COMMODORE

Commodore Sachbuchreihe Band 6

COMMODORE plus/4 ROM – LISTING

COMMODORE Plus/4
ROM – LISTING

Commodore Sachbuchreihe Band 6

COMMODORE Plus/4 ROM – LISTING

Commodore

Nachdruck, auch auszugsweise, nur mit schriftlicher Genehmigung von COMMODORE.
Artikel-Nr. 584000/10.84 Änderungen vorbehalten. ISBN-Nr. 3-89133-006-5

Commodore 16 , 116 , plus/4 Speicherbelegung (Systemadressen)

Hex. Adr.	Dez. Adr.	Label	Beschreibung
$0000	0	PDIR	Datenrichtungsreg. des 7501
$0001	1	PORT	Ein-/ Ausgabe-Port des 7501
$0002	2	SRCHTK	Flag für Schleifen
$0003-0004	3-4	ZPVEC1	Neue Startadresse (Renumber)
$0005-0006	5-6	ZPVEC2	Schrittweite (Renumber)
$0007	7	CHARAC	Gesuchtes Zeichen
$0008	8	ENDCHR	Flag für Anführungszeichen-Modus
$0009	9	TRMPOS	TAB - Spaltenzähler
$000A	10	VERCK	Flag: 0 = Load, 1 = Verify
$000B	11	COUNT	Zeiger für Eingabepuffer, Anzahl der Elemente
$000C	12	DIMFLG	Flag für Standard-DIM
$000D	13	VALTYP	Datentyp: $FF=String, $00=Numerisch
$000E	14	INTFLG	Datentyp: $80=Integer, $00=Fließkomma
$000F	15	DORES	Flag für DATA/LIST
$0010	16	SUBFLG	Flag: Element/FNx-Flag
$0011	17	INPFLG	Flag: $00=INPUT, $40=GET, $98=READ
$0012	18	TANSGN	Flag: Vorzeichen des ATN
$0013	19	CHANNL	Flag: Input-Prompt
$0014-0015	20-21	LINNUM	
$0016	22	TEMPPT	Zeiger auf temporären Stringstapel
$0017-0018	23-24	LASTPT	Letzter temporärer Stringvektor
$0019-0021	25-33	TEMPST	Stapel für temporäre Strings
$0022-0023	34-35	INDEX1	Bereich für Hilfszeiger 1
$0024-0025	36-37	INDEX2	Bereich für Hilfszeiger 2
$0026	38	RESHO	Bereich für Produkt bei Multiplikation
$0027	39	RESMOH	" " "
$0028	40	RESMO	" " "
$0029	41	RESLO	" " "
$002A	42		" " "
$002B-002C	43-44	TXTTAB	Zeiger auf Basic-Anfang
$002D-002E	45-46	VARTAB	Zeiger auf Variablen-Anfang
$002F-0030	47-48	ARYTAB	Zeiger auf Beginn der Arrays
$0031-0032	49-50	STREND	Zeiger auf Ende der Arrays (+1)
$0033-0034	51-52	FRETOP	Zeiger auf Stringspeicher
$0035-0036	53-54	FRESPC	Hilfszeiger für Strings
$0037-0038	55-56	MEMSIZ	Zeiger auf Speichergrenze
$0039-003A	57-58	CURLIN	Laufende Basiczeilennummer
$003B-003C	59-60	TXTPTR	Textpointer
$003D-003E	61-62	FNDPNT	Zeiger auf Basic-Statement für CONT
$003F-0040	63-64	DATLIN	Nummer der aktuellen DATA-Zeile
$0041-0042	65-66	DATPTR	Adresse des aktuellen DATA-Elementes
$0043-0044	67-68	INPPTR	Sprungvektor für INPUT
$0045-0046	69-70	VARNAM	Aktueller Variablenname
$0047-0048	71-72	VARPNT	Adresse der aktuellen Variablen
$0049-004A	73-74	FORPNT	Variablenzeiger für FOR...NEXT
$004B-004c	75-76	OPPTR	Zwischenspeicher für Basic-Zeiger
$004D	77	OPMASK	Akkumulator für Vergleichssymbole
$004E-004F	78-79	DEFPNT	Arbeitsbereich (Zeiger usw.)
$0050-0051	80-81	DSCPNT	" "

Commodore 16 , 116 , plus/4 Speicherbelegung (Systemadressen)

Hex. Adr.	Dez. Adr.	Label	Beschreibung
$0052	82		Arbeitsbereich (Zeiger usw.)
$0053	83	HELPER	
$0054-0056	84-86	JMPER	Sprungvektor für Funktionen
$0057-0060	87-96	TEMPF1	Bereich für numerische Operationen
$0061	97	FACEXP	Fließkomma-Akkumulator 1 (FAC): Exponent
$0062-0065	98-101	FACHO	Fließkomma-Akkumulator 1 (FAC): Mantisse
$0066	102	FACSGN	Fließkomma-Akkumulator 1 (FAC): Vorzeichen
$0067	103	SGNFLG	Zeiger für Polynom-Auswertung
$0068	104	BITS	Fließkomma-Akkumulator 1 Überlauf
$0069-006E	105-110	ARGEXP	Fließkomma-Akkumulator 2 Exponent usw.
$006F	111	ARISGN	Vorzeichenvergleich Akku 1 mit Akku 2
$0070	112	FACOV	Fließkomma-Akkumulator 1 niederwertige Stelle (Rundung)
$0071-0072	113-114	FBUFPT	Kassettenpuffer Länge/Zeiger
$0073-0074	115-116	AUTINC	Automatisches Zeileninkrement 0=AUS
$0075	117	MVDFLG	Grafik-Flag
$0076-0078	118-120	KEYNUM	Arbeitsbereich
$0079-007B	121-123	DSDESC	Darstellung von DS$
$007C-007D	124-125	TOS	Basic-Pseudo-Stack-Pointer
$007E-008F	126-143	TMPTON	Arbeitsbereich (Sound)
$0090	144	STATUS	Statuswort ST
$0091	145	STKEY	Flag: Stoptaste / RVS-taste
$0094	148	C3PO	Flag: Serieller Bus - Ausgabe Zeichenpuffer
$0095	149	BSOUR	Zeichenspeicher - Serieller Bus
$0096	150	RSAV	Zwischenspeicher
$0097	151	LDTND	Anzahl der geöffneten Files
$0098	152	DFLTN	Eingabegerät (Normalwert: 0)
$0099	153	DFLTO	Ausgabegerät (Normalwert: 3)
$009A	154	MSGFLG	Flag: $80=Direkt- $00=Programm-Modus
$009B-009C	155-156	SAL	Zeiger für Kassettenpuffer / Scrolling
$009D-009E	157-158	EAL	Zeiger auf Ende des Programms
$009F-00A0	159-160	T1	Temporärer Datenspeicher
$00A1-00A2	161-162	T2	" "
$00A3-00A5	163-165	TIME	Echtzeit-Uhr
$00A6	166	R2D2	Register für seriellen Bus
$00A7	167	TPBYTE	Register für Kassettenroutine
$00A8	168	BSOUR1	Register für seriellen Bus
$00A9	169	FPVERR	Temporärer Farb-Vektor
$00AA	170	DCOUNT	Register für Kassettenroutine
$00AB	171	FNLEN	Anzahl der Zeichen im Filenamen
$00AC	172	LA	Aktuelle logische File-Nummer
$00AD	173	SA	Aktuelle Sekundär-Adresse
$00AE	174	FA	Aktuelle Geräte-Nummer
$00AF-00B0	175-176	FNADR	Zeiger auf Filenamen
$00B1	177	ERRSUM	
$00B2-00B3	178-179	STAL	I/O Startadresse

Commodore 16 , 116 , plus/4 Speicherbelegung (Systemadressen)

Hex. Adr.	Dez. Adr.	Label	Beschreibung
$00B4-00B5	180-181	MEMUSS	Basis-Ladeadresse
$00B6-00B7	182-183	TAPEBS	Ladeendadresse (Kassette)
$00B8-00B9	184-185		
$00BA-00BB	186-187	WRBASE	Zeiger auf Zeichen im Kassettenpuffer
$00BC-00BD	188-189	IMPARM	
$00BE-00BF	190-191	FETPTR	Register für Long-Fetch-Routine
$00c0-00C1	192-193	SEDSAL	Register für Scrolling
$00C2	194	RVS	RVS-Flag (Bildschirm)
$00C3	195	INDX	Zeiger: Ende der Zeile (Input)
$00C4-00C5	196-197	LSXP	Cursorposition (Input), Reihe/Spalte
$00C6	198	SFDX	Tastaturabfrage ($40=keine Taste)
$00C7	199	CRSW	Flag: Eingabe Bildschirm/Tastatur
$00C8-00C9	200-201	PNT	Zeiger auf Bildschirmzeile
$00CA	202	PNTR	Cursorposition in akt. Bildschirmzeile
$00CB	203	QTSW	Flag: Anführungszeichen-Modus ($00 = nein)
$00CC	204	SEDT1	Länge der aktuellen Bildschirmzeile
$00CD	205	TBLX	Zeile, in der sich der Cursor befindet
$00CE	206	DATAX	Letztes Zeichen (I/O)
$00CF	207	INSRT	Anzahl der Zeichen (Insert-Modus)
$00D0-00D7	208-215		Reserviert für Sprachsynthesizer
$00D8-00E8	216-232		Reserviert für Anwendungssoftware
$00E9	233	CIRSEG	Arbeitsbereich
$00EA-00EB	234-235	USER	Bildschirm-Editor (Farbe)
$00EC-00EE	236-238	KEYTAB	Arbeitsbereich (Bildschirm)
$00EF	239	NDX	Anzahl der Zeichen im Tastaturpuffer
$00F0	240	STPFLG	
$00F1-00F2	241-242	TO	Register für Monitor
$00F3	243	CHRPTR	
$00F4	244	BUFEND	
$00F5	245	CHKSUM	Register für Prüfsumme
$00F6	246	LENGTH	
$00F7	247	PASS	
$00F8	248	TYPE	
$00F9	249	USEKDY	
$00FA	250	XSTOP	Register für X bei Stoptastentest
$00FB	251	CURBNK	Aktuelle Bank-Konfiguration
$00FC	252	XON	
$00FD	253	XOFF	
$00FE	254	SEDT2	Arbeitsregister (Editor)
$00FF	255	LOFBUF	
$0100-01FF	256-511		Prozessorstack
$0200-0258	512-600	BUF	Eingabepuffer
$0259-025C	601-604		Basic-Puffer
$025D-02AC	605-684		Basic-/DOS-Arbeitsbereich
$025D	605	XCNT	DOS-Schleifenzähler
$025E-026D	606-621	FNBUFR	Bereich für Filenamen
$026E	622	DOSF1L	1. Filename (Länge)
$026F	623	DOSDS1	DOS (Laufwerk 1)
$0270-0271	624-625	DOSF1A	1. Filename (Adresse)

Commodore 16 , 116 , plus/4 Speicherbelegung (Systemspeicher)

Hex. Adr.	Dez. Adr.	Label	Beschreibung
$0272	626	DOSF2L	2. Filename (Länge)
$0273	627	DOSFA	DOS (Laufwerk 2)
$0274-0275	628-629	DOSF2FA	2. Filename (Adresse)
$0276	630	DOSLA	DOS logische Adresse
$0277	631	DOSFA	DOS (Geräteadresse)
$0278	632	DOSSA	DOS (Sekundäradresse)
$0279-027A	633-634	DODDID	DOS (Disketten-ID)
$027B	635	DIDCHK	ID-Flag
$027C	636	DOSSTR	DOS (Ausgabepuffer)
$027D-02AC	637-684	DOSSPC	DOS (Arbeitsbereich)
$02AD-02B0	685-688		Grafik-Cursor
$02B1-02B4	689-692		Grafik-Cursor (Register)
$02B5-02CB	693-715		Grafik-Register
$02CC-02E3	716-739		Print-Using, Grafik-Arbeitsbereich
$02E4	740	CHRPAG	High-Byte-Adresse des Charakter-ROM
$02EB	747	TRCFLG	Trace-Flag
$02F0	752	VTEMP4	Anzahl der Grafik-Parameter
$02F1	753	VTEMP5	Parameter: Relativ (1), Absolut (0)
$02F2-02F3	754-755	ADRAY1	Fließkomma-Vektor
$02F4-02F5	756-757	ADRAY2	Integer-Vektor
$0300-0301	768-769	IERROR	Sprungvektor: Fehlermeldung
$0302-0303	770-771	IMAIN	Sprungvektor: Basic-Warmstart
$0304-0305	772-773	ICRNCH	Sprungvektor: Token-Generierung
$0306-0307	774-775	IQPLOP	Sprungvektor: Keywort erzeugen
$0308-0309	776-777	IGONE	Sprungvektor: Hauptschleife
$030A-030B	778-779	IEVAL	Sprungvektor: Eval
$030C-030D	780-781	IESCLK	Sprungvektor: Token-Generierung (User)
$030E-030F	782-783	IESCPR	Sprungvektor: Keywort erzeugen
$0310-0311	784-785	IESCEX	Sprungvektor: User-Token bearbeiten
$0312-0313	786-787	ITIME	Sprungvektor: Interrupt (Uhr)
$0314-0315	788-789	CINV	Sprungvektor: Interrupt
$0316-0317	790-791	CBINV	Sprungvektor: Break Interrupt
$0318-0319	792-793	IOPEN	Sprungvektor: Open
$031A-031B	794-795	ICLOSE	Sprungvektor: Close
$031C-031D	796-797	ICHKIN	Sprungvektor: Kanal für Eingabe öffnen
$031E-031F	798-799	ICKOUT	Sprungvektor: Kanal für Ausgabe öffnen
$0320-0321	800-801	ICLRCH	Sprungvektor: I/O zurücksetzen
$0322-0323	802-803	IBASIN	Sprungvektor: Input
$0324-0325	804-805	IBSOUT	Sprungvektor: Ausgabe
$0326-0327	806-807	ISTOP	Sprungvektor: Abfrage der Stoptaste
$0328-0329	808-809	IGETIN	Sprungvektor: Getin-Routine
$032A-032B	810-811	ICLALL	Sprungvektor: Schließen aller Files
$032C-032D	812-813	USRCMD	Sprungvektor: Monitor Break
$032E-032F	814-815	ILOAD	Sprungvektor: Lade-Routine
$0330-0331	816-817	ISAVE	Sprungvektor:Save-Routine
$0332-03F2	818-1010	TAPBUF	Kassettenpuffer
$03F3-03F4	1011-1012	WRLEN	Datenzähler (Write)
$03F5-03F6	1013-1014	RDCNT	Datenzähler (Read)
$03F7-0436	1015-1078	INPQUE	RS-232-Input-Puffer (64 Byte)
$0437-0454	1079-1108	ESTAKL	

