sta keysiz,y     funktionstasten (20bytes)
1965  e841  a8             tay
1966  e842  f0 1a          beq kyinok
1967  e844  b5 01          lda i6509,x      6509 indirection register
1968  e846  85 c6          sta sedeal       addrl scroll-zeiger ende
1969  e848  b5 02          lda usrpok,x     jmp code für 'usr'-routine
1970  e84a  85 c7          sta sedeal+1     addrh scroll-zeiger ende
1971  e84c  88        kyinlp dey
1972  e84d  b5 03          lda usrpok+1,x   addrl  'usr'-routine
1973  e84f  85 01          sta i6509        6509 indirection register
1974  e851  b1 c6          lda (sedeal),y   addrl scroll-zeiger ende
1975  e853  20 91 e2       jsr pagres       segment zurücksetzen
1976  e856  20 82 e2       jsr pagst2       segment funktionstasten setzen
1977  e859  91 c4          sta (sedsal),y   addrl scroll-zeiger start
1978  e85b  98             tya
1979  e85c  d0 ee          bne kyinlp
1980  e85e  20 91 e2  kyinok jsr pagres      segment zurücksetzen
1981  e861  18             clc
1982  e862  68        kyxit  pla
1983  e863  58             cli
1984  e864  60             rts
1985  e865
1986  e865
1987  e865  ===> tast. lesen, in puffer schreiben <===
1988  e865
1989  e865  a0 ff      key    ldy #$ff
1990  e867  84 e0            sty modkey       flag shift/control-taste
1991  e869  84 e1            sty norkey       flag normale taste
1992  e86b  c8               iny
1993  e86c  8c 01 df         sty tri2+pb      6525 triport 2: port reg. b
1994  e86f  8c 00 df         sty tri2+pa      6525 triport 2: port reg. a
1995  e872  20 1e e9         jsr getkey       tastaturabfrage über triport 2
1996  e875  29 3f            and #$3f
1997  e877  49 3f            eor #$3f
```

```
zeile adr.  obj.-code   source-code

1998  e879  d0 03           bne keyx1
1999  e87b  4c f3 e8        jmp nulxit
2000  e87e
2001  e87e
2002  e87e  a9 ff     keyx1  lda #$ff
2003  e880  8d 00 df        sta tri2+pa      6525 triport 2: port reg. a
2004  e883  0a              asl a
2005  e884  8d 01 df        sta tri2+pb      6525 triport 2: port reg. b
2006  e887  20 1e e9        jsr getkey       tastaturabfrage über triport 2
2007  e88a  48              pha
2008  e88b  85 e0           sta modkey       flag shift/control-taste
2009  e88d  09 30           ora #$30
2010  e88f  d0 03           bne line01
2011  e891  20 1e e9  linelp jsr getkey      tastaturabfrage über triport 2
2012  e894  a2 05     line01 ldx #$05
2013  e896  4a        kyloop lsr a
2014  e897  90 10           bcc havkey       tastenwert aus tabelle lesen
2015  e899  c8              iny
2016  e89a  ca              dex
2017  e89b  10 f9           bpl kyloop
2018  e89d  38              sec
2019  e89e  2e 01 df        rol tri2+pb      6525 triport 2: port reg. b
2020  e8a1  2e 00 df        rol tri2+pa      6525 triport 2: port reg. a
2021  e8a4  b0 eb           bcs linelp
2022  e8a6  68              pla
2023  e8a7  90 4a           bcc nulxit
2024  e8a9
2025  e8a9
2026  e8a9  ===> tastenwert aus tabelle lesen <===
2027  e8a9
2028  e8a9  84 e1     havkey sty norkey       flag normale taste
2029  e8ab  be 29 ea         ldx normtb,y     code-tab. für tastatur (-shift)
2030  e8ae  68              pla
2031  e8af  0a              asl a
2032  e8b0  0a              asl a
2033  e8b1  0a              asl a
2034  e8b2  90 0e           bcc doctl
2035  e8b4  30 0f           bmi havasc       tastenwert in tast.-puffer
2036  e8b6  be 89 ea        ldx shfttb,y     code-tab. für tastatur (+shift)
2037  e8b9  a5 cc           lda grmode       flag für grafik/text-modus
2038  e8bb  f0 08           beq havasc       tastenwert in tast.-puffer
2039  e8bd  be e9 ea        ldx shftgr,y     code-tab. für tastatur
                                             (graphik+shift)
2040  e8c0  d0 03           bne havasc       tastenwert in tast.-puffer
2041  e8c2  be 49 eb  doctl  ldx ctltbl,y     code-tab. für tastatur
                                             (control-modus)
2042  e8c5
2043  e8c5
2044  e8c5  ===> tastenwert in tast.-puffer <===
2045  e8c5
2046  e8c5  e0 ff     havasc cpx #$ff
2047  e8c7  f0 2c           beq keyxit
2048  e8c9  e0 e0           cpx #$e0
2049  e8cb  90 09           bcc notfun
2050  e8cd  98              tya
2051  e8ce  48              pha
2052  e8cf  20 27 e9        jsr funjmp       aufruf funktionstasten (ind.-vektor)
2053  e8d2  68              pla
```

```
zeile adr.  obj.-code   source-code

2054  e8d3  a8              tay
2055  e8d4  b0 1f           bcs keyxit
2056  e8d6  8a       notfun txa
2057  e8d7  c4 cd           cpy lstx          index letztes zeichen
2058  e8d9  f0 27           beq dorat         repeat-schleife
2059  e8db  a2 13           ldx #$13
2060  e8dd  86 d8           stx delay         zeit bis zum einsetzen des repeat
2061  e8df  a6 d1           ldx ndx           anzahl bytes im tastaturpuffer
2062  e8e1  e0 09           cpx #$09
2063  e8e3  f0 0e           beq nulxit
2064  e8e5  c0 59           cpy #$59
2065  e8e7  d0 29           bne savkey
2066  e8e9  e0 08           cpx #$08
2067  e8eb  f0 06           beq nulxit
2068  e8ed  9d ab 03        sta keyd,x        tastaturpuffer (10 bytes)
2069  e8f0  e8              inx
2070  e8f1  d0 1f           bne savkey
2071  e8f3  a0 ff     nulxit ldy #$ff
2072  e8f5  84 cd     keyxit sty lstx         index letztes zeichen
2073  e8f7  a2 7f     keyxt2 ldx #$7f
2074  e8f9  8e 00 df        stx tri2+pa       6525 triport 2: port reg. a
2075  e8fc  a2 ff           ldx #$ff
2076  e8fe  8e 01 df        stx tri2+pb       6525 triport 2: port reg. b
2077  e901  60              rts
2078  e902
2079  e902
2080  e902  ===> repeat-schleife <===
2081  e902
2082  e902  c6 d8     dorat  dec delay         zeit bis zum einsetzen des repeat
2083  e904  10 f1           bpl keyxt2
2084  e906  e6 d8           inc delay         zeit bis zum einsetzen des repeat
2085  e908  c6 d7           dec rptcnt        zeit zwischen zeichen bei repeat
2086  e90a  10 eb           bpl keyxt2
2087  e90c  e6 d7           inc rptcnt        zeit zwischen zeichen bei repeat
2088  e90e  a6 d1           ldx ndx           anzahl bytes im tastaturpuffer
2089  e910  d0 e5           bne keyxt2
2090  e912  9d ab 03  savkey sta keyd,x       tastaturpuffer (10 bytes)
2091  e915  e8              inx
2092  e916  86 d1           stx ndx           anzahl bytes im tastaturpuffer
2093  e918  a2 03           ldx #$03
2094  e91a  86 d7           stx rptcnt        zeit zwischen zeichen bei repeat
2095  e91c  d0 d7           bne keyxit
2096  e91e
2097  e91e
2098  e91e  ===> tastaturabfrage über triport 2 <===
2099  e91e
2100  e91e  ad 02 df  getkey lda tri2+pc      6525 triport 2: port reg. c
2101  e921  cd 02 df        cmp tri2+pc       6525 triport 2: port reg. c
2102  e924  d0 f8           bne getkey        tastaturabfrage über triport 2
2103  e926  60              rts
2104  e927
2105  e927
2106  e927  6c b5 03  funjmp jmp (funvec)     addrl ind. jmp für funktionstasten
2107  e92a
2108  e92a
2109  e92a  c4 cd           cpy lstx          index letztes zeichen
2110  e92c  f0 19           beq funrts
2111  e92e  a5 d1           lda ndx           anzahl bytes im tastaturpuffer
```

```
zeile adr.  obj.-code   source-code

2112  e930  05 d6            ora kyndx        zeichenzähler für pgm tastenbefehl
2113  e932  d0 13            bne funrts
2114  e934  8d 9d 03         sta keyidx       index programmierbare befehle
2115  e937  8a               txa
2116  e938  29 1f            and #$1f
2117  e93a  a8                tay
2118  e93b  b9 83 03         lda keysiz,y     funktionstasten (20bytes)
2119  e93e  85 d6            sta kyndx        zeichenzähler für pgm tastenbefehl
2120  e940  20 49 e9         jsr findky       funktiontasten-adr. in ac,xr
2121  e943  85 c2            sta keypnt       addrl akt. pgm tastenpuffer
2122  e945  86 c3            stx keypnt+1     addrh akt. pgm tastenpuffer
2123  e947  38        funrts sec
2124  e948  60               rts
2125  e949
2126  e949
2127  e949  ===> funktiontasten-adr. in ac,xr <===
2128  e949
2129  e949  a5 c0     findky lda pkybuf       addrl start programmierbare tasten
2130  e94b  a6 c1            ldx pkybuf+1     addrh start programmierbare tasten
2131  e94d  18        findlp clc
2132  e94e  88               dey
2133  e94f  30 08            bmi fndout
2134  e951  79 83 03         adc keysiz,y     funktionstasten (20bytes)
2135  e954  90 f7            bcc findlp
2136  e956  e8               inx
2137  e957  d0 f4            bne findlp
2138  e959  60        fndout rts
2139  e95a
2140  e95a
2141  e95a  ===> tabulator holen <===
2142  e95a
2143  e95a  98        gettab tya
2144  e95b  29 07            and #$07
2145  e95d  aa               tax
2146  e95e  bd 67 ec         lda keyend,x     bit-tabelle für zeilen-link
2147  e961  8d 9c 03         sta bitmsk       bit mask/funkt.tasten temporär
2148  e964  98               tya
2149  e965  4a               lsr a
2150  e966  4a               lsr a
2151  e967  4a               lsr a
2152  e968  aa               tax
2153  e969  bd a1 03         lda tab,x        tabstop flags (max.10byt/80bit)
2154  e96c  2c 9c 03         bit bitmsk       bit mask/funkt.tasten temporär
2155  e96f  60               rts
2156  e970
2157  e970
2158  e970  ===> bearbeitung von esc-funktionen <===
2159  e970
2160  e970  29 7f     escape and #$7f
2161  e972  38               sec
2162  e973  e9 41            sbc #$41
2163  e975  c9 1a            cmp #$1a
2164  e977  90 01            bcc escgo        esc-sequenz-adr. auf stack
2165  e979
2166  e979
2167  e979  ===> esc-sequenz abbrechen (esc, x) <===
2168  e979
2169  e979  60        escrts rts
```

```
zeile adr.  obj.-code   source-code


2170  e97a
2171  e97a
2172  e97a  ===> esc-sequenz-adr. auf stack <===
2173  e97a
2174  e97a  0a        escgo  asl a
2175  e97b  aa               tax
2176  e97c  bd 86 e9         lda escvct+1,x   adress-tabelle der esc-routinen (esc
                                               a-z)
2177  e97f  48               pha
2178  e980  bd 85 e9         lda escvct,x     adress-tabelle der esc-routinen (esc
                                               a-z)
2179  e983  48               pha
2180  e984  60               rts
2181  e985
2182  e985                   .end
2183  e985                   .lib editor3
```

```
zeile adr.  obj.-code   source-code


2185  e985
2186  e985  ===> adress-tabelle der esc-routinen (esc a-z) <===
2187  e985
2188  e985  22 ea     escvct .word auton-1     'auto insert'-modus ein (esc, a)
2189  e987  ba e9            .word sethtb-1    cursorpos- = untere rechte
                                               bildschirm-ecke (esc, b)
2190  e989  1f ea            .word autoff-1    'auto insert'-modus aus (esc, c)
2191  e98b  6c e6            .word dline-1     aktuelle zeile löschen (esc, d)
2192  e98d  ee e9            .word setcr4-1    festen cursor ein (esc, e)
2193  e98f  e5 e9            .word setcr2-1    blinkenden cursor ein (esc, f)
2194  e991  d5 e9            .word bellon-1    glocke am zeilenende ein (esc, g)
2195  e993  d7 e9            .word bellof-1    glocke am zeilenende aus (esc, h)
2196  e995  57 e6            .word iline-1     zeile unter aktueller zeile einfügen
                                               (esc, i)
2197  e997  31 e5            .word fndfst-1    cursor an zeilenanfang (esc, j)
2198  e999  43 e5            .word fndend-1    cursor an zeilenende (esc, k)
2199  e99b  e2 e6            .word scrsw0-1    bildschirm-scroll ein (esc, l)
2200  e99d  e4 e6            .word scrsw1-1    bildschirm-scroll aus (esc, m)
2201  e99f  04 ea            .word nrmscr-1    bildschirm 'revers' auf 'normal'
                                               (esc, n)
2202  e9a1  a1 e3            .word toqm-1      'inst-','quote-','rvs'-mode aus
2203  e9a3  a8 e6            .word etosol-1    zeile von cursor bis anfang löschen
                                               (esc, p)
2204  e9a5  93 e6            .word etoeol-1    zeile von cursor bis ende löschen
                                               (esc, q)
2205  e9a7  f5 e9            .word revscr-1    bildschirm 'normal' auf 'revers'
                                               (esc, r)
2206  e9a9  eb e9            .word setcr3-1    vollen cursor ein (esc, s)
2207  e9ab  b8 e9            .word sethtt-1    cursorpos. = obere linke
                                               bildschirm-ecke (esc, t)
2208  e9ad  db e9            .word setcr1-1    unterstrich-cursor ein (esc, u)
2209  e9af  bc e6            .word suup-1      bildschirm aufwärts rollen (esc, v)
2210  e9b1  ca e6            .word sddn-1      bildschirm abwärts rollen (esc, w)
2211  e9b3  78 e9            .word escrts-1    esc-sequenz abbrechen (esc, x)
2212  e9b5  07 ea            .word nrmset-1    standard zeichengen. ein (esc, y)
2213  e9b7  f8 e9            .word usrset-1    benutzer zeichengen. ein (esc, z)
2214  e9b9
2215  e9b9
2216  e9b9  ===> cursorpos. = obere linke bildschirm-ecke (esc, t) <===
2217  e9b9
2218  e9b9  18        sethtt clc
2219  e9ba  24               .byte $24
2220  e9bb
2221  e9bb
2222  e9bb  ===> cursorpos- = untere rechte bildschirm-ecke (esc, b) <===
2223  e9bb
2224  e9bb  38        sethtb sec
2225  e9bc
2226  e9bc
2227  e9bc  ===> bs-fenster links oben ($0f) / rechts unten ($8f) <===
2228  e9bc
2229  e9bc  a6 cb     window ldx pntr          cursor spalte
2230  e9be  a5 ca            lda tblx          cursor zeile
2231  e9c0  90 0f            bcc settps        ac = obere, xr = linke
                                               bildschirm-ecke
2232  e9c2
2233  e9c2
2234  e9c2  ===> ac = untere, xr = rechte bildschirm-ecke <===
```

```
zeile adr.  obj.-code   source-code

2235  e9c2
2236  e9c2  85 dd     setbts sta scbot        unterste zeile bildschirm (0-25)
2237  e9c4  86 df            stx scrt         rechter rand bildschirm
2238  e9c6  60               rts
2239  e9c7
2240  e9c7
2241  e9c7  ===> bs-fenster auf volle grösse (home-home) <===
2242  e9c7
2243  e9c7  a9 18     sreset lda #$18
2244  e9c9  a2 4f            ldx #$4f
2245  e9cb  20 c2 e9         jsr setbts       ac = untere, xr = rechte
                                              bildschirm-ecke
2246  e9ce  a9 00            lda #$00
2247  e9d0  aa               tax
2248  e9d1
2249  e9d1
2250  e9d1  ===> ac = obere, xr = linke bildschirm-ecke <===
2251  e9d1
2252  e9d1  85 dc     settps sta sctop        oberste zeile bildschirm (0-25)
2253  e9d3  86 de            stx sclf         linker rand bildschirm
2254  e9d5  60               rts
2255  e9d6
2256  e9d6
2257  e9d6  ===> glocke am zeilenende ein (esc, g) <===
2258  e9d6
2259  e9d6  a9 00     bellon lda #$00
2260  e9d8
2261  e9d8
2262  e9d8  ===> glocke am zeilenende aus (esc, h) <===
2263  e9d8
2264  e9d8  8d 9f 03  bellof sta bellmd       flag für glocke am zeilenende
2265  e9db  60               rts
2266  e9dc
2267  e9dc
2268  e9dc  ===> unterstrich-cursor ein (esc, u) <===
2269  e9dc
2270  e9dc  a9 0b     setcr1 lda #$0b
2271  e9de  2c 02 df         bit tri2+pc      6525 triport 2: port reg. c
2272  e9e1  30 05            bmi setung
2273  e9e3  a9 06            lda #$06
2274  e9e5  2c               .byte $2c
2275  e9e6
2276  e9e6
2277  e9e6  ===> blinkenden cursor ein (esc, f) <===
2278  e9e6
2279  e9e6  a9 60     setcr2 lda #$60
2280  e9e8  05 d4     setung ora config       cursor type / char. vor blinken
2281  e9ea  d0 07            bne setcr5
2282  e9ec
2283  e9ec
2284  e9ec  ===> vollen cursor ein (esc, s) <===
2285  e9ec
2286  e9ec  a9 f0     setcr3 lda #$f0
2287  e9ee  2c               .byte $2c
2288  e9ef
2289  e9ef
2290  e9ef  ===> festen cursor ein (esc, e) <===
2291  e9ef
```

```
zeile adr.  obj.-code   source-code

2292  e9ef  a9 0f     setcr4 lda #$0f
2293  e9f1  25 d4            and config       cursor type / char. vor blinken
2294  e9f3  85 d4     setcr5 sta config       cursor type / char. vor blinken
2295  e9f5  60               rts
2296  e9f6
2297  e9f6
2298  e9f6  ===> bildschirm 'normal' auf 'revers' (esc, r) <===
2299  e9f6
2300  e9f6  a9 20     revscr lda #$20
2301  e9f8  2c               .byte $2c
2302  e9f9
2303  e9f9
2304  e9f9  ===> benutzer zeichengen. ein (esc, z) <===
2305  e9f9
2306  e9f9  a9 10     usrset lda #$10
2307  e9fb  a2 0e            ldx #$0e
2308  e9fd  8e 00 d8         stx vdc+adreg    6845 vdc: adress-reg.
2309  ea00  0d 01 d8         ora vdc+dareg    6845 vdc: daten-reg.
2310  ea03  d0 0d            bne setscr
2311  ea05
2312  ea05
2313  ea05  ===> bildschirm 'revers' auf 'normal' (esc, n) <===
2314  ea05
2315  ea05  a9 df     nrmscr lda #$df
2316  ea07  2c               .byte $2c
2317  ea08
2318  ea08
2319  ea08  ===> standard zeichengen. ein (esc, y) <===
2320  ea08
2321  ea08  a9 ef     nrmset lda #$ef
2322  ea0a  a2 0e            ldx #$0e
2323  ea0c  8e 00 d8         stx vdc+adreg    6845 vdc: adress-reg.
2324  ea0f  2d 01 d8         and vdc+dareg    6845 vdc: daten-reg.
2325  ea12  8d 01 d8  setscr sta vdc+dareg    6845 vdc: daten-reg.
2326  ea15  29 30            and #$30
2327  ea17  a2 0c            ldx #$0c
2328  ea19  8e 00 d8         stx vdc+adreg    6845 vdc: adress-reg.
2329  ea1c  8d 01 d8         sta vdc+dareg    6845 vdc: daten-reg.
2330  ea1f  60               rts
2331  ea20
2332  ea20
2333  ea20  ===> 'auto insert'-modus aus (esc, c) <===
2334  ea20
2335  ea20  a9 00     autoff lda #$00
2336  ea22  2c               .byte $2c
2337  ea23
2338  ea23
2339  ea23  ===> 'auto insert'-modus ein (esc, a) <===
2340  ea23
2341  ea23  a9 ff     auton  lda #$ff
2342  ea25  8d 9a 03         sta insflg       auto insert flag
2343  ea28  60               rts
2344  ea29
2345  ea29
2346  ea29  ===> code-tab. für tastatur (-shift) <===
2347  ea29
2348  ea29  e0        normtb .byte $e0, $1b, $09, $ff, $00, $01
2348  ea2a  1b
```

```
zeile adr.  obj.-code   source-code

2348  ea2b  09
2348  ea2c  ff
2348  ea2d  00
2348  ea2e  01
2349  ea2f  e1              .byte $e1, $31, $51, $41, $5a, $ff
2349  ea30  31
2349  ea31  51
2349  ea32  41
2349  ea33  5a
2349  ea34  ff
2350  ea35  e2              .byte $e2, $32, $57, $53, $58, $43
2350  ea36  32
2350  ea37  57
2350  ea38  53
2350  ea39  58
2350  ea3a  43
2351  ea3b  e3              .byte $e3, $33, $45, $44, $46, $56
2351  ea3c  33
2351  ea3d  45
2351  ea3e  44
2351  ea3f  46
2351  ea40  56
2352  ea41  e4              .byte $e4, $34, $52, $54, $47, $42
2352  ea42  34
2352  ea43  52
2352  ea44  54
2352  ea45  47
2352  ea46  42
2353  ea47  e5              .byte $e5, $35, $36, $59, $48, $4e
2353  ea48  35
2353  ea49  36
2353  ea4a  59
2353  ea4b  48
2353  ea4c  4e
2354  ea4d  e6              .byte $e6, $37, $55, $4a, $4d, $20
2354  ea4e  37
2354  ea4f  55
2354  ea50  4a
2354  ea51  4d
2354  ea52  20
2355  ea53  e7              .byte $e7, $38, $49, $4b, $2c, $2e
2355  ea54  38
2355  ea55  49
2355  ea56  4b
2355  ea57  2c
2355  ea58  2e
2356  ea59  e8              .byte $e8, $39, $4f, $4c, $3b, $2f
2356  ea5a  39
2356  ea5b  4f
2356  ea5c  4c
2356  ea5d  3b
2356  ea5e  2f
2357  ea5f  e9              .byte $e9, $30, $2d, $50, $5b, $27
2357  ea60  30
2357  ea61  2d
2357  ea62  50
2357  ea63  5b
2357  ea64  27
```

```
zeile adr.  obj.-code   source-code

2358  ea65  11              .byte $11, $3d, $5f, $5d, $0d, $de
2358  ea66  3d
2358  ea67  5f
2358  ea68  5d
2358  ea69  0d
2358  ea6a  de
2359  ea6b  91              .byte $91, $9d, $1d, $14, $02, $ff
2359  ea6c  9d
2359  ea6d  1d
2359  ea6e  14
2359  ea6f  02
2359  ea70  ff
2360  ea71  13              .byte $13, $3f, $37, $34, $31, $30
2360  ea72  3f
2360  ea73  37
2360  ea74  34
2360  ea75  31
2360  ea76  30
2361  ea77  12              .byte $12, $04, $38, $35, $32, $2e
2361  ea78  04
2361  ea79  38
2361  ea7a  35
2361  ea7b  32
2361  ea7c  2e
2362  ea7d  8e              .byte $8e, $2a, $39, $36, $33, $30
2362  ea7e  2a
2362  ea7f  39
2362  ea80  36
2362  ea81  33
2362  ea82  30
2363  ea83  03              .byte $03, $2f, $2d, $2b, $0d, $ff
2363  ea84  2f
2363  ea85  2d
2363  ea86  2b
2363  ea87  0d
2363  ea88  ff
2364  ea89
2365  ea89
2366  ea89  ===> code-tab. für tastatur (+shift) <===
2367  ea89
2368  ea89  ea        shfttb .byte $ea, $1b, $89, $ff, $00, $01
2368  ea8a  1b
2368  ea8b  89
2368  ea8c  ff
2368  ea8d  00
2368  ea8e  01
2369  ea8f  eb              .byte $eb, $21, $d1, $c1, $da, $ff
2369  ea90  21
2369  ea91  d1
2369  ea92  c1
2369  ea93  da
2369  ea94  ff
2370  ea95  ec              .byte $ec, $40, $d7, $d3, $d8, $c3
2370  ea96  40
2370  ea97  d7
2370  ea98  d3
2370  ea99  d8
2370  ea9a  c3
```

```
zeile adr.  obj.-code   source-code

2371  ea9b  ed              .byte $ed, $23, $c5, $c4, $c6, $d6
2371  ea9c  23
2371  ea9d  c5
2371  ea9e  c4
2371  ea9f  c6
2371  eaa0  d6
2372  eaa1  ee              .byte $ee, $24, $d2, $d4, $c7, $c2
2372  eaa2  24
2372  eaa3  d2
2372  eaa4  d4
2372  eaa5  c7
2372  eaa6  c2
2373  eaa7  ef              .byte $ef, $25, $5e, $d9, $c8, $ce
2373  eaa8  25
2373  eaa9  5e
2373  eaaa  d9
2373  eaab  c8
2373  eaac  ce
2374  eaad  f0              .byte $f0, $26, $d5, $ca, $cd, $a0
2374  eaae  26
2374  eaaf  d5
2374  eab0  ca
2374  eab1  cd
2374  eab2  a0
2375  eab3  f1              .byte $f1, $2a, $c9, $cb, $3c, $3e
2375  eab4  2a
2375  eab5  c9
2375  eab6  cb
2375  eab7  3c
2375  eab8  3e
2376  eab9  f2              .byte $f2, $28, $cf, $cc, $3a, $3f
2376  eaba  28
2376  eabb  cf
2376  eabc  cc
2376  eabd  3a
2376  eabe  3f
2377  eabf  f3              .byte $f3, $29, $2d, $d0, $5b, $22
2377  eac0  29
2377  eac1  2d
2377  eac2  d0
2377  eac3  5b
2377  eac4  22
2378  eac5  11              .byte $11, $2b, $5c, $5d, $8d, $de
2378  eac6  2b
2378  eac7  5c
2378  eac8  5d
2378  eac9  8d
2378  eaca  de
2379  eacb  91              .byte $91, $9d, $1d, $94, $82, $ff
2379  eacc  9d
2379  eacd  1d
2379  eace  94
2379  eacf  82
2379  ead0  ff
2380  ead1  93              .byte $93, $3f, $37, $34, $31, $30
2380  ead2  3f
2380  ead3  37
2380  ead4  34
```