Commodore 16 , 116 , plus/4 Speicherbelegung (Systemadressen)

Hex. Adr.	Dez. Adr.	Label	Beschreibung
$0455-0472	1109-1138	ESTAKH	
$0473-0478	1139-1144	CHRGET	Chrget-Routine
$0455-0484	1145-1156	CHRGOT	Chrgot-Routine
$0494-04DC	1172-1244		Bankswitching-Routinen
$04E7-04EA	1255-1258	PUFILL	Print-Using-Parameter
$04EF-04F6	1263-1270	ERRNUM	Trap- und Error-Flags
$0500-0502	1280-1282	USRPOK	USR-Sprungbefehl
$0503-0507	1283-1287	RNDX	Startwert für RND
$0508	1288	DEJAVU	Flag für Kalt- oder Warmstart
$0509-0512	1289-1298	LAT	Tabelle der logische Filenummern
$0513-051C	1299-1308	FAT	Tabelle der Gerätenummern
$051D-0526	1309-1318	SAT	Tabelle der Sekundäradressen
$0527-0530	1319-1328	KEYD	Tastaturpuffer
$0531-0532	1329-1330	MEMSTR	Basic-Anfang
$0533-0534	1331-1332	MSIZ	Basic-Ende
$0539	1337	TPTR	Zeiger: Kassettenpuffer
$053A	1338	FLTYPE	Typ des Kassettenfiles
$0540	1344	RPTFLG	Tasten-Wiederholfunktion $80=alle,$40=keine,$00=nur DEL,CUR,SPC
$0541-0542	1345-1346		Zähler für Wiederholfunktion
$0543	1347	SHFLAG	Shift-Flag
$0544	1348	LSTSHF	
$0545-0546	1349-1350	KEYLOG	Sprungvektor: Keylog
$0547	1351	MODE	Text-/Grafik-Flag
$0548	1352	AUTODN	Scroll-Flag
$054B-0551	1355-1372	FORMAT	Cpu-Arbeitsbereich
$0552-0557	1362-1367	PCH	Cpu-Register
$0558-055C	1368-1372		Cpu-Arbeitsbereich
$055D	1373	KYNDX	Zähler für Funktionstasten
$055E	1374	KEYIDX	Zeiger: Text, Funktionstasten
$055F-05E6	1375-1510	KEYBUF	Speicher für Funktionstasten
$05EC-06EB	1516-1571	SAVRAM	1 Page, Bankingroutinen
$05EC-05EF	1516-1519	PAT	Adresstabelle
$05F0-05F1	1520-1521	LNGJMP	Long-Jump (Adresse)
$05F2	1522	FETARG	Long-Jump (Akkumulator)
$05F3	1523	FETXRG	Long-Jump (X-Register)
$05F4	1524	FETSRG	Long-Jump (Status-Register)
$05F5-065D	1525-1629	AREAS	RAM-Bereich für Bankswitching
$065E-06EB	1630-1771	ASPECH	RAM-Bereich für Sprach-Synthesizer
$06EC-07AF	1792-1967	STKTOP	Basic-Pseudo-Stack
$07B0-07CC	1968-1996	WROUT	Kassetten-Arbeitsbereich
$07CD-07D0	1997-2000		RS-232-Arbeitsbereich
$07D1	2001	INQFPT	RS-232: Pointer auf Anf. Eingabepuffer
$07D2	2002	INQRPT	RS-232: Pointer auf Ende Eingabepuffer
$07D3	2003	INQCNT	Anzahl der Zeichen im Eingabepuffer
$07E5	2021	SCBOT	Bildschirm: Unterer Rand
$07E6	2022	SCTOP	Bildschirm: Oberer Rand
$07E7	2023	SCLF	Bildschirm: Linker Rand
$07E8	2024	SCRT	Bildschirm: Rechter Rand
$07EA	2026	INSFLG	$FF=Automatisches Einfügen

Commodore 16 , 116 , plus/4 Speicherbelegung (Systemspeicher)

Hex. Adr.	Dez. Adr.	Label	Beschreibung
$07EE-07F1	2030-2033	BITABL	
$07F2	2034	SAREG	A-Reg. retten bei Sys-Kommando
$07F3	2035	SXREG	X-Reg. retten bei Sys-Kommando
$07F4	2036	SYREG	Y-Reg. retten bei Sys-Kommando
$07F5	2037	SPREG	Status-Reg. retten bei Sys-Kom.
$07F6	2038	LSTX	Register für Tastaturabfrage
$07F7	2039	STPDSB	Flag: CTRL-S:$00=offen,$06=gesperrt
$07F8	2040	RAMROM	RAM-ROM-Umschaltung für Monitor 0=ROM, $80=RAM
$07FC	2044	LSEM	Motorsteuerung
$0800-0BFF	2048-3071	TEDATR	Farbspeicher (Text)
$0C00-0FE7	3072-4071	TEDSCN	Bildschirmspeicher (Text)
Grafik-Bildschirm wurde noch nicht aufgerufen:			
$1000-FCFF	4096-64767	BASBGN	Basic-RAM bei plus/4
$1000-3FFF	4096-16383	BASBGN	Basic-RAM bei C-16/116
Grafik-Bildschirm wurde einmal aufgerufen:			
$4000-FCFF	16384-64776		Basic-RAM bei plus/4
$1000-17FF	4096-6143		Basic-RAM bei c-16/116
$1800-1BFF	6144-7167		Luminanztabelle (Grafik)
$1C00-1FFF	7168-8191		Farbtabelle (Grafik)
$2000-3FFF	8192-16383		Grafikbildschirm

```
32768 ;------------------------------  B A S I C  S T A R T
32768
32768 a32768  jmp a32793             ; ---> basic kaltstart
32771
32771 a32771  jmp a32778             ; ---> basic warmstart
32774
32774 a32774  .by 0
32775 a32775  .by 'cbm'
32778 ;
32778 ;------------------------------  W A R M S T A R T
32778
32778 a32778  jsr a65484             ; ---> clrchn
32781         jsr a35544             ; ---> clear stack
32784         sta *a19               ; i/o kanal = tastatur
32786         jsr a51145             ; ---> graphik abschalten
32789         cli
32790 a32790  jmp a34430             ; ---> ready.
32793
32793 ;------------------------------  K A L T S T A R T
32793
32793 a32793  jsr a33047             ; ---> vektoren einrichten
32796         jsr a32814             ; ---> basic reset
32799         jsr a32962             ; ---> logo ausgeben
32802         jsr a64756             ; ---> moduln initialisieren
32805         ldx #251               ; lässt die oberen 4 stackbytes
32807         txs                    ; f zeilenkopf frei
32808         bne a32790             ; immer ->
32810
32810 ;------------------------------  V E K T O R E N
32810
32810 a32810  .si a39025             ; (a754) gk -> int wandlung
32812         .si a38001             ; (a756) int -> gk wandlung
32814
32814 ;------------------------------  B A S I C - R E S E T
32814
32814 a32814  lda #76                ; jmp-kode
32816         sta *a84               ; funktionen-aufruf
32818         sta a1280              ; usr-jump
32821         lda #l,a39196          ; adresse von 'illegal quantity'
32823         ldy #h,a39196
32825         sta a1281              ; l,usr-adresse
32828         sty a1282              ; h,...
32831         ldx #a32814-a32810-1
32833 a32833  lda a32810,x           ; 2 vektoren einrichten
32836         sta a754,x
32839         dex
32840         bpl a32833
32842         ldx #a33109-a33058-1
32844 a32844  lda a33059-1,x         ; chrget-routine und allgemeine
32847         sta a1139-1,x          ; routine f lda (...),y aus ram
32850         dex                    ; nach unten kopieren
32851         bne a32844                                          en
32853         stx *a104              ; vorstelle f register-verschieb
32855         stx *a19               ; i/o-kanal := tastatur          stack
32857         stx *a24               ; h,zeiger auf letzten string im
32859         stx a747               ; trace abschalten
32862         stx a4096              ; anfang des programmspeichers
32865         txa                    ; = 0
32866         ldx #3
```

```
32868 a32868  sta *a115-1,x           ; auto und graphik-speicher
32870         sta a742-1,x            ; flags für graphik
32873         dex
32874         bne a32868
32876         nop
32877         stx a1283               ; 1. stelle der zufallszahl
32880         inx                     ; = 1
32881         stx a512-3              ; link-adresse f basic-zeilen
32884         stx a512-4              ; vorbesetzen
32887         ldx a1339               ; farbe & helligkeit für zeichen
32890         stx *a134
32892         ldx #%01100110          ; farbe & helligkeit für
32894         stx *a133               ; multicolor-graphik
32896         ldx #a25                ; anfang des string-stacks
32898         stx *a22                ; string-stackzeiger
32900         ldx #l,a4096+1          ; l,basic text bereich
32902         ldy #h,a4096+1          ; h,...
32904         stx *a43                ; l,progr anfangszeiger
32906         sty *a44                ; h,...
32908         ldx #5
32910         stx *a34                ; zähler f kopierroutine
32912         lda #h,53248            ; basisadresse des zeichensatzes 1
32914         sta a740                ; für char speichern
32917         ldx #2
32919 a32919  lda a1331-1,x           ; memtop: speicher-obergrenze
32922         sta *a55-1,x            ; als obergrenze für basic
32924         sta *a51-1,x            ; und anfang des stringbereichs
32926         dex                     ; speichern
32927         bne a32919
32929 a32929  ldy #0
32931 a32931  lda a33095,y            ; routinen f indirekten zugriff
32934         sta a1189,x             ; auf ram nach unten kopieren
32937         inx
32938         iny
32939         cpy #11
32941         bcc a32931
32943         ldy *a34
32945         lda a32956,y
32948         sta a1183,x
32951         dec *a34
32953         bpl a32929
32955         rts
32956
32956 a32956  .by a100 a95 a111 a36 a34 a59   ; adr f lda (...),y
32962
32962 ;---------------------------  E I N S C H A L T M E L D U N G ,   N E W
32962
32962 a32962  lda *a43                ; l,progr anfangszeiger
32964         ldy *a44                ; h,...
32966         jsr a35107              ; ---> platz im speicher prüfen
32969         jsr a65359              ; ---> meldung ausgeben
32972         .by 147 13 ' commodore basic v3.5 ' 0
32997         lda *a55                ; l,basic obergrenze
32999         sec
33000         sbc *a43                ; - l,progr anfangszeiger
33002         tax                     ; = l, freier speicher
33003         lda *a56                ; h,basic obergrenze
33005         sbc *a44                ; - h,progr anfangszeiger
33007         jsr a42079              ; ---> (x,a) ausgeben
33010         jsr a65359              ; ---> meldung ausgeben
33013         .by ' bytes free' 13 0
33026         jmp a35451              ; ---> new
```

```
33029
33029 ;------------------------------  B A S I C  V E K T O R E N
33029
33029 a33029  .si a34438              ; (a768) fehlerausgang
33031         .si a34578              ; (a770) ready-schleife
33033         .si a35158              ; (a772) zeile tokenisieren
33035         .si a35694              ; (a774) token in keywort wandeln
33037         .si a35798              ; (a776) basic routinenaufruf
33039         .si a37911              ; (a778) formelauswertung
33041         .si a35178              ; (a780) keywort in token 254 wandeln
33043         .si a35720              ; (a782) token 254 listen
33045         .si a35979              ; (a784) token 254 ausführen
33047
33047 a33047  ldx #a33047-a33029-1    ; vektoren einrichten
33049 a33049  lda a33029,x
33052         sta a768,x
33055         dex
33056         bpl a33049
33058 a33058  rts
33059
33059 ;------------------------------  C H R G E T  K O P I E
33059
33059 ; der bereich a33059,...,a33108 wird in den ram-bereich ab a1139 kopiert
33059
33059 a33059  inc *a59                ; chrget-routine
33061         bne a33065              ; wird nach unten kopiert
33063         inc *a60
33065 a33065  sei
33066         sta a65343              ; schalter auf ram
33069         ldy #0
33071         lda (a59),y
33073         sta a65342              ; schalter auf rom
33076         cli
33077         cmp #58                 ; ziffer: carry = 0
33079         bcs a33091              ; sonst:  carry = 1
33081         cmp #'
33083         beq a33059
33085         sec
33086         sbc #48
33088         sec
33089         sbc #208
33091 a33091  rts
33092
33092         sta a1180               ; allgemeine routine f indirekten
33095 a33095  sei                     ; zugriff auf ram, wird in den
33096         sta a65343              ; bereich ab 1172 kopiert.
33099         lda (a0),y              ; 1180 ist dann die adresse des operanden
33101         sta a65342
33104         cli
33105         rts
33106
33106         .by 0 0 0               ; konstante 0 für systemvariablen
33109
33109 ;------------------------------  L D A  (...),Y  A U S  R A M
33109
33109 a33109  lda #a67                ; adresse f lda (...),y
33111         bne a33163
33113
33113 a33113  lda #a78
33115         bne a33163
```

```
33117 a33117  lda #a20
33119         bne a33163
33121
33121 a33121  lda #a71
33123         bne a33163
33125
33125 a33125  lda #a78
33127         bne a33163
33129
33129 a33129  lda #a92
33131         bne a33163
33133
33133 a33133  lda #a95
33135         bne a33163
33137
33137 a33137  lda #a61
33139         bne a33163
33141
33141 a33141  lda #a87
33143         bne a33163
33145
33145 a33145  lda #a89
33147         bne a33163
33149
33149 a33149  lda #a98
33151         bne a33163
33153
33153 a33153  lda #a80
33155         bne a33163
33157
33157 a33157  lda #a108
33159         bne a33163
33161
33161 a33161  lda #a90
33163 a33163  jmp a1172              ; ---> lda (adr),y  (aus ram)
33166
33166 ;------------------------------  B A S I C  K E Y W O E R T E R
33166
33166 a33166  .by 'enD'              ; 128
33169         .by 'foR'              ; 129
33172         .by 'nexT'             ; 130
33176         .by 'datA'             ; 131
33180         .by 'input' 163        ; 132
33186         .by 'inpuT'            ; 133
33191         .by 'diM'              ; 134
33194         .by 'reaD'             ; 135
33198         .by 'leT'              ; 136
33201         .by 'gotO'             ; 137
33205         .by 'ruN'              ; 138
33208         .by 'iF'               ; 139
33210         .by 'restorE'          ; 140
33217         .by 'gosuB'            ; 141
33222         .by 'returN'           ; 142
33228         .by 'reM'              ; 143
33231         .by 'stoP'             ; 144
33235         .by 'oN'               ; 145
33237         .by 'waiT'             ; 146
33241         .by 'loaD'             ; 147
33245         .by 'savE'             ; 148
33249         .by 'verifY'           ; 149
```

```
33255       .by 'deF'               ; 150
33258       .by 'pokE'              ; 151
33262       .by 'print' 163         ; 152
33268       .by 'prinT'             ; 153
33273       .by 'conT'              ; 154
33277       .by 'lisT'              ; 155
33281       .by 'clR'               ; 156
33284       .by 'cmD'               ; 157
33287       .by 'syS'               ; 158
33290       .by 'opeN'              ; 159
33294       .by 'closE'             ; 160
33299       .by 'geT'               ; 161
33302       .by 'neW'               ; 162
33305       .by 'tab' 168           ; 163
33309       .by 'tO'                ; 164
33311       .by 'fN'                ; 165
33313       .by 'spc' 168           ; 166
33317       .by 'theN'              ; 167
33321       .by 'noT'               ; 168
33324       .by 'steP'              ; 169
33328       .by 171                 ; 170  +
33329       .by 173                 ; 171  -
33330       .by 170                 ; 172  *
33331       .by 175                 ; 173  /
33332       .by 222                 ; 174
33333       .by 'anD'               ; 175
33336       .by 'oR'                ; 176
33338       .by 190                 ; 177  >
33339       .by 189                 ; 178  =
33340       .by 188                 ; 179  <
33341       .by 'sgN'               ; 180
33344       .by 'inT'               ; 181
33347       .by 'abS'               ; 182
33350       .by 'usR'               ; 183
33353       .by 'frE'               ; 184
33356       .by 'poS'               ; 185
33359       .by 'sqR'               ; 186
33362       .by 'rnD'               ; 187
33365       .by 'loG'               ; 188
33368       .by 'exP'               ; 189
33371       .by 'coS'               ; 190
33374       .by 'siN'               ; 191
33377       .by 'taN'               ; 192
33380       .by 'atN'               ; 193
33383       .by 'peeK'              ; 194
33387       .by 'leN'               ; 195
33390       .by 'str' 164           ; 196
33394       .by 'vaL'               ; 197
33397       .by 'asC'               ; 198
33400       .by 'chr' 164           ; 199
33404       .by 'left' 164          ; 200
33409       .by 'right' 164         ; 201
33415       .by 'mid' 164           ; 202
33419       .by 'gO'                ; 203
33421       .by 'rgR'               ; 204
33424       .by 'rclR'              ; 205
33428       .by 'rluM'              ; 206
33432       .by 'joY'               ; 207
33435       .by 'rdoT'              ; 208
33439       .by 'deC'               ; 209
```

```
33442          .by 'hex' 164            ; 210
33446          .by 'err' 164            ; 211
33450          .by 'instR'              ; 212
33455          .by 'elsE'               ; 213
33459          .by 'resumE'             ; 214
33465          .by 'traP'               ; 215
33469          .by 'troN'               ; 216
33473          .by 'troff'              ; 217
33478          .by 'sounD'              ; 218
33483          .by 'voL'                ; 219
33486          .by 'autO'               ; 220
33490          .by 'pudeF'              ; 221
33495          .by 'graphiC'            ; 222
33502          .by 'painT'              ; 223
33507          .by 'chaR'               ; 224
33511          .by 'boX'                ; 225
33514          .by 'circlE'             ; 226
33520          .by 'gshapE'             ; 227
33526          .by 'sshapE'             ; 228
33532          .by 'draW'               ; 229
33536          .by 'locatE'             ; 230
33542          .by 'coloR'              ; 231
33547          .by 'scnclR'             ; 232
33553          .by 'scalE'              ; 233
33558          .by 'helP'               ; 234
33562          .by 'dO'                 ; 235
33564          .by 'looP'               ; 236
33568          .by 'exiT'               ; 237
33572          .by 'directorY'          ; 238
33581          .by 'dsavE'              ; 239
33586          .by 'dloaD'              ; 240
33591          .by 'headeR'             ; 241
33597          .by 'scratcH'            ; 242
33604          .by 'collecT'            ; 243
33611          .by 'copY'               ; 244
33615          .by 'renamE'             ; 245
33621          .by 'backuP'             ; 246
33627          .by 'deletE'             ; 247
33633          .by 'renumbeR'           ; 248
33641          .by 'keY'                ; 249
33644          .by 'monitoR'            ; 250
33651          .by 'usinG'              ; 251