```
zeile adr.  obj.-code   source-code


2380  ead5  31
2380  ead6  30
2381  ead7  92              .byte $92, $84, $38, $35, $32, $2e
2381  ead8  84
2381  ead9  38
2381  eada  35
2381  eadb  32
2381  eadc  2e
2382  eadd  0e              .byte $0e, $2a, $39, $36, $33, $30
2382  eade  2a
2382  eadf  39
2382  eae0  36
2382  eae1  33
2382  eae2  30
2383  eae3  83              .byte $83, $2f, $2d, $2b, $8d, $ff
2383  eae4  2f
2383  eae5  2d
2383  eae6  2b
2383  eae7  8d
2383  eae8  ff
2384  eae9
2385  eae9
2386  eae9  ===> code-tab. für tastatur (graphik+shift) <===
2387  eae9
2388  eae9  ea        shftgr .byte $ea, $1b, $89, $ff, $00, $01
2388  eaea  1b
2388  eaeb  89
2388  eaec  ff
2388  eaed  00
2388  eaee  01
2389  eaef  eb              .byte $eb, $21, $d1, $c1, $da, $ff
2389  eaf0  21
2389  eaf1  d1
2389  eaf2  c1
2389  eaf3  da
2389  eaf4  ff
2390  eaf5  ec              .byte $ec, $40, $d7, $d3, $d8, $c0
2390  eaf6  40
2390  eaf7  d7
2390  eaf8  d3
2390  eaf9  d8
2390  eafa  c0
2391  eafb  ed              .byte $ed, $23, $c5, $c4, $c6, $c3
2391  eafc  23
2391  eafd  c5
2391  eafe  c4
2391  eaff  c6
2391  eb00  c3
2392  eb01  ee              .byte $ee, $24, $d2, $d4, $c7, $c2
2392  eb02  24
2392  eb03  d2
2392  eb04  d4
2392  eb05  c7
2392  eb06  c2
2393  eb07  ef              .byte $ef, $25, $5e, $d9, $c8, $dd
2393  eb08  25
2393  eb09  5e
2393  eb0a  d9
```

```
zeile adr.  obj.-code   source-code

2393  eb0b  c8
2393  eb0c  dd
2394  eb0d  f0              .byte $f0, $26, $d5, $ca, $cd, $a0
2394  eb0e  26
2394  eb0f  d5
2394  eb10  ca
2394  eb11  cd
2394  eb12  a0
2395  eb13  f1              .byte $f1, $2a, $c9, $cb, $3c, $3e
2395  eb14  2a
2395  eb15  c9
2395  eb16  cb
2395  eb17  3c
2395  eb18  3e
2396  eb19  f2              .byte $f2, $28, $cf, $d6, $3a, $3f
2396  eb1a  28
2396  eb1b  cf
2396  eb1c  d6
2396  eb1d  3a
2396  eb1e  3f
2397  eb1f  f3              .byte $f3, $29, $2d, $d0, $5b, $22
2397  eb20  29
2397  eb21  2d
2397  eb22  d0
2397  eb23  5b
2397  eb24  22
2398  eb25  11              .byte $11, $2b, $5c, $5d, $8d, $de
2398  eb26  2b
2398  eb27  5c
2398  eb28  5d
2398  eb29  8d
2398  eb2a  de
2399  eb2b  91              .byte $91, $9d, $1a, $94, $82, $ff
2399  eb2c  9d
2399  eb2d  1a
2399  eb2e  94
2399  eb2f  82
2399  eb30  ff
2400  eb31  93              .byte $93, $3f, $37, $34, $31, $30
2400  eb32  3f
2400  eb33  37
2400  eb34  34
2400  eb35  31
2400  eb36  30
2401  eb37  92              .byte $92, $04, $38, $35, $32, $2e
2401  eb38  04
2401  eb39  38
2401  eb3a  35
2401  eb3b  32
2401  eb3c  2e
2402  eb3d  0e              .byte $0e, $2a, $39, $36, $33, $30
2402  eb3e  2a
2402  eb3f  39
2402  eb40  36
2402  eb41  33
2402  eb42  30
2403  eb43  83              .byte $83, $2f, $2d, $2b, $8d, $ff
2403  eb44  2f
```

```
zeile adr.  obj.-code   source-code


2403  eb45  2d
2403  eb46  2b
2403  eb47  8d
2403  eb48  ff
2404  eb49
2405  eb49
2406  eb49  ===> code-tab. für tastatur (control-modus) <===
2407  eb49
2408  eb49  ff        ctltbl .byte $ff, $ff, $ff, $ff, $ff, $ff
2408  eb4a  ff
2408  eb4b  ff
2408  eb4c  ff
2408  eb4d  ff
2408  eb4e  ff
2409  eb4f  ff               .byte $ff, $a1, $11, $01, $1a, $ff
2409  eb50  a1
2409  eb51  11
2409  eb52  01
2409  eb53  1a
2409  eb54  ff
2410  eb55  ff               .byte $ff, $a2, $17, $13, $18, $03
2410  eb56  a2
2410  eb57  17
2410  eb58  13
2410  eb59  18
2410  eb5a  03
2411  eb5b  ff               .byte $ff, $a3, $05, $04, $06, $16
2411  eb5c  a3
2411  eb5d  05
2411  eb5e  04
2411  eb5f  06
2411  eb60  16
2412  eb61  ff               .byte $ff, $a4, $12, $14, $07, $02
2412  eb62  a4
2412  eb63  12
2412  eb64  14
2412  eb65  07
2412  eb66  02
2413  eb67  ff               .byte $ff, $a5, $a7, $19, $08, $0e
2413  eb68  a5
2413  eb69  a7
2413  eb6a  19
2413  eb6b  08
2413  eb6c  0e
2414  eb6d  ff               .byte $ff, $be, $15, $0a, $0d, $ff
2414  eb6e  be
2414  eb6f  15
2414  eb70  0a
2414  eb71  0d
2414  eb72  ff
2415  eb73  ff               .byte $ff, $bb, $09, $0b, $ce, $ff
2415  eb74  bb
2415  eb75  09
2415  eb76  0b
2415  eb77  ce
2415  eb78  ff
2416  eb79  ff               .byte $ff, $bf, $0f, $0c, $dc, $ff
2416  eb7a  bf
```

```
zeile adr.  obj.-code   source-code


2416  eb7b  0f
2416  eb7c  0c
2416  eb7d  dc
2416  eb7e  ff
2417  eb7f  ff              .byte $ff, $ac, $bc, $10, $cc, $a8
2417  eb80  ac
2417  eb81  bc
2417  eb82  10
2417  eb83  cc
2417  eb84  a8
2418  eb85  ff              .byte $ff, $a9, $df, $ba, $ff, $a6
2418  eb86  a9
2418  eb87  df
2418  eb88  ba
2418  eb89  ff
2418  eb8a  a6
2419  eb8b  ff              .byte $ff, $ff, $ff, $ff, $ff, $ff
2419  eb8c  ff
2419  eb8d  ff
2419  eb8e  ff
2419  eb8f  ff
2419  eb90  ff
2420  eb91  ff              .byte $ff, $b7, $b4, $b1, $b0, $ad
2420  eb92  b7
2420  eb93  b4
2420  eb94  b1
2420  eb95  b0
2420  eb96  ad
2421  eb97  ff              .byte $ff, $b8, $b5, $b2, $ae, $bd
2421  eb98  b8
2421  eb99  b5
2421  eb9a  b2
2421  eb9b  ae
2421  eb9c  bd
2422  eb9d  ff              .byte $ff, $b9, $b6, $b3, $db, $ff
2422  eb9e  b9
2422  eb9f  b6
2422  eba0  b3
2422  eba1  db
2422  eba2  ff
2423  eba3  ff              .byte $ff, $af, $aa, $ab, $ff
2423  eba4  af
2423  eba5  aa
2423  eba6  ab
2423  eba7  ff
2424  eba8  ff              .byte $ff
2425  eba9
2426  eba9
2427  eba9  ===> befehlstext 'dload"* -cr- run' <===
2428  eba9
2429  eba9  44        runtb .byte 'd', $cc
2429  ebaa  cc
2430  ebab  22 2a           .byte '"*', $0d
2430  ebad  0d
2431  ebae  52 55 4e        .byte 'run', $0d
2431  ebb1  0d
2432  ebb2
2433  ebb2
```

```
zeile adr.  obj.-code   source-code


2434  ebb2  ===> tab. anfangs-addrl bs-zeilen <===
2435  ebb2
2436  ebb2  00        ldtb2  .byte <linz0     bildschirm-adr. zeile  1
2437  ebb3  50               .byte <linz1     bildschirm-adr. zeile  2
2438  ebb4  a0               .byte <linz2     bildschirm-adr. zeile  3
2439  ebb5  f0               .byte <linz3     bildschirm-adr. zeile  4
2440  ebb6  40               .byte <linz4     bildschirm-adr. zeile  5
2441  ebb7  90               .byte <linz5     bildschirm-adr. zeile  6
2442  ebb8  e0               .byte <linz6     bildschirm-adr. zeile  7
2443  ebb9  30               .byte <linz7     bildschirm-adr. zeile  8
2444  ebba  80               .byte <linz8     bildschirm-adr. zeile  9
2445  ebbb  d0               .byte <linz9     bildschirm-adr. zeile 10
2446  ebbc  20               .byte <linz10    bildschirm-adr. zeile 11
2447  ebbd  70               .byte <linz11    bildschirm-adr. zeile 12
2448  ebbe  c0               .byte <linz12    bildschirm-adr. zeile 13
2449  ebbf  10               .byte <linz13    bildschirm-adr. zeile 14
2450  ebc0  60               .byte <linz14    bildschirm-adr. zeile 15
2451  ebc1  b0               .byte <linz15    bildschirm-adr. zeile 16
2452  ebc2  00               .byte <linz16    bildschirm-adr. zeile 17
2453  ebc3  50               .byte <linz17    bildschirm-adr. zeile 18
2454  ebc4  a0               .byte <linz18    bildschirm-adr. zeile 19
2455  ebc5  f0               .byte <linz19    bildschirm-adr. zeile 20
2456  ebc6  40               .byte <linz20    bildschirm-adr. zeile 21
2457  ebc7  90               .byte <linz21    bildschirm-adr. zeile 22
2458  ebc8  e0               .byte <linz22    bildschirm-adr. zeile 23
2459  ebc9  30               .byte <linz23    bildschirm-adr. zeile 24
2460  ebca  80               .byte <linz24    bildschirm-adr. zeile 25
2461  ebcb
2462  ebcb
2463  ebcb  ===> tab. anfangs-addrh der bs-zeilen <===
2464  ebcb
2465  ebcb  d0        ldtb1  .byte >linz0     bildschirm-adr. zeile  1
2466  ebcc  d0               .byte >linz1     bildschirm-adr. zeile  2
2467  ebcd  d0               .byte >linz2     bildschirm-adr. zeile  3
2468  ebce  d0               .byte >linz3     bildschirm-adr. zeile  4
2469  ebcf  d1               .byte >linz4     bildschirm-adr. zeile  5
2470  ebd0  d1               .byte >linz5     bildschirm-adr. zeile  6
2471  ebd1  d1               .byte >linz6     bildschirm-adr. zeile  7
2472  ebd2  d2               .byte >linz7     bildschirm-adr. zeile  8
2473  ebd3  d2               .byte >linz8     bildschirm-adr. zeile  9
2474  ebd4  d2               .byte >linz9     bildschirm-adr. zeile 10
2475  ebd5  d3               .byte >linz10    bildschirm-adr. zeile 11
2476  ebd6  d3               .byte >linz11    bildschirm-adr. zeile 12
2477  ebd7  d3               .byte >linz12    bildschirm-adr. zeile 13
2478  ebd8  d4               .byte >linz13    bildschirm-adr. zeile 14
2479  ebd9  d4               .byte >linz14    bildschirm-adr. zeile 15
2480  ebda  d4               .byte >linz15    bildschirm-adr. zeile 16
2481  ebdb  d5               .byte >linz16    bildschirm-adr. zeile 17
2482  ebdc  d5               .byte >linz17    bildschirm-adr. zeile 18
2483  ebdd  d5               .byte >linz18    bildschirm-adr. zeile 19
2484  ebde  d5               .byte >linz19    bildschirm-adr. zeile 20
2485  ebdf  d6               .byte >linz20    bildschirm-adr. zeile 21
2486  ebe0  d6               .byte >linz21    bildschirm-adr. zeile 22
2487  ebe1  d6               .byte >linz22    bildschirm-adr. zeile 23
2488  ebe2  d7               .byte >linz23    bildschirm-adr. zeile 24
2489  ebe3  d7               .byte >linz24    bildschirm-adr. zeile 25
2490  ebe4
2491  ebe4
```

```
zeile adr.  obj.-code   source-code


2492  ebe4  ===> adress-tab. der control-routinen <===
2493  ebe4
2494  ebe4  10 e3     ctable .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2495  ebe6  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2496  ebe8  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2497  ebea  1b e6            .word stprun-1   dload"*" 'cr' 'run' 'cr' ($83)
2498  ebec  b9 e4            .word ce-1       löscht letzte eingegebene zahl
2499  ebee  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2500  ebf0  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2501  ebf2  8c e4            .word bell-1     glocke anschlagen ($07 oder $87)
2502  ebf4  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2503  ebf6  59 e3            .word tabit-1    tab ($09) - setzen/löschen ($89)
2504  ebf8  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2505  ebfa  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2506  ebfc  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2507  ebfe  93 e3            .word nxt1-1     'carriage return' ($0d oder $8d)
2508  ec00  4c e2            .word ctext-1    umsch. text ($0e) - graphik ($8e)
2509  ec02  bb e9            .word window-1   bs-fenster links oben ($0f) / rechts
                                              unten ($8f)
2510  ec04  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2511  ec06  13 e3            .word cdnup-1    cursor unten ($11) - oben ($91)
2512  ec08  43 e3            .word rvsf-1     revers on ($12) - off ($92)
2513  ec0a  49 e3            .word homclr-1   cursor home ($13) - bs clear ($93)
2514  ec0c  ad e5 .          .word delins-1   zeichen löschen ($14), einfügen
                                              ($94)
2515  ec0e  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2516  ec10  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2517  ec12  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2518  ec14  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2519  ec16  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2520  ec18  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2521  ec1a  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2522  ec1c  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2523  ec1e  30 e3            .word crtlf-1    cursor rechts ($1d) - links ($9d)
2524  ec20  10 e3            .word cuser-1    user control-funktionen
                                              (ind.-vektor)
2525  ec22  10 e3            .word cuser-1
2526  ec24
2527  ec24
2528  ec24  ===> länge der funktionstasten-befehle <===
```

```
zeile adr.  obj.-code   source-code


2529  ec24
2530  ec24  05        keylen .byte $05, $04, $06, $06, $05, $06
2530  ec25  04
2530  ec26  06
2530  ec27  06
2530  ec28  05
2530  ec29  06
2531  ec2a  04               .byte $04, $09, $07
2531  ec2b  09
2531  ec2c  07
2532  ec2d  05               .byte $05        tab.1 funktionstastenbefehle -1
2533  ec2e
2534  ec2e
2535  ec2e  ===> tabelle der funktionstasten befehle (t1) <===
2536  ec2e
2537  ec2e  50 52 49  keydef .byte 'print'
2537  ec31  4e 54
2538  ec33  4c 49 53         .byte 'list'
2538  ec36  54
2539  ec37  44 4c 4f         .byte 'dload"'
2539  ec3a  41 44 22
2540  ec3d  44 53 41         .byte 'dsave"'
2540  ec40  56 45 22
2541  ec43  44 4f 50         .byte 'dopen'
2541  ec46  45 4e
2542  ec48  44 43 4c         .byte 'dclose'
2542  ec4b  4f 53 45
2543  ec4e  43 4f 50         .byte 'copy'
2543  ec51  59
2544  ec52  44 49 52         .byte 'directory'
2544  ec55  45 43 54
2544  ec58  4f 52 59
2545  ec5b  53 43 52         .byte 'scratch'
2545  ec5e  41 54 43
2545  ec61  48
2546  ec62  43 48 52         .byte 'chr$('
2546  ec65  24 28
2547  ec67
2548  ec67
2549  ec67  ===> bit-tabelle für zeilen-link <===
2550  ec67
2551  ec67  80        keyend .byte $80, $40, $20, $10, $08, $04
2551  ec68  40
2551  ec69  20
2551  ec6a  10
2551  ec6b  08
2551  ec6c  04
2552  ec6d  02               .byte $02, $01
2552  ec6e  01
2553  ec6f
2554  ec6f
2555  ec6f  ===> codetab. 1 für bs-crt 6845 <===
2556  ec6f
2557  ec6f  6c        atext  .byte $6c
2558  ec70  50               .byte $50        bildfenster horizontal: 80 zeichen
2559  ec71  53               .byte $53        horizontal synch pos.: 83
2560  ec72  0f               .byte $0f        hori/verti synch widths: 0/15
2561  ec73  19               .byte $19        bildschirm vertikal: 25 zeilen
```

```
zeile adr.  obj.-code   source-code

2562  ec74  03              .byte $03        feinjustierung vertikal: 3
2563  ec75  19              .byte $19        bs-fenster vertikal: 25 zeilen
2564  ec76  19              .byte $19        vertical synch pos.: 25
2565  ec77  00              .byte $00        steuerreg.: alle reg. binär
2566  ec78  0d              .byte $0d        abtastzeilen/zeichen: 13
2567  ec79  60              .byte $60        abtastzeile cursor-obergrenze: 0,
                                             blinken ($6x)
2568  ec7a  0d              .byte $0d        abtastzeile cursor-untergrenze: 13
2569  ec7b  00              .byte $00, $00   bildschirm start adr. im bs-ram
2569  ec7c  00
2570  ec7d  00              .byte $00, $00   cursor-pos. im bildschirmram
2570  ec7e  00
2571  ec7f  00              .byte $00, $00   lichtgriffel-adr. bildschirmram
2571  ec80  00
2572  ec81
2573  ec81
2574  ec81  ===> codetab. 2 für bs-crt 6845 <===
2575  ec81
2576  ec81  7e        ntext .byte $7e
2577  ec82  50              .byte $50        bildfenster horizontal: 80 zeichen
2578  ec83  62              .byte $62        horizontal synch pos.: 98
2579  ec84  0a              .byte $0a        hori/vert synch widths: 0/10
2580  ec85  1f              .byte $1f        bildschirm vertikal: 25 zeilen
2581  ec86  06              .byte $06        feinjustierung vertikal: 6
2582  ec87  19              .byte $19        bs-fenster vertikal: 25 zeilen
2583  ec88  1c              .byte $1c        vertical synch pos.: 28
2584  ec89  00              .byte $00        steuerreg.: alle reg. binär
2585  ec8a  07              .byte $07        abtastzeilen/zeichen: 7
2586  ec8b  00              .byte $00        abtastzeile cursor-obergrenze: 0,
                                             cursor fest
2587  ec8c  07              .byte $07        abtastzeile cursor-untergrenze: 7
2588  ec8d  00              .byte $00, $00   bildschirm start adr. im bs-ram
2588  ec8e  00
2589  ec8f  00              .byte $00, $00   cursor-pos. im bildschirmram
2589  ec90  00
2590  ec91  00              .byte $00, $00   lichtgriffel-adr. im bs-ram
2590  ec92  00
2591  ec93
2592  ec93
2593  ec93  ===> codetab. 3 für bs-crt 6845 <===
2594  ec93
2595  ec93  7f        ptext .byte $7f
2596  ec94  50              .byte $50        bs-fenster horizontal: 80 zeichen
2597  ec95  60              .byte $60        horizontal synch pos.: 96
2598  ec96  0a              .byte $0a        hori/verti synch widths: 0/10
2599  ec97  26              .byte $26        bildschirm vertikal: 38 zeilen
2600  ec98  01              .byte $01        feinjustierung vertikal: 1
2601  ec99  19              .byte $19        bs-fenster vertikal: 25 zeilen
2602  ec9a  1e              .byte $1e        vertical synch pos.: 30
2603  ec9b  00              .byte $00        steuerreg.: alle reg. binär
2604  ec9c  07              .byte $07        abtastzeilen/zeichen: 7
2605  ec9d  00              .byte $00        abtastzeile cursor-obergrenze: 0,
                                             cursor fest
2606  ec9e  07              .byte $07        abtastzeile cursor-untergrenze: 7
2607  ec9f  00              .byte $00, $00   bildschirm start adr. im bs-ram
2607  eca0  00
2608  eca1  00              .byte $00, $00   cursor-pos. im bildschirmram
2608  eca2  00
```

```
zeile adr.  obj.-code   source-code


2609  eca3  00              .byte $00, $00   lichtgriffel-adr. im bs-ram
2609  eca4  00
2610  eca5
2611  eca5
2612  eca5  ===> prüfbyte rom $e000-$ffff <===
2613  eca5
2614  eca5  65        cksume .byte $65
2615  eca6
2616  eca6
2617  eca6                 * = *+$5a
2618  ed00
2619  ed00
2620  ed00  ===> anzahl zeilen bs=1, dann pla, pla <===
2621  ed00
2622  ed00  a6 dd     bszei1 ldx scbot        unterste zeile bildschirm (0-25)
2623  ed02  e4 dc            cpx sctop        oberste zeile bildschirm (0-25)
2624  ed04  d0 02            bne bszeib
2625  ed06  68               pla
2626  ed07  68               pla
2627  ed08  2c 9b 03  bszeib bit scrdis       scroll disable flag
2628  ed0b  60               rts
2629  ed0c
2630  ed0c                   .end
2631  ed0c                   .lib monitor
```

```
zeile adr.  obj.-code   source-code


2633  ed0c
2634  ed0c                    * = *+$f4
2635  ee00
2636  ee00  ===> monitor-einsprung 'evect' <===
2637  ee00
2638  ee00  20 fb f9  monon   jsr ioinit      i/o register init.
2639  ee03  20 a2 fb          jsr restor      standard-vektoren für jsr und
                                              interrupts einrichten
2640  ee06  20 04 e0          jsr jcint
2641  ee09
2642  ee09
2643  ee09  ===> monitor-kaltstart (basic aus) <===
2644  ee09
2645  ee09  20 cc ff  monoff  jsr kclrch      ein-/ausgabekanal schliessen
2646  ee0c  a9 5a             lda #$5a
2647  ee0e  a2 00             ldx #$00
2648  ee10  a0 ee             ldy #$ee
2649  ee12  20 ca fb          jsr vreset
2650  ee15  58                cli
2651  ee16
2652  ee16
2653  ee16  ===> monitor-einsprung 'sys 60950' call <===
2654  ee16
2655  ee16  a9 c0     timc    lda #$c0
2656  ee18  8d 61 03          sta msgflg      message flag
2657  ee1b  a9 40             lda #$40
2658  ee1d  85 bd             sta tmpc        ablage letzter monitor-befehl
2659  ee1f  d0 10             bne b3
2660  ee21
2661  ee21
2662  ee21  ===> monitor-einsprung 'break-vektor' <===
2663  ee21
2664  ee21  20 cc ff  timb    jsr kclrch      ein-/ausgabekanal schliessen
2665  ee24  a9 53             lda #$53
2666  ee26  85 bd             sta tmpc        ablage letzter monitor-befehl
2667  ee28  d8                cld
2668  ee29  a2 05             ldx #$05
2669  ee2b  68        b1      pla
2670  ee2c  95 ae             sta pch,x       addrh programmzähler pc
2671  ee2e  ca                dex
2672  ee2f  10 fa             bpl b1
2673  ee31  a5 01     b3      lda i6509       6509 indirection register
2674  ee33  85 b5             sta xi6509      alte indirection bank
2675  ee35  ad 00 03          lda cinv        addrl irq vector
2676  ee38  85 b8             sta invl        addrl user interrupt-vektor
2677  ee3a  ad 01 03          lda cinv+1      addrh irq vector
2678  ee3d  85 b7             sta invh        addrh user interrupt-vektor
2679  ee3f  ba                tsx
2680  ee40  86 b4             stx sp          stack-pointer sp
2681  ee42  58                cli
2682  ee43  a9 08             lda #$08
2683  ee45  85 bf             sta ddisk       lfd. disk-primäradr. für monitor
2684  ee47  a4 bd             ldy tmpc        ablage letzter monitor-befehl
2685  ee49  20 1c f2          jsr spmsg       test errorflag, ausgabe
                                              systemmeldung
2686  ee4c  a9 52             lda #$52
2687  ee4e  d0 27             bne s0          monitor-befehl lesen (einsprung von
                                              usrcmd)
```

```
zeile adr.  obj.-code   source-code


2688  ee50
2689  ee50
2690  ee50  ===> mon-error, ausgabe '?','cr','.' <===
2691  ee50
2692  ee50  20 22 ef  erropr jsr outqst       ausgabe ascii 'fragezeichen'
2693  ee53  68                pla
2694  ee54  68                pla
2695  ee55  a9 c0             lda #$c0
2696  ee57  8d 61 03          sta msgflg       message flag
2697  ee5a  a9 00             lda #$00
2698  ee5c  85 90             sta fnadr        addrl akt. filename
2699  ee5e  a9 02             lda #$02
2700  ee60  85 91             sta fnadr+1      addrh akt. filename
2701  ee62  a9 0f             lda #$0f
2702  ee64  85 92             sta fnadr+2      bank akt. filename
2703  ee66  20 27 ef          jsr crlf         ausgabe mon-prompter 'cr' + '.'
2704  ee69
2705  ee69
2706  ee69  ===> sprung monitor-erweiterung <===
2707  ee69
2708  ee69  20 cf ff  st1     jsr kbasin       eingabe 1 char vom aktiven kanal in
                                               ac (chn-vector)
2709  ee6c  c9 2e             cmp #$2e
2710  ee6e  f0 f9             beq st1          sprung monitor-erweiterung
2711  ee70  c9 20             cmp #$20
2712  ee72  f0 f5             beq st1          sprung monitor-erweiterung
2713  ee74  6c 1e 03          jmp (usrcmd)     addrl monitor erweiterung-vector
2714  ee77
2715  ee77
2716  ee77  ===> monitor-befehl lesen (einsprung von usrcmd) <===
2717  ee77
2718  ee77  a2 00      s0     ldx #$00
2719  ee79  86 9d             stx fnlen        länge akt. filename
2720  ee7b  a8                tay
2721  ee7c  a9 ee             lda #$ee
2722  ee7e  48                pha
2723  ee7f  a9 54             lda #$54
2724  ee81  48                pha
2725  ee82  98                tya
2726  ee83
2727  ee83
2728  ee83  ===> mon-befehl mit tabelle vergleichen <===
2729  ee83
2730  ee83  dd d5 ee  s1      cmp cmds,x       tabelle monitor-befehle/adressen
2731  ee86  d0 10             bne s2           nächster mon-befehl
2732  ee88  8d 66 03          sta savx         kernalvariable temporär
2733  ee8b  bd d6 ee          lda cmds+1,x
2734  ee8e  85 b9             sta tmp1         addrl monitor zwei-byte zeiger 1
2735  ee90  bd d7 ee          lda cmds+1+1,x
2736  ee93  85 ba             sta tmp1+1       addrh monitor zwei-byte zeiger 1
2737  ee95  6c b9 00          jmp (tmp1)       addrl monitor zwei-byte zeiger 1
2738  ee98
2739  ee98
2740  ee98  ===> nächster mon-befehl <===
2741  ee98
2742  ee98  e8        s2      inx
2743  ee99  e8                inx
2744  ee9a  e8                inx
```