33656          .by 'untiL'              ; 252
33661          .by 'whilE'              ; 253
33666          .by 0
33667
33667 ;------------------------------  B A S I C   R O U T I N E N   A D R
33667
33667 a33667   .si a36058-1             ; 128  end
33669          .si a44490-1             ; 129  for
33671          .si a37524-1             ; 130  next
33673          .si a36272-1             ; 131  data
33675          .si a37102-1             ; 132  input#
33677          .si a37128-1             ; 133  input
33679          .si a38555-1             ; 134  dim
33681          .si a37199-1             ; 135  read
33683          .si a36476-1             ; 136  let
33685          .si a36173-1             ; 137  goto
33687          .si a35772-1             ; 138  run
33689          .si a36321-1             ; 139  if
```

```
33691   .si a35994-1    ; 140  restore
33693   .si a36140-1    ; 141  gosub
33695   .si a36227-1    ; 142  return
33697   .si a36363-1    ; 143  rem
33699   .si a36056-1    ; 144  stop
33701   .si a36379-1    ; 145  on
33703   .si a40554-1    ; 146  wait
33705   .si a42995-1    ; 147  load
33707   .si a42974-1    ; 148  save
33709   .si a42992-1    ; 149  verify
33711   .si a39581-1    ; 150  def
33713   .si a40466-1    ; 151  poke
33715   .si a36832-1    ; 152  print#
33717   .si a36864-1    ; 153  print
33719   .si a36099-1    ; 154  cont
33721   .si a35583-1    ; 155  list
33723   .si a35480-1    ; 156  clr
33725   .si a36838-1    ; 157  cmd
33727   .si a42933-1    ; 158  sys
33729   .si a43085-1    ; 159  open
33731   .si a43098-1    ; 160  close
33733   .si a37048-1    ; 161  get
33735   .si a35449-1    ; 162  new
33737   .si a36363-1    ; 213  else
33739   .si a46144-1    ; 214  resume
33741   .si a46123-1    ; 215  trap
33743   .si a46674-1    ; 216  tron
33745   .si a46677-1    ; 217  troff
33747   .si a47177-1    ; 218  sound
33749   .si a47293-1    ; 219  vol
33751   .si a46797-1    ; 220  auto
33753   .si a46404-1    ; 221  pudef
33755   .si a50627-1    ; 222  graphic
33757   .si a47313-1    ; 223  paint
33759   .si a47572-1    ; 224  char
33761   .si a47842-1    ; 225  box
33763   .si a49182-1    ; 226  circle
33765   .si a48437-1    ; 227  gshape
33767   .si a48681-1    ; 228  sshape
33769   .si a50393-1    ; 229  draw
33771   .si a50447-1    ; 230  locate
33773   .si a50458-1    ; 231  color
33775   .si a50535-1    ; 232  scnclr
33777   .si a50616-1    ; 233  scale
33779   .si a46824-1    ; 234  help
33781   .si a46423-1    ; 235  do
33783   .si a46595-1    ; 236  loop
33785   .si a46508-1    ; 237  exit
33787   .si a51388-1    ; 238  directory
33789   .si a51521-1    ; 239  dsave
33791   .si a51537-1    ; 240  dload
33793   .si a51560-1    ; 241  header
33795   .si a51612-1    ; 242  scratch
33797   .si a51660-1    ; 243  collect
33799   .si a51674-1    ; 244  copy
33801   .si a51700-1    ; 245  rename
33803   .si a51712-1    ; 246  backup
33805   .si a44634-1    ; 247  delete
33807   .si a43919-1    ; 248  renumber
33809   .si a46889-1    ; 249  key
33811   .si a65362-1    ; 250  monitor
```

```
33813 a33813  .si a41662              ; 180  sgn
33815         .si a41816              ; 181  int
33817         .si a41693              ; 182  abs
33819         .se a1280               ; 183  usr
33821         .si a39522              ; 184  fre
33823         .si a39549              ; 185  pos
33825         .si a42468              ; 186  sqr
33827         .si a42759              ; 187  rnd
33829         .si a40990              ; 188  log
33831         .si a42592              ; 189  exp
33833         .si a43632              ; 190  cos
33835         .si a43639              ; 191  sin
33837         .si a43712              ; 192  tan
33839         .si a43802              ; 193  atn
33841         .si a40442              ; 194  peek
33843         .si a40289              ; 195  len
33845         .si a39782              ; 196  str$
33847         .si a40339              ; 197  val
33849         .si a40304              ; 198  asc
33851         .si a40123              ; 199  chr$
33853         .si a40143              ; 200  left$
33855         .si a40195              ; 201  right$
33857         .si a40213              ; 202  mid$
33859         .si a49017              ; 204  rgr
33861         .si a49029              ; 205  rclr
33863         .si a49031              ; 206  rlum
33865         .si a49089              ; 207  joy
33867         .si a49149              ; 208  rdot
33869         .si a40475              ; 209  dec
33871         .si a46343              ; 210  hex$
33873         .si a46270              ; 211  err$
33875
33875 a33875  .by 121                 ; 170  +
33876         .si a40606-1
33878         .by 121                 ; 171  -
33879         .si a40583-1
33881         .by 123                 ; 172  *
33882         .si a41083-1
33884         .by 123                 ; 173  /
33885         .si a41367-1
33887         .by 127                 ; 174
33888         .si a42478-1
33890         .by 80                  ; 175  and
33891         .si a38395-1
33893         .by 70                  ; 176  or
33894         .si a38392-1
33896         .by 125                 ;      +/-
33897         .si a42535-1
33899         .by 90                  ; 168  not
33900         .si a37989-1
33902         .by 100                 ; 177  >   178  =   179  <
33903         .si a38440-1
33905
33905 ;-------------------------------  M E L D U N G E N
33905
33905 a33905  .by 'too many fileS'    ;         1
33919         .by 'file opeN'         ;         2
33928         .by 'file not opeN'     ;         3
33941         .by 'file not founD'    ;         4
33955         .by 'device not presenT'        ; 5
```

```
33973          .by 'not input filE'    ;          6
33987          .by 'not output filE'   ;          7
34002          .by 'missing file namE'         ;  8
34019          .by 'illegal device numbeR'     ;  9
34040          .by 'next without foR'  ;         10
34056          .by 'syntaX'            ;         11
34062          .by 'return without gosuB'      ; 12
34082          .by 'out of datA'       ;         13
34093          .by 'illegal quantitY'  ;         14
34109          .by 'overflow'          ;         15
34117          .by 'out of memorY'     ;         16
34130          .by 'undef' 39 'd statemenT'    ; 17
34147          .by 'bad subscripT'     ;         18
34160          .by 'redim' 39 'd arraY'        ; 19
34173          .by 'division by zerO'  ;         20
34189          .by 'illegal direcT'    ;         21
34203          .by 'type mismatcH'     ;         22
34216          .by 'string too lonG'   ;         23
34231          .by 'file datA'         ;         24
34240          .by 'formula too compleX'       ; 25
34259          .by 'can' 39 't continuE'       ; 26
34273          .by 'undef' 39 'd functioN'     ; 27
34289          .by 'verifY'            ;         28
34295          .by 'loaD'              ;         29
34299          .by 'break' 0 160       ;         30
34306          .by 'can' 39 't resumE'         ; 31
34318          .by 'loop not founD'    ;         32
34332          .by 'loop without dO'   ;         33
34347          .by 'direct mode onlY'  ;         34
34363          .by 'no graphics areA'  ;         35
34379          .by 'bad disK'          ;         36
34387
34387 a34387   tax                     ; nummer der meldung
34388          ldy #0
34390          lda #l,a33905           ; l,tabelle der meldungen
34392          sta *a36
34394          lda #h,a33905           ; h,...
34396          sta *a37
34398 a34398   dex                     ; richtige meldung gefunden?
34399          bmi a34429              ; ja: ->
34401 a34401   lda (a36),y             ; nächste meldung überlesen
34403          pha
34404          inc *a36
34406          bne a34410
34408          inc *a37
34410 a34410   pla                     ; letztes zeichen der meldung?
34411          bpl a34401              ; nein: ->
34413          bmi a34398              ; ja: ->
34415
34415 a34415   jsr a65359              ; ---> meldung ausgeben
34418          .by 13 10 'ready.' 13 10 0
34429 a34429   rts
34430
34430 ;------------------------------   A U S G A N G
34430
34430 a34430   ldx #128                ; 'ready.'
34432          .by 44
34433 a34433   ldx #16                 ; 'out of memory'
34435 a34435   jmp (a768)              ; ---> fehlerausgang (a34438)
```

```
34438 a34438  txa
34439         bmi a34563              ; 'ready.' ausgeben ->
34441         stx a1263               ; fehlerkode
34444         bit *a129               ; run-modus?
34446         bpl a34501              ; nein: ->
34448         ldy #1
34450 a34450  lda a57,y               ; zeilennummer
34453         sta a1264,y             ; und
34456         lda a603,y              ; adresse
34459         sta a1269,y             ; für trap und resume speichern
34462         dey
34463         bpl a34450
34465         cpx #17                 ; 'undefined statement'?
34467         beq a34501              ; ja: ->
34469         ldy a1267               ; on-error-flag
34472         iny                     ; gesetzt?
34473         beq a34501              ; nein: ->
34475         dey
34476         sty *a21                ; h,zeilennr f fehlerbehandlung
34478         sty a1268               ; merken während fehlerbearbeitung
34481         ldy a1266               ; l,...
34484         sty *a20
34486         ldx #255                ; on-error-flag rücksetzen
34488         stx a1267
34491         ldx a1271               ; stack-zeiger für trap
34494         txs
34495         jsr a36201              ; ---> fehlerbehandlung
34498         jmp a35804              ; ---> kommandoschleife
34501
34501 a34501  dex
34502         txa
34503         pha
34504         lda #0
34506         sta *a131               ; graphik-modus abschalten
34508         jsr a51145              ; ---> normale speicheraufteilung
34511         pla
34512         jsr a34387              ; ---> fehlerzeiger positionieren
34515         jsr a65484              ; ---> clrch
34518         lda #0
34520         sta *a19                ; i/o kanal = tastatur
34522         jsr a36926              ; ---> neue zeile
34525         jsr a37040              ; ---> '?' ausgeben
34528         ldy #0
34530 a34530  lda (a36),y
34532         pha
34533         and #127                ; bit 7 löschen
34535         jsr a37042              ; ---> zeichen ausgeben
34538         iny
34539         pla
34540         bpl a34530
34542         jsr a35544              ; ---> clear stack
34545         jsr a65359              ; ---> meldung ausgeben
34548         .by ' error' 0
34555 a34555  ldy *a58                ; direktmodus?
34557         iny
34558         beq a34563              ; ja: ->
34560         jsr a42067              ; ---> ' in ' und zeilennummer ausgeben
34563 a34563  jsr a34415              ; ---> 'ready.' ausgeben
34566         lda #128                ; ermöglicht ausgaben
34568         jsr a65424              ; ---> setmsg
34571         lda #0
34573         sta *a129               ; flag für runmodus rücksetzen
```

```
34575 ;------------------------------  R E A D Y
34575
34575 a34575  jmp (a770)              ; ---> ready (a34578)
34578
34578 a34578  ldx #255                ; direktmodus herstellen
34580         stx *a58
34582         jsr a34906              ; ---> eingabeschleife
34585         stx *a59                ; l,textzeiger vor puffer setzen
34587         sty *a60                ; h,...
34589         jsr a1139               ; ---> chrget
34592         tax
34593 a34593  beq a34575              ; leere eingabe: ->
34595         bcc a34606              ; zeilennummer eingegeben: ->
34597         jsr a35155              ; ---> eingabe tokenisieren
34600         jsr a1145               ; ---> chrgot
34603         jmp a35801              ; ---> direktanweisung ausführen
34606
34606 a34606  jsr a36414              ; ---> zeilennummer übernehmen
34609         jsr a35155              ; ---> zeile tokenisieren
34612         sty *a11                ; zeilenlänge
34614         jsr a35389              ; ---> zeile suchen
34617         bcc a34693              ; nicht vorhanden: ->
34619
34619 ;------------------------------  Z E I L E  L ö S C H E N
34619
34619         ldy #1
34621         jsr a1233               ; ---> h,linkadresse holen
34624         sta *a35
34626         lda *a45                ; l,variablenanfang
34628         sta *a34
34630         lda *a96                ; h,zeilenanfang
34632         sta *a37
34634         dey
34635         jsr a1233               ; ---> l,linkadresse holen
34638         clc
34639         sbc *a95                ; - l,zeilenanfang
34641         eor #255                ; invertiert
34643         clc                     ; = -zeilenlänge
34644         adc *a45                ; + l,variablenanfang
34646         sta *a45                ; = l,neuer variablenanfang
34648         sta *a36
34650         lda *a46                ; h,variablenanfang
34652         adc #255
34654         sta *a46
34656         sbc *a96                ; h,zeilenanfang
34658         tax                     ; zahl der zu verschiebenden pages
34659         sec
34660         lda *a95                ; l,zeilenanfang
34662         sbc *a45                ; - l,variablenanfang
34664         tay                     ; = länge des restabschnitts
34665         bcs a34670              ; kleiner als 256: ->
34667         inx
34668         dec *a37                ; zeiger f transport vorbereiten
34670 a34670  clc
34671         adc *a34
34673         bcc a34678
34675         dec *a35
34677         clc
```

```
34678 a34678  jsr a1200               ; ---> lda (a34),y
34681         sta (a36),y
34683         iny                     ; ende der page?
34684         bne a34678              ; nein: ->
34686         inc *a35                ; h,quellzeiger
34688         inc *a37                ; h,zielzeiger
34690         dex                     ; page-zähler
34691         bne a34678              ; nicht null: ->
34693 a34693  jsr a35482              ; ---> clr
34696         jsr a34840              ; ---> linkadressen erzeugen
34699         ldy #0
34701         jsr a1189               ; ---> erstes zeichen im puffer holen
34704         beq a34593              ; endemarke: ready ->
34706
34706 ;------------------------------  Z E I L E  E I N F Ü G E N
34706
34706         clc
34707         lda *a45                ; l,variablenanfang
34709         ldy *a46                ; h,...
34711         sta *a90
34713         sty *a91
34715         adc *a11                ; + zeilenlänge
34717         bcc a34720
34719         iny
34720 a34720  clc
34721         adc #4                  ; + länge des zeilenkopfes
34723         bcc a34726
34725         iny
34726 a34726  sta *a88                ; l,neues programmende
34728         sty *a89                ; h,...
34730         jsr a35008              ; ---> verschieberoutine
34733         ldy #0
34735         lda #1
34737         sta (a95),y             ; l,linkadresse
34739         iny
34740         sta (a95),y             ; h,...
34742         iny
34743         lda *a20                ; l,zeilennummer
34745         sta (a95),y
34747         lda *a21                ; h,...
34749         iny
34750         sta (a95),y
34752         iny
34753         tya
34754         clc
34755         adc *a95
34757         sta *a95                ; zeiger hinter zeilenkopf setzen
34759         bcc a34763
34761         inc *a96
34763 a34763  lda *a49
34765         ldy *a50
34767         sta *a45                ; l,neuer variablenanfang
34769         sty *a46                ; h,...
34771         ldy *a11                ; zeilenlänge
34773         dey
34774 a34774  jsr a1189               ; ---> lda (a59),y
34777         sta (a95),y             ; zeile in freigemachten bereich
34779         dey                     ; kopieren
34780         bpl a34774
34782         jsr a34840              ; ---> linkadressen erzeugen
```

```
34785          jsr a35475             ; ---> textzeiger rücksetzen, clr
34788          lda *a115              ; l,auto-schrittweite
34790          ora *a116              ; h,...
34792          beq a34837             ; beide null: ->
34794          lda *a20               ; l,zeilennummer
34796          clc
34797          adc *a115              ; + l,schrittweite
34799          sta *a99               ; = l,neue zeilennummer
34801          lda *a21               ; h,zeilennummer
34803          adc *a116              ; + h,schrittweite
34805          sta *a98               ; = h,neue zeilennummer
34807          ldx #144               ; exponent
34809          sec                    ; verhindert invertieren
34810          jsr a41678             ; ---> integer in gk-zahl wandeln
34813          jsr a42095             ; ---> zahlstring erzeugen
34816          ldx #0
34818 a34818   lda a256+1,x           ; zahlstring aus puffer
34821          beq a34829             ; ende: ->
34823          sta a1319,x            ; in tastaturpuffer
34826          inx
34827          bne a34818
34829 a34829   lda #29                ; 'kursor nach rechts'
34831          sta a1319,x            ; in tastaturpuffer
34834          inx
34835          stx *a239              ; pufferindex
34837 a34837   jmp a34575             ; ---> ready
34840
34840 ;--------------------------   L I N K A D R E S S E N   E R Z E U G E N
34840
34840 a34840   lda *a43               ; l,programmanfang
34842          ldy *a44               ; h,...
34844          sta *a34
34846          sty *a35
34848          clc
34849 a34849   ldy #0
34851          jsr a1200              ; ---> l,linkadresse holen
34854          bne a34862
34856          iny
34857          jsr a1200              ; ---> h,...
34860          beq a34905             ; beide null: programmende ->
34862 a34862   ldy #4
34864 a34864   iny
34865          jsr a1200              ; ---> zeichen aus zeile holen
34868          bne a34864             ; zeilenende? nein: ->
34870          iny
34871          tya
34872          adc *a34
34874          tax
34875          ldy #0
34877          sta (a34),y            ; l,linkadresse
34879          tya
34880          adc *a35
34882          iny
34883          sta (a34),y            ; h,...
34885          stx *a34
34887          sta *a35
34889          bcc a34849             ; immer ->
```

```
34891 a34891  clc
34892         lda *a34                ; l,programmende
34894         ldy *a35                ; h,...
34896         adc #2                  ; + 2
34898         bcc a34901
34900         iny
34901 a34901  sta *a45                ; l,variablenanfang
34903         sty *a46                ; h,...
34905 a34905  rts
34906
34906 ;------------------------------  E I N G A B E S C H L E I F E
34906
34906 a34906  ldx #0                  ; eingabezeiger
34908 a34908  jsr a42897              ; ---> zeichen vom bildschirm holen
34911         cmp #13                 ; zeilenende?
34913         beq a34926              ; ja: ->
34915         sta a512,x              ; zeichen in puffer speichern
34918         inx                     ; eingabezeiger erhöhen
34919         cpx #89                 ; schon 89 zeichen eingegeben?
34921         bcc a34908              ; nein: ->
34923         jmp a52300              ; ---> 'string too long'
34926
34926 a34926  jmp a36913              ; ---> puffer abschliessen
34929
34929 ;------------------------------  S U C H E  I M  B A S I C - S T A C K
34929
34929 a34929  jsr a42848              ; ---> (a61,a62) := (a124,a125)
34932 a34932  lda *a61                ; l,stackzeiger
34934         cmp #l,a1968
34936         bne a34944
34938         lda *a62                ; h,...
34940         cmp #h,a1968
34942         beq a35005              ; stack leer: ->
34944 a34944  ldy #0
34946         lda *a2                 ; gesuchtes token
34948         cmp #129                ; 'for'?
34950         bne a34979              ; nein: ->
34952         cmp (a61),y             ; token im stack?
34954         bne a35007              ; nein: ->
34956         ldy #2
34958         lda *a74                ; for-next variable
34960         cmp #255                ; bei 'next' angegeben?
34962         beq a35007              ; nein: ->
34964         cmp (a61),y             ; gleicher name im stack?
34966         bne a34975              ; nein: ->
34968         dey
34969         lda *a73
34971         cmp (a61),y
34973         beq a35007              ; ja: ->
34975 a34975  ldx #18                 ; stacklänge bei 'for'
34977         bne a34993
34979
34979 a34979  lda (a61),y             ; token im stack
34981         cmp *a2                 ; = gesuchtes token?
34983         beq a35007              ; ja: ->
34985         ldx #18                 ; stacklänge bei 'for'
34987         cmp #129                ; 'for'
34989         beq a34993
34991         ldx #5                  ; stacklänge bei 'gosub'
```

```
34993 a34993  txa
34994         clc
34995         adc *a61                ; stackzeiger erhöhen
34997         sta *a61
34999         bcc a34932
35001         inc *a62
35003         bne a34932
35005
35005 a35005  ldy #1                  ; setzt z-bit zurück
35007 a35007  rts
35008
35008 ;------------------------------  B L O C K T R A N S P O R T
35008
35008 a35008  jsr a35107              ; ---> speicherplatz prüfen
35011         sta *a49                ; l,neues variablenende
35013         sty *a50                ; h,...
35015         sec
35016         lda *a90                ; l,quellbereich ende + 1
35018         sbc *a95                ; - l,quellbereich anfang
35020         sta *a34                ; = blocklänge mod 256
35022         tay
35023         lda *a91                ; h,quellbereich ende + 1
35025         sbc *a96                ; - h,quellbereich anfang
35027         tax                     ; = anzahl der vollen pages
35028         inx
35029         tya
35030         beq a35069              ; keine teilpage: ->
35032         lda *a90                ; l,quellbereich ende + 1
35034         sec
35035         sbc *a34                ; - länge der teilpage
35037         sta *a90                ; = l,ende letzte volle quellpage + 1
35039         bcs a35044
35041         dec *a91                ; h,...
35043         sec
35044 a35044  lda *a88                ; l,zielbereich ende + 1
35046         sbc *a34                ; - länge der teilpage
35048         sta *a88                ; = l,ende letzte volle zielpage + 1
35050         bcs a35061
35052         dec *a89                ; h,...
35054         bcc a35061
35056
35056 a35056  jsr a33161              ; ---> lda (a90),y
35059         sta (a88),y
35061 a35061  dey                     ; teilpage fertig?
35062         bne a35056              ; nein: ->
35064         jsr a33161              ; ---> lda (a90),y
35067         sta (a88),y             ; letztes byte der teilpage
35069 a35069  dec *a91                ; h,quellzeiger
35071         dec *a89                ; h,zielzeiger
35073         dex                     ; page-zähler dekrementieren
35074         bne a35061              ; noch nicht null: ->
35076         rts
35077
35077 ;--------------------------  P L A T Z  I M  B A S I C  S T A C K  ?
35077
35077 a35077  sty a2036               ; platzbedarf (bytes)
35080         sec
35081         lda *a124               ; l,stack zeiger
35083         sbc a2036               ; - platzbedarf
35086         sta *a124               ; = l,neuer stack zeiger
```

```
35088          lda *a125              ; h,...
35090          sbc #0
35092          sta *a125
35094          cmp #h,a1772           ; stack-untergrenze unterschritten?
35096          bcc a35152
35098          bne a35106
35100          lda *a124
35102          cmp #l,a1772
35104          bcc a35152             ; ja: fehler ->
35106 a35106   rts
35107
35107 ;------------------------------  P L A T Z   I M   S P E I C H E R  ?
35107
35107 a35107   cpy *a52               ; (a,y) < string-anfangsadresse?
35109          bcc a35151
35111          bne a35117
35113          cmp *a51
35115          bcc a35151             ; ja: fertig ->
35117 a35117   pha                    ; a retten
35118          ldx #a96-a87           ; zellen a87 bis a96 retten
35120          tya                    ; y retten
35121 a35121   pha
35122          lda *a87,x             ; zellen a87 bis a96 retten
35124          dex
35125          bpl a35121
35127          jsr a43348             ; ---> garbage collect
35130          ldx #247
35132 a35132   pla
35133          sta *a97,x             ; zellen a87 bis a96 wiederherstellen
35135          inx
35136          bmi a35132
35138          pla                    ; y wiederherstellen
35139          tay
35140          pla                    ; a wiederherstellen
35141          cpy *a52               ; string-anfangsadresse
35143          bcc a35151             ; jetzt > (a,y)?
35145          bne a35152
35147          cmp *a51
35149          bcs a35152             ; nein: fehler ->
35151 a35151   rts
35152
35152 a35152   jmp a34433             ; ---> 'out of memory'
35155
35155 ;------------------------------  T O K E N I S I E R E N
35155
35155 a35155   jmp (a772)             ; ---> zeile tokenisieren (a35158)
35158
35158 a35158   lda *a59               ; programmzeiger
35160          pha                    ; retten
35161          lda *a60
35163          pha
35164 a35164   jsr a1145              ; ---> chrgot
35167          jmp a35173             ; --->
35170
35170 a35170   jsr a1139              ; ---> chrget
35173 a35173   bcc a35170             ; ziffer: weiterlesen ->
35175          jmp (a780)             ; ---> user-keywort (a35178)
```

```
35178 a35178  bcc a35284              ; cc: user-keywort ->
35180         cmp #0                  ; endemarke?
35182         beq a35269              ; ja: ->
35184         cmp #':                 ; trennzeichen?
35186         beq a35170              ; ja: ->
35188         cmp #'?                 ; '?'
35190         bne a35196              ; nein: ->
35192         lda #153                ; durch 'print'-token ersetzen
35194         bne a35242
35196
35196 a35196  cmp #128                ; shift-zeichen?
35198         bcc a35211              ; nein: ->
35200         cmp #255                ; pi?
35202         beq a35170              ; ja: ->
35204         ldy #1
35206         jsr a35306              ; ---> zeilenrest um (y) vorziehen
35209         beq a35164              ; immer ->
35211
35211 a35211  cmp #'"
35213         bne a35228
35215 a35215  jsr a1139               ; ---> chrget
35218         cmp #0                  ; endemarke?
35220         beq a35269              ; ja: ->
35222         cmp #'"
35224         beq a35170
35226         bne a35215
35228
35228 a35228  jsr a35331              ; ---> keywort tokenisieren
35231         bcc a35170              ; nicht gefunden: ->
35233         cpy #0
35235         beq a35240              ; kein offset: ->
35237         jsr a35306              ; ---> zeilenrest um (y) vorziehen
35240 a35240  lda *a11                ; token
35242 a35242  ldy #0
35244         sta (a59),y             ; speichern
35246         cmp #143                ; 'rem'-token?
35248         beq a35263              ; ja: ->
35250         cmp #131                ; 'data'-token?
35252         bne a35170              ; nein: ->
35254         jsr a1139               ; ---> chrget
35257         jsr a36272              ; ---> progzgr auf nächstes trennzchn
35260         jmp a35164              ; ---> zurück an den anfang
35263
35263 a35263  jsr a1139               ; ---> chrget
35266         jsr a36363              ; ---> progzgr auf zeilenende
35269 a35269  ldx *a59
35271         pla                     ; programmzeiger wiederherstellen
35272         sta *a60
35274         pla
35275         sta *a59
35277         sec
35278         txa
35279         sbc *a59
35281         tay
35282         iny                     ; zeilenlänge
35283         rts
```

```
35284 ;------------------------------  U S E R - T O K E N  E I N B A U E N
35284
35284 a35284  pha                     ; user-zeichen auf stack
35285         dey
35286         dey
35287         jsr a35306              ; ---> zeilenrest um (y) vorziehen
35290         ldy #0
35292         lda #254                ; user-token 254
35294         sta (a59),y
35296         iny
35297         pla                     ; und user-zeichen
35298         sta (a59),y             ; in zeile bringen
35300         jsr a1139               ; ---> chrget
35303         jmp a35170              ; ---> weitermachen
35306
35306 ;------------------------------  Z E I L E N R E S T  V O R Z I E H E N
35306
35306 a35306  clc
35307         tya                     ; offset
35308         adc *a59                ; + textzeiger
35310         sta *a34                ; = quellzeiger
35312         lda *a60
35314         adc #0
35316         sta *a35
35318         ldy #0
35320 a35320  jsr a1200               ; ---> lda (a34),y
35323         sta (a59),y
35325         iny
35326         cmp #0                  ; zeilenende?
35328         bne a35320              ; nein: ->
35330         rts
35331
35331 ;------------------------------  K E Y W O R T  S U C H E N
35331
35331 a35331  lda #h,a33166           ; anfang der keywort-tabelle
35333         ldy #l,a33166
35335         sta *a35
35337         sty *a34
35339         ldy #0
35341         sty *a11                ; wortzähler
35343         dey
35344 a35344  iny
35345 a35345  jsr a1189               ; ---> lda (a59),y: zeichen aus text
35348         sec
35349         sbc (a34),y             ; - kode aus tabelle
35351         beq a35344              ; = 0: vergleich fortsetzen ->
35353         cmp #128                ; wortende?
35355         beq a35384              ; ja: ->
35357 a35357  lda (a34),y             ; zeichen aus tabelle