```
zeile adr.  obj.-code   source-code

2745  ee9b  e0 24           cpx #$24
2746  ee9d  90 e4           bcc s1          mon-befehl mit tabelle vergleichen
2747  ee9f  a2 00           ldx #$00
2748  eea1
2749  eea1
2750  eea1  ===> eingabe name mon-disk-befehl <===
2751  eea1
2752  eea1  c9 0d     s3    cmp #$0d
2753  eea3  f0 0d           beq s4          kein mon-befehl, befehl von disk
2754  eea5  c9 20           cmp #$20
2755  eea7  f0 09           beq s4          kein mon-befehl, befehl von disk
2756  eea9  9d 00 02        sta buf,x       basic input-puffer -$2ff
2757  eeac  20 cf ff        jsr kbasin      eingabe 1 char vom aktiven kanal in
                                            ac (chn-vector)
2758  eeaf  e8              inx
2759  eeb0  d0 ef           bne s3          eingabe name mon-disk-befehl
2760  eeb2
2761  eeb2
2762  eeb2  ===> kein mon-befehl, befehl von disk <===
2763  eeb2
2764  eeb2  85 bd     s4    sta tmpc        ablage letzter monitor-befehl
2765  eeb4  8a              txa
2766  eeb5  f0 1d           beq s6
2767  eeb7  85 9d           sta fnlen       länge akt. filename
2768  eeb9  a9 40           lda #$40
2769  eebb  8d 61 03        sta msgflg      message flag
2770  eebe  a5 bf           lda ddisk       lfd. disk-primäradr. für monitor
2771  eec0  85 9f           sta fa          akt. primäradresse (geräte-nummer)
2772  eec2  a9 0f           lda #$0f
2773  eec4  85 01           sta i6509       6509 indirection register
2774  eec6  a2 ff           ldx #$ff
2775  eec8  a0 ff           ldy #$ff
2776  eeca  20 d5 ff        jsr kload       einlesen vom log. file
2777  eecd  b0 05           bcs s6
2778  eecf  a5 bd           lda tmpc        ablage letzter monitor-befehl
2779  eed1  6c 99 00        jmp (stal)      addrl anfang akt. load/store
2780  eed4
2781  eed4
2782  eed4  60        s6    rts
2783  eed5
2784  eed5
2785  eed5  ===> tabelle monitor-befehle/adressen <===
2786  eed5
2787  eed5  3a        cmds  .byte ':'
2788  eed6  f5 ef           .word altm
2789  eed8  3b              .byte ';'
2790  eed9  cb ef           .word altr      mon-befehl ';' = registerzeile ins
                                            ram
2791  eedb  52              .byte 'r'
2792  eedc  4c ef           .word dsplyr    mon-befehl 'r' = registerausgabe
2793  eede  4d              .byte 'm'
2794  eedf  8f ef           .word dsplym    mon-befehl 'm' = druckt memoryzeile
                                            (16byt)
2795  eee1  47              .byte 'g'
2796  eee2  10 f0           .word go        mon-befehl 'g' goto xxxxxx
2797  eee4  4c              .byte 'l'
2798  eee5  4a f0           .word ld        mon-befehl 'l'=load, 's'=save
2799  eee7  53              .byte 's'
```

```
zeile adr.  obj.-code   source-code

2800  eee8  4a f0           .word ld         mon-befehl 'l'=load, 's'=save
2801  eeea  56              .byte 'v'
2802  eeeb  e1 ef           .word view       mon-befehl 'v' = bankswitch
2803  eeed  40              .byte '@'
2804  eeee  65 f1           .word disk       mon-befehl '.' = disk-kurzbefehle
2805  eef0  5a              .byte 'z'
2806  eef1  72 ff           .word kipcgo     mon-befehl 'z' umsch. auf
                                             coprozessor
2807  eef3  58              .byte 'x'
2808  eef4  f9 ee           .word xeit       mon-befehl 'x'=ende
2809  eef6  55              .byte 'u'
2810  eef7  eb ef           .word unitd      mon-befehl 'u' = ändern diskadresse
2811  eef9
2812  eef9
2813  eef9  ===> mon-befehl 'x'=ende <===
2814  eef9
2815  eef9  68        xeit  pla
2816  eefa  68              pla
2817  eefb  78              sei
2818  eefc  6c f8 03        jmp (evect)      addrl warmstart-vector
2819  eeff
2820  eeff
2821  eeff  ===> adr. von tmp1 nach pcl <===
2822  eeff
2823  eeff  a5 b9     putp  lda tmp1         addrl monitor zwei-byte zeiger 1
2824  ef01  85 af           sta pcl          addrl programmzähler pc
2825  ef03  a5 ba           lda tmp1+1       addrh monitor zwei-byte zeiger 1
2826  ef05  85 ae           sta pch          addrh programmzähler pc
2827  ef07  60              rts
2828  ef08
2829  ef08
2830  ef08  ===> sr, ac setzen, bank= $f <===
2831  ef08
2832  ef08  a9 b0     setr  lda #$b0
2833  ef0a  85 b9           sta tmp1         addrl monitor zwei-byte zeiger 1
2834  ef0c  a9 00           lda #$00
2835  ef0e  85 ba           sta tmp1+1       addrh monitor zwei-byte zeiger 1
2836  ef10  a9 0f           lda #$0f
2837  ef12  85 01           sta i6509        6509 indirection register
2838  ef14  a9 05           lda #$05
2839  ef16  60              rts
2840  ef17
2841  ef17
2842  ef17  ===> ausgabe 'cr' + '.', ac + space <===
2843  ef17
2844  ef17  48        altrit pha
2845  ef18  20 27 ef        jsr crlf         ausgabe mon-prompter 'cr' + '.'
2846  ef1b  68              pla
2847  ef1c  20 d2 ff        jsr kbsout       ausgabe 1 char auf akt. kanal
2848  ef1f
2849  ef1f
2850  ef1f  ===> ausgabe ascii 'space' <===
2851  ef1f
2852  ef1f  a9 20     space lda #$20
2853  ef21  2c              .byte $2c
2854  ef22
2855  ef22
2856  ef22  ===> ausgabe ascii 'fragezeichen' <===
```

```
zeile adr.  obj.-code   source-code


2857  ef22
2858  ef22  a9 3f     outqst lda #$3f
2859  ef24  4c d2 ff         jmp kbsout       ausgabe 1 char auf akt. kanal
2860  ef27
2861  ef27
2862  ef27  ===> ausgabe mon-prompter 'cr' + '.' <===
2863  ef27
2864  ef27  a9 0d     crlf    lda #$0d
2865  ef29  20 d2 ff          jsr kbsout       ausgabe 1 char auf akt. kanal
2866  ef2c  a9 2e             lda #$2e
2867  ef2e  4c d2 ff          jmp kbsout       ausgabe 1 char auf akt. kanal
2868  ef31
2869  ef31
2870  ef31  ===> kopf für cpu-register ausgabe <===
2871  ef31
2872  ef31  0d        regk    .byte $0d
2873  ef32  20 20 20          .byte '   pc  irq  sr ac xr yr sp'
2873  ef35  50 43 20
2873  ef38  20 49 52
2873  ef3b  51 20 20
2873  ef3e  53 52 20
2873  ef41  41 43 20
2873  ef44  58 52 20
2873  ef47  59 52 20
2873  ef4a  53 50
2874  ef4c
2875  ef4c
2876  ef4c  ===> mon-befehl 'r' = registerausgabe <===
2877  ef4c
2878  ef4c  a2 00     dsplyr ldx #$00
2879  ef4e
2880  ef4e
2881  ef4e  ===> cpu-register ab xr ausgeben <===
2882  ef4e
2883  ef4e  bd 31 ef  d2      lda regk,x       kopf für cpu-register ausgabe
2884  ef51  20 d2 ff          jsr kbsout       ausgabe 1 char auf akt. kanal
2885  ef54  e8                inx
2886  ef55  e0 1b             cpx #$1b
2887  ef57  d0 f5             bne d2           cpu-register ab xr ausgeben
2888  ef59  a9 3b             lda #$3b
2889  ef5b  20 17 ef          jsr altrit       ausgabe 'cr' + '.', ac + space
2890  ef5e  a6 ae             ldx pch          addrh programmzähler pc
2891  ef60  a4 af             ldy pcl          addrl programmzähler pc
2892  ef62  20 f6 f0          jsr wroa         ausgabe xr/yr als vier ascii-bytes
2893  ef65  20 1f ef          jsr space        ausgabe ascii 'space'
2894  ef68  a6 b7             ldx invh         addrh user interrupt-vektor
2895  ef6a  a4 b8             ldy invl         addrl user interrupt-vektor
2896  ef6c  20 f6 f0          jsr wroa         ausgabe xr/yr als vier ascii-bytes
2897  ef6f  20 08 ef          jsr setr         sr, ac setzen, bank= $f
2898  ef72
2899  ef72
2900  ef72  ===> druckt spc, hexdump ab $69, länge in $bd <===
2901  ef72
2902  ef72  85 bd     dm      sta tmpc         ablage letzter monitor-befehl
2903  ef74  a0 00             ldy #$00
2904  ef76  84 9d             sty fnlen        länge akt. filename
2905  ef78
2906  ef78
```

```
zeile adr.  obj.-code   source-code

2907  ef78  ===> druckt spc, hexdump ab $69 + yr <===
2908  ef78
2909  ef78  20 1f ef  dm1    jsr space        ausgabe ascii 'space'
2910  ef7b  b1 b9            lda (tmp1),y     addrl monitor zwei-byte zeiger 1
2911  ef7d  20 fb f0         jsr wrob         ausgabe ac als ascii-doppelbyte
2912  ef80  e6 b9            inc tmp1         addrl monitor zwei-byte zeiger 1
2913  ef82  d0 06            bne dm2
2914  ef84  e6 ba            inc tmp1+1       addrh monitor zwei-byte zeiger 1
2915  ef86  d0 02            bne dm2
2916  ef88  c6 9d            dec fnlen        länge akt. filename
2917  ef8a  c6 bd     dm2    dec tmpc         ablage letzter monitor-befehl
2918  ef8c  d0 ea            bne dm1          druckt spc, hexdump ab $69 + yr
2919  ef8e  60               rts
2920  ef8f
2921  ef8f
2922  ef8f  ===> mon-befehl 'm' = druckt memoryzeile (16byt) <===
2923  ef8f
2924  ef8f  20 40 f0  dsplym jsr rdoae        4 zeichen nach hex in tmp1,
                                              bcs=error
2925  ef92  20 13 f1         jsr t2t2         tmp1 mit tmp2 vertauschen
2926  ef95  20 23 f1         jsr rdoa         4 zeichen nach hex in tmp1
2927  ef98  90 08            bcc dsp123
2928  ef9a  a5 bb            lda tmp2         addrl monitor zwei-byte zeiger 2
2929  ef9c  85 b9            sta tmp1         addrl monitor zwei-byte zeiger 1
2930  ef9e  a5 bc            lda tmp2+1       addrh monitor zwei-byte zeiger 2
2931  efa0  85 ba            sta tmp1+1       addrh monitor zwei-byte zeiger 1
2932  efa2  20 13 f1  dsp123 jsr t2t2         tmp1 mit tmp2 vertauschen
2933  efa5  20 e1 ff  dsp1   jsr kstop        stop-taste lesen
2934  efa8  f0 20            beq begs1
2935  efaa  a9 3a            lda #$3a
2936  efac  20 17 ef         jsr altrit       ausgabe 'cr' + '.', ac + space
2937  efaf  a6 ba            ldx tmp1+1       addrh monitor zwei-byte zeiger 1
2938  efb1  a4 b9            ldy tmp1         addrl monitor zwei-byte zeiger 1
2939  efb3  20 f6 f0         jsr wroa         ausgabe xr/yr als vier ascii-bytes
2940  efb6  a9 10            lda #$10
2941  efb8  20 72 ef         jsr dm           druckt spc, hexdump ab $69, länge in
                                              $bd
2942  efbb  a5 9d            lda fnlen        länge akt. filename
2943  efbd  d0 0b            bne begs1
2944  efbf  38               sec
2945  efc0  a5 bb            lda tmp2         addrl monitor zwei-byte zeiger 2
2946  efc2  e5 b9            sbc tmp1         addrl monitor zwei-byte zeiger 1
2947  efc4  a5 bc            lda tmp2+1       addrh monitor zwei-byte zeiger 2
2948  efc6  e5 ba            sbc tmp1+1       addrh monitor zwei-byte zeiger 1
2949  efc8  b0 db            bcs dsp1
2950  efca  60        begs1  rts
2951  efcb
2952  efcb
2953  efcb  ===> mon-befehl ';' = registerzeile ins ram <===
2954  efcb
2955  efcb  20 40 f0  altr   jsr rdoae        4 zeichen nach hex in tmp1,
                                              bcs=error
2956  efce  20 ff ee         jsr putp         adr. von tmp1 nach pcl
2957  efd1  20 40 f0         jsr rdoae        4 zeichen nach hex in tmp1,
                                              bcs=error
2958  efd4  a5 b9            lda tmp1         addrl monitor zwei-byte zeiger 1
2959  efd6  85 b8            sta invl         addrl user interrupt-vektor
2960  efd8  a5 ba            lda tmp1+1       addrh monitor zwei-byte zeiger 1
```

```
zeile adr.  obj.-code   source-code

2961  efda  85 b7           sta invh         addrh user interrupt-vektor
2962  efdc  20 08 ef        jsr setr         sr, ac setzen, bank= $f
2963  efdf  d0 19           bne a4
2964  efe1
2965  efe1
2966  efe1  ===> mon-befehl 'v' = bankswitch <===
2967  efe1
2968  efe1  20 3a f0  view   jsr rdobe        2 zeichen (1byt) nach hex in ac,
                                              bcs=error
2969  efe4  c9 10           cmp #$10
2970  efe6  b0 5f           bcs errl         mon-error, ausgabe '?','cr','.'
2971  efe8  85 01           sta i6509        6509 indirection register
2972  efea  60              rts
2973  efeb
2974  efeb
2975  efeb  ===> mon-befehl 'u' = ändern diskadresse <===
2976  efeb
2977  efeb  20 3a f0  unitd  jsr rdobe        2 zeichen (1byt) nach hex in ac,
                                              bcs=error
2978  efee  c9 20           cmp #$20
2979  eff0  b0 55           bcs errl         mon-error, ausgabe '?','cr','.'
2980  eff2  85 bf           sta ddisk        lfd. disk-primäradr. für monitor
2981  eff4  60              rts
2982  eff5
2983  eff5
2984  eff5  ===> mon-befehl ':' = memoryzeile ins ram (16 byt) <===
2985  eff5
2986  eff5  20 40 f0  altm   jsr rdoae        4 zeichen nach hex in tmp1,
                                              bcs=error
2987  eff8  a9 10           lda #$10
2988  effa  85 bd     a4    sta tmpc         ablage letzter monitor-befehl
2989  effc  20 30 f1  a5    jsr rdob         2 zeichen (1byt) nach hex in ac
2990  efff  b0 0e           bcs a9
2991  f001  a0 00           ldy #$00
2992  f003  91 b9           sta (tmp1),y     addrl monitor zwei-byte zeiger 1
2993  f005  e6 b9           inc tmp1         addrl monitor zwei-byte zeiger 1
2994  f007  d0 02           bne a6
2995  f009  e6 ba           inc tmp1+1       addrh monitor zwei-byte zeiger 1
2996  f00b  c6 bd     a6    dec tmpc         ablage letzter monitor-befehl
2997  f00d  d0 ed           bne a5
2998  f00f  60        a9    rts
2999  f010
3000  f010
3001  f010  ===> mon-befehl 'g' goto xxxxxx <===
3002  f010
3003  f010  20 5f f1  go     jsr rdoc         eingabe 1 char.in ac, test cr
3004  f013  f0 06           beq g1
3005  f015  20 40 f0        jsr rdoae        4 zeichen nach hex in tmp1,
                                              bcs=error
3006  f018  20 ff ee        jsr putp         adr. von tmp1 nach pcl
3007  f01b  a6 b4     g1    ldx sp           stack-pointer sp
3008  f01d  9a              txs
3009  f01e  78              sei
3010  f01f  a5 b7           lda invh         addrh user interrupt-vektor
3011  f021  8d 01 03        sta cinv+1       addrh irq vector
3012  f024  a5 b8           lda invl         addrl user interrupt-vektor
3013  f026  8d 00 03        sta cinv         addrl irq vector
3014  f029  a5 b5           lda xi6509       alte indirection bank
```

```
zeile adr.  obj.-code   source-code

3015  f02b  85 01           sta i6509        6509 indirection register
3016  f02d  a2 00           ldx #$00
3017  f02f  b5 ae     g2    lda pch,x        addrh programmzähler pc
3018  f031  48              pha
3019  f032  e8              inx
3020  f033  e0 06           cpx #$06
3021  f035  d0 f8           bne g2
3022  f037  4c a5 fc        jmp prend        irq-ende, stack nach yr,xr,ac
3023  f03a
3024  f03a
3025  f03a  ===> 2 zeichen (1byt) nach hex in ac, bcs=error <===
3026  f03a
3027  f03a  20 30 f1  rdobe  jsr rdob         2 zeichen (1byt) nach hex in ac
3028  f03d  b0 06           bcs errlpl
3029  f03f  60        rdoxit rts
3030  f040
3031  f040
3032  f040  ===> 4 zeichen nach hex in tmp1, bcs=error <===
3033  f040
3034  f040  20 23 f1  rdoae  jsr rdoa         4 zeichen nach hex in tmp1
3035  f043  90 fa           bcc rdoxit
3036  f045  68        errlpl pla
3037  f046  68              pla
3038  f047  4c 50 ee  errl   jmp erropr       mon-error, ausgabe '?','cr','.'
3039  f04a
3040  f04a
3041  f04a  ===> mon-befehl 'l'=load, 's'=save <===
3042  f04a
3043  f04a  a0 01     ld     ldy #$01
3044  f04c  84 9f            sty fa           akt. primäradresse (geräte-nummer)
3045  f04e  88               dey
3046  f04f  a9 ff            lda #$ff
3047  f051  85 b9            sta tmp1         addrl monitor zwei-byte zeiger 1
3048  f053  85 ba            sta tmp1+1       addrh monitor zwei-byte zeiger 1
3049  f055  a5 01            lda i6509        6509 indirection register
3050  f057  85 be            sta t6509        laufende 6509 indirection bank
3051  f059  a9 0f            lda #$0f
3052  f05b  85 01            sta i6509        6509 indirection register
3053  f05d  20 5f f1  l1     jsr rdoc         eingabe 1 char.in ac, test cr
3054  f060  f0 1c            beq l5
3055  f062  c9 20            cmp #$20
3056  f064  f0 f7            beq l1
3057  f066  c9 22            cmp #$22
3058  f068  d0 dd     l2     bne errl         mon-error, ausgabe '?','cr','.'
3059  f06a  20 5f f1  l3     jsr rdoc         eingabe 1 char.in ac, test cr
3060  f06d  f0 0f            beq l5
3061  f06f  c9 22            cmp #$22
3062  f071  f0 1d            beq l8
3063  f073  91 90            sta (fnadr),y    addrl akt. filename
3064  f075  e6 9d            inc fnlen        länge akt. filename
3065  f077  c8               iny
3066  f078  c0 10            cpy #$10
3067  f07a  f0 cb            beq errl         mon-error, ausgabe '?','cr','.'
3068  f07c  d0 ec            bne l3
3069  f07e  ad 66 03  l5     lda savx         kernalvariable temporär
3070  f081  c9 4c            cmp #$4c
3071  f083  d0 e3            bne l2
3072  f085  a5 be            lda t6509        laufende 6509 indirection bank
```

```
zeile adr.  obj.-code   source-code

3073  f087  29 0f           and #$0f
3074  f089  a6 b9           ldx tmp1        addrl monitor zwei-byte zeiger 1
3075  f08b  a4 ba           ldy tmp1+1      addrh monitor zwei-byte zeiger 1
3076  f08d  4c d5 ff        jmp kload       einlesen vom log. file
3077  f090
3078  f090
3079  f090  20 5f f1  18    jsr rdoc        eingabe 1 char.in ac, test cr
3080  f093  f0 e9           beq 15
3081  f095  c9 2c           cmp #$2c
3082  f097  d0 cf     19    bne 12
3083  f099  20 3a f0        jsr rdobe       2 zeichen (1byt) nach hex in ac,
                                            bcs=error
3084  f09c  85 9f           sta fa          akt. primäradresse (geräte-nummer)
3085  f09e  20 5f f1        jsr rdoc        eingabe 1 char.in ac, test cr
3086  f0a1  f0 db           beq 15
3087  f0a3  c9 2c           cmp #$2c
3088  f0a5  d0 f0     112   bne 19
3089  f0a7  20 3a f0        jsr rdobe       2 zeichen (1byt) nach hex in ac,
                                            bcs=error
3090  f0aa  c9 10           cmp #$10
3091  f0ac  b0 45           bcs 115         mon-error, ausgabe '?','cr','.'
3092  f0ae  85 be           sta t6509       laufende 6509 indirection bank
3093  f0b0  85 9b           sta stas        bank anfang akt. load/store
3094  f0b2  20 40 f0        jsr rdoae       4 zeichen nach hex in tmp1,
                                            bcs=error
3095  f0b5  a5 b9           lda tmp1        addrl monitor zwei-byte zeiger 1
3096  f0b7  85 99           sta stal        addrl anfang akt. load/store
3097  f0b9  a5 ba           lda tmp1+1      addrh monitor zwei-byte zeiger 1
3098  f0bb  85 9a           sta stah        addrh anfang akt. load/store
3099  f0bd  20 5f f1        jsr rdoc        eingabe 1 char.in ac, test cr
3100  f0c0  f0 bc           beq 15
3101  f0c2  c9 2c           cmp #$2c
3102  f0c4  d0 2d           bne 115         mon-error, ausgabe '?','cr','.'
3103  f0c6  20 3a f0        jsr rdobe       2 zeichen (1byt) nach hex in ac,
                                            bcs=error
3104  f0c9  c9 10           cmp #$10
3105  f0cb  b0 26           bcs 115         mon-error, ausgabe '?','cr','.'
3106  f0cd  85 98           sta eas         bank ende akt. load/store
3107  f0cf  20 40 f0        jsr rdoae       4 zeichen nach hex in tmp1,
                                            bcs=error
3108  f0d2  a5 b9           lda tmp1        addrl monitor zwei-byte zeiger 1
3109  f0d4  85 96           sta eal         addrl ende akt. load/store
3110  f0d6  a5 ba           lda tmp1+1      addrh monitor zwei-byte zeiger 1
3111  f0d8  85 97           sta eah         addrh ende akt. load/store
3112  f0da  20 cf ff  120   jsr kbasin      eingabe 1 char vom aktiven kanal in
                                            ac (chn-vector)
3113  f0dd  c9 20           cmp #$20
3114  f0df  f0 f9           beq 120
3115  f0e1  c9 0d           cmp #$0d
3116  f0e3  d0 c0     114   bne 112
3117  f0e5  ad 66 03        lda savx        kernalvariable temporär
3118  f0e8  c9 53           cmp #$53
3119  f0ea  d0 f7           bne 114
3120  f0ec  a2 99           ldx #$99
3121  f0ee  a0 96           ldy #$96
3122  f0f0  4c d8 ff        jmp ksave       abspeichern auf log. file
3123  f0f3
3124  f0f3
```

```
zeile adr.  obj.-code   source-code

3125  f0f3  4c 50 ee  l15    jmp erropr       mon-error, ausgabe '?','cr','.'
3126  f0f6
3127  f0f6
3128  f0f6  ===> ausgabe xr/yr als vier ascii-bytes <===
3129  f0f6
3130  f0f6  8a         wroa   txa
3131  f0f7  20 fb f0           jsr wrob         ausgabe ac als ascii-doppelbyte
3132  f0fa  98                 tya
3133  f0fb
3134  f0fb
3135  f0fb  ===> ausgabe ac als ascii-doppelbyte <===
3136  f0fb
3137  f0fb  48         wrob   pha
3138  f0fc  4a                lsr a
3139  f0fd  4a                lsr a
3140  f0fe  4a                lsr a
3141  f0ff  4a                lsr a
3142  f100  20 07 f1          jsr ascii        umwandlung 1 hexbyte nach ascii
3143  f103  aa                tax
3144  f104  68                pla
3145  f105  29 0f             and #$0f
3146  f107
3147  f107
3148  f107  ===> umwandlung 1 hexbyte nach ascii <===
3149  f107
3150  f107  18         ascii  clc
3151  f108  69 f6             adc #$f6
3152  f10a  90 02             bcc asc1
3153  f10c  69 06             adc #$06
3154  f10e  69 3a      asc1   adc #$3a
3155  f110  4c d2 ff          jmp kbsout       ausgabe 1 char auf akt. kanal
3156  f113
3157  f113
3158  f113  ===> tmp1 mit tmp2 vertauschen <===
3159  f113
3160  f113  a2 02      t2t2   ldx #$02
3161  f115  b5 b8      t2t21  lda invl,x       addrl user interrupt-vektor
3162  f117  48                pha
3163  f118  b5 ba             lda tmp1+1,x     addrh monitor zwei-byte zeiger 1
3164  f11a  95 b8             sta invl,x       addrl user interrupt-vektor
3165  f11c  68                pla
3166  f11d  95 ba             sta tmp1+1,x     addrh monitor zwei-byte zeiger 1
3167  f11f  ca                dex
3168  f120  d0 f3             bne t2t21
3169  f122  60                rts
3170  f123
3171  f123
3172  f123  ===> 4 zeichen nach hex in tmp1 <===
3173  f123
3174  f123  20 30 f1  rdoa   jsr rdob          2 zeichen (1byt) nach hex in ac
3175  f126  b0 07             bcs rdoa2
3176  f128  85 ba             sta tmp1+1       addrh monitor zwei-byte zeiger 1
3177  f12a  20 30 f1          jsr rdob         2 zeichen (1byt) nach hex in ac
3178  f12d  85 b9             sta tmp1         addrl monitor zwei-byte zeiger 1
3179  f12f  60         rdoa2  rts
3180  f130
3181  f130
3182  f130  ===> 2 zeichen (1byt) nach hex in ac <===
```