35359         bmi a35364              ; endemarke: ->
35361         iny
35362         bne a35357              ; immer ->
35364
35364 a35364  iny
35365         inc *a11                ; wortzähler erhöhen
35367         clc
35368         tya
35369         adc *a34
35371         sta *a34                ; zeiger auf nächstes wort setzen
35373         bcc a35377
35375         inc *a35
```

```
35377 a35377  clc
35378         ldy #0
35380         lda (a34),y             ; zeichen aus tabelle
35382         bne a35345              ; tabelle noch nicht zu ende: ->
35384 a35384  ora *a11                ; bit 7 setzen, wenn wort gefunden
35386         sta *a11
35388         rts
35389
35389 ;-------------------------------  Z E I L E   S U C H E N
35389
35389 a35389  lda *a43                ; l,programmanfang
35391         ldx *a44                ; h,...
35393 a35393  ldy #1
35395         sta *a95
35397         stx *a96
35399         jsr a1233               ; ---> lda (a95),y
35402         beq a35447              ; programmende: ->
35404         iny
35405         iny
35406         jsr a1233               ; ---> lda (a95),y: h,zeilennummer
35409         sta *a120
35411         lda *a21                ; h,gesuchte zeilennummer
35413         cmp *a120               ; h,gelesene zeilennummer
35415         bcc a35448              ; gesuchte kleiner: ->
35417         beq a35422              ; gleich: ->
35419         dey                     ; y := 2
35420         bne a35436              ; immer ->
35422
35422 a35422  dey
35423         jsr a1233               ; ---> lda (a95),y: l,zeilennummer
35426         sta *a120
35428         lda *a20                ; l,gesuchte zeilennummer
35430         cmp *a120               ; l,gelesene zeilennummer
35432         bcc a35448              ; gesuchte kleiner: ->
35434         beq a35448              ; gleich: ->
35436 a35436  dey
35437         jsr a1233               ; ---> lda (a95),y: h,linkadresse
35440         tax
35441         dey
35442         jsr a1233               ; ---> lda (a95),y: l,linkadresse
35445         bcs a35393              ; immer ->
35447
35447 a35447  clc                     ; flag 'zeile nicht gefunden'
35448 a35448  rts
35449
35449 ;-------------------------------  N E W
35449
35449 a35449  bne a35448              ; kein trennzeichen folgt: fehler ->
35451 a35451  lda #0
35453         tay
35454         sta (a43),y             ; am programmanfang die
35456         iny                     ; endemarke '00' anbringen
35457         sta (a43),y
35459         sta a747                ; trace abschalten
35462         lda *a43                ; programmanfang
35464         clc
35465         adc #2                  ; + 2
35467         sta *a45                ; = variablenanfang
35469         lda *a44
35471         adc #0
35473         sta *a46
```

```
35475 a35475  jsr a35569          ; ---> programmzeiger rücksetzen
35478         lda #0              ; verhindert branch
35480
35480 ;-------------------------   C L R
35480
35480 a35480  bne a35564          ; kein trennzeichen folgt: fehler ->
35482 a35482  jsr a65511          ; ---> clall schliesst alle kanäle
35485         ldy #0
35487         sty *a121           ; ds$ löschen
35489         dey
35490         sty a1267           ; on-error-flag rücksetzen
35493         sty a1264           ; l,fehlerzeile f 'trap'
35496         sty a1265           ; h,...
35499         sty a1263           ; fehlerkode
35502         lda *a55            ; l,basic-obergrenze
35504         ldy *a56            ; h,...
35506         sta *a51            ; l,stringbereich-anfang
35508         sty *a52            ; h,...
35510         lda #l,a1968        ; l,basic stack ende
35512         ldy #h,a1968        ; h,...
35514         sta *a124           ; l,basic stack zeiger
35516         sty *a125           ; h,...
35518         lda *a45            ; l,variablenanfang
35520         ldy *a46            ; h,...
35522         sta *a47            ; l,arrays-anfang
35524         sty *a48            ; h,...
35526         sta *a49            ; l,variablenende
35528         sty *a50            ; h,...
35530         ldx #3
35532 a35532  lda a35565,x        ; pudef-zeichen
35535         sta a1255,x         ; wiederherstellen
35538         dex
35539         bpl a35532
35541 a35541  jsr a36017          ; ---> restore
35544 a35544  ldx #25             ; string stack zeiger
35546         stx *a22            ; rücksetzen
35548         pla                 ; letzte rücksprungadresse
35549         tay                 ; vom stack nehmen
35550         pla
35551         ldx #250            ; stack rücksetzen
35553         txs
35554         pha                 ; letzte rücksprungadresse
35555         tya                 ; wieder auf stack legen
35556         pha
35557         lda #0
35559         sta a604            ; log adresse f ausgabe und
35562         sta *a16            ; integer-sperre rücksetzen
35564 a35564  rts
35565
35565 a35565  .by ' ,.$'          ; zeichen f 'pudef'
35569
35569 ;-------------------------   P R O G Z G R  R U E C K S E T Z E N
35569
35569 a35569  clc
35570         lda *a43            ; programmanfang
35572         adc #255            ; - 1
35574         sta *a59            ; = programmzeiger
35576         lda *a44
35578         adc #255
35580         sta *a60
35582         rts
```

```
35583 ;------------------------------  L I S T
35583
35583 a35583  jsr a44746              ; ---> bereichsangabe analysieren
35586 a35586  ldy #1
35588         jsr a1233               ; ---> lda (a95),y: h,linkadresse
35591         bne a35599
35593         dey
35594         jsr a1233               ; ---> lda (a95),y: l,linkadresse
35597         beq a35645              ; programmende: ->
35599 a35599  jsr a36032              ; ---> stoptaste prüfen
35602         jsr a36926              ; ---> neue zeile
35605         ldy #2
35607         jsr a1233               ; ---> lda (a95),y: l,zeilennummer
35610         tax
35611         iny
35612         jsr a1233               ; ---> lda (a95),y: h,zeilennummer
35615         cmp *a21                ; endzeile
35617         bne a35623
35619         cpx *a20
35621         beq a35625              ; erreicht: ->
35623 a35623  bcs a35645              ; überschritten: ->
35625 a35625  jsr a35648              ; ---> zeile listen
35628         ldy #0
35630         jsr a1233               ; ---> lda (a95),y: l,linkadresse
35633         tax
35634         iny
35635         jsr a1233               ; ---> lda (a95),y: h,linkadresse
35638         stx *a95                ; l,laufzeiger
35640         sta *a96                ; h,...
35642         jmp a35586              ; ---> weiterlisten
35645
35645 a35645  jmp a36926              ; ---> neue zeile
35648
35648 a35648  ldy #3
35650         sty *a73                ; y merken
35652         sty *a15                ; textflag rücksetzen
35654         jsr a42079              ; ---> zeilennummer ausgeben
35657         lda #'
35659 a35659  ldy *a73                ; y erinnern
35661         and #127                ; bit 7 löschen
35663 a35663  jsr a37042              ; ---> zeichen ausgeben
35666         cmp #'"
35668         bne a35676              ; kein gänsefuss: ->
35670         lda *a15                ; textflag flippen
35672         eor #255
35674         sta *a15
35676 a35676  iny                     ; zeiger in zeile erhöhen
35677         beq a35645              ; zeile ohne ende: abbruch ->
35679         bit *a83                ; help-listing?
35681         bpl a35686              ; nein: ->
35683         jsr a46860              ; ---> blinkflag an fehlerstelle setzen
35686 a35686  jsr a1233               ; ---> lda (a95),y
35689         beq a35771              ; zeilenende: ->
35691
35691 ;------------------------------  K E Y W O R T  E R Z E U G E N
35691
35691 a35691  jmp (a774)              ; ---> keywort erzeugen (a35694)
```

```
35694 a35694  bpl a35663              ; kein token: ->
35696         cmp #255                ; pi?
35698         beq a35663              ; ja: ->
35700         bit *a15                ; text?
35702         bmi a35663              ; ja: ->
35704         cmp #254                ; user-token?
35706         bne a35731              ; nein: ->
35708         iny
35709         jsr a1233               ; ---> lda (a95),y: nächstes zeichen
35712         beq a35726              ; zeilenende: ->
35714         sty *a73                ; y merken
35716         sec
35717         jmp (a782)              ; ---> user-aufruf (a35720)
35720
35720 a35720  bcs a35663              ; keine user-verzweigung: fehler ->
35722         ldy #0                  ; offset
35724         beq a35762              ; immer ->
35726
35726 a35726  dey                     ; offset zurücksetzen
35727         lda #254                ; user-token 254 wiederherstellen
35729         bne a35663              ; immer ->
35731
35731 a35731  tax                     ; token als zähler
35732         sty *a73                ; y merken
35734         ldy #h,a33166           ; anfang der keywort-tabelle
35736         sty *a35                ; in laufzeiger
35738         ldy #l,a33166
35740         sty *a34
35742         ldy #0
35744 a35744  dex                     ; zähler dekrementieren
35745         bpl a35762              ; kleiner als 128: ->
35747 a35747  lda (a34),y             ; zeichen aus tabelle
35749         pha
35750         inc *a34                ; laufzeiger erhöhen
35752         bne a35756
35754         inc *a35
35756 a35756  pla
35757         bpl a35747              ; kein wortende: ->
35759         bmi a35744              ; wortende: ->
35761
35761 a35761  iny
35762 a35762  lda (a34),y             ; zeichen aus tabelle
35764         bmi a35659              ; wortende: ->
35766         jsr a37042              ; ---> zeichen ausgeben
35769         bne a35761              ; immer ->
35771
35771 a35771  rts
35772
35772 ;------------------------------  R U N
35772
35772 a35772  bne a35780              ; zeilennummer folgt: ->
35774         jsr a36128              ; ---> dir-modus, auto, meldungen aus
35777         jmp a35475              ; ---> progzgr rücksetzen, clr
35780
35780 a35780  jsr a35482              ; ---> clr
35783         jsr a1145               ; ---> chrgot
35786         jsr a36173              ; ---> goto
35789         jsr a36128              ; ---> dir-modus, auto, meldungen aus
35792         jmp a35804              ; ---> hauptschleife
```

```
35795 ;------------------------------  H A U P T S C H L E I F E
35795
35795 a35795  jmp (a776)              ; ---> hauptschleife (a35798)
35798
35798 a35798  jsr a1139               ; ---> chrget
35801 a35801  jsr a35877              ; ---> basic-routinen-aufruf
35804 a35804  jsr a36032              ; ---> stoptaste prüfen
35807         bit *a129               ; run-modus?
35809         bpl a35818              ; nein: ->
35811         jsr a35866              ; ---> progrzeiger f cont u trap merken
35814         tsx                     ; stack zeiger
35815         stx a1271               ; für trap merken
35818 a35818  ldy #0
35820         jsr a1189               ; ---> lda (a59),y: erstes zeichen
35823         beq a35828              ; endemarke: ->
35825         jmp a35987              ; ---> auf ':' prüfen
35828
35828 a35828  bit *a129               ; run-modus?
35830         bpl a35863              ; nein: ->
35832         ldy #2
35834         jsr a1189               ; ---> lda (a59),y: h,linkadr
35837         beq a35863              ; programmende: ->
35839         iny
35840         jsr a1189               ; ---> lda (a59),y: l,zeilennummer
35843         sta *a57                ; merken
35845         iny
35846         jsr a1189               ; ---> lda (a59),y: h,zeilennummer
35849         sta *a58                ; merken
35851         tya
35852         clc
35853         adc *a59                ; progrzeiger hinter zeilenkopf setzen
35855         sta *a59
35857         bcc a35795
35859         inc *a60
35861         bne a35795              ; und zurück zum anfang ->
35863
35863 a35863  jmp a34430               ; ---> ready.
35866
35866 a35866  lda *a59                ; l,programmzeiger
35868         ldy *a60                ; h,...
35870         sta a603                ; l,aktuelle zeilenadresse
35873         sty a604                ; h,...
35876 a35876  rts
35877
35877 a35877  beq a35876              ; endezeichen: ->
35879         bit a747                ; trace eingeschaltet?
35882         bpl a35903              ; nein: ->
35884         bit *a129               ; run-modus?
35886         bpl a35903              ; nein: ->
35888         pha
35889         lda #91                 ; 'eckige klammer auf'
35891         jsr a37042              ; ---> ausgeben
35894         jsr a42075              ; ---> zeilennummer ausgeben
35897         lda #93                 ; 'eckige klammer zu'
35899         jsr a37042              ; ---> ausgeben
35902         pla
35903 a35903  cmp #254                ; user-token?
35905         beq a35970              ; ja: ->
35907         cmp #203                ; 'go'?
35909         beq a35956              ; ja: ->
```

```
35911          cmp #202                ; 'mid$'?
35913          beq a35947              ; ja: ->
35915          cmp #251                ; funktionstoken?
35917          bcs a35981
35919          cmp #163
35921          bcc a35929
35923          cmp #213
35925          bcc a35981              ; ja: fehler ->
35927          sbc #50                 ; token um 50 vermindern
35929 a35929   sec
35930          sbc #128
35932          bcc a35984              ; kein token: 'let' ->
35934          asl a                   ; ergebnis verdoppeln
35935          tay                     ; ergibt zeiger in adressentabelle
35936          lda a33667+1,y          ; adresse auf stack legen
35939          pha
35940          lda a33667,y
35943          pha
35944          jmp a1139               ; ---> chrget, 'rts' ruft routine
35947
35947 a35947   lda #h,a46683-1         ; aufruf der routine 'mid$(...) ='
35949          pha
35950          lda #l,a46683-1
35952          pha
35953 a35953   jmp a1139               ; ---> chrget
35956
35956 a35956   jsr a1139               ; ---> chrget
35959          cmp #164                ; 'to'-token?
35961          bne a35981              ; nein: fehler ->
35963          jsr a1139               ; ---> chrget
35966          jmp a36173              ; ---> goto
35969
35969          .by 0
35970
35970 a35970   jsr a1139               ; ---> chrget
35973          beq a35981              ; zeilenende: fehler ->
35975          sec
35976          jmp (a784)              ; ---> user-token bearbeiten (a35979)
35979
35979 a35979   bcc a35953              ; user-token bearbeitet: ->
35981 a35981   jmp a38049              ; ---> 'syntax error'
35984
35984 a35984   jmp a36476              ; ---> let
35987
35987 a35987   cmp #':
35989          bne a35981              ; kein trennzeichen: fehler ->