```
zeile adr.  obj.-code   source-code

3183  f130
3184  f130  a9 00     rdob   lda #$00
3185  f132  8d 00 01         sta stack        6509 cpu-stack
3186  f135  20 5f f1         jsr rdoc         eingabe 1 char.in ac, test cr
3187  f138  f0 19            beq rdob4
3188  f13a  c9 20            cmp #$20
3189  f13c  f0 f2            beq rdob         2 zeichen (1byt) nach hex in ac
3190  f13e  20 54 f1         jsr hexit        ac nach hex umwandeln
3191  f141  0a               asl a
3192  f142  0a               asl a
3193  f143  0a               asl a
3194  f144  0a               asl a
3195  f145  8d 00 01         sta stack        6509 cpu-stack
3196  f148  20 5f f1         jsr rdoc         eingabe 1 char.in ac, test cr
3197  f14b  f0 06            beq rdob4
3198  f14d  20 54 f1         jsr hexit        ac nach hex umwandeln
3199  f150  0d 00 01         ora stack        6509 cpu-stack
3200  f153  60        rdob4  rts
3201  f154
3202  f154
3203  f154  ===> ac nach hex umwandeln <===
3204  f154
3205  f154  c9 3a     hexit  cmp #$3a
3206  f156  08               php
3207  f157  29 0f            and #$0f
3208  f159  28               plp
3209  f15a  90 02            bcc hex09
3210  f15c  69 08            adc #$08
3211  f15e  60        hex09  rts
3212  f15f
3213  f15f
3214  f15f  ===> eingabe 1 char.in ac, test cr <===
3215  f15f
3216  f15f  20 cf ff  rdoc   jsr kbasin       eingabe 1 char vom aktiven kanal in
                                               ac (chn-vector)
3217  f162  c9 0d            cmp #$0d
3218  f164  60               rts
3219  f165
3220  f165
3221  f165  ===> mon-befehl ' '= disk-kurzbefehle <===
3222  f165
3223  f165  a9 00     disk   lda #$00
3224  f167  85 9c            sta status       i/o operation status
3225  f169  85 9d            sta fnlen        länge akt. filename
3226  f16b  a6 bf            ldx ddisk        lfd. disk-primäradr. für monitor
3227  f16d  a0 0f            ldy #$0f
3228  f16f  20 43 fb         jsr setlfs       log., geräte- und sekundäradr.
                                               aktualisieren
3229  f172  18               clc
3230  f173  20 c0 ff         jsr kopen        log. file öffnen/befehl ausgeben
3231  f176  b0 42            bcs disk30
3232  f178  20 5f f1         jsr rdoc         eingabe 1 char.in ac, test cr
3233  f17b  f0 1d            beq disk20
3234  f17d  48               pha
3235  f17e  a2 00            ldx #$00
3236  f180  20 c9 ff         jsr kckout       ausgabekanal öffnen
3237  f183  68               pla
3238  f184  b0 34            bcs disk30
```

```
zeile adr.  obj.-code   source-code

3239  f186  90 03           bcc disk15
3240  f188  20 cf ff  disk10 jsr kbasin       eingabe 1 char vom aktiven kanal in
                                               ac (chn-vector)
3241  f18b  c9 0d      disk15 cmp #$0d
3242  f18d  08               php
3243  f18e  20 d2 ff         jsr kbsout       ausgabe 1 char auf akt. kanal
3244  f191  a5 9c             lda status       i/o operation status
3245  f193  d0 21             bne disk28
3246  f195  28               plp
3247  f196  d0 f0             bne disk10
3248  f198  f0 20             beq disk30
3249  f19a  20 27 ef  disk20 jsr crlf          ausgabe mon-prompter 'cr' + '.'
3250  f19d  a2 00             ldx #$00
3251  f19f  20 c6 ff          jsr kchkin       eingabekanal öffnen
3252  f1a2  b0 16             bcs disk30
3253  f1a4  20 5f f1  disk25 jsr rdoc          eingabe 1 char.in ac, test cr
3254  f1a7  08               php
3255  f1a8  20 d2 ff          jsr kbsout       ausgabe 1 char auf akt. kanal
3256  f1ab  a5 9c             lda status       i/o operation status
3257  f1ad  29 bf             and #$bf
3258  f1af  d0 05             bne disk28
3259  f1b1  28               plp
3260  f1b2  d0 f0             bne disk25
3261  f1b4  f0 04             beq disk30
3262  f1b6  68        disk28 pla
3263  f1b7  20 45 f9          jsr error5       ausgabe 'i/o error: 5'
3264  f1ba  20 cc ff  disk30 jsr kclrch        ein-/ausgabekanal schliessen
3265  f1bd  a9 00             lda #$00
3266  f1bf  18               clc
3267  f1c0  4c c3 ff          jmp kclose       logisches file schliessen
3268  f1c3
3269  f1c3                   .end
3270  f1c3                   .lib kernal1
```

```
zeile adr.  obj.-code   source-code


3272  f1c3
3273  f1c3  ===> tabelle der betriebssystem-meldungen <===
3274  f1c3
3275  f1c3  0d        ms1     .byte $0d
3276  f1c4  49 2f 4f          .byte 'i/o error ', $a3, $0d
3276  f1c7  20 45 52
3276  f1ca  52 4f 52
3276  f1cd  20
3276  f1ce  a3
3276  f1cf  0d
3277  f1d0  53 45 41          .byte 'searching', $a0
3277  f1d3  52 43 48
3277  f1d6  49 4e 47
3277  f1d9  a0
3278  f1da  46 4f 52          .byte 'for', $a0, $0d
3278  f1dd  a0
3278  f1de  0d
3279  f1df  4c 4f 41          .byte 'loadin', $c7, $0d
3279  f1e2  44 49 4e
3279  f1e5  c7
3279  f1e6  0d
3280  f1e7  53 41 56          .byte 'saving', $a0, $0d
3280  f1ea  49 4e 47
3280  f1ed  a0
3280  f1ee  0d
3281  f1ef  56 45 52          .byte 'verifyin', $c7, $0d
3281  f1f2  49 46 59
3281  f1f5  49 4e
3281  f1f7  c7
3281  f1f8  0d
3282  f1f9  46 4f 55          .byte 'found', $a0, $0d
3282  f1fc  4e 44
3282  f1fe  a0
3282  f1ff  0d
3283  f200  4f 4b             .byte 'ok', $8d, $0d
3283  f202  8d
3283  f203  0d
3284  f204  2a 2a 20          .byte '** monitor 1.0 **', $8d, $0d
3284  f207  4d 4f 4e
3284  f20a  49 54 4f
3284  f20d  52 20 31
3284  f210  2e 30 20
3284  f213  2a 2a
3284  f215  8d
3284  f216  0d
3285  f217  42 52 45          .byte 'brea', $cb
3285  f21a  41
3285  f21b  cb
3286  f21c
3287  f21c
3288  f21c  ===> test errorflag, ausgabe systemmeldung <===
3289  f21c
3290  f21c  2c 61 03  spmsg   bit msgflg       message flag
3291  f21f  10 0d             bpl msg10
3292  f221
3293  f221
3294  f221  ===> ausgabe systemmeldung, offset=yr <===
3295  f221
```

```
zeile adr.  obj.-code   source-code

3296  f221  b9 c3 f1  msg    lda ms1,y        tabelle der betriebssystem-meldungen
3297  f224  08              php
3298  f225  29 7f            and #$7f
3299  f227  20 d2 ff         jsr kbsout       ausgabe 1 char auf akt. kanal
3300  f22a  c8              iny
3301  f22b  28              plp
3302  f22c  10 f3           bpl msg          ausgabe systemmeldung, offset=yr
3303  f22e  18        msg10 clc
3304  f22f  60              rts
3305  f230
3306  f230
3307  f230  ===> talk auf iec-bus ausgeben <===
3308  f230
3309  f230  09 40     ntalk  ora #$40
3310  f232  d0 02            bne list1
3311  f234
3312  f234
3313  f234  ===> listen auf iec-bus ausgeben <===
3314  f234
3315  f234  09 20     nlistn ora #$20
3316  f236  48        list1  pha
3317  f237  a9 3f            lda #$3f
3318  f239  8d 03 de         sta tri1+ddpa    6525 triport 1: data direct.reg. a
3319  f23c  a9 ff            lda #$ff
3320  f23e  8d 00 dc         sta cia+pra      6526 cia: periph. data reg.a (pra)
3321  f241  8d 02 dc         sta cia+ddra     6526 cia: data direct.reg.a (ddra)
3322  f244  a9 fa            lda #$fa
3323  f246  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3324  f249  a5 aa            lda c3po         ieee puffer flag
3325  f24b  10 1b            bpl list2
3326  f24d  ad 00 de         lda tri1+pa      6525 triport 1: port reg. a
3327  f250  29 df            and #$df
3328  f252  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3329  f255  a5 ab            lda bsour        ieee zeichen puffer
3330  f257  20 b9 f2         jsr tbyte        ac auf iec-bus (ohne eof-flag)
3331  f25a  a5 aa            lda c3po         ieee puffer flag
3332  f25c  29 7f            and #$7f
3333  f25e  85 aa            sta c3po         ieee puffer flag
3334  f260  ad 00 de         lda tri1+pa      6525 triport 1: port reg. a
3335  f263  09 20            ora #$20
3336  f265  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3337  f268  ad 00 de  list2  lda tri1+pa      6525 triport 1: port reg. a
3338  f26b  29 f7            and #$f7
3339  f26d  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3340  f270  68               pla
3341  f271  4c b9 f2         jmp tbyte        ac auf iec-bus (ohne eof-flag)
3342  f274
3343  f274
3344  f274  ===> sekundäradr. nach listen ausgeben <===
3345  f274
3346  f274  20 b9 f2  nsecnd jsr tbyte        ac auf iec-bus (ohne eof-flag)
3347  f277  ad 00 de  scatn  lda tri1+pa      6525 triport 1: port reg. a
3348  f27a  09 08            ora #$08
3349  f27c  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3350  f27f  60               rts
3351  f280
3352  f280
3353  f280  ===> akt. sekundäradr. auf iec-bus ausgeben <===
```

```
zeile adr.  obj.-code   source-code

3354  f280
3355  f280  20 b9 f2  ntksa  jsr tbyte        ac auf iec-bus (ohne eof-flag)
3356  f283  a9 39     tkatn  lda #$39
3357  f285  2d 00 de         and tri1+pa      6525 triport 1: port reg. a
3358  f288  8d 00 de  setlns sta tri1+pa      6525 triport 1: port reg. a
3359  f28b  a9 c7             lda #$c7
3360  f28d  8d 03 de         sta tri1+ddpa    6525 triport 1: data direct.reg. a
3361  f290  a9 00             lda #$00
3362  f292  8d 02 dc         sta cia+ddra     6526 cia: data direct.reg.a (ddra)
3363  f295  f0 e0             beq scatn
3364  f297
3365  f297
3366  f297  ===> ac auf iec-bus (mit eof-flag) <===
3367  f297
3368  f297  48        nciout pha
3369  f298  a5 aa             lda c3po         ieee puffer flag
3370  f29a  10 07             bpl ci1
3371  f29c  a5 ab             lda bsour        ieee zeichen puffer
3372  f29e  20 b9 f2          jsr tbyte        ac auf iec-bus (ohne eof-flag)
3373  f2a1  a5 aa             lda c3po         ieee puffer flag
3374  f2a3  09 80     ci1     ora #$80
3375  f2a5  85 aa             sta c3po         ieee puffer flag
3376  f2a7  68                pla
3377  f2a8  85 ab             sta bsour        ieee zeichen puffer
3378  f2aa  60                rts
3379  f2ab
3380  f2ab
3381  f2ab  ===> untalk auf iec-bus ausgeben <===
3382  f2ab
3383  f2ab  a9 5f     nuntlk lda #$5f
3384  f2ad  d0 02             bne unls1
3385  f2af
3386  f2af
3387  f2af  ===> unlisten auf iec-bus ausgeben <===
3388  f2af
3389  f2af  a9 3f     nunlsn lda #$3f
3390  f2b1  20 36 f2  unls1  jsr list1
3391  f2b4  a9 f9             lda #$f9
3392  f2b6  4c 88 f2          jmp setlns
3393  f2b9
3394  f2b9
3395  f2b9  ===> ac auf iec-bus (ohne eof-flag) <===
3396  f2b9
3397  f2b9  49 ff     tbyte  eor #$ff
3398  f2bb  8d 00 dc         sta cia+pra      6526 cia: periph. data reg.a (pra)
3399  f2be  ad 00 de         lda tri1+pa      6525 triport 1: port reg. a
3400  f2c1  09 12             ora #$12
3401  f2c3  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3402  f2c6  2c 00 de         bit tri1+pa      6525 triport 1: port reg. a
3403  f2c9  50 09             bvc tby2
3404  f2cb  10 07             bpl tby2
3405  f2cd  a9 80             lda #$80
3406  f2cf  20 5f fb          jsr udst         'ora' ac in status
3407  f2d2  d0 30             bne tby7
3408  f2d4  ad 00 de  tby2   lda tri1+pa      6525 triport 1: port reg. a
3409  f2d7  10 fb             bpl tby2
3410  f2d9  29 ef             and #$ef
3411  f2db  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
```

```
zeile adr.  obj.-code   source-code

3412  f2de  20 6f f3  tby3   jsr timero      timer für time-out auf iec-i/o
                                              einschalten
3413  f2e1  90 01             bcc tby4
3414  f2e3  38        tby3t   sec
3415  f2e4  2c 00 de  tby4    bit tri1+pa     6525 triport 1: port reg. a
3416  f2e7  70 13             bvs tby6
3417  f2e9  ad 0d dc          lda cia+icr     6526 cia: irq control reg. (icr)
3418  f2ec  29 02             and #$02
3419  f2ee  f0 f4             beq tby4
3420  f2f0  ad 5e 03          lda timout      ieee timeout enable flag
3421  f2f3  30 e9             bmi tby3
3422  f2f5  90 ec             bcc tby3t
3423  f2f7  a9 01             lda #$01
3424  f2f9  20 5f fb          jsr udst        'ora' ac in status
3425  f2fc  ad 00 de  tby6    lda tri1+pa     6525 triport 1: port reg. a
3426  f2ff  09 10             ora #$10
3427  f301  8d 00 de          sta tri1+pa     6525 triport 1: port reg. a
3428  f304  a9 ff     tby7    lda #$ff
3429  f306  8d 00 dc          sta cia+pra     6526 cia: periph. data reg.a (pra)
3430  f309  60                rts
3431  f30a
3432  f30a
3433  f30a  ===> byte von iec nach ac <===
3434  f30a
3435  f30a  ad 00 de  nrbyte lda tri1+pa      6525 triport 1: port reg. a
3436  f30d  29 b9             and #$b9
3437  f30f  09 81             ora #$81
3438  f311  8d 00 de          sta tri1+pa     6525 triport 1: port reg. a
3439  f314  20 6f f3  rby1    jsr timero      timer für time-out auf iec-i/o
                                              einschalten
3440  f317  90 01             bcc rby2
3441  f319  38        rby1t   sec
3442  f31a  ad 00 de  rby2    lda tri1+pa     6525 triport 1: port reg. a
3443  f31d  29 10             and #$10
3444  f31f  f0 1e             beq rby4
3445  f321  ad 0d dc          lda cia+icr     6526 cia: irq control reg. (icr)
3446  f324  29 02             and #$02
3447  f326  f0 f2             beq rby2
3448  f328  ad 5e 03          lda timout      ieee timeout enable flag
3449  f32b  30 e7             bmi rby1
3450  f32d  90 ea             bcc rby1t
3451  f32f  a9 02             lda #$02
3452  f331  20 5f fb          jsr udst        'ora' ac in status
3453  f334  ad 00 de          lda tri1+pa     6525 triport 1: port reg. a
3454  f337  29 3d             and #$3d
3455  f339  8d 00 de          sta tri1+pa     6525 triport 1: port reg. a
3456  f33c  a9 0d             lda #$0d
3457  f33e  60                rts
3458  f33f
3459  f33f
3460  f33f  ad 00 de  rby4    lda tri1+pa     6525 triport 1: port reg. a
3461  f342  29 7f             and #$7f
3462  f344  8d 00 de          sta tri1+pa     6525 triport 1: port reg. a
3463  f347  29 20             and #$20
3464  f349  d0 05             bne rby5
3465  f34b  a9 40             lda #$40
3466  f34d  20 5f fb          jsr udst        'ora' ac in status
3467  f350  ad 00 dc  rby5    lda cia+pra     6526 cia: periph. data reg.a (pra)
```

```
zeile adr.  obj.-code   source-code

3468  f353  49 ff            eor #$ff
3469  f355  48               pha
3470  f356  ad 00 de         lda tri1+pa      6525 triport 1: port reg. a
3471  f359  09 40            ora #$40
3472  f35b  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3473  f35e  ad 00 de  rby7   lda tri1+pa      6525 triport 1: port reg. a
3474  f361  29 10            and #$10
3475  f363  f0 f9            beq rby7
3476  f365  ad 00 de         lda tri1+pa      6525 triport 1: port reg. a
3477  f368  29 bf            and #$bf
3478  f36a  8d 00 de         sta tri1+pa      6525 triport 1: port reg. a
3479  f36d  68               pla
3480  f36e  60               rts
3481  f36f
3482  f36f
3483  f36f  ===> timer für time-out auf iec-i/o einschalten <===
3484  f36f
3485  f36f  a9 ff     timero lda #$ff
3486  f371  8d 07 dc         sta cia+tbhi     6526 cia: timer b hi reg. (tb hi)
3487  f374  a9 11            lda #$11
3488  f376  8d 0f dc         sta cia+crb      6526 cia: control reg. b (crb)
3489  f379  ad 0d dc         lda cia+icr      6526 cia: irq control reg. (icr)
3490  f37c  18               clc
3491  f37d  60               rts
3492  f37e
3493  f37e
3494  f37e  4c 51 f9         jmp error9       ausgabe 'i/o error: 9'
3495  f381
3496  f381
3497  f381  ===> open rs232-schnittstelle <===
3498  f381
3499  f381  20 2e f4  opn232 jsr rst232       status rs232-schnittstelle
3500  f384  a0 00            ldy #$00
3501  f386  c4 9d     opn020 cpy fnlen        länge akt. filename
3502  f388  f0 0b            beq opn030
3503  f38a  20 92 fe         jsr fnadry       filename 'bank x' in ac, offset yr
3504  f38d  99 76 03         sta m51ctr,y     rs232: kopie 6551 control-reg.
3505  f390  c8               iny
3506  f391  c0 04            cpy #$04
3507  f393  d0 f1            bne opn020
3508  f395  ad 76 03  opn030 lda m51ctr       rs232: kopie 6551 control-reg.
3509  f398  8d 03 dd         sta acia+ctr     6551 acia: control-reg.
3510  f39b  ad 77 03         lda m51cdr       rs232: kopie 6551 command-reg.
3511  f39e  29 f2            and #$f2
3512  f3a0  09 02            ora #$02
3513  f3a2  8d 02 dd         sta acia+cdr     6551 acia: command-reg.
3514  f3a5  18               clc
3515  f3a6  a5 a0            lda sa           akt. sekundäradresse
3516  f3a8  29 02            and #$02
3517  f3aa  f0 15            beq opn045
3518  f3ac  ad 7d 03         lda ridbe        rs232: eingabe end zeiger
3519  f3af  8d 7c 03         sta ridbs        rs232: eingabe start zeiger
3520  f3b2  a5 a8            lda ribuf+2      bank rs232 eingabe-puffer
3521  f3b4  29 f0            and #$f0
3522  f3b6  f0 09            beq opn045
3523  f3b8  20 fc f3         jsr req256
3524  f3bb  85 a8            sta ribuf+2      bank rs232 eingabe-puffer
3525  f3bd  86 a6            stx ribuf        addrl rs232 eingabe-puffer
```

```
zeile adr.  obj.-code   source-code

3526  f3bf  84 a7           sty ribuf+1      addrh rs232 eingabe-puffer
3527  f3c1  90 03     opn045 bcc opn050
3528  f3c3  4c 53 f9         jmp errorx       fehler ausgabe-routine
3529  f3c6
3530  f3c6
3531  f3c6  60        opn050 rts
3532  f3c7
3533  f3c7
3534  f3c7  ===> umw. ascii-code nach cbm-code <===
3535  f3c7
3536  f3c7  c9 41     toasci cmp #$41
3537  f3c9  90 10            bcc toa020
3538  f3cb  c9 5b            cmp #$5b
3539  f3cd  b0 02            bcs toa010
3540  f3cf  09 20            ora #$20
3541  f3d1  c9 c1     toa010 cmp #$c1
3542  f3d3  90 06            bcc toa020
3543  f3d5  c9 db            cmp #$db
3544  f3d7  b0 02            bcs toa020
3545  f3d9  29 7f            and #$7f
3546  f3db  60        toa020 rts
3547  f3dc
3548  f3dc
3549  f3dc  ===> umw. cbm-code nach ascii-code <===
3550  f3dc
3551  f3dc  c9 41     tocbm  cmp #$41
3552  f3de  90 10            bcc toc020
3553  f3e0  c9 5b            cmp #$5b
3554  f3e2  b0 02            bcs toc010
3555  f3e4  09 80            ora #$80
3556  f3e6  c9 61     toc010 cmp #$61
3557  f3e8  90 06            bcc toc020
3558  f3ea  c9 7b            cmp #$7b
3559  f3ec  b0 02            bcs toc020
3560  f3ee  29 df            and #$df
3561  f3f0  60        toc020 rts
3562  f3f1
3563  f3f1
3564  f3f1  ad 02 dd  xon232 lda acia+cdr     6551 acia: command-reg.
3565  f3f4  09 09            ora #$09
3566  f3f6  29 fb            and #$fb
3567  f3f8  8d 02 dd         sta acia+cdr     6551 acia: command-reg.
3568  f3fb  60               rts
3569  f3fc
3570  f3fc
3571  f3fc  a2 00     req256 ldx #$00
3572  f3fe  a0 01            ldy #$01
3573  f400  8a        tttop  txa
3574  f401  38               sec
3575  f402  49 ff            eor #$ff
3576  f404  6d 55 03         adc hiadr        addrl ende des system speichers
3577  f407  aa               tax
3578  f408  98               tya
3579  f409  49 ff            eor #$ff
3580  f40b  6d 56 03         adc hiadr+1      addrh ende des system speichers
3581  f40e  a8               tay
3582  f40f  ad 57 03         lda hiadr+2      bank ende des system speichers
3583  f412  b0 06            bcs top010
```

```
zeile adr.  obj.-code   source-code

3584  f414  a9 ff            lda #$ff
3585  f416  09 40     topbad ora #$40
3586  f418  38               sec
3587  f419  60               rts
3588  f41a
3589  f41a
3590  f41a  cc 5c 03  top010 cpy memsiz+1     addrh ende des benutzer speichers
3591  f41d  90 f7            bcc topbad
3592  f41f  d0 05            bne topxit
3593  f421  ec 5b 03         cpx memsiz       addrl ende des benutzer speichers
3594  f424  90 f0            bcc topbad
3595  f426  8e 55 03  topxit stx hiadr        addrl ende des system speichers
3596  f429  8c 56 03         sty hiadr+1      addrh ende des system speichers
3597  f42c  18               clc
3598  f42d  60               rts
3599  f42e
3600  f42e
3601  f42e  ===> status rs232-schnittstelle <===
3602  f42e
3603  f42e  08        rst232 php
3604  f42f  78               sei
3605  f430  ad 01 dd         lda acia+srsn    6551 acia: reset-write und status
                                              read-reg.
3606  f433  29 60            and #$60
3607  f435  8d 7a 03         sta rsstat       rs232: akt. rs232-status
3608  f438  8d 7b 03         sta dcdsr        rs232: letzter dcd/dsr wert
3609  f43b  28               plp
3610  f43c  60               rts
3611  f43d
3612  f43d
3613  f43d  ===> eingabe 1 char vom aktiven kanal in ac (queue-vector) <===
3614  f43d
3615  f43d  a5 a1     ngetin lda dftln         vorgabe input-grätenummer
3616  f43f  d0 0c            bne gn10         test eingabekanal =2 (rs232)
3617  f441  a5 d1            lda ndx          anzahl bytes im tastaturpuffer
3618  f443  05 d6            ora kyndx        zeichenzähler für pgm tastenbefehl
3619  f445  f0 53            beq gn20
3620  f447  78               sei
3621  f448  20 07 e0         jsr jlp2
3622  f44b  18               clc
3623  f44c  60               rts
3624  f44d
3625  f44d
3626  f44d  ===> test eingabekanal =2 (rs232) <===
3627  f44d
3628  f44d  c9 02     gn10   cmp #$02
3629  f44f  f0 03            beq gn232        zeichen von rs232 in ac
3630  f451  4c cf ff         jmp kbasin       eingabe 1 char vom aktiven kanal in
                                              ac (chn-vector)
3631  f454
3632  f454
3633  f454  ===> zeichen von rs232 in ac <===
3634  f454
3635  f454  8c 65 03  gn232  sty xsav         kernalvariable temporär
3636  f457  8e 66 03         stx savx         kernalvariable temporär
3637  f45a  ac 7c 03         ldy ridbs        rs232: eingabe start zeiger
3638  f45d  cc 7d 03         cpy ridbe        rs232: eingabe end zeiger
3639  f460  d0 16            bne gn15
```