35991          jmp a35795              ; ---> zurück in die hauptschleife
35994
35994 ;------------------------------  R E S T O R E
35994
35994 a35994   beq a36017              ; ohne zeilennummer: ->
35996          jsr a40417              ; ---> zeilennummer holen
35999          sty *a20                ; l,zeilennummer
36001          sta *a21                ; h,...
36003          jsr a35389              ; ---> zeile suchen
36006          bcs a36011              ; gefunden: ->
36008          jmp a36239              ; ---> 'undefined statement'
36011
36011 a36011   lda *a95                ; l,zeilenanfang
36013          ldy *a96                ; h,...
36015          bcs a36022              ; immer ->
```

```
36017 a36017  sec
36018         lda *a43                ; l,programmanfang
36020         ldy *a44                ; h,...
36022 a36022  sbc #1                  ; - 1
36024         bcs a36027
36026         dey
36027 a36027  sta *a65                ; l,datazeiger
36029         sty *a66                ; h,...
36031 a36031  rts
36032
36032 ;-------------------------------  S T O P   T R A P
36032
36032 a36032  jsr a65505              ; ---> stoptaste gedrückt?
36035         bne a36031              ; nein: ->
36037         php
36038         ldy a1267               ; on-error-flag
36041         iny                     ; gesetzt?
36042         beq a36055              ; nein: ->
36044 a36044  jsr a65505              ; ---> stoptaste gedrückt?
36047         beq a36044              ; ja: warten ->
36049         plp
36050         ldx #30                 ; 'break'
36052         jmp a34435              ; ---> fehlerausgang
36055
36055 a36055  plp
36056
36056 ;-------------------------------  S T O P
36056
36056 a36056  bcs a36059
36058
36058 ;-------------------------------  E N D
36058
36058 a36058  clc
36059 a36059  bne a36031
36061         bit *a129               ; run-modus?
36063         bpl a36078              ; nein: ->
36065         jsr a35866              ; ---> programmzeiger speichern
36068         lda *a57                ; l,zeilennummer
36070         ldy *a58                ; h,...
36072         sta a601                ; l,zeilennummer f 'cont'
36075         sty a602                ; h,...
36078 a36078  pla                     ; rücksprungadresse vom stack nehmen
36079         pla
36080         bcc a36096              ; 'end': ->
36082         jsr a65359              ; ---> meldung ausgeben
36085         .by 13 10 'break' 0
36093         jmp a34555              ; ---> 'in ...' ausgeben, ready.
36096
36096 a36096  jmp a34430              ; ---> ready.
36099
36099 ;-------------------------------  C O N T
36099
36099 a36099  bne a36031              ; kein trennzeichen: rts ->
36101         ldx #26                 ; 'can't continue'
36103         ldy a604                ; cont-sperre gesetzt?
36106         bne a36111              ; nein: ->
36108         jmp a34435              ; ---> fehlerausgang
```

```
36111 a36111  lda a603                ; programmzeiger wiederherstellen
36114         sta *a59
36116         sty *a60
36118         lda a601                ; zeilennummer wiederherstellen
36121         ldy a602
36124         sta *a57
36126         sty *a58
36128 a36128  lda #128
36130         sta *a129               ; run-modus einschalten
36132         asl a                   ; a := 0
36133         sta *a115               ; auto abschalten
36135         sta *a116
36137         jmp a65424              ; ---> setmsg: meldungen abschalten
36140
36140 ;-------------------------------  G O S U B
36140
36140 a36140  ldy #5
36142         jsr a35077              ; ---> platz im basic stack?
36145         dey
36146         lda *a60                ; programmzeiger
36148         sta (a124),y
36150         dey
36151         lda *a59
36153         sta (a124),y
36155         dey
36156         lda *a58                ; und zeilennummer
36158         sta (a124),y
36160         dey
36161         lda *a57
36163         sta (a124),y
36165         dey
36166         lda #141                ; und 'gosub'-token
36168         sta (a124),y            ; auf basic-stack legen
36170         jsr a1145               ; ---> chrgot
36173
36173 ;-------------------------------  G O T O
36173
36173 a36173  jsr a36414              ; ---> zeilennummer holen
36176         jsr a36289              ; ---> y := offset zum zeilenende
36179         sec
36180         lda *a57                ; aktuelle zeilennummer
36182         sbc *a20
36184         lda *a58
36186         sbc *a21                ; grösser als gerufene zeilennummer?
36188         bcs a36201              ; ja: ->
36190         tya
36191         sec
36192         adc *a59                ; programmzeiger auf nächste zeile
36194         ldx *a60
36196         bcc a36205
36198         inx
36199         bcs a36205
36201
36201 a36201  lda *a43                ; programmanfang
36203         ldx *a44
36205 a36205  jsr a35393              ; ---> zeile ab adresse (a,x) suchen
36208         bcc a36239              ; nicht gefunden: ->
36210         lda *a95                ; zeilenadresse
36212         sbc #1                  ; - 1
36214         sta *a59                ; in programmzeiger bringen
```

```
36216          lda *a96
36218          sbc #0
36220          sta *a60
36222          bit *a129              ; run-modus?
36224          bpl a36128             ; nein: dir-modus, auto, meldungen aus ->
36226          rts
36227
36227 ;------------------------------  R E T U R N
36227
36227 a36227   lda #141               ; 'gosub'-token
36229          sta *a2                ; f stacksuche merken
36231          jsr a34929             ; ---> suche im basic stack
36234          beq a36244             ; gefunden: ->
36236          ldx #12                ; 'return without gosub'
36238          .by 44
36239 a36239   ldx #17                ; 'undefined statement'
36241          jmp a34435             ; ---> fehlerausgang
36244
36244 a36244   jsr a42857             ; ---> stackzeiger kopieren
36247          ldy #5                 ; zahl der bytes im stack
36249          jsr a42866             ; ---> zeiger um y erhöhen
36252          dey
36253          lda (a61),y            ; programmzeiger
36255          sta *a60
36257          dey
36258          lda (a61),y
36260          sta *a59
36262          dey
36263          lda (a61),y            ; zeilennummer
36265          jsr a52607             ; ---> rucksack
36268          lda (a61),y
36270          sta *a57
36272
36272 ;------------------------------  D A T A
36272
36272 a36272   jsr a36286             ; ---> y := offset zum trennzeichen
36275 a36275   tya
36276          clc
36277          adc *a59               ; programmzeiger auf
36279          sta *a59               ; nächstes trennzeichen setzen
36281          bcc a36285
36283          inc *a60
36285 a36285   rts
36286
36286 a36286   ldx #':                ; trennzeichen
36288          .by 44
36289 a36289   ldx #0                 ; zeilenendemarke
36291          stx *a7
36293          ldy #0
36295          sty *a8
36297 a36297   lda *a8
36299          ldx *a7
36301          sta *a7
36303          stx *a8
36305 a36305   jsr a1189              ; ---> lda (a59),y: zeichen aus zeile
36308          beq a36285             ; endemarke: ->
36310          cmp *a8                ; trennzeichen bzw. endemarke?
36312          beq a36285             ; ja: ->
36314          iny
36315          cmp #'"                ; gänsefuss?
36317          bne a36305             ; nein: ->
36319          beq a36297             ; sonst trennzeichen unterdrücken
```

```
36321 ;------------------------------   I F
36321
36321 a36321  jsr a37676              ; ---> ausdruck auswerten
36324         jsr a1145               ; ---> chrgot
36327         cmp #137                ; 'goto'-token?
36329         beq a36336              ; ja: ->
36331         lda #167                ; 'then'-token
36333         jsr a38035              ; ---> syntax check
36336 a36336  lda *a97                ; exponent von fac
36338         bne a36368              ; bedingung erfüllt: ->
36340 a36340  jsr a36272              ; ---> progrzgr auf trennzeichen
36343         ldy #0
36345         jsr a1189               ; ---> lda (a59),y: zeichen aus zeile
36348         beq a36363              ; endezeichen: ->
36350         jsr a1139               ; ---> chrget
36353         cmp #213                ; 'else'-token?
36355         bne a36340              ; nein: ->
36357         jsr a1139               ; ---> chrget
36360         jmp a36368              ; --->
36363
36363 ;------------------------------   R E M ,   E L S E
36363
36363 a36363  jsr a36289              ; ---> progrzgr auf zeilenende
36366         beq a36275              ; immer ->
36368
36368 a36368  jsr a1145               ; ---> chrgot
36371         bcs a36376              ; keine zeilennummer: ->
36373         jmp a36173              ; ---> goto
36376
36376 a36376  jmp a35877              ; ---> routinenaufruf
36379
36379 ;------------------------------   O N
36379
36379 a36379  jsr a40324              ; ---> byteausdruck auswerten
36382         pha                     ; nächstes zeichen
36383         cmp #141                ; 'gosub'-token?
36385         beq a36394              ; ja: ->
36387 a36387  cmp #137                ; 'goto'-token?
36389         beq a36394              ; ja: ->
36391         jmp a38049              ; ---> syntax error
36394
36394 a36394  dec *a101               ; parameter dekrementieren
36396         bne a36402              ; noch nicht null: ->
36398         pla                     ; nächstes zeichen
36399         jmp a35903              ; ---> hauptschleife
36402
36402 a36402  jsr a1139               ; ---> chrget
36405         jsr a36414              ; ---> zeilennummer holen
36408         cmp #',                 ; folgt komma?
36410         beq a36394              ; ja: ->
36412         pla                     ; stackzeiger wiederherstellen
36413 a36413  rts
36414
36414 ;--------------------   Z E I L E N N U M M E R  IN ADRESSFORMAT WANDELN
36414
36414 a36414  ldx #0                   ; vorbesetzung
36416         stx *a8                 ; flag f 'zeilennummer vorhanden'
36418         stx *a20                ; l,zeilennummer
36420         stx *a21                ; h,...
```

```
36422 a36422  bcs a36413              ; keine ziffer im accu: fertig ->
36424         inc *a8                 ; flag setzen
36426         sbc #47                 ; zeilennummer in a20,a21 generieren
36428         sta *a7
36430         lda *a21
36432         sta *a34
36434         cmp #25                 ; zeilennummer > 63999?
36436         bcs a36387              ; ja: fehler ->
36438         lda *a20
36440         asl a
36441         rol *a34
36443         asl a
36444         rol *a34
36446         adc *a20
36448         sta *a20
36450         lda *a34
36452         adc *a21
36454         sta *a21
36456         asl *a20
36458         rol *a21
36460         lda *a20
36462         adc *a7
36464         sta *a20
36466         bcc a36470
36468         inc *a21
36470 a36470  jsr a1139               ; ---> chrget
36473         jmp a36422              ; ---> zurück zum schleifenanfang
36476
36476 ;-------------------------------  L E T
36476
36476 a36476  jsr a38565              ; ---> variable suchen oder anlegen
36479         sta *a73                ; l,variablenzeiger
36481         sty *a74                ; h,...
36483         lda #178                ; '='-token
36485         jsr a38035              ; ---> syntax check
36488         lda *a14                ; integer flag
36490         pha
36491         lda *a13                ; string flag
36493         pha
36494         jsr a37676              ; ---> ausdruck auswerten
36497         pla                     ; string flag
36498         rol a                   ; setzt carry bei string
36499         jsr a37659              ; ---> variablentyp prüfen
36502         bne a36528              ; string: ->
36504         pla
36505 a36505  bpl a36525              ; gk: ->
36507         jsr a41632              ; ---> fac runden
36510         jsr a39046              ; ---> gk-integer wandlung
36513         ldy #0                  ; integer in variable schreiben
36515         lda *a100
36517         sta (a73),y
36519         iny
36520         lda *a101
36522         sta (a73),y
36524         rts
36525
36525 a36525  jmp a41557              ; ---> fac in variable übertragen
```

```
36528 ;------------------------------  S T R I N G Z U W E I S U N G
36528
36528 ; die kommentare beziehen sich auf die zuweisung n$ = a$.
36528 ; mit d(..) werden die deskriptoren, mit z(..) die zeiger bezeichnet.
36528
36528 a36528  pla
36529 a36529  ldy *a74                ; h,z(n$)
36531         cpy #4                 ; deskriptor in page 4?
36533         bne a36649             ; nein: ->
36535
36535 ;------------------------------  Z U W E I S U N G  A N  T I $
36535
36535         jsr a40014             ; ---> frestr
36538         cmp #6                 ; stringlänge = 6?
36540         bne a36604             ; nein: illegal quantity ->
36542         ldy #0
36544         sty *a97               ; fac exponent
36546         sty *a102              ; fac vorzeichen
36548 a36548  sty *a113
36550         jsr a36596             ; ---> prüfen ob ziffer
36553         jsr a41314             ; ---> fac := 10 * fac
36556         inc *a113              ; stellenzähler erhöhen
36558         ldy *a113
36560         jsr a36596             ; ---> prüfen ob ziffer
36563         jsr a41617             ; ---> arg := fac
36566         tax
36567         beq a36574             ; fac = 0: ->
36569         inx
36570         txa
36571         jsr a41325             ; ---> fac := fac + arg
36574 a36574  ldy *a113              ; stellenzähler
36576         iny                    ; erhöhen
36577         cpy #6                 ; schon 6 stellen?
36579         bne a36548             ; nein: ->
36581         jsr a41314             ; ---> fac := 10 * fac
36584         jsr a41767             ; ---> gk-integer wandlung
36587         ldx *a100
36589         ldy *a99
36591         lda *a101
36593         jmp a65499             ; ---> settim: uhr stellen
36596
36596 a36596  jsr a1200              ; ---> lda (a34),y
36599         jsr a1157              ; ---> ziffer?
36602         bcc a36607             ; ja: ->
36604 a36604  jmp a39196             ; ---> 'illegal quantity'
36607
36607 a36607  sbc #47                ; 47+1 (carry = 0!) subtrahieren
36609         jmp a41994             ; ---> ergebnis nach fac und arg
36612
36612 ;------------------------------
36612
36612 a36612  pla                    ; h,adr(a$)
36613         iny
36614 a36614  cmp *a52               ; h,stringbereich-anfang
36616         bcc a36642             ; a$ nicht im stringbereich: ->
36618         bne a36628
36620         dey
36621         jsr a1244              ; ---> lda (a100),y: l,adr(a$)