```
zeile adr.  obj.-code   source-code

3640  f462  ad 02 dd        lda acia+cdr     6551 acia: command-reg.
3641  f465  29 fd           and #$fd
3642  f467  09 01           ora #$01
3643  f469  8d 02 dd        sta acia+cdr     6551 acia: command-reg.
3644  f46c  ad 7a 03        lda rsstat       rs232: akt. rs232-status
3645  f46f  09 10           ora #$10
3646  f471  8d 7a 03        sta rsstat       rs232: akt. rs232-status
3647  f474  a9 00           lda #$00
3648  f476  f0 1c           beq gnexit
3649  f478  ad 7a 03  gn15  lda rsstat       rs232: akt. rs232-status
3650  f47b  29 ef           and #$ef
3651  f47d  8d 7a 03        sta rsstat       rs232: akt. rs232-status
3652  f480  a6 01           ldx i6509        6509 indirection register
3653  f482  a5 a8           lda ribuf+2      bank rs232 eingabe-puffer
3654  f484  85 01           sta i6509        6509 indirection register
3655  f486  b1 a6           lda (ribuf),y    addrl rs232 eingabe-puffer
3656  f488  86 01           stx i6509        6509 indirection register
3657  f48a  ee 7c 03        inc ridbs        rs232: eingabe start zeiger
3658  f48d  24 a0           bit sa           akt. sekundäradresse
3659  f48f  10 03           bpl gnexit
3660  f491  20 dc f3        jsr tocbm        umw. cbm-code nach ascii-code
3661  f494  ac 65 03  gnexit ldy xsav        kernalvariable temporär
3662  f497  ae 66 03        ldx savx         kernalvariable temporär
3663  f49a  18        gn20  clc
3664  f49b  60              rts
3665  f49c
3666  f49c
3667  f49c  ===> eingabe 1 char vom aktiven kanal in ac (chn-vector) <===
3668  f49c
3669  f49c  a5 a1     nbasin lda dftln        vorgabe input-grätenummer
3670  f49e  d0 0b           bne bn10
3671  f4a0  a5 cb           lda pntr         cursor spalte
3672  f4a2  85 ce           sta lstp         screen edit startposition
3673  f4a4  a5 ca           lda tblx         cursor zeile
3674  f4a6  85 cf           sta lsxp         letzte position (zeile)
3675  f4a8  4c b5 f4        jmp bn15
3676  f4ab
3677  f4ab
3678  f4ab  c9 03     bn10  cmp #$03
3679  f4ad  d0 0b           bne bn20
3680  f4af  85 d0           sta crsw         flag 'cr' bei bs-eingabe
3681  f4b1  a5 df           lda scrt         rechter rand bildschirm
3682  f4b3  85 d5           sta indx         position letztes zeichen in zeile
3683  f4b5  20 0a e0  bn15  jsr jloop5
3684  f4b8  18              clc
3685  f4b9  60              rts
3686  f4ba
3687  f4ba
3688  f4ba  b0 07     bn20  bcs bn30
3689  f4bc  c9 02           cmp #$02
3690  f4be  f0 10           beq bn50
3691  f4c0  20 5a fe        jsr xtape
3692  f4c3  a5 9c     bn30  lda status       i/o operation status
3693  f4c5  f0 04           beq bn35
3694  f4c7  a9 0d     bn31  lda #$0d
3695  f4c9  18        bn32  clc
3696  f4ca  60        bn33  rts
3697  f4cb
```

```
zeile adr.  obj.-code   source-code


3698  f4cb
3699  f4cb  20 a5 ff  bn35   jsr kacptr       ein byte von iec-bus nach accu
3700  f4ce  18               clc
3701  f4cf  60               rts
3702  f4d0
3703  f4d0
3704  f4d0  20 e4 ff  bn50   jsr kgetin       eingabe 1 char vom aktiven kanal in
                                              ac (queue-vector)
3705  f4d3  b0 f5            bcs bn33
3706  f4d5  c9 00            cmp #$00
3707  f4d7  d0 f0            bne bn32
3708  f4d9  ad 7a 03         lda rsstat       rs232: akt. rs232-status
3709  f4dc  29 10            and #$10
3710  f4de  f0 e9            beq bn32
3711  f4e0  ad 7a 03         lda rsstat       rs232: akt. rs232-status
3712  f4e3  29 60            and #$60
3713  f4e5  d0 e0            bne bn31
3714  f4e7  20 e1 ff         jsr kstop        stop-taste lesen
3715  f4ea  d0 e4            bne bn50
3716  f4ec  38               sec
3717  f4ed  60               rts
3718  f4ee
3719  f4ee
3720  f4ee  ===> ausgabe 1 char auf akt. kanal <===
3721  f4ee
3722  f4ee  48         nbsout pha
3723  f4ef  a5 a2            lda dflto        vorgabe output-gerätenummer
3724  f4f1  c9 03            cmp #$03
3725  f4f3  d0 06            bne bo10
3726  f4f5  68               pla
3727  f4f6  20 0d e0         jsr jprt         zeichen aus ac auf bildschirm
3728  f4f9  18               clc
3729  f4fa  60               rts
3730  f4fb
3731  f4fb
3732  f4fb  90 06      bo10  bcc bo20
3733  f4fd  68               pla
3734  f4fe  20 a8 ff         jsr kciout       ein byte aus accu auf iec-bus
3735  f501  18               clc
3736  f502  60               rts
3737  f503
3738  f503
3739  f503  c9 02      bo20  cmp #$02
3740  f505  f0 0a            beq bo50         zeichen von ac nach rs232
3741  f507  68               pla
3742  f508  20 5a fe         jsr xtape
3743  f50b  68         rstbo pla
3744  f50c  90 02            bcc rstor1
3745  f50e  a9 00            lda #$00
3746  f510  60         rstor1 rts
3747  f511
3748  f511
3749  f511  ===> zeichen von ac nach rs232 <===
3750  f511
3751  f511  8e 63 03  bo50   stx t1           kernalvariable temporär
3752  f514  8c 64 03         sty t2           kernalvariable temporär
3753  f517  ad 7a 03         lda rsstat       rs232: akt. rs232-status
3754  f51a  29 60            and #$60
```

```
zeile adr.  obj.-code   source-code

3755  f51c  d0 22           bne bo90
3756  f51e  68              pla
3757  f51f  24 a0           bit sa            akt. sekundäradresse
3758  f521  10 03           bpl bo80
3759  f523  20 c7 f3        jsr toasci        umw. ascii-code nach cbm-code
3760  f526  8d 00 dd  bo80  sta acia+drsn     6551 acia: data reg. write =
                                              transmit, read = receive
3761  f529  48              pha
3762  f52a  ad 7a 03  bo60  lda rsstat        rs232: akt. rs232-status
3763  f52d  29 60           and #$60
3764  f52f  d0 0f           bne bo90
3765  f531  ad 01 dd        lda acia+srsn     6551 acia: reset-write und status
                                              read-reg.
3766  f534  29 10           and #$10
3767  f536  d0 08           bne bo90
3768  f538  20 e1 ff        jsr kstop         stop-taste lesen
3769  f53b  d0 ed           bne bo60
3770  f53d  38              sec
3771  f53e  b0 cb           bcs rstbo
3772  f540  68        bo90  pla
3773  f541  ae 63 03        ldx t1            kernalvariable temporär
3774  f544  ac 64 03        ldy t2            kernalvariable temporär
3775  f547  18              clc
3776  f548  60              rts
3777  f549
3778  f549
3779  f549  ===> eingabe-kanal öffnen <===
3780  f549
3781  f549  20 3e f6  nchkin jsr lookup       status auf 0, zeiger file-tabelle
                                              erhöhen
3782  f54c  f0 03           beq jx310
3783  f54e  4c 3f f9        jmp error3        ausgabe 'i/o error: 3'
3784  f551
3785  f551
3786  f551  20 50 f6  jx310  jsr jz100        filetabellen in la,fa,sa kopieren
3787  f554  a5 9f           lda fa            akt. primäradresse (geräte-nummer)
3788  f556  f0 2e           beq jx320
3789  f558  c9 03           cmp #$03
3790  f55a  f0 2a           beq jx320
3791  f55c  b0 2c           bcs jx330
3792  f55e  c9 02           cmp #$02
3793  f560  d0 1e           bne jx315
3794  f562  a5 a0           lda sa            akt. sekundäradresse
3795  f564  29 02           and #$02
3796  f566  f0 1b           beq jx316
3797  f568  2d 02 dd        and acia+cdr      6551 acia: command-reg.
3798  f56b  f0 0f           beq jx312
3799  f56d  49 ff           eor #$ff
3800  f56f  2d 02 dd        and acia+cdr      6551 acia: command-reg.
3801  f572  09 01           ora #$01
3802  f574  48              pha
3803  f575  20 2e f4        jsr rst232        status rs232-schnittstelle
3804  f578  68              pla
3805  f579  8d 02 dd        sta acia+cdr      6551 acia: command-reg.
3806  f57c  a9 02     jx312  lda #$02
3807  f57e  d0 06           bne jx320
3808  f580  20 5a fe  jx315  jsr xtape
3809  f583  4c 48 f9  jx316  jmp error6        ausgabe 'i/o error: 6'
```

```
zeile adr.  obj.-code   source-code

3810  f586
3811  f586
3812  f586  85 a1     jx320  sta dftln        vorgabe input-grätenummer
3813  f588  18               clc
3814  f589  60               rts
3815  f58a
3816  f58a
3817  f58a  aa         jx330  tax
3818  f58b  20 b4 ff          jsr ktalk       talk auf iec-bus ausgeben
3819  f58e  a5 a0             lda sa          akt. sekundäradresse
3820  f590  10 06             bpl jx340
3821  f592  20 83 f2          jsr tkatn
3822  f595  4c 9b f5          jmp jx350
3823  f598
3824  f598
3825  f598  20 96 ff  jx340  jsr ktksa        auf iec-bus sekundäradr. nach talk
                                              ausgeben
3826  f59b  8a         jx350  txa
3827  f59c  24 9c             bit status      i/o operation status
3828  f59e  10 e6             bpl jx320
3829  f5a0  4c 45 f9          jmp error5      ausgabe 'i/o error: 5'
3830  f5a3
3831  f5a3
3832  f5a3  ===> ausgabe-kanal öffnen <===
3833  f5a3
3834  f5a3  20 3e f6  nckout jsr lookup       status auf 0, zeiger file-tabelle
                                              erhöhen
3835  f5a6  f0 03             beq ck5
3836  f5a8  4c 3f f9          jmp error3      ausgabe 'i/o error: 3'
3837  f5ab
3838  f5ab
3839  f5ab  20 50 f6  ck5    jsr jz100        filetabellen in la,fa,sa kopieren
3840  f5ae  a5 9f             lda fa          akt. primäradresse (geräte-nummer)
3841  f5b0  d0 03             bne ck10
3842  f5b2  4c 4b f9  ck20   jmp error7       ausgabe 'i/o error: 7'
3843  f5b5
3844  f5b5
3845  f5b5  c9 03     ck10   cmp #$03
3846  f5b7  f0 18             beq ck30
3847  f5b9  b0 1a             bcs ck40
3848  f5bb  c9 02             cmp #$02
3849  f5bd  d0 0f             bne ck15
3850  f5bf  a5 a0             lda sa          akt. sekundäradresse
3851  f5c1  4a                lsr a
3852  f5c2  90 ee             bcc ck20
3853  f5c4  20 2e f4          jsr rst232      status rs232-schnittstelle
3854  f5c7  20 f1 f3          jsr xon232
3855  f5ca  a9 02             lda #$02
3856  f5cc  d0 03             bne ck30
3857  f5ce  20 5a fe  ck15   jsr xtape
3858  f5d1  85 a2     ck30   sta dflto        vorgabe output-gerätenummer
3859  f5d3  18               clc
3860  f5d4  60               rts
3861  f5d5
3862  f5d5
3863  f5d5  aa         ck40   tax
3864  f5d6  20 b1 ff          jsr klistn      listen auf iec-bus ausgeben
3865  f5d9  a5 a0             lda sa          akt. sekundäradresse
```

```
zeile adr.  obj.-code   source-code

3866  f5db  10 05           bpl ck50
3867  f5dd  20 77 f2        jsr scatn
3868  f5e0  d0 03           bne ck60
3869  f5e2  20 93 ff  ck50  jsr ksecnd      auf iec-bus sekundäradr. nach listen
                                           ausgeben
3870  f5e5  8a        ck60  txa
3871  f5e6  24 9c           bit status      i/o operation status
3872  f5e8  10 e7           bpl ck30
3873  f5ea  4c 45 f9        jmp error5      ausgabe 'i/o error: 5'
3874  f5ed
3875  f5ed
3876  f5ed  ===> logisches file schliessen <===
3877  f5ed
3878  f5ed  08        nclose php
3879  f5ee  20 43 f6        jsr jltlk
3880  f5f1  f0 03           beq jx110
3881  f5f3  28              plp
3882  f5f4  18              clc
3883  f5f5  60              rts
3884  f5f6
3885  f5f6
3886  f5f6  20 50 f6  jx110 jsr jz100       filetabellen in la,fa,sa kopieren
3887  f5f9  28              plp
3888  f5fa  8a              txa
3889  f5fb  48              pha
3890  f5fc  90 1f           bcc jx150
3891  f5fe  a5 9f           lda fa          akt. primäradresse (geräte-nummer)
3892  f600  f0 1b           beq jx150
3893  f602  c9 03           cmp #$03
3894  f604  f0 17           beq jx150
3895  f606  b0 12           bcs jx120
3896  f608  c9 02           cmp #$02
3897  f60a  d0 07           bne jx115
3898  f60c  a9 00           lda #$00
3899  f60e  8d 02 dd        sta acia+cdr    6551 acia: command-reg.
3900  f611  f0 0a           beq jx150
3901  f613  68        jx115 pla
3902  f614  20 1e f6        jsr jx151
3903  f617  20 5a fe        jsr xtape
3904  f61a  20 bf f8  jx120 jsr clsei
3905  f61d  68        jx150 pla
3906  f61e  aa        jx151 tax
3907  f61f  ce 60 03        dec ldtnd       zeiger in file-tabellen
3908  f622  ec 60 03        cpx ldtnd       zeiger in file-tabellen
3909  f625  f0 15           beq jx160
3910  f627  ac 60 03        ldy ldtnd       zeiger in file-tabellen
3911  f62a  b9 34 03        lda lat,y       tabelle akt. log. file # (10 byte)
3912  f62d  9d 34 03        sta lat,x       tabelle akt. log. file # (10 byte)
3913  f630  b9 3e 03        lda fat,y       tabelle akt. primäradr.
3914  f633  9d 3e 03        sta fat,x       tabelle akt. primäradr.
3915  f636  b9 48 03        lda sat,y       tabelle akt. sekundäradr.
3916  f639  9d 48 03        sta sat,x       tabelle akt. sekundäradr.
3917  f63c  18        jx160 clc
3918  f63d  60              rts
3919  f63e
3920  f63e
3921  f63e  ===> status auf 0, zeiger file-tabelle erhöhen <===
3922  f63e
```

```
zeile adr.  obj.-code   source-code

3923  f63e  a9 00     lookup lda #$00
3924  f640  85 9c            sta status      i/o operation status
3925  f642  8a               txa
3926  f643  ae 60 03  jltlk  ldx ldtnd       zeiger in file-tabellen
3927  f646  ca        jx600  dex
3928  f647  30 2d            bmi lkups4
3929  f649  dd 34 03         cmp lat,x       tabelle akt. log. file # (10 byte)
3930  f64c  d0 f8            bne jx600
3931  f64e  18               clc
3932  f64f  60               rts
3933  f650
3934  f650
3935  f650  ===> filetabellen in la,fa,sa kopieren <===
3936  f650
3937  f650  bd 34 03  jz100  lda lat,x       tabelle akt. log. file # (10 byte)
3938  f653  85 9e            sta la          akt. logische filenummer
3939  f655  bd 3e 03         lda fat,x       tabelle akt. primäradr.
3940  f658  85 9f            sta fa          akt. primäradresse (geräte-nummer)
3941  f65a  bd 48 03         lda sat,x       tabelle akt. sekundäradr.
3942  f65d  85 a0            sta sa          akt. sekundäradresse
3943  f65f  60               rts
3944  f660
3945  f660
3946  f660  ===> file-parameter aufgrund sekundäradr. suchen <===
3947  f660
3948  f660  98        lkupsa tya
3949  f661  ae 60 03         ldx ldtnd       zeiger in file-tabellen
3950  f664  ca        lkups2 dex
3951  f665  30 0f            bmi lkups4
3952  f667  dd 48 03         cmp sat,x       tabelle akt. sekundäradr.
3953  f66a  d0 f8            bne lkups2
3954  f66c  18               clc
3955  f66d  20 50 f6  lkups3 jsr jz100       filetabellen in la,fa,sa kopieren
3956  f670  a8               tay
3957  f671  a5 9e            lda la          akt. logische filenummer
3958  f673  a6 9f            ldx fa          akt. primäradresse (geräte-nummer)
3959  f675  60               rts
3960  f676
3961  f676
3962  f676  38        lkups4 sec
3963  f677  60               rts
3964  f678
3965  f678
3966  f678  ===> file-parameter aufgrund logischer adr. suchen <===
3967  f678
3968  f678  aa        lkupla tax
3969  f679  20 3e f6         jsr lookup      status auf 0, zeiger file-tabelle
                                             erhöhen
3970  f67c  90 ef            bcc lkups3
3971  f67e  60               rts
3972  f67f
3973  f67f                   .end
3974  f67f                   .lib kernal2
```

```
zeile adr.  obj.-code   source-code


3976  f67f
3977  f67f  ===> alle logischen files schliessen <===
3978  f67f
3979  f67f  6e 65 03  nclall ror xsav          kernalvariable temporär
3980  f682  8d 66 03         sta savx          kernalvariable temporär
3981  f685  ae 60 03  ncl010 ldx ldtnd         zeiger in file-tabellen
3982  f688  ca        ncl020 dex
3983  f689  30 1b             bmi nclrch       alle kanäle schliessen
3984  f68b  2c 65 03          bit xsav         kernalvariable temporär
3985  f68e  10 08             bpl ncl030
3986  f690  ad 66 03          lda savx         kernalvariable temporär
3987  f693  dd 3e 03          cmp fat,x        tabelle akt. primäradr.
3988  f696  d0 f0             bne ncl020
3989  f698  bd 34 03  ncl030 lda lat,x         tabelle akt. log. file # (10 byte)
3990  f69b  38                sec
3991  f69c  20 c3 ff          jsr kclose       logisches file schliessen
3992  f69f  90 e4             bcc ncl010
3993  f6a1  a9 00             lda #$00
3994  f6a3  8d 60 03          sta ldtnd        zeiger in file-tabellen
3995  f6a6
3996  f6a6
3997  f6a6  ===> alle kanäle schliessen <===
3998  f6a6
3999  f6a6  a2 03     nclrch ldx #$03
4000  f6a8  e4 a2             cpx dflto        vorgabe output-gerätenummer
4001  f6aa  b0 03             bcs jx750
4002  f6ac  20 ae ff          jsr kunlsn       unlisten auf iec-bus ausgeben
4003  f6af  e4 a1     jx750  cpx dftln         vorgabe input-grätenummer
4004  f6b1  b0 03             bcs clall2
4005  f6b3  20 ab ff          jsr kuntlk       untalk auf iec-bus ausgeben
4006  f6b6  a2 03     clall2 ldx #$03
4007  f6b8  86 a2             stx dflto        vorgabe output-gerätenummer
4008  f6ba  a9 00             lda #$00
4009  f6bc  85 a1             sta dftln        vorgabe input-grätenummer
4010  f6be  60                rts
4011  f6bf
4012  f6bf
4013  f6bf  ===> log. file öffnen/befehl ausgeben <===
4014  f6bf
4015  f6bf  90 03     nopen  bcc op000
4016  f6c1  4c 3a f7          jmp tranr        sek.adr.15 + name auf ieee
4017  f6c4
4018  f6c4
4019  f6c4  a6 9e     op000  ldx la            akt. logische filenummer
4020  f6c6  20 3e f6          jsr lookup       status auf 0, zeiger file-tabelle
                                               erhöhen
4021  f6c9  d0 03             bne op100
4022  f6cb  4c 3c f9          jmp error2       ausgabe 'i/o error: 2'
4023  f6ce
4024  f6ce
4025  f6ce  ae 60 03  op100  ldx ldtnd         zeiger in file-tabellen
4026  f6d1  e0 0a             cpx #$0a
4027  f6d3  90 03             bcc op110
4028  f6d5  4c 39 f9          jmp error1       ausgabe 'i/o error: 1'
4029  f6d8
4030  f6d8
4031  f6d8  ee 60 03  op110  inc ldtnd         zeiger in file-tabellen
4032  f6db  a5 9e             lda la           akt. logische filenummer
```

```
zeile adr.  obj.-code   source-code

4033  f6dd  9d 34 03        sta lat,x        tabelle akt. log. file # (10 byte)
4034  f6e0  a5 a0           lda sa           akt. sekundäradresse
4035  f6e2  09 60           ora #$60
4036  f6e4  85 a0           sta sa           akt. sekundäradresse
4037  f6e6  9d 48 03        sta sat,x        tabelle akt. sekundäradr.
4038  f6e9  a5 9f           lda fa           akt. primäradresse (geräte-nummer)
4039  f6eb  9d 3e 03        sta fat,x        tabelle akt. primäradr.
4040  f6ee  f0 15           beq op175
4041  f6f0  c9 03           cmp #$03
4042  f6f2  f0 11           beq op175
4043  f6f4  90 05           bcc op150
4044  f6f6  20 07 f7        jsr openi        open ieee-kanal, name senden
4045  f6f9  90 0a           bcc op175
4046  f6fb  c9 02     op150 cmp #$02
4047  f6fd  d0 03           bne op152
4048  f6ff  4c 81 f3        jmp opn232       open rs232-schnittstelle
4049  f702
4050  f702
4051  f702  20 5a fe  op152 jsr xtape
4052  f705  18        op175 clc
4053  f706  60              rts
4054  f707
4055  f707
4056  f707  ===> open ieee-kanal, name senden <===
4057  f707
4058  f707  a5 a0     openi lda sa           akt. sekundäradresse
4059  f709  30 2d           bmi op50
4060  f70b  a4 9d           ldy fnlen        länge akt. filename
4061  f70d  f0 29           beq op50
4062  f70f  a5 9f           lda fa           akt. primäradresse (geräte-nummer)
4063  f711  20 b1 ff        jsr klistn       listen auf iec-bus ausgeben
4064  f714  a5 a0           lda sa           akt. sekundäradresse
4065  f716  09 f0           ora #$f0
4066  f718
4067  f718
4068  f718  ===> ac=sek.adr. + name auf ieee <===
4069  f718
4070  f718  20 93 ff  openib jsr ksecnd      auf iec-bus sekundäradr. nach listen
                                             ausgeben
4071  f71b  a5 9c           lda status       i/o operation status
4072  f71d  10 05           bpl op35
4073  f71f  68              pla
4074  f720  68              pla
4075  f721  4c 45 f9        jmp error5       ausgabe 'i/o error: 5'
4076  f724
4077  f724
4078  f724  a5 9d     op35  lda fnlen        länge akt. filename
4079  f726  f0 0d           beq op45
4080  f728  a0 00           ldy #$00
4081  f72a  20 92 fe  op40  jsr fnadry       filename 'bank x' in ac, offset yr
4082  f72d  20 a8 ff        jsr kciout       ein byte aus accu auf iec-bus
4083  f730  c8              iny
4084  f731  c4 9d           cpy fnlen        länge akt. filename
4085  f733  d0 f5           bne op40
4086  f735  20 ae ff  op45  jsr kunlsn       unlisten auf iec-bus ausgeben
4087  f738  18        op50  clc
4088  f739  60              rts
4089  f73a
```

```
zeile adr.  obj.-code   source-code

4090  f73a
4091  f73a  ===> sek.adr.15 + name auf ieee <===
4092  f73a
4093  f73a  a5 9f     tranr  lda fa          akt. primäradresse (geräte-nummer)
4094  f73c  20 b1 ff         jsr klistn      listen auf iec-bus ausgeben
4095  f73f  a9 6f            lda #$6f
4096  f741  85 a0            sta sa          akt. sekundäradresse
4097  f743  4c 18 f7         jmp openib      ac=sek.adr. + name auf ieee
4098  f746
4099  f746
4100  f746  ===> einlesen vom log. file <===
4101  f746
4102  f746  8e 6f 03  nload  stx relsal      addrl variable start-load-adr.
4103  f749  8c 70 03         sty relsah      addrh variable start-load-adr.
4104  f74c  8d 5f 03         sta verck       load/verify flag
4105  f74f  8d 71 03         sta relsag      bank variable start-load-adr.
4106  f752  a9 00            lda #$00
4107  f754  85 9c            sta status      i/o operation status
4108  f756  a5 9f            lda fa          akt. primäradresse (geräte-nummer)
4109  f758  d0 03            bne ld20
4110  f75a  4c 51 f9  ld10   jmp error9      ausgabe 'i/o error: 9'
4111  f75d
4112  f75d
4113  f75d  c9 03     ld20   cmp #$03
4114  f75f  f0 f9            beq ld10
4115  f761  b0 03            bcs ld22
4116  f763  4c 10 f8         jmp ld100
4117  f766
4118  f766
4119  f766  a9 60     ld22   lda #$60
4120  f768  85 a0            sta sa          akt. sekundäradresse
4121  f76a  a4 9d            ldy fnlen       länge akt. filename
4122  f76c  d0 03            bne ld25
4123  f76e  4c 4e f9         jmp error8      ausgabe 'i/o error: 8'
4124  f771
4125  f771
4126  f771  20 1b f8  ld25   jsr luking
4127  f774  20 07 f7         jsr openi       open ieee-kanal, name senden
4128  f777  a5 9f            lda fa          akt. primäradresse (geräte-nummer)
4129  f779  20 b4 ff         jsr ktalk       talk auf iec-bus ausgeben
4130  f77c  a5 a0            lda sa          akt. sekundäradresse
4131  f77e  20 96 ff         jsr ktksa       auf iec-bus sekundäradr. nach talk
                                             ausgeben
4132  f781  20 a5 ff         jsr kacptr      ein byte von iec-bus nach accu
4133  f784  85 96            sta eal         addrl ende akt. load/store
4134  f786  85 99            sta stal        addrl anfang akt. load/store
4135  f788  a5 9c            lda status      i/o operation status
4136  f78a  4a               lsr a
4137  f78b  4a               lsr a
4138  f78c  90 03            bcc ld30
4139  f78e  4c 42 f9         jmp error4      ausgabe 'i/o error: 4'
4140  f791
4141  f791
4142  f791  20 a5 ff  ld30   jsr kacptr      ein byte von iec-bus nach accu
4143  f794  85 97            sta eah         addrh ende akt. load/store
4144  f796  85 9a            sta stah        addrh anfang akt. load/store
4145  f798  20 40 f8         jsr loding
4146  f79b  ad 71 03         lda relsag      bank variable start-load-adr.
```

```
zeile adr.  obj.-code   source-code

4147  f79e  85 98          sta eas         bank ende akt. load/store
4148  f7a0  85 9b          sta stas        bank anfang akt. load/store
4149  f7a2  ad 6f 03       lda relsal      addrl variable start-load-adr.
4150  f7a5  2d 70 03       and relsah      addrh variable start-load-adr.
4151  f7a8  c9 ff          cmp #$ff
4152  f7aa  f0 0e          beq ld40
4153  f7ac  ad 6f 03       lda relsal      addrl variable start-load-adr.
4154  f7af  85 96          sta eal         addrl ende akt. load/store
4155  f7b1  85 99          sta stal        addrl anfang akt. load/store
4156  f7b3  ad 70 03       lda relsah      addrh variable start-load-adr.
4157  f7b6  85 97          sta eah         addrh ende akt. load/store
4158  f7b8  85 9a          sta stah        addrh anfang akt. load/store
4159  f7ba  a9 fd    ld40  lda #$fd
4160  f7bc  25 9c          and status      i/o operation status
4161  f7be  85 9c          sta status      i/o operation status
4162  f7c0  20 e1 ff       jsr kstop       stop-taste lesen
4163  f7c3  d0 03          bne ld45
4164  f7c5  4c b3 f8       jmp break
4165  f7c8
4166  f7c8
4167  f7c8  20 a5 ff ld45  jsr kacptr      ein byte von iec-bus nach accu
4168  f7cb  aa             tax
4169  f7cc  a5 9c          lda status      i/o operation status
4170  f7ce  4a             lsr a
4171  f7cf  4a             lsr a
4172  f7d0  b0 e8          bcs ld40
4173  f7d2  8a             txa
4174  f7d3  a6 01          ldx i6509       6509 indirection register
4175  f7d5  a4 98          ldy eas         bank ende akt. load/store
4176  f7d7  84 01          sty i6509       6509 indirection register
4177  f7d9  a0 00          ldy #$00
4178  f7db  2c 5f 03       bit verck       load/verify flag
4179  f7de  10 0e          bpl ld60-2
4180  f7e0  85 93          sta sal         addrl akt. load/store adresse
4181  f7e2  b1 96          lda (eal),y     addrl ende akt. load/store
4182  f7e4  c5 93          cmp sal         addrl akt. load/store adresse
4183  f7e6  f0 08          beq ld60
4184  f7e8  a9 10          lda #$10
4185  f7ea  20 5f fb       jsr udst        'ora' ac in status
4186  f7ed  ad 91 96       lda pulst1
4187  f7f0  86 01    ld60  stx i6509       6509 indirection register
4188  f7f2  e6 96          inc eal         addrl ende akt. load/store
4189  f7f4  d0 0a          bne ld64
4190  f7f6  e6 97          inc eah         addrh ende akt. load/store
4191  f7f8  d0 06          bne ld64
4192  f7fa  e6 98          inc eas         bank ende akt. load/store
4193  f7fc  a9 02          lda #$02
4194  f7fe  85 96          sta eal         addrl ende akt. load/store
4195  f800  24 9c    ld64  bit status      i/o operation status
4196  f802  50 b6          bvc ld40
4197  f804  20 ab ff       jsr kuntlk      untalk auf iec-bus ausgeben
4198  f807  20 bf f8       jsr clsei
4199  f80a  4c 13 f8       jmp ld180
4200  f80d
4201  f80d
4202  f80d  4c 42 f9       jmp error4      ausgabe 'i/o error: 4'
4203  f810
4204  f810
```

```
zeile adr.  obj.-code   source-code

4205  f810  20 5a fe  ld100  jsr xtape
4206  f813  18        ld180  clc
4207  f814  a5 98             lda eas          bank ende akt. load/store
4208  f816  a6 96             ldx eal          addrl ende akt. load/store
4209  f818  a4 97             ldy eah          addrh ende akt. load/store
4210  f81a  60                 rts
4211  f81b
4212  f81b
4213  f81b  2c 61 03  luking bit msgflg       message flag
4214  f81e  10 1f             bpl ld115
4215  f820  a0 0c             ldy #$0c
4216  f822  20 1c f2          jsr spmsg        test errorflag, ausgabe
                                               systemmeldung
4217  f825  a5 9d             lda fnlen        länge akt. filename
4218  f827  f0 16             beq ld115
4219  f829  a0 17             ldy #$17
4220  f82b  20 1c f2          jsr spmsg        test errorflag, ausgabe
                                               systemmeldung
4221  f82e  a4 9d     outfn   ldy fnlen        länge akt. filename
4222  f830  f0 0d             beq ld115
4223  f832  a0 00             ldy #$00
4224  f834  20 92 fe  ld110  jsr fnadry        filename 'bank x' in ac, offset yr
4225  f837  20 d2 ff          jsr kbsout       ausgabe 1 char auf akt. kanal
4226  f83a  c8                iny
4227  f83b  c4 9d             cpy fnlen        länge akt. filename
4228  f83d  d0 f5             bne ld110
4229  f83f  60        ld115  rts
4230  f840
4231  f840
4232  f840  a0 1b     loding ldy #$1b
4233  f842  ad 5f 03          lda verck        load/verify flag
4234  f845  10 02             bpl ld410
4235  f847  a0 2b             ldy #$2b
4236  f849  4c 1c f2  ld410  jmp spmsg        test errorflag, ausgabe
                                               systemmeldung
4237  f84c
4238  f84c
4239  f84c  ===> abspeichern auf log. file <===
4240  f84c
4241  f84c  b5 00     nsave  lda e6509,x       6509 execution register
4242  f84e  85 99            sta stal         addrl anfang akt. load/store
4243  f850  b5 01            lda i6509,x      6509 indirection register
4244  f852  85 9a            sta stah         addrh anfang akt. load/store
4245  f854  b5 02            lda usrpok,x     jmp code für 'usr'-routine
4246  f856  85 9b            sta stas         bank anfang akt. load/store
4247  f858  98               tya
4248  f859  aa               tax
4249  f85a  b5 00            lda e6509,x      6509 execution register
4250  f85c  85 96            sta eal          addrl ende akt. load/store
4251  f85e  b5 01            lda i6509,x      6509 indirection register
4252  f860  85 97            sta eah          addrh ende akt. load/store
4253  f862  b5 02            lda usrpok,x     jmp code für 'usr'-routine
4254  f864  85 98            sta eas          bank ende akt. load/store
4255  f866  a5 9f            lda fa           akt. primäradresse (geräte-nummer)
4256  f868  d0 03            bne sv20
4257  f86a  4c 51 f9  sv10   jmp error9       ausgabe 'i/o error: 9'
4258  f86d
4259  f86d
```

```
zeile adr.  obj.-code   source-code

4260  f86d  c9 03     sv20   cmp #$03
4261  f86f  f0 f9            beq sv10
4262  f871  90 63            bcc sv100
4263  f873  a9 61            lda #$61
4264  f875  85 a0            sta sa          akt. sekundäradresse
4265  f877  a4 9d            ldy fnlen       länge akt. filename
4266  f879  d0 03            bne sv25
4267  f87b  4c 4e f9         jmp error8      ausgabe 'i/o error: 8'
4268  f87e
4269  f87e
4270  f87e  20 07 f7  sv25   jsr openi       open ieee-kanal, name senden
4271  f881  20 d9 f8         jsr saving
4272  f884  a5 9f            lda fa          akt. primäradresse (geräte-nummer)
4273  f886  20 b1 ff         jsr klistn      listen auf iec-bus ausgeben
4274  f889  a5 a0            lda sa          akt. sekundäradresse
4275  f88b  20 93 ff         jsr ksecnd      auf iec-bus sekundäradr. nach listen
                                             ausgeben
4276  f88e  a6 01            ldx i6509       6509 indirection register
4277  f890  20 62 fe         jsr rd300       'stah'-vektor nach 'sah'-vektor und
                                             bank# nach i6509
4278  f893  a5 93            lda sal         addrl akt. load/store adresse
4279  f895  20 a8 ff         jsr kciout      ein byte aus accu auf iec-bus
4280  f898  a5 94            lda sah         addrh akt. load/store adresse
4281  f89a  20 a8 ff         jsr kciout      ein byte aus accu auf iec-bus
4282  f89d  a0 00            ldy #$00
4283  f89f  20 71 fe  sv30   jsr cmpste      vektor 'sal' - 'eal'
4284  f8a2  b0 16            bcs sv50
4285  f8a4  b1 93            lda (sal),y     addrl akt. load/store adresse
4286  f8a6  20 a8 ff         jsr kciout      ein byte aus accu auf iec-bus
4287  f8a9  20 7f fe         jsr incsal
4288  f8ac  20 e1 ff         jsr kstop       stop-taste lesen
4289  f8af  d0 ee            bne sv30
4290  f8b1  86 01            stx i6509       6509 indirection register
4291  f8b3  20 bf f8  break  jsr clsei
4292  f8b6  a9 00            lda #$00
4293  f8b8  38               sec
4294  f8b9  60               rts
4295  f8ba
4296  f8ba
4297  f8ba  86 01     sv50   stx i6509       6509 indirection register
4298  f8bc  20 ae ff         jsr kunlsn      unlisten auf iec-bus ausgeben
4299  f8bf  24 a0     clsei  bit sa          akt. sekundäradresse
4300  f8c1  30 11            bmi sv110
4301  f8c3  a5 9f            lda fa          akt. primäradresse (geräte-nummer)
4302  f8c5  20 b1 ff         jsr klistn      listen auf iec-bus ausgeben
4303  f8c8  a5 a0            lda sa          akt. sekundäradresse
4304  f8ca  29 ef            and #$ef
4305  f8cc  09 e0            ora #$e0
4306  f8ce  20 93 ff         jsr ksecnd      auf iec-bus sekundäradr. nach listen
                                             ausgeben
4307  f8d1  20 ae ff         jsr kunlsn      unlisten auf iec-bus ausgeben
4308  f8d4  18        sv110  clc
4309  f8d5  60        sv115  rts
4310  f8d6
4311  f8d6
4312  f8d6  20 5a fe  sv100  jsr xtape
4313  f8d9  ad 61 03  saving lda msgflg      message flag
4314  f8dc  10 f7            bpl sv115
```

```
zeile adr.  obj.-code   source-code

4315  f8de  a0 23           ldy #$23
4316  f8e0  20 1c f2        jsr spmsg       test errorflag, ausgabe
                                            systemmeldung
4317  f8e3  4c 2e f8        jmp outfn
4318  f8e6
4319  f8e6
4320  f8e6  ===> zeit lesen (am/pm,h,min,s,1/10 s) <===
4321  f8e6
4322  f8e6  ad 08 dc  rdtim  lda cia+tod10   6526 cia: uhr: 1/10 sek.-reg.
4323  f8e9  48               pha
4324  f8ea  48               pha
4325  f8eb  0a               asl a
4326  f8ec  0a               asl a
4327  f8ed  0a               asl a
4328  f8ee  29 60            and #$60
4329  f8f0  0d 0b dc         ora cia+todhr    6526 cia: uhr: stunde-am/pm-reg.
4330  f8f3  a8               tay
4331  f8f4  68               pla
4332  f8f5  6a               ror a
4333  f8f6  6a               ror a
4334  f8f7  29 80            and #$80
4335  f8f9  0d 09 dc         ora cia+todsec   6526 cia: uhr: sek.-reg.
4336  f8fc  85 93            sta sal          addrl akt. load/store adresse
4337  f8fe  6a               ror a
4338  f8ff  29 80            and #$80
4339  f901  0d 0a dc         ora cia+todmin   6526 cia: uhr: min.-reg.
4340  f904  aa               tax
4341  f905  68               pla
4342  f906  cd 08 dc         cmp cia+tod10    6526 cia: uhr: 1/10 sek.-reg.
4343  f909  d0 db            bne rdtim        zeit lesen (am/pm,h,min,s,1/10 s)
4344  f90b  a5 93            lda sal          addrl akt. load/store adresse
4345  f90d  60               rts
4346  f90e
4347  f90e
4348  f90e  ===> zeit setzen (am/pm,h,min,s,1/10 s) <===
4349  f90e
4350  f90e  48         settim pha
4351  f90f  48               pha
4352  f910  6a               ror a
4353  f911  29 80            and #$80
4354  f913  0d 0f dc         ora cia+crb      6526 cia: control reg. b (crb)
4355  f916  8d 0f dc         sta cia+crb      6526 cia: control reg. b (crb)
4356  f919  98               tya
4357  f91a  2a               rol a
4358  f91b  2a               rol a
4359  f91c  26 93            rol sal          addrl akt. load/store adresse
4360  f91e  2a               rol a
4361  f91f  26 93            rol sal          addrl akt. load/store adresse
4362  f921  8a               txa
4363  f922  2a               rol a
4364  f923  26 93            rol sal          addrl akt. load/store adresse
4365  f925  68               pla
4366  f926  2a               rol a
4367  f927  26 93            rol sal          addrl akt. load/store adresse
4368  f929  8c 0b dc         sty cia+todhr    6526 cia: uhr: stunde-am/pm-reg.
4369  f92c  8e 0a dc         stx cia+todmin   6526 cia: uhr: min.-reg.
4370  f92f  68               pla
4371  f930  8d 09 dc         sta cia+todsec   6526 cia: uhr: sek.-reg.
```

```
zeile adr.  obj.-code   source-code

4372  f933  a5 93           lda sal          addrl akt. load/store adresse
4373  f935  8d 08 dc        sta cia+tod10    6526 cia: uhr: 1/10 sek.-reg.
4374  f938  60              rts
4375  f939
4376  f939
4377  f939  ===> ausgabe 'i/o error: 1' <===
4378  f939
4379  f939  a9 01     error1 lda #$01
4380  f93b  2c               .byte $2c
4381  f93c
4382  f93c
4383  f93c  ===> ausgabe 'i/o error: 2' <===
4384  f93c
4385  f93c  a9 02     error2 lda #$02
4386  f93e  2c               .byte $2c
4387  f93f
4388  f93f
4389  f93f  ===> ausgabe 'i/o error: 3' <===
4390  f93f
4391  f93f  a9 03     error3 lda #$03
4392  f941  2c               .byte $2c
4393  f942
4394  f942
4395  f942  ===> ausgabe 'i/o error: 4' <===
4396  f942
4397  f942  a9 04     error4 lda #$04
4398  f944  2c               .byte $2c
4399  f945
4400  f945
4401  f945  ===> ausgabe 'i/o error: 5' <===
4402  f945
4403  f945  a9 05     error5 lda #$05
4404  f947  2c               .byte $2c
4405  f948
4406  f948
4407  f948  ===> ausgabe 'i/o error: 6' <===
4408  f948
4409  f948  a9 06     error6 lda #$06
4410  f94a  2c               .byte $2c
4411  f94b
4412  f94b
4413  f94b  ===> ausgabe 'i/o error: 7' <===
4414  f94b
4415  f94b  a9 07     error7 lda #$07
4416  f94d  2c               .byte $2c
4417  f94e
4418  f94e
4419  f94e  ===> ausgabe 'i/o error: 8' <===
4420  f94e
4421  f94e  a9 08     error8 lda #$08
4422  f950  2c               .byte $2c
4423  f951
4424  f951
4425  f951  ===> ausgabe 'i/o error: 9' <===
4426  f951
4427  f951  a9 09     error9 lda #$09
4428  f953
4429  f953
```

```
zeile adr.  obj.-code   source-code


4430  f953  ===> fehler ausgabe-routine <===
4431  f953
4432  f953  48        errorx pha
4433  f954  20 cc ff         jsr kclrch      ein-/ausgabekanal schliessen
4434  f957  a0 00            ldy #$00
4435  f959  2c 61 03         bit msgflg      message flag
4436  f95c  50 0a            bvc erexit
4437  f95e  20 21 f2         jsr msg         ausgabe systemmeldung, offset=yr
4438  f961  68               pla
4439  f962  48               pha
4440  f963  09 30            ora #$30
4441  f965  20 d2 ff         jsr kbsout      ausgabe 1 char auf akt. kanal
4442  f968  68        erexit pla
4443  f969  38               sec
4444  f96a  60               rts
4445  f96b
4446  f96b
4447  f96b  ===> stop-taste überprüfen <===
4448  f96b
4449  f96b  a5 a9     nstop  lda stkey        stop tasten flag
4450  f96d  29 01            and #$01
4451  f96f  d0 07            bne stop2
4452  f971  08               php
4453  f972  20 cc ff         jsr kclrch      ein-/ausgabekanal schliessen
4454  f975  85 d1            sta ndx         anzahl bytes im tastaturpuffer
4455  f977  28               plp
4456  f978  60        stop2  rts
4457  f979
4458  f979
4459  f979  ===> system-zeit aktualisieren <===
4460  f979
4461  f979  ad 02 df  udtim  lda tri2+pc      6525 triport 2: port reg. c
4462  f97c  4a               lsr a
4463  f97d  b0 12            bcs udexit
4464  f97f  a9 fe            lda #$fe
4465  f981  8d 01 df         sta tri2+pb      6525 triport 2: port reg. b
4466  f984  a9 10            lda #$10
4467  f986  2d 02 df         and tri2+pc      6525 triport 2: port reg. c
4468  f989  d0 01            bne udttt
4469  f98b  38               sec
4470  f98c  a9 ff     udttt  lda #$ff
4471  f98e  8d 01 df         sta tri2+pb      6525 triport 2: port reg. b
4472  f991  2a        udexit rol a
4473  f992  85 a9            sta stkey        stop tasten flag
4474  f994  60               rts
4475  f995
4476  f995
4477  f995  ===> test-bytes für cbm-rom <===
4478  f995
4479  f995  c2        patall .byte $c2, $cd
4479  f996  cd
4480  f997
4481  f997
4482  f997  ===> system reset routine <===
4483  f997
4484  f997  a2 fe     start  ldx #$fe
4485  f999  78               sei
4486  f99a  9a               txs
```

```
zeile adr.  obj.-code   source-code

4487  f99b  d8              cld
4488  f99c  a9 a5           lda #$a5
4489  f99e  cd fa 03        cmp evect+2      warmstart-variable
4490  f9a1  d0 07           bne scold        system-kaltstart
4491  f9a3  ad fb 03        lda evect+3      warmstart-variable
4492  f9a6  c9 5a           cmp #$5a
4493  f9a8  f0 4e           beq swarm        system-warmstart
4494  f9aa
4495  f9aa
4496  f9aa  ===> system-kaltstart <===
4497  f9aa
4498  f9aa  a9 06     scold lda #$06
4499  f9ac  85 96           sta eal          addrl ende akt. load/store
4500  f9ae  a9 00           lda #$00
4501  f9b0  85 97           sta eah          addrh ende akt. load/store
4502  f9b2  8d f8 03        sta evect        addrl warmstart-vector
4503  f9b5  a2 30           ldx #$30
4504  f9b7  a0 03     sloop0 ldy #$03
4505  f9b9  a5 97           lda eah          addrh ende akt. load/store
4506  f9bb  30 18           bmi sloop2
4507  f9bd  18              clc
4508  f9be  69 10           adc #$10
4509  f9c0  85 97           sta eah          addrh ende akt. load/store
4510  f9c2  e8              inx
4511  f9c3  8a              txa
4512  f9c4  d1 96           cmp (eal),y      addrl ende akt. load/store
4513  f9c6  d0 ef           bne sloop0
4514  f9c8  88              dey
4515  f9c9  b1 96     sloop1 lda (eal),y     addrl ende akt. load/store
4516  f9cb  88              dey
4517  f9cc  30 0a           bmi sloop3
4518  f9ce  d9 95 f9        cmp patall,y     test-bytes für cbm-rom
4519  f9d1  f0 f6           beq sloop1
4520  f9d3  d0 e2           bne sloop0
4521  f9d5  a0 e0     sloop2 ldy #$e0
4522  f9d7  2c              .byte $2c
4523  f9d8  a4 97     sloop3 ldy eah          addrh ende akt. load/store
4524  f9da  8c f9 03        sty evect+1      addrh warmstart-vector
4525  f9dd  aa              tax
4526  f9de  10 18           bpl swarm        system-warmstart
4527  f9e0  20 fb f9        jsr ioinit       i/o register init.
4528  f9e3  a9 f0           lda #$f0
4529  f9e5  85 c1           sta pkybuf+1     addrh start programmierbare tasten
4530  f9e7  20 04 e0        jsr jcint
4531  f9ea  20 88 fa        jsr ramtas       ram-test, zeiger init.
4532  f9ed  20 a2 fb        jsr restor       standard-vektoren für jsr und
                                             interrupts einrichten
4533  f9f0  20 04 e0        jsr jcint
4534  f9f3  a9 a5           lda #$a5
4535  f9f5  8d fa 03        sta evect+2      warmstart-variable
4536  f9f8  6c f8 03  swarm  jmp (evect)     system-warmstart
4537  f9fb
4538  f9fb
4539  f9fb  ===> i/o register init. <===
4540  f9fb
4541  f9fb  a9 f3     ioinit lda #$f3
4542  f9fd  8d 06 de        sta tri1+creg    6525 triport 1: control reg.
4543  fa00  a0 ff           ldy #$ff
```

```
zeile adr.  obj.-code   source-code

4544  fa02  8c 05 de        sty tri1+ddpc    6525 triport 1: data direct.reg. c
4545  fa05  a9 5c           lda #$5c
4546  fa07  8d 01 de        sta tri1+pb      6525 triport 1: port reg. b
4547  fa0a  a9 7d           lda #$7d
4548  fa0c  8d 04 de        sta tri1+ddpb    6525 triport 1: data direct.reg. b
4549  fa0f  a9 3d           lda #$3d
4550  fa11  8d 00 de        sta tri1+pa      6525 triport 1: port reg. a
4551  fa14  a9 3f           lda #$3f
4552  fa16  8d 03 de        sta tri1+ddpa    6525 triport 1: data direct.reg. a
4553  fa19  8c 00 df        sty tri2+pa      6525 triport 2: port reg. a
4554  fa1c  8c 01 de        sty tri1+pb      6525 triport 1: port reg. b
4555  fa1f  8c 03 df        sty tri2+ddpa    6525 triport 2: data direct.reg. a
4556  fa22  8c 04 df        sty tri2+ddpb    6525 triport 2: data direct.reg. b
4557  fa25  4e 00 df        lsr tri2+pa      6525 triport 2: port reg. a
4558  fa28  c8              iny
4559  fa29  8c 02 df        sty tri2+pc      6525 triport 2: port reg. c
4560  fa2c  8c 05 df        sty tri2+ddpc    6525 triport 2: data direct.reg. c
4561  fa2f  a9 7f           lda #$7f
4562  fa31  8d 0d dc        sta cia+icr      6526 cia: irq control reg. (icr)
4563  fa34  8c 02 dc        sty cia+ddra     6526 cia: data direct.reg.a (ddra)
4564  fa37  8c 03 dc        sty cia+ddrb     6526 cia: data direct.reg.b (ddrb)
4565  fa3a  8c 0f dc        sty cia+crb      6526 cia: control reg. b (crb)
4566  fa3d  8d 08 dc        sta cia+tod10    6526 cia: uhr: 1/10 sek.-reg.
4567  fa40  8c 02 de        sty tri1+pc      6525 triport 1: port reg. c
4568  fa43  ad 02 de  io100  lda tri1+pc      6525 triport 1: port reg. c
4569  fa46  6a              ror a
4570  fa47  90 fa           bcc io100
4571  fa49  8c 02 de        sty tri1+pc      6525 triport 1: port reg. c
4572  fa4c  a2 00           ldx #$00
4573  fa4e  e8        io110 inx
4574  fa4f  d0 fd           bne io110
4575  fa51  c8              iny
4576  fa52  ad 02 de        lda tri1+pc      6525 triport 1: port reg. c
4577  fa55  6a              ror a
4578  fa56  90 f6           bcc io110
4579  fa58  c0 1b           cpy #$1b
4580  fa5a  90 03           bcc io120
4581  fa5c  a9 88           lda #$88
4582  fa5e  2c              .byte $2c
4583  fa5f  a9 08     io120 lda #$08
4584  fa61  8d 0e dc        sta cia+cra      6526 cia: control reg. a (cra)
4585  fa64  ad 0d db        lda ipcia+icr    6526 cia: irq control reg. (icr)
4586  fa67  a9 90           lda #$90
4587  fa69  8d 0d db        sta ipcia+icr    6526 cia: irq control reg. (icr)
4588  fa6c  a9 40           lda #$40
4589  fa6e  8d 01 db        sta ipcia+prb    6526 cia: periph.data reg. b (prb)
4590  fa71  8e 02 db        stx ipcia+ddra   6526 cia: data direct.reg.a (ddra)
4591  fa74  8e 0f db        stx ipcia+crb    6526 cia: control reg. b (crb)
4592  fa77  8e 0e db        stx ipcia+cra    6526 cia: control reg. a (cra)
4593  fa7a  a9 48           lda #$48
4594  fa7c  8d 03 db        sta ipcia+ddrb   6526 cia: data direct.reg.b (ddrb)
4595  fa7f  a9 01           lda #$01
4596  fa81  0d 01 de        ora tri1+pb      6525 triport 1: port reg. b
4597  fa84  8d 01 de        sta tri1+pb      6525 triport 1: port reg. b
4598  fa87  60              rts
4599  fa88
4600  fa88
4601  fa88  ===> ram-test, zeiger init. <===
```

```
zeile adr.  obj.-code   source-code

4602  fa88
4603  fa88  a9 00     ramtas lda #$00
4604  fa8a  aa               tax
4605  fa8b  95 02     px1    sta usrpok,x     jmp code für 'usr'-routine
4606  fa8d  ea               nop              *** vom control-file eingefügt ***
4607  fa8e  9d 00 02         sta buf,x        basic input-puffer -$2ff
4608  fa91  9d f8 02         sta buf+$f8,x    basic input-puffer +$f8
4609  fa94  e8               inx
4610  fa95  d0 f4            bne px1
4611  fa97  a9 01            lda #$01
4612  fa99  85 01            sta i6509        6509 indirection register
4613  fa9b  8d 5a 03         sta memstr+2     bank anfang des benutzer speichers
4614  fa9e  8d 54 03         sta lowadr+2     bank anfang des system speichers
4615  faa1  a9 02            lda #$02
4616  faa3  8d 58 03         sta memstr       addrl anfang des benutzer speichers
4617  faa6  8d 52 03         sta lowadr       addrl anfang des system speichers
4618  faa9  c6 01            dec i6509        6509 indirection register
4619  faab  e6 01     sizlop inc i6509        6509 indirection register
4620  faad  a5 01            lda i6509        6509 indirection register
4621  faaf  c9 0f            cmp #$0f
4622  fab1  f0 24            beq size
4623  fab3  a0 02            ldy #$02
4624  fab5  b1 93     siz100 lda (sal),y      addrl akt. load/store adresse
4625  fab7  aa               tax
4626  fab8  a9 55            lda #$55
4627  faba  91 93            sta (sal),y      addrl akt. load/store adresse
4628  fabc  b1 93            lda (sal),y      addrl akt. load/store adresse
4629  fabe  c9 55            cmp #$55
4630  fac0  d0 15            bne size
4631  fac2  0a               asl a
4632  fac3  91 93            sta (sal),y      addrl akt. load/store adresse
4633  fac5  b1 93            lda (sal),y      addrl akt. load/store adresse
4634  fac7  c9 aa            cmp #$aa
4635  fac9  d0 0c            bne size
4636  facb  8a               txa
4637  facc  91 93            sta (sal),y      addrl akt. load/store adresse
4638  face  c8               iny
4639  facf  d0 e4            bne siz100
4640  fad1  e6 94            inc sah          addrh akt. load/store adresse
4641  fad3  d0 e0            bne siz100
4642  fad5  f0 d4            beq sizlop
4643  fad7  a6 01     size   ldx i6509        6509 indirection register
4644  fad9  ca               dex
4645  fada  8a               txa
4646  fadb  a2 ff            ldx #$ff
4647  fadd  a0 fd            ldy #$fd
4648  fadf  8d 57 03         sta hiadr+2      bank ende des system speichers
4649  fae2  8c 56 03         sty hiadr+1      addrh ende des system speichers
4650  fae5  8e 55 03         stx hiadr        addrl ende des system speichers
4651  fae8  a0 fa            ldy #$fa
4652  faea  18               clc
4653  faeb  20 78 fb         jsr memtop       höchste ram-adr. lesen/schreiben
4654  faee  c6 a8            dec ribuf+2      bank rs232 eingabe-puffer
4655  faf0  c6 a5            dec tape1+2      bank kassettenpuffer
4656  faf2  a9 5d            lda #$5d
4657  faf4  8d 6a 03         sta itape        addrl indir. cassette
4658  faf7  a9 fe            lda #$fe
4659  faf9  8d 6b 03         sta itape+1      addrh indir. cassette
```