36624         cmp *a51               ; l,stringbereich-anfang
36626         bcc a36642             ; a$ nicht im stringbereich: ->
```

```
36628 a36628  ldy *a101              ; h,z(a$)
36630         cpy *a46               ; h,variablenanfang
36632         bcc a36642             ; d(a$) nicht in variablentabelle: ->
36634         bne a36672
36636         lda *a100              ; l,z(a$)
36638         cmp *a45
36640         bcs a36672             ; d(a$) nicht in variablentabelle: ->
36642 a36642  lda *a100              ; l,z(a$)
36644         ldy *a101              ; h,...
36646         jmp a36702             ; --->
36649
36649 a36649  ldy #2
36651         jsr a1244              ; ---> lda (a100),y: h,adr(a$)
36654         cmp *a123              ; = h,adr(ds$)?
36656         bne a36614             ; nein: ->
36658         pha
36659         dey
36660         jsr a1244              ; ---> lda (a100),y: l,adr(a$)
36663         cmp *a122              ; = l,adr(ds$)?
36665         bne a36612             ; nein: ->
36667         lda *a121              ; länge von ds$
36669         beq a36612             ; = 0: ->
36671         pla                    ; stack bereinigen
36672
36672 ;------------------------------  S T R I N G   K O P I E R E N
36672
36672 a36672  ldy #0
36674         jsr a1244              ; ---> lda (a100),y: länge von a$
36677         jsr a39764             ; ---> platz reservieren
36680         lda *a80               ; l,z(a$)
36682         ldy *a81               ; h,...
36684         sta *a111
36686         sty *a112
36688         jsr a39963             ; ---> a$ in stringbereich kopieren
36691         lda *a111              ; l,z(a$)
36693         ldy *a112              ; h,...
36695         jsr a40106             ; ---> d(a$) aus string stack nehmen
36698         lda #l,a97
36700         ldy #h,a97
36702 a36702  sta *a80
36704         sty *a81
36706         sta *a34
36708         sty *a35
36710         jsr a40106             ; ---> d(a$) aus string stack nehmen
36713         jsr a36764             ; ---> prüfen, ob r-zeiger vorhanden
36716         bcc a36729             ; nein: ->
36718         ldy #0
36720         lda *a73               ; z(n$)
36722         sta (a34),y            ; in r-zeiger kopieren
36724         iny
36725         lda *a74
36727         sta (a34),y
36729 a36729  lda *a73
36731         sta *a34
36733         lda *a74
36735         sta *a35
36737         jsr a36764             ; ---> alten inhalt von n$ löschen?
36740         bcc a36751             ; nein: ->
36742         dey
```

```
36743          lda #255                ; r-zeiger von n$ := länge, 255
36745          sta (a34),y
36747          dey
36748          txa
36749          sta (a34),y
36751 a36751   ldy #2
36753 a36753   lda #a80
36755          jsr a1172               ; ---> lda (a),y
36758          sta (a73),y             ; d(n$) := d(a$)
36760          dey
36761          bpl a36753
36763          rts
36764
36764 ;---------------  S T R I N G   A U F   R - Z E I G E R   P R U E F E N
36764
36764 a36764   ldy #0
36766          jsr a1200               ; ---> lda (a34),y
36769          pha
36770          beq a36829              ; stringlänge = 0: ->
36772          iny
36773          jsr a1200               ; ---> lda (a34),y: l,stringadresse
36776          tax
36777          iny
36778          jsr a1200               ; ---> lda (a34),y: h,stringadresse
36781          cmp *a56                ; string oberhalb der
36783          bcc a36791              ; basic-obergrenze?
36785          bne a36829
36787          cpx *a55
36789          bcs a36829              ; ja: ->
36791 a36791   jsr a1200               ; ---> lda (a34),y: h,stringadresse
36794          cmp *a52                ; string unterhalb des
36796          bcc a36829              ; stringbereichs?
36798          bne a36804
36800          cpx *a51
36802          bcc a36829              ; ja: ->
36804 a36804   cmp *a123               ; string identisch mit ds$?
36806          bne a36812
36808          cpx *a122
36810          beq a36829              ; ja: ->
36812 a36812   stx *a34                ; zeiger (a34,a35)
36814          sta *a35                ; auf r-zeiger setzen
36816          pla
36817          tax
36818          clc
36819          adc *a34
36821          sta *a34
36823          bcc a36827
36825          inc *a35
36827 a36827   sec                     ; flag f 'r-zeiger vorhanden'
36828          rts
36829
36829 a36829   pla
36830          clc                     ; flag f 'kein r-zeiger'
36831          rts
36832
36832 ;------------------------------  P R I N T #
36832
36832 a36832   jsr a36838              ; ---> cmd
36835          jmp a37118              ; ---> clrch
```

```
36838 ;------------------------------  C M D
36838
36838 a36838  jsr a40324              ; ---> log adresse holen
36841         beq a36848              ; trennzeichen folgt: ->
36843         lda #',
36845         jsr a38035              ; ---> syntax check
36848 a36848  php
36849         stx *a19                ; i/o-kanal
36851         jsr a42903              ; ---> ckout
36854         plp
36855         jmp a36864              ; ---> print
36858
36858 ;------------------------------
36858
36858 a36858  jsr a37003              ; ---> string ausgeben
36861 a36861  jsr a1145               ; ---> chrgot
36864
36864 ;------------------------------  P R I N T
36864
36864 a36864  beq a36926              ; trennzeichen folgt: ->
36866         cmp #251                ; 'using'-token?
36868         bne a36873              ; nein: ->
36870         jmp a44791              ; ---> print using
36873
36873 a36873  beq a36942
36875         cmp #163                ; 'tab('-token?
36877         beq a36959              ; ja: ->
36879         cmp #166                ; 'spc('-token?
36881         clc
36882         beq a36959              ; ja: ->
36884         cmp #',
36886         beq a36943
36888         cmp #';
36890         beq a36986
36892         jsr a37676              ; ---> ausdruck auswerten
36895         bit *a13                ; string?
36897         bmi a36858              ; ja: ->
36899         jsr a42095              ; ---> fac in zahlenstring wandeln
36902         jsr a39796              ; ---> string übernehmen
36905         jsr a37003              ; ---> string ausgeben
36908         jsr a37030              ; ---> 'kursor nach rechts' ausgeben
36911         bne a36861              ; immer ->
36913
36913 ;----------------  E I N G A B E P U F F E R  A B S C H L I E S S E N
36913
36913 a36913  lda #0                  ; endemarke
36915         sta a512,x              ; in puffer schreiben
36918         ldx #l,a512-1
36920         ldy #h,a512-1
36922         lda *a19                ; i/o-kanal
36924         bne a36942              ; nicht tastatur: ->
36926 a36926  lda #13                 ; cr-kode
36928         jsr a37042              ; ---> ausgeben
36931         bit *a19                ; log adresse
36933         bpl a36940              ; kleiner als 128: ->
36935         lda #10                 ; lf-kode
36937         jsr a37042              ; ---> ausgeben
36940 a36940  eor #255
36942 a36942  rts
```

```
36943 ;------------------------------  Z E H N E R T A B
36943
36943 a36943  sec
36944         jsr a65520              ; ---> plot: kursorspalte nach y
36947         tya                     ; von kursorspalte
36948         sec
36949 a36949  sbc #10                 ; so oft 10 subtrahieren,
36951         bcs a36949              ; bis ergebnis negativ.
36953         eor #255                ; negieren des ergebnisses ergibt zahl
36955         adc #1                  ; der schritte bis tab-position
36957         bne a36981              ; immer ->
36959
36959 ;------------------------------  T A B ,  S P C
36959
36959 a36959  php                     ; tab: carry = 1, spc: carry = 0
36960         sec
36961         jsr a65520              ; ---> plot: kursorspalte nach y
36964         sty *a9                 ; kursorspalte merken
36966         jsr a40321              ; ---> parameter holen
36969         cmp #')                 ; nächstes zeichen ')'?
36971         bne a36992              ; nein: syntax error ->
36973         plp                     ; tab/spc-flag
36974         bcc a36982              ; spc: ->
36976         txa                     ; parameterwert
36977         sbc *a9                 ; - kursorspalte
36979         bcc a36986              ; negativ: fertig ->
36981 a36981  tax                     ; differenz als zähler verwenden
36982 a36982  inx
36983 a36983  dex                     ; und so oft 'kursor nach rechts'
36984         bne a36995              ; bis zähler null
36986 a36986  jsr a1139               ; ---> chrget
36989         jmp a36873              ; ---> print
36992
36992 a36992  jmp a38049              ; ---> 'syntax error'
36995
36995 a36995  jsr a37030              ; ---> 'kursor nach rechts' ausgeben
36998         bne a36983              ; immer ->
37000
37000 ;------------------------------  S T R I N G  A U S G E B E N
37000
37000 a37000  jsr a39796              ; ---> string übernehmen
37003 a37003  jsr a40014              ; ---> frestr
37006         tax                     ; stringlänge
37007         ldy #0
37009         inx
37010 a37010  dex
37011         beq a36942              ; string fertig ausgegeben: ->
37013         jsr a1200               ; ---> lda (a34),y: zeichen aus string
37016         jsr a37042              ; ---> ausgeben
37019         iny
37020         cmp #13                 ; cr-kode?
37022         bne a37010              ; nein: ->
37024         jsr a36940              ; ---> zeichen invertieren
37027         jmp a37010              ; ---> nächstes zeichen
```

```
37030 ;-------------------------------  Z E I C H E N A U S G A B E
37030
37030 a37030  lda *a19                ; i/o-kanal
37032         beq a37037              ; tastatur: ->
37034 a37034  lda #'
37036         .by 44
37037 a37037  lda #29                 ; 'kursor nach rechts'
37039         .by 44
37040 a37040  lda #'?
37042 a37042  jsr a42891              ; ---> ausgeben
37045         and #255
37047         rts
37048
37048 ;-------------------------------  G E T
37048
37048 a37048  jsr a39558              ; ---> fehler wenn direktmodus
37051         sta *a128
37053         cmp #'#
37055         beq a37067
37057         cmp #249                ; 'key'-token?
37059         bne a37083
37061         jsr a1139               ; ---> chrget
37064         jmp a37083              ; --->
37067
37067 a37067  jsr a1139               ; ---> chrget
37070         jsr a40324              ; ---> log adresse holen
37073         lda #',
37075         jsr a38035              ; ---> syntax check
37078         stx *a19                ; i/o-kanal := log adresse
37080         jsr a42918              ; ---> chkin
37083 a37083  ldx #1,a512+1
37085         ldy #h,a512+1
37087         lda #0                  ; endemarke
37089         sta a512+1              ; hinter erstes Pufferbyte
37092         lda #64                 ; flagwert f 'get'
37094         jsr a37208              ; ---> zeichen in variable bringen
37097         ldx *a19                ; i/o-kanal
37099         bne a37120              ; nicht tastatur: ->
37101         rts
37102
37102 ;-------------------------------  I N P U T #
37102
37102 a37102  jsr a40324              ; ---> log adresse holen
37105         lda #",
37107         jsr a38035              ; ---> syntax check
37110         stx *a19                ; i/o-kanal := log adresse
37112         jsr a42918              ; ---> chkin
37115         jsr a37143              ; ---> input
37118 a37118  lda *a19                ; i/o-kanal
37120 a37120  jsr a65484              ; ---> clrch
37123         ldx #0                  ; i/o-kanal := tastatur
37125         stx *a19
37127         rts
37128
37128 ;-------------------------------  I N P U T
37128
37128 a37128  cmp #'"
37130         bne a37143
37132         jsr a37966              ; ---> fragestring übernehmen
37135         lda #';
37137         jsr a38035              ; ---> syntax check
37140         jsr a37003              ; ---> string ausgeben
```

```
37143 a37143  jsr a39558              ; ---> fehler wenn direktmodus
37146         lda #',
37148         sta a512-1
37151 a37151  jsr a37186              ; ---> '?' ausgeben, eingabeschleife
37154         lda *a19                ; i/o-kanal
37156         beq a37171              ; tastatur: ->
37158         jsr a65463              ; ---> readst
37161         and #2                  ; timeout?
37163         beq a37171              ; nein: ->
37165         jsr a37118              ; ---> clrch, tastatur einschalten
37168         jmp a36272              ; ---> progrzgr auf trennzeichen
37171
37171 a37171  lda a512                ; erstes zeichen im puffer
37174         bne a37206              ; kein endezeichen: ->
37176         lda *a19                ; i/o-kanal
37178         bne a37151              ; nicht tastatur: ->
37180         jsr a36286              ; ---> nächstes trennzeichen suchen
37183         jmp a36275              ; ---> progrzeiger auf trennzeichen
37186
37186 a37186  lda *a19                ; i/o-kanal
37188         bne a37196              ; nicht tastatur: ->
37190         jsr a37040              ; ---> '?' ausgeben
37193         jsr a37034              ; ---> 'kursor nach rechts' ausgeben
37196 a37196  jmp a34906              ; ---> eingabeschleife
37199
37199 ;-------------------------------  R E A D
37199
37199 a37199  ldx *a65                ; l,data zeiger
37201         ldy *a66                ; h,...
37203         lda #152                ; flagwert f 'read'
37205         .by 44
37206 a37206  lda #0                  ; flagwert f 'input'
37208 a37208  sta *a17                ; eingabeflag
37210         stx *a67                ; l,eingabezeiger
37212         sty *a68                ; h,...
37214 a37214  jsr a38565              ; ---> variable suchen oder anlegen
37217         sta *a73                ; l,variablenzeiger
37219         sty *a74                ; h,...
37221         ldx #1
37223 a37223  lda *a59,x              ; programmzeiger
37225         sta *a75,x              ; retten
37227         lda *a67,x              ; und durch eingabezeiger ersetzen
37229         sta *a59,x
37231         dex
37232         bpl a37223
37234         jsr a1145               ; ---> chrgot
37237         bne a37288              ; kein endezeichen: ->
37239         bit *a17                ; eingabeflag
37241         bvc a37269              ; nicht 'get': ->
37243         lda *a128               ; letztes token
37245         cmp #249                ; 'key'?
37247         bne a37257              ; nein: ->
37249 a37249  jsr a42927              ; ---> getin
37252         tax                     ; tastenkode eingegeben?
37253         beq a37249              ; nein: warten ->
37255         bne