```
zeile adr.  obj.-code   source-code

4660  fafc  60              rts
4661  fafd
4662  fafd
4663  fafd  ===> tabelle der standard-vektoren <===
4664  fafd
4665  fafd  e9 fb     jmptab .word yirq       interrupt-routine (irq-vector)
4666  faff  21 ee            .word timb       monitor-einsprung 'break-vektor'
4667  fb01  aa fc            .word panic      nmi-routine (nmi-vector)
4668  fb03  bf f6            .word nopen      log. file öffnen/befehl ausgeben
4669  fb05  ed f5            .word nclose     logisches file schliessen
4670  fb07  49 f5            .word nchkin     eingabe-kanal öffnen
4671  fb09  a3 f5            .word nckout     ausgabe-kanal öffnen
4672  fb0b  a6 f6            .word nclrch     alle kanäle schliessen
4673  fb0d  9c f4            .word nbasin     eingabe 1 char vom aktiven kanal in
                                              ac (chn-vector)
4674  fb0f  ee f4            .word nbsout     ausgabe 1 char auf akt. kanal
4675  fb11  6b f9            .word nstop      stop-taste überprüfen
4676  fb13  3d f4            .word ngetin     eingabe 1 char vom aktiven kanal in
                                              ac (queue-vector)
4677  fb15  7f f6            .word nclall     alle logischen files schliessen
4678  fb17  46 f7            .word nload      einlesen vom log. file
4679  fb19  4c f8            .word nsave      abspeichern auf log. file
4680  fb1b  77 ee            .word s0         monitor-befehl lesen (einsprung von
                                              usrcmd)
4681  fb1d  1f e0            .word jescrt     bearbeitung von escape-funktionen
4682  fb1f  1f e0            .word jescrt     bearbeitung von escape-funktionen
4683  fb21  74 f2            .word nsecnd     sekundäradr. nach listen ausgeben
4684  fb23  80 f2            .word ntksa      akt. sekundäradr. auf iec-bus
                                              ausgeben
4685  fb25  0a f3            .word nrbyte     byte von iec nach ac
4686  fb27  97 f2            .word nciout     ac auf iec-bus (mit eof-flag)
4687  fb29  ab f2            .word nuntlk     untalk auf iec-bus ausgeben
4688  fb2b  af f2            .word nunlsn     unlisten auf iec-bus ausgeben
4689  fb2d  34 f2            .word nlistn     listen auf iec-bus ausgeben
4690  fb2f  30 f2            .word ntalk      talk auf iec-bus ausgeben
4691  fb31  6c 04 03  tabend jmp (nminv)      ind. sprung nmi-vector (von $fffa)
4692  fb34
4693  fb34                   .end
4694  fb34                   .lib kernal3
```

```
zeile adr.  obj.-code   source-code


4696  fb34
4697  fb34  ===> adr. des filenamens eintragen <===
4698  fb34
4699  fb34  85 9d     setnam sta fnlen        länge akt. filename
4700  fb36  b5 00            lda e6509,x     6509 execution register
4701  fb38  85 90            sta fnadr       addrl akt. filename
4702  fb3a  b5 01            lda i6509,x     6509 indirection register
4703  fb3c  85 91            sta fnadr+1     addrh akt. filename
4704  fb3e  b5 02            lda usrpok,x    jmp code für 'usr'-routine
4705  fb40  85 92            sta fnadr+2     bank akt. filename
4706  fb42  60               rts
4707  fb43
4708  fb43
4709  fb43  ===> log., geräte- und sekundäradr. aktualisieren <===
4710  fb43
4711  fb43  85 9e     setlfs sta la          akt. logische filenummer
4712  fb45  86 9f            stx fa          akt. primäradresse (geräte-nummer)
4713  fb47  84 a0            sty sa          akt. sekundäradresse
4714  fb49  60               rts
4715  fb4a
4716  fb4a
4717  fb4a  ===> system-status lesen/schreiben <===
4718  fb4a
4719  fb4a  90 18     readst bcc storst      ac in status bzw rs232-status
4720  fb4c  a5 9f            lda fa          akt. primäradresse (geräte-nummer)
4721  fb4e  c9 02            cmp #$02
4722  fb50  d0 0b            bne readss      fehler-status setzen
4723  fb52  ad 7a 03         lda rsstat      rs232: akt. rs232-status
4724  fb55  48               pha
4725  fb56  a9 00            lda #$00
4726  fb58  f0 11            beq statxt      stack in rs232-status
4727  fb5a
4728  fb5a
4729  fb5a  ===> fehlerflag und status setzen <===
4730  fb5a
4731  fb5a  8d 61 03  setmsg sta msgflg      message flag
4732  fb5d
4733  fb5d
4734  fb5d  ===> fehler-status setzen <===
4735  fb5d
4736  fb5d  a5 9c     readss lda status      i/o operation status
4737  fb5f
4738  fb5f
4739  fb5f  ===> 'ora' ac in status <===
4740  fb5f
4741  fb5f  05 9c     udst   ora status      i/o operation status
4742  fb61  85 9c            sta status      i/o operation status
4743  fb63  60               rts
4744  fb64
4745  fb64
4746  fb64  ===> ac in status bzw rs232-status <===
4747  fb64
4748  fb64  48        storst pha
4749  fb65  a5 9f            lda fa          akt. primäradresse (geräte-nummer)
4750  fb67  c9 02            cmp #$02
4751  fb69  d0 05            bne storss      stack in status
4752  fb6b
4753  fb6b
```

```
zeile adr.  obj.-code   source-code

4754  fb6b  ===> stack in rs232-status <===
4755  fb6b
4756  fb6b  68        statxt pla
4757  fb6c  8d 7a 03         sta rsstat       rs232: akt. rs232-status
4758  fb6f  60               rts
4759  fb70
4760  fb70
4761  fb70  ===> stack in status <===
4762  fb70
4763  fb70  68        storss pla
4764  fb71  85 9c            sta status       i/o operation status
4765  fb73  60               rts
4766  fb74
4767  fb74
4768  fb74  ===> timeout iec-bus ein/aus <===
4769  fb74
4770  fb74  8d 5e 03  settmo sta timout       ieee timeout enable flag
4771  fb77  60               rts
4772  fb78
4773  fb78
4774  fb78  ===> höchste ram-adr. lesen/schreiben <===
4775  fb78
4776  fb78  90 09     memtop bcc settop       xr,yr,ac in zeiger höchste ram-adr.
4777  fb7a  ad 5d 03         lda memsiz+2     bank ende des benutzer speichers
4778  fb7d  ae 5b 03         ldx memsiz       addrl ende des benutzer speichers
4779  fb80  ac 5c 03         ldy memsiz+1     addrh ende des benutzer speichers
4780  fb83
4781  fb83
4782  fb83  ===> xr,yr,ac in zeiger höchste ram-adr. <===
4783  fb83
4784  fb83  8e 5b 03  settop stx memsiz       addrl ende des benutzer speichers
4785  fb86  8c 5c 03         sty memsiz+1     addrh ende des benutzer speichers
4786  fb89  8d 5d 03         sta memsiz+2     bank ende des benutzer speichers
4787  fb8c  60               rts
4788  fb8d
4789  fb8d
4790  fb8d  ===> niedrigste ram-adr. lesen/schreiben <===
4791  fb8d
4792  fb8d  90 09     membot bcc setbot       xr,yr,ac in zeiger anfang ram-adr.
4793  fb8f  ad 5a 03         lda memstr+2     bank anfang des benutzer speichers
4794  fb92  ae 58 03         ldx memstr       addrl anfang des benutzer speichers
4795  fb95  ac 59 03         ldy memstr+1     addrh anfang des benutzer speichers
4796  fb98
4797  fb98
4798  fb98  ===> xr,yr,ac in zeiger anfang ram-adr. <===
4799  fb98
4800  fb98  8e 58 03  setbot stx memstr       addrl anfang des benutzer speichers
4801  fb9b  8c 59 03         sty memstr+1     addrh anfang des benutzer speichers
4802  fb9e  8d 5a 03         sta memstr+2     bank anfang des benutzer speichers
4803  fba1  60               rts
4804  fba2
4805  fba2
4806  fba2  ===> standard-vektoren für jsr und interrupts einrichten <===
4807  fba2
4808  fba2  a2 fd     restor ldx #$fd
4809  fba4  a0 fa            ldy #$fa
4810  fba6  a9 0f            lda #$0f
4811  fba8  18               clc
```

```
zeile adr.  obj.-code   source-code

4812  fba9
4813  fba9
4814  fba9  ===> lesen/schreiben der akt. system-ram-vektoren <===
4815  fba9
4816  fba9  86 93     vector stx sal          addrl akt. load/store adresse
4817  fbab  84 94            sty sah          addrh akt. load/store adresse
4818  fbad  a6 01            ldx i6509        6509 indirection register
4819  fbaf  85 01            sta i6509        6509 indirection register
4820  fbb1  90 0a            bcc vect50
4821  fbb3  a0 33            ldy #$33
4822  fbb5  b9 00 03  vect20 lda cinv,y       addrl irq vector
4823  fbb8  91 93            sta (sal),y      addrl akt. load/store adresse
4824  fbba  88               dey
4825  fbbb  10 f8            bpl vect20
4826  fbbd  a0 33     vect50 ldy #$33
4827  fbbf  b1 93     vect60 lda (sal),y      addrl akt. load/store adresse
4828  fbc1  99 00 03         sta cinv,y       addrl irq vector
4829  fbc4  88               dey
4830  fbc5  10 f8            bpl vect60
4831  fbc7  86 01            stx i6509        6509 indirection register
4832  fbc9  60               rts
4833  fbca
4834  fbca
4835  fbca  8e f8 03  vreset stx evect        addrl warmstart-vector
4836  fbcd  8c f9 03         sty evect+1      addrh warmstart-vector
4837  fbd0  a9 5a            lda #$5a
4838  fbd2  8d fb 03         sta evect+3      warmstart-variable
4839  fbd5  60               rts
4840  fbd6
4841  fbd6
4842  fbd6  ===> interrupt-routine (irq) <===
4843  fbd6
4844  fbd6  48        nirq   pha
4845  fbd7  8a               txa
4846  fbd8  48               pha
4847  fbd9  98               tya
4848  fbda  48               pha
4849  fbdb  ba               tsx
4850  fbdc  bd 04 01         lda stack+4,x    6509 cpu-stack
4851  fbdf  29 10            and #$10
4852  fbe1  d0 03            bne brkirq       sprung auf brk-vector (im irq)
4853  fbe3  6c 00 03         jmp (cinv)       addrl irq vector
4854  fbe6
4855  fbe6
4856  fbe6  6c 02 03  brkirq jmp (cbinv)      sprung auf brk-vector (im irq)
4857  fbe9
4858  fbe9
4859  fbe9  ===> interrupt-routine (irq-vector) <===
4860  fbe9
4861  fbe9  a5 01     yirq   lda i6509        6509 indirection register
4862  fbeb  48               pha
4863  fbec  d8               cld
4864  fbed  ad 07 de         lda tri1+air     6525 triport 1: active irq reg.
4865  fbf0  d0 03            bne irq000
4866  fbf2  4c a2 fc         jmp prendn       irq-ende, stack nach i6509,yr,xr,ac
4867  fbf5
4868  fbf5
4869  fbf5  c9 10     irq000 cmp #$10
```

```
zeile adr.  obj.-code   source-code

4870  fbf7  f0 03           beq irq002
4871  fbf9  4c 5b fc        jmp irq100
4872  fbfc
4873  fbfc
4874  fbfc  ad 01 dd  irq002 lda acia+srsn    6551 acia: reset-write und status
                                              read-reg.
4875  fbff  aa              tax
4876  fc00  29 60           and #$60
4877  fc02  a8              tay
4878  fc03  4d 7b 03        eor dcdsr         rs232: letzter dcd/dsr wert
4879  fc06  f0 0d           beq irq004
4880  fc08  98              tya
4881  fc09  8d 7b 03        sta dcdsr         rs232: letzter dcd/dsr wert
4882  fc0c  0d 7a 03        ora rsstat        rs232: akt. rs232-status
4883  fc0f  8d 7a 03        sta rsstat        rs232: akt. rs232-status
4884  fc12  4c 9f fc        jmp irq900
4885  fc15
4886  fc15
4887  fc15  8a        irq004 txa
4888  fc16  29 08           and #$08
4889  fc18  f0 26           beq irq010
4890  fc1a  ac 7d 03        ldy ridbe         rs232: eingabe end zeiger
4891  fc1d  c8              iny
4892  fc1e  cc 7c 03        cpy ridbs         rs232: eingabe start zeiger
4893  fc21  d0 04           bne irq005
4894  fc23  a9 08           lda #$08
4895  fc25  d0 13           bne irq007
4896  fc27  8c 7d 03  irq005 sty ridbe        rs232: eingabe end zeiger
4897  fc2a  88              dey
4898  fc2b  a6 a8           ldx ribuf+2       bank rs232 eingabe-puffer
4899  fc2d  86 01           stx i6509         6509 indirection register
4900  fc2f  ae 01 dd        ldx acia+srsn     6551 acia: reset-write und status
                                              read-reg.
4901  fc32  ad 00 dd        lda acia+drsn     6551 acia: data reg. write =
                                              transmit, read = receive
4902  fc35  91 a6           sta (ribuf),y     addrl rs232 eingabe-puffer
4903  fc37  8a              txa
4904  fc38  29 07           and #$07
4905  fc3a  0d 7a 03  irq007 ora rsstat        rs232: akt. rs232-status
4906  fc3d  8d 7a 03        sta rsstat        rs232: akt. rs232-status
4907  fc40  ad 01 dd  irq010 lda acia+srsn    6551 acia: reset-write und status
                                              read-reg.
4908  fc43  29 10           and #$10
4909  fc45  f0 11           beq irq090
4910  fc47  ad 02 dd        lda acia+cdr      6551 acia: command-reg.
4911  fc4a  29 0c           and #$0c
4912  fc4c  c9 04           cmp #$04
4913  fc4e  d0 08           bne irq090
4914  fc50  a9 f3           lda #$f3
4915  fc52  2d 02 dd        and acia+cdr      6551 acia: command-reg.
4916  fc55  8d 02 dd        sta acia+cdr      6551 acia: command-reg.
4917  fc58  4c 9f fc  irq090 jmp irq900
4918  fc5b
4919  fc5b
4920  fc5b  c9 08     irq100 cmp #$08
4921  fc5d  d0 0a           bne irq110
4922  fc5f  ad 0d db        lda ipcia+icr     6526 cia: irq control reg. (icr)
4923  fc62  58              cli
```

```
zeile adr.  obj.-code   source-code

4924  fc63  20 48 fd         jsr ipserv
4925  fc66  4c 9f fc         jmp irq900
4926  fc69
4927  fc69
4928  fc69  58        irq110 cli
4929  fc6a  c9 04            cmp #$04
4930  fc6c  d0 0c            bne irq200
4931  fc6e  ad 0d dc         lda cia+icr      6526 cia: irq control reg. (icr)
4932  fc71  0d 69 03         ora alarm        flag für irq von 6526
4933  fc74  8d 69 03         sta alarm        flag für irq von 6526
4934  fc77  4c 9f fc         jmp irq900
4935  fc7a
4936  fc7a
4937  fc7a  c9 02     irq200 cmp #$02
4938  fc7c  d0 03            bne irq300       tastatur lesen, ti stellen im irq
4939  fc7e  4c 9f fc         jmp irq900
4940  fc81
4941  fc81
4942  fc81  ===> tastatur lesen, ti stellen im irq <===
4943  fc81
4944  fc81  20 13 e0  irq300 jsr jkey         tastatur lesen/in puffer schreiben
4945  fc84  20 79 f9         jsr udtim        system-zeit aktualisieren
4946  fc87  ad 01 de         lda tri1+pb      6525 triport 1: port reg. b
4947  fc8a  10 09            bpl irq310
4948  fc8c  a0 00            ldy #$00
4949  fc8e  8c 75 03         sty cas1         cassette switchflag
4950  fc91  09 40            ora #$40
4951  fc93  d0 07            bne irq320
4952  fc95  ac 75 03  irq310 ldy cas1         cassette switchflag
4953  fc98  d0 05            bne irq900
4954  fc9a  29 bf            and #$bf
4955  fc9c  8d 01 de  irq320 sta tri1+pb      6525 triport 1: port reg. b
4956  fc9f  8d 07 de  irq900 sta tri1+air     6525 triport 1: active irq reg.
4957  fca2
4958  fca2
4959  fca2  ===> irq-ende, stack nach i6509,yr,xr,ac <===
4960  fca2
4961  fca2  68        prendn pla
4962  fca3  85 01            sta i6509        6509 indirection register
4963  fca5
4964  fca5
4965  fca5  ===> irq-ende, stack nach yr,xr,ac <===
4966  fca5
4967  fca5  68        prend  pla
4968  fca6  a8               tay
4969  fca7  68               pla
4970  fca8  aa               tax
4971  fca9  68               pla
4972  fcaa
4973  fcaa
4974  fcaa  ===> nmi-routine (nmi-vector) <===
4975  fcaa
4976  fcaa  40        panic  rti
4977  fcab
4978  fcab
4979  fcab  ad 00 08  iprqst lda ipb+ipccmd   coprozessor-puffer: command
4980  fcae  29 7f            and #$7f
4981  fcb0  a8               tay
```

```
zeile adr.  obj.-code   source-code

4982  fcb1  20 21 fe          jsr getpar
4983  fcb4  a9 04             lda #$04
4984  fcb6  2d 01 db          and ipcia+prb    6526 cia: periph.data reg. b (prb)
4985  fcb9  d0 f0             bne iprqst
4986  fcbb  a9 08             lda #$08
4987  fcbd  0d 01 db          ora ipcia+prb    6526 cia: periph.data reg. b (prb)
4988  fcc0  8d 01 db          sta ipcia+prb    6526 cia: periph.data reg. b (prb)
4989  fcc3  ea                nop
4990  fcc4  ad 01 db          lda ipcia+prb    6526 cia: periph.data reg. b (prb)
4991  fcc7  aa                tax
4992  fcc8  29 04             and #$04
4993  fcca  f0 0c             beq ipr100
4994  fccc  8a                txa
4995  fccd  49 08             eor #$08
4996  fccf  8d 01 db          sta ipcia+prb    6526 cia: periph.data reg. b (prb)
4997  fcd2  8a                txa
4998  fcd3  ea                nop
4999  fcd4  ea                nop
5000  fcd5  ea                nop
5001  fcd6  d0 d3             bne iprqst
5002  fcd8  a9 ff     ipr100  lda #$ff
5003  fcda  8d 02 db          sta ipcia+ddra   6526 cia: data direct.reg.a (ddra)
5004  fcdd  ad 00 08          lda ipb+ipccmd   coprozessor-puffer: command
5005  fce0  8d 00 db          sta ipcia+pra    6526 cia: periph.data reg. a (pra)
5006  fce3  20 08 fe          jsr frebus
5007  fce6  ad 01 db          lda ipcia+prb    6526 cia: periph.data reg. b (prb)
5008  fce9  29 bf             and #$bf
5009  fceb  8d 01 db          sta ipcia+prb    6526 cia: periph.data reg. b (prb)
5010  fcee  09 40             ora #$40
5011  fcf0  58                cli
5012  fcf1  ea                nop
5013  fcf2  ea                nop
5014  fcf3  ea                nop
5015  fcf4  8d 01 db          sta ipcia+prb    6526 cia: periph.data reg. b (prb)
5016  fcf7  20 ee fd          jsr waithi
5017  fcfa  a9 00             lda #$00
5018  fcfc  8d 02 db          sta ipcia+ddra   6526 cia: data direct.reg.a (ddra)
5019  fcff  20 f6 fd          jsr acklo
5020  fd02  20 e6 fd          jsr waitlo
5021  fd05  a0 00             ldy #$00
5022  fd07  f0 1d             beq ipr250
5023  fd09  a9 ff     ipr200  lda #$ff
5024  fd0b  8d 02 db          sta ipcia+ddra   6526 cia: data direct.reg.a (ddra)
5025  fd0e  b9 05 08          lda ipb+ipcdat,y coprozessor-puffer: data puffer
5026  fd11  8d 00 db          sta ipcia+pra    6526 cia: periph.data reg. a (pra)
5027  fd14  20 ff fd          jsr ackhi
5028  fd17  20 ee fd          jsr waithi
5029  fd1a  a9 00             lda #$00
5030  fd1c  8d 02 db          sta ipcia+ddra   6526 cia: data direct.reg.a (ddra)
5031  fd1f  20 f6 fd          jsr acklo
5032  fd22  20 e6 fd          jsr waitlo
5033  fd25  c8                iny
5034  fd26  cc 03 08  ipr250  cpy ipb+ipcin    coprozessor-puffer: # input bytes
5035  fd29  d0 de             bne ipr200
5036  fd2b  a0 00             ldy #$00
5037  fd2d  f0 13             beq ipr350
5038  fd2f  20 ff fd  ipr300  jsr ackhi
5039  fd32  20 ee fd          jsr waithi
```

```
zeile adr.  obj.-code   source-code

5040  fd35  ad 00 db        lda ipcia+pra     6526 cia: periph.data reg. a (pra)
5041  fd38  99 05 08        sta ipb+ipcdat,y coprozessor-puffer: data puffer
5042  fd3b  20 f6 fd        jsr acklo
5043  fd3e  20 e6 fd        jsr waitlo
5044  fd41  c8              iny
5045  fd42  cc 04 08  ipr350 cpy ipb+ipcout   coprozessor-puffer: # output bytes
5046  fd45  d0 e8           bne ipr300
5047  fd47  60              rts
5048  fd48
5049  fd48
5050  fd48  a9 00     ipserv lda #$00
5051  fd4a  8d 02 db        sta ipcia+ddra    6526 cia: data direct.reg.a (ddra)
5052  fd4d  ad 00 db        lda ipcia+pra     6526 cia: periph.data reg. a (pra)
5053  fd50  8d 00 08        sta ipb+ipccmd    coprozessor-puffer: command
5054  fd53  29 7f           and #$7f
5055  fd55  a8              tay
5056  fd56  20 21 fe        jsr getpar
5057  fd59  98              tya
5058  fd5a  0a              asl a
5059  fd5b  a8              tay
5060  fd5c  b9 10 08        lda ipjtab,y      coprozessor-jump tabelle
5061  fd5f  8d 01 08        sta ipb+ipcjmp    coprozessor-puffer: jump addrl
5062  fd62  c8              iny
5063  fd63  b9 10 08        lda ipjtab,y      coprozessor-jump tabelle
5064  fd66  8d 02 08        sta ipb+ipcjmp+1 coprozessor-puffer: jump addrh
5065  fd69  20 ff fd        jsr ackhi
5066  fd6c  20 e6 fd        jsr waitlo
5067  fd6f  a0 00           ldy #$00
5068  fd71  cc 03 08  ips100 cpy ipb+ipcin    coprozessor-puffer: # input bytes
5069  fd74  f0 15           beq ips200
5070  fd76  20 f6 fd        jsr acklo
5071  fd79  20 ee fd        jsr waithi
5072  fd7c  ad 00 db        lda ipcia+pra     6526 cia: periph.data reg. a (pra)
5073  fd7f  99 05 08        sta ipb+ipcdat,y coprozessor-puffer: data puffer
5074  fd82  20 ff fd        jsr ackhi
5075  fd85  20 e6 fd        jsr waitlo
5076  fd88  c8              iny
5077  fd89  d0 e6           bne ips100
5078  fd8b  2c 00 08  ips200 bit ipb+ipccmd   coprozessor-puffer: command
5079  fd8e  30 33           bmi ips500
5080  fd90  a9 fd           lda #$fd
5081  fd92  48              pha
5082  fd93  a9 98           lda #$98
5083  fd95  48              pha
5084  fd96  6c 01 08        jmp (ipb+ipcjmp) coprozessor-puffer: jump addrl
5085  fd99
5086  fd99
5087  fd99  20 f6 fd  ips300 jsr acklo
5088  fd9c  a0 00           ldy #$00
5089  fd9e  f0 1d           beq ips350
5090  fda0  20 ee fd  ips310 jsr waithi
5091  fda3  a9 ff           lda #$ff
5092  fda5  8d 02 db        sta ipcia+ddra    6526 cia: data direct.reg.a (ddra)
5093  fda8  b9 05 08        lda ipb+ipcdat,y coprozessor-puffer: data puffer
5094  fdab  8d 00 db        sta ipcia+pra     6526 cia: periph.data reg. a (pra)
5095  fdae  20 ff fd        jsr ackhi
5096  fdb1  20 e6 fd        jsr waitlo
5097  fdb4  a9 00           lda #$00
```

```
zeile adr.  obj.-code   source-code

5098  fdb6  8d 02 db        sta ipcia+ddra   6526 cia: data direct.reg.a (ddra)
5099  fdb9  20 f6 fd        jsr acklo
5100  fdbc  c8              iny
5101  fdbd  cc 04 08  ips350 cpy ipb+ipcout   coprozessor-puffer: # output bytes
5102  fdc0  d0 de            bne ips310
5103  fdc2  60        ips400 rts
5104  fdc3
5105  fdc3
5106  fdc3  a9 fd     ips500 lda #$fd
5107  fdc5  48               pha
5108  fdc6  a9 ce            lda #$ce
5109  fdc8  48               pha
5110  fdc9  20 11 fe         jsr getbus
5111  fdcc  6c 01 08         jmp (ipb+ipcjmp) coprozessor-puffer: jump addrl
5112  fdcf
5113  fdcf
5114  fdcf  20 08 fe  ips600 jsr frebus
5115  fdd2  ad 04 08         lda ipb+ipcout   coprozessor-puffer: # output bytes
5116  fdd5  8d 03 08         sta ipb+ipcin    coprozessor-puffer: # input bytes
5117  fdd8  8d 00 08         sta ipb+ipccmd   coprozessor-puffer: command
5118  fddb  a9 00            lda #$00
5119  fddd  8d 04 08         sta ipb+ipcout   coprozessor-puffer: # output bytes
5120  fde0  20 ab fc         jsr iprqst
5121  fde3  4c c2 fd         jmp ips400
5122  fde6
5123  fde6
5124  fde6  ad 01 db  waitlo lda ipcia+prb    6526 cia: periph.data reg. b (prb)
5125  fde9  29 04            and #$04
5126  fdeb  d0 f9            bne waitlo
5127  fded  60               rts
5128  fdee
5129  fdee
5130  fdee  ad 01 db  waithi lda ipcia+prb    6526 cia: periph.data reg. b (prb)
5131  fdf1  29 04            and #$04
5132  fdf3  f0 f9            beq waithi
5133  fdf5  60               rts
5134  fdf6
5135  fdf6
5136  fdf6  ad 01 db  acklo  lda ipcia+prb    6526 cia: periph.data reg. b (prb)
5137  fdf9  29 f7            and #$f7
5138  fdfb  8d 01 db         sta ipcia+prb    6526 cia: periph.data reg. b (prb)
5139  fdfe  60               rts
5140  fdff
5141  fdff
5142  fdff  a9 08     ackhi  lda #$08
5143  fe01  0d 01 db         ora ipcia+prb    6526 cia: periph.data reg. b (prb)
5144  fe04  8d 01 db         sta ipcia+prb    6526 cia: periph.data reg. b (prb)
5145  fe07  60               rts
5146  fe08
5147  fe08
5148  fe08  ad 01 de  frebus lda tri1+pb      6525 triport 1: port reg. b
5149  fe0b  29 ef            and #$ef
5150  fe0d  8d 01 de         sta tri1+pb      6525 triport 1: port reg. b
5151  fe10  60               rts
5152  fe11
5153  fe11
5154  fe11  ad 01 db  getbus lda ipcia+prb    6526 cia: periph.data reg. b (prb)
5155  fe14  29 02            and #$02
```

```
zeile adr.  obj.-code   source-code

5156  fe16  f0 f9           beq getbus
5157  fe18  ad 01 de        lda tri1+pb      6525 triport 1: port reg. b
5158  fe1b  09 10           ora #$10
5159  fe1d  8d 01 de        sta tri1+pb      6525 triport 1: port reg. b
5160  fe20  60              rts
5161  fe21
5162  fe21
5163  fe21  b9 10 09  getpar lda ipptab,y    coprozessor-parameter tabelle
5164  fe24  48              pha
5165  fe25  29 0f           and #$0f
5166  fe27  8d 03 08        sta ipb+ipcin    coprozessor-puffer: # input bytes
5167  fe2a  68              pla
5168  fe2b  4a              lsr a
5169  fe2c  4a              lsr a
5170  fe2d  4a              lsr a
5171  fe2e  4a              lsr a
5172  fe2f  8d 04 08        sta ipb+ipcout   coprozessor-puffer: # output bytes
5173  fe32  60              rts
5174  fe33
5175  fe33
5176  fe33  a2 ff     ipcgo  ldx #$ff
5177  fe35  86 01            stx i6509        6509 indirection register
5178  fe37  ad 01 de         lda tri1+pb      6525 triport 1: port reg. b
5179  fe3a  29 ef            and #$ef
5180  fe3c  8d 01 de         sta tri1+pb      6525 triport 1: port reg. b
5181  fe3f  ea               nop
5182  fe40  ad 01 db         lda ipcia+prb    6526 cia: periph.data reg. b (prb)
5183  fe43  6a               ror a
5184  fe44  b0 01            bcs ipcgx
5185  fe46  60               rts
5186  fe47
5187  fe47
5188  fe47  a9 00     ipcgx  lda #$00
5189  fe49  78               sei
5190  fe4a  8d 01 db         sta ipcia+prb    6526 cia: periph.data reg. b (prb)
5191  fe4d  a9 40            lda #$40
5192  fe4f  ea               nop
5193  fe50  ea               nop
5194  fe51  ea               nop
5195  fe52  ea               nop
5196  fe53  8d 01 db         sta ipcia+prb    6526 cia: periph.data reg. b (prb)
5197  fe56  58               cli
5198  fe57  4c 57 fe  iploop jmp iploop
5199  fe5a
5200  fe5a
5201  fe5a  6c 6a 03  xtape  jmp (itape)      addrl indir. cassette
5202  fe5d
5203  fe5d
5204  fe5d  ===> pla, pla, ausg. 'i/0 error: 5' <===
5205  fe5d
5206  fe5d  68        nocass pla
5207  fe5e  68               pla
5208  fe5f  4c 45 f9         jmp error5       ausgabe 'i/o error: 5'
5209  fe62
5210  fe62
5211  fe62  ===> 'stah'-vektor nach 'sah'-vektor und bank# nach i6509 <===
5212  fe62
5213  fe62  a5 9a     rd300  lda stah         addrh anfang akt. load/store
```

```
zeile adr.  obj.-code   source-code

5214  fe64  85 94           sta sah          addrh akt. load/store adresse
5215  fe66  a5 99           lda stal         addrl anfang akt. load/store
5216  fe68  85 93           sta sal          addrl akt. load/store adresse
5217  fe6a  a5 9b           lda stas         bank anfang akt. load/store
5218  fe6c  85 95           sta sas          bank akt. load/store adresse
5219  fe6e  85 01           sta i6509        6509 indirection register
5220  fe70  60              rts
5221  fe71
5222  fe71
5223  fe71  ===> vektor 'sal' - 'eal' <===
5224  fe71
5225  fe71  38        cmpste sec
5226  fe72  a5 93           lda sal          addrl akt. load/store adresse
5227  fe74  e5 96           sbc eal          addrl ende akt. load/store
5228  fe76  a5 94           lda sah          addrh akt. load/store adresse
5229  fe78  e5 97           sbc eah          addrh ende akt. load/store
5230  fe7a  a5 95           lda sas          bank akt. load/store adresse
5231  fe7c  e5 98           sbc eas          bank ende akt. load/store
5232  fe7e  60              rts
5233  fe7f
5234  fe7f
5235  fe7f  e6 93     incsal inc sal          addrl akt. load/store adresse
5236  fe81  d0 0e           bne incr20
5237  fe83  e6 94           inc sah          addrh akt. load/store adresse
5238  fe85  d0 0a           bne incr20
5239  fe87  e6 95           inc sas          bank akt. load/store adresse
5240  fe89  a5 95           lda sas          bank akt. load/store adresse
5241  fe8b  85 01           sta i6509        6509 indirection register
5242  fe8d  a9 02           lda #$02
5243  fe8f  85 93           sta sal          addrl akt. load/store adresse
5244  fe91  60        incr20 rts
5245  fe92
5246  fe92
5247  fe92  ===> filename 'bank x' in ac, offset yr <===
5248  fe92
5249  fe92  a6 01     fnadry ldx i6509        6509 indirection register
5250  fe94  a5 92           lda fnadr+2      bank akt. filename
5251  fe96  85 01           sta i6509        6509 indirection register
5252  fe98  b1 90           lda (fnadr),y    addrl akt. filename
5253  fe9a  86 01           stx i6509        6509 indirection register
5254  fe9c  60              rts
5255  fe9d
5256  fe9d
5257  fe9d  85 01     txjmp  sta i6509        6509 indirection register
5258  fe9f  8a               txa
5259  fea0  18               clc
5260  fea1  69 02            adc #$02
5261  fea3  90 01            bcc txjmp1
5262  fea5  c8               iny
5263  fea6  aa        txjmp1 tax
5264  fea7  98               tya
5265  fea8  48               pha
5266  fea9  8a               txa
5267  feaa  48               pha
5268  feab  20 19 ff         jsr ipinit       'ipoint' auf $0100, yr=$ff
5269  feae  a9 fe            lda #$fe
5270  feb0  91 ac            sta (ipoint),y   addrl ram indirekt-pointer
5271  feb2  08        exsub  php
```

```
zeile adr.  obj.-code   source-code

5272  feb3  78              sei
5273  feb4  48              pha
5274  feb5  8a              txa
5275  feb6  48              pha
5276  feb7  98              tya
5277  feb8  48              pha
5278  feb9  20 19 ff        jsr ipinit        'ipoint' auf $0100, yr=$ff
5279  febc  a8              tay
5280  febd  a5 00           lda e6509         6509 execution register
5281  febf  20 2a ff        jsr putag         ac nach (ipoint),y  yr=yr-1
5282  fec2  a9 04           lda #$04
5283  fec4  a2 ff           ldx #$ff
5284  fec6  20 24 ff        jsr putaxs        xr, ac nach (ipoint),y yr=yr-2
5285  fec9  ba              tsx
5286  feca  bd 05 01        lda stack+5,x     6509 cpu-stack
5287  fecd  38              sec
5288  fece  e9 03           sbc #$03
5289  fed0  48              pha
5290  fed1  bd 06 01        lda stack+6,x     6509 cpu-stack
5291  fed4  e9 00           sbc #$00
5292  fed6  aa              tax
5293  fed7  68              pla
5294  fed8  20 24 ff        jsr putaxs        xr, ac nach (ipoint),y yr=yr-2
5295  fedb  98              tya
5296  fedc  38        excomm sec
5297  fedd  e9 04           sbc #$04
5298  fedf  8d ff 01        sta buf-1         basic input-puffer -1
5299  fee2  a8              tay
5300  fee3  a2 04           ldx #$04
5301  fee5  68        exsu10 pla
5302  fee6  c8              iny
5303  fee7  91 ac           sta (ipoint),y    addrl ram indirekt-pointer
5304  fee9  ca              dex
5305  feea  d0 f9           bne exsu10
5306  feec  ac ff 01        ldy buf-1         basic input-puffer -1
5307  feef  a9 2d           lda #$2d
5308  fef1  a2 ff           ldx #$ff
5309  fef3  20 24 ff        jsr putaxs        xr, ac nach (ipoint),y yr=yr-2
5310  fef6  68              pla
5311  fef7  68              pla
5312  fef8  ba              tsx
5313  fef9  8e ff 01        stx buf-1         basic input-puffer -1
5314  fefc  98              tya
5315  fefd  aa              tax
5316  fefe  9a              txs
5317  feff  a5 01           lda i6509         6509 indirection register
5318  ff01  4c f6 ff        jmp kgbye         ac in 6509 execution register
5319  ff04
5320  ff04
5321  ff04  ea              nop
5322  ff05
5323  ff05
5324  ff05  ===> rückkehr-adresse wenn 'rts' <===
5325  ff05
5326  ff05  08        excrts php
5327  ff06  08              php
5328  ff07  78              sei
5329  ff08  48              pha
```

```
zeile adr.  obj.-code   source-code


5330  ff09  8a              txa
5331  ff0a  48              pha
5332  ff0b  98              tya
5333  ff0c  48              pha
5334  ff0d  ba              tsx
5335  ff0e  bd 06 01        lda stack+6,x     6509 cpu-stack
5336  ff11  85 01           sta i6509         6509 indirection register
5337  ff13  20 19 ff        jsr ipinit        'ipoint' auf $0100, yr=$ff
5338  ff16  4c dc fe        jmp excomm
5339  ff19
5340  ff19
5341  ff19  ===> 'ipoint' auf $0100, yr=$ff <===
5342  ff19
5343  ff19  a0 01     ipinit ldy #$01
5344  ff1b  84 ad            sty ipoint+1     addrh ram indirekt-pointer
5345  ff1d  88               dey
5346  ff1e  84 ac            sty ipoint       addrl ram indirekt-pointer
5347  ff20  88               dey
5348  ff21  b1 ac            lda (ipoint),y   addrl ram indirekt-pointer
5349  ff23  60               rts
5350  ff24
5351  ff24
5352  ff24  ===> xr, ac nach (ipoint),y yr=yr-2 <===
5353  ff24
5354  ff24  48        putaxs pha
5355  ff25  8a               txa
5356  ff26  91 ac            sta (ipoint),y   addrl ram indirekt-pointer
5357  ff28  88               dey
5358  ff29  68               pla
5359  ff2a
5360  ff2a
5361  ff2a  ===> ac nach (ipoint),y  yr=yr-1 <===
5362  ff2a
5363  ff2a  91 ac     putag  sta (ipoint),y   addrl ram indirekt-pointer
5364  ff2c  88               dey
5365  ff2d  60               rts
5366  ff2e
5367  ff2e
5368  ff2e  68               pla
5369  ff2f  a8               tay
5370  ff30  68               pla
5371  ff31  aa               tax
5372  ff32  68               pla
5373  ff33  28               plp
5374  ff34  60               rts
5375  ff35
5376  ff35
5377  ff35  08               php
5378  ff36  6c fa ff         jmp (hanmi)      6509 hardware-nmi-vektor
5379  ff39
5380  ff39
5381  ff39  00               brk
5382  ff3a  ea               nop
5383  ff3b  60               rts
5384  ff3c
5385  ff3c
5386  ff3c  58               cli
5387  ff3d  60               rts
```

```
zeile adr.  obj.-code   source-code

5388  ff3e
5389  ff3e
5390  ff3e
5391  ff3e
5392  ff3e                    * = *+$2e
5393  ff6c
5394  ff6c
5395  ff6c  4c 9d fe  knwsys jmp txjmp        transfer-of-execution jumper
5396  ff6f
5397  ff6f
5398  ff6f  4c ca fb  kvrese jmp vreset       netz ein/aus reset-vector
5399  ff72
5400  ff72
5401  ff72  4c 33 fe  kipcgo jmp ipcgo        mon-befehl 'z' umsch. auf
                                              coprozessor
5402  ff75
5403  ff75
5404  ff75  4c 22 e0  kfunky jmp jfunky       vector für funktionstasten
5405  ff78
5406  ff78
5407  ff78  4c ab fc  kipcrq jmp iprqst       vector für ipc-anforderung
5408  ff7b
5409  ff7b
5410  ff7b  4c fb f9  kioini jmp ioinit       initialisierung ein-/ausgabe
5411  ff7e
5412  ff7e
5413  ff7e  4c 04 e0  kcint  jmp jcint        initialisierung bildschirm
5414  ff81
5415  ff81
5416  ff81  4c 00 f4  kalloc jmp tttop        allocation-routine
5417  ff84
5418  ff84
5419  ff84  4c a9 fb  kvecto jmp vector       ein-/ausgabe-vectoren lesen /
                                              schreiben
5420  ff87
5421  ff87
5422  ff87  4c a2 fb  kresto jmp restor       standard ein-/ausgabe-vectoren
                                              setzen
5423  ff8a
5424  ff8a
5425  ff8a  4c 60 f6  klkups jmp lkupsa       geräteadr. über sekundäradr. suchen
5426  ff8d
5427  ff8d
5428  ff8d  4c 78 f6  klkupl jmp lkupla       geräte- sekundäradr. über log. file#
                                              suchen
5429  ff90
5430  ff90
5431  ff90  4c 5a fb  kstmsg jmp setmsg       meldung des operat.syst. ausgeben
5432  ff93
5433  ff93
5434  ff93  6c 24 03  ksecnd jmp (isecnd)     auf iec-bus sekundäradr. nach listen
                                              ausgeben
5435  ff96
5436  ff96
5437  ff96  6c 26 03  ktksa  jmp (itksa)      auf iec-bus sekundäradr. nach talk
                                              ausgeben
5438  ff99
5439  ff99
```

```
zeile adr.  obj.-code   source-code

5440  ff99  4c 78 fb  kmemtp jmp memtop       höchste speichergrenze
                                              lesen/schreiben
5441  ff9c
5442  ff9c
5443  ff9c  4c 8d fb  kmembt jmp membot       unterste speichergrenze
                                              lesen/schreiben
5444  ff9f
5445  ff9f
5446  ff9f  4c 13 e0  kscnky jmp jkey         tastatur abfragen
5447  ffa2
5448  ffa2
5449  ffa2  4c 74 fb  ksetmo jmp settmo       zeitüberwachung iec-bus ein
5450  ffa5
5451  ffa5
5452  ffa5  6c 28 03  kacptr jmp (iacptr)     ein byte von iec-bus nach accu
5453  ffa8
5454  ffa8
5455  ffa8  6c 2a 03  kciout jmp (iciout)     ein byte aus accu auf iec-bus
5456  ffab
5457  ffab
5458  ffab  6c 2c 03  kuntlk jmp (iuntlk)     untalk auf iec-bus ausgeben
5459  ffae
5460  ffae
5461  ffae  6c 2e 03  kunlsn jmp (iunlsn)     unlisten auf iec-bus ausgeben
5462  ffb1
5463  ffb1
5464  ffb1  6c 30 03  klistn jmp (ilistn)     listen auf iec-bus ausgeben
5465  ffb4
5466  ffb4
5467  ffb4  6c 32 03  ktalk  jmp (italk)      talk auf iec-bus ausgeben
5468  ffb7
5469  ffb7
5470  ffb7  4c 4a fb  kreast jmp readst       ein/ausgabe-status lesen / schreiben
5471  ffba
5472  ffba
5473  ffba  4c 43 fb  kstlfs jmp setlfs       log. filenummer, geräte- und
                                              sek.adr. eintragen
5474  ffbd
5475  ffbd
5476  ffbd  4c 34 fb  kstnam jmp setnam       länge und adr. des filenamens
                                              eintragen
5477  ffc0
5478  ffc0
5479  ffc0  6c 06 03  kopen  jmp (iopen)      log. file öffnen/befehl ausgeben
5480  ffc3
5481  ffc3
5482  ffc3  6c 08 03  kclose jmp (iclose)     logisches file schliessen
5483  ffc6
5484  ffc6
5485  ffc6  6c 0a 03  kchkin jmp (ichkin)     eingabekanal öffnen
5486  ffc9
5487  ffc9
5488  ffc9  6c 0c 03  kckout jmp (ickout)     ausgabekanal öffnen
5489  ffcc
5490  ffcc
5491  ffcc  6c 0e 03  kclrch jmp (iclrch)     ein-/ausgabekanal schliessen
5492  ffcf
5493  ffcf
```

```
zeile adr.  obj.-code   source-code

5494  ffcf  6c 10 03  kbasin jmp (ibasin)     eingabe 1 char vom aktiven kanal in
                                               ac (chn-vector)
5495  ffd2
5496  ffd2
5497  ffd2  6c 12 03  kbsout jmp (ibsout)     ausgabe 1 char auf akt. kanal
5498  ffd5
5499  ffd5
5500  ffd5  6c 1a 03  kload  jmp (iload)      einlesen vom log. file
5501  ffd8
5502  ffd8
5503  ffd8  6c 1c 03  ksave  jmp (isave)      abspeichern auf log. file
5504  ffdb
5505  ffdb
5506  ffdb  4c 0e f9  ksttim jmp settim       interne uhr stellen
5507  ffde
5508  ffde
5509  ffde  4c e6 f8  krdtim jmp rdtim        interne uhr ablesen
5510  ffe1
5511  ffe1
5512  ffe1  6c 14 03  kstop  jmp (istop)      stop-taste lesen
5513  ffe4
5514  ffe4
5515  ffe4  6c 16 03  kgetin jmp (igetin)     eingabe 1 char vom aktiven kanal in
                                               ac (queue-vector)
5516  ffe7
5517  ffe7
5518  ffe7  6c 18 03  kclall jmp (iclall)     alle logischen files schliessen
5519  ffea
5520  ffea
5521  ffea  4c 79 f9  kudtim jmp udtim        interne uhr bedienen
5522  ffed
5523  ffed
5524  ffed  4c 10 e0  kscror jmp jscror       bildschirm
5525  fff0
5526  fff0
5527  fff0  4c 19 e0  kplot  jmp jplot        cursor-koordinaten lesen/schreiben
5528  fff3
5529  fff3
5530  fff3  4c 1c e0          jmp jiobas      basisadr. i/o-gerät nach xr/yr
5531  fff6
5532  fff6
5533  fff6  ===> ac in 6509 execution register <===
5534  fff6
5535  fff6  85 00     kgbye  sta e6509        6509 execution register
5536  fff8  60               rts
5537  fff9
5538  fff9
5539  fff9  ===> prüfbyte rom $e000-$ffff <===
5540  fff9
5541  fff9  01        checkf .byte $01
5542  fffa
5543  fffa
5544  fffa  ===> 6509 hardware-nmi-vektor <===
5545  fffa
5546  fffa  31 fb     hanmi  .word tabend     ind. sprung nmi-vector (von $fffa)
5547  fffc
5548  fffc
5549  fffc  ===> 6509 hardware-reset-vektor <===
```

```
zeile adr.  obj.-code   source-code


5550  fffc
5551  fffc  97 f9      hares  .word start       system reset routine
5552  fffe
5553  fffe
5554  fffe  ===> 6509 hardware-irq-vektor <===
5555  fffe
5556  fffe  d6 fb      hairq  .word nirq        interrupt-routine (irq)
5557  0000
5558  0000                    .end
5559  0000
5560  0000                    .end



fehler in pass 1 = 0000
fehler in pass 2 = 0000

assemblierung ende
```

5. TEIL

CROSSREFERENZLISTE 2

CBM 610/710

OPERATINGSYSTEM

```
irq010 $fc40  4889  4907
irq090 $fc58  4909  4913  4917
irq100 $fc5b  4871  4920
irq110 $fc69  4921  4928
irq200 $fc7a  4930  4937
irq300 $fc81  4938  4944
irq310 $fc95  4947  4952
irq320 $fc9c  4951  4955
irq900 $fc9f  4884  4917  4925  4934  4939  4953  4956
isave  $031c    96  5503
isecnd $0324   100  5434
istop  $0314    92  5512
italk  $0332   107  5467
itape  $036a   124  4657  4659  5201
itksa  $0326   101  5437
iunlsn $032e   105  5461
iuntlk $032c   104  5458
jcint  $e004   574  2640  4530  4533  5413
jescrt $e01f   592  4681  4682
jfunky $e022   594  5404
jiobas $e01c   590  5530
jkey   $e013   584  4944  5446
jloop5 $e00a   578  3683
jlp2   $e007   576  3621
jltlk  $f643  3879  3926
jmonon $e000   570
jmptab $fafd  4665
jmvcur $e016   586
jplot  $e019   588  5527
jprt   $e00d   580  3727
jscror $e010   582  5524
jx110  $f5f6  3880  3886
jx115  $f613  3897  3901
jx120  $f61a  3895  3904
jx150  $f61d  3890  3892  3894  3900  3905
jx151  $f61e  3902  3906
jx160  $f63c  3909  3917
jx310  $f551  3782  3786
jx312  $f57c  3798  3806
jx315  $f580  3793  3808
jx316  $f583  3796  3809
jx320  $f586  3788  3790  3807  3812  3828
jx330  $f58a  3791  3817
jx340  $f598  3820  3825
jx350  $f59b  3822  3826
jx600  $f646  3927  3930
jx750  $f6af  4001  4003
jz100  $f650  3786  3839  3886  3937  3955
kacptr $ffa5  3699  4132  4142  4167  5452
kalloc $ff81   658  5416
kbasin $ffcf  2708  2757  3112  3216  3240  3630  5494
kbsout $ffd2  1809  1823  1825  1828  1843  1845  1851  1853  1872  2847
              2859  2865  2867  2884  3155  3243  3255  3299  4225  4441
              5497
kchkin $ffc6  3251  5485
kcint  $ff7e  5413
kciout $ffa8  3734  4082  4279  4281  4286  5455
kckout $ffc9  3236  5488
kclall $ffe7  5518
kclose $ffc3  3267  3991  5482
kclrch $ffcc  2645  2664  3264  4433  4453  5491
keword $e7a6  1808  1887
```

```
l14     $f0e3  3116  3119
l15     $f0f3  3091  3102  3105  3125
l2      $f068  3058  3071  3082
l20     $f0da  3112  3114
l3      $f06a  3059  3068
l5      $f07e  3054  3060  3069  3080  3086  3100
l8      $f090  3062  3079
l9      $f097  3082  3088
la      $009e    24  3938  3957  4019  4032  4711
lat     $0334   108  3911  3912  3929  3937  3989  4033
ld      $f04a  2798  2800  3043
ld10    $f75a  4110  4114
ld100   $f810  4116  4205
ld110   $f834  4224  4228
ld115   $f83f  4214  4218  4222  4229
ld180   $f813  4199  4206
ld20    $f75d  4109  4113
ld22    $f766  4115  4119
ld25    $f771  4122  4126
ld30    $f791  4138  4142
ld40    $f7ba  4152  4159  4172  4196
ld410   $f849  4234  4236
ld45    $f7c8  4163  4167
ld60    $f7f0  4179  4183  4187
ld64    $f800  4189  4191  4195
ldtb1   $ebcb   716  1250  2465
ldtb2   $ebb2   714  1248  2436
ldtnd   $0360   117  3907  3908  3910  3926  3949  3981  3994  4025  4031
lf      $000a   473
line01  $e894  2010  2012
linelp  $e891  2011  2021
linrts  $e66c  1685  1688
lintmp  $0398   139   790   800   842
linz0   $d000   164  2436  2465
linz1   $d050   165  2437  2466
linz10  $d320   174  2446  2475
linz11  $d370   175  2447  2476
linz12  $d3c0   176  2448  2477
linz13  $d410   177  2449  2478
linz14  $d460   178  2450  2479
linz15  $d4b0   179  2451  2480
linz16  $d500   180  2452  2481
linz17  $d550   181  2453  2482
linz18  $d5a0   182  2454  2483
linz19  $d5f0   183  2455  2484
linz2   $d0a0   166  2438  2467
linz20  $d640   184  2456  2485
linz21  $d690   185  2457  2486
linz22  $d6e0   186  2458  2487
linz23  $d730   187  2459  2488
linz24  $d780   188  2460  2489
linz3   $d0f0   167  2439  2468
linz4   $d140   168  2440  2469
linz5   $d190   169  2441  2470
linz6   $d1e0   170  2442  2471
linz7   $d230   171  2443  2472
linz8   $d280   172  2444  2473
linz9   $d2d0   173  2445  2474
list1   $f236  3310  3316  3390
list2   $f268  3325  3337
listky  $e6ff  1791  1797
listlp  $e701  1798  1856
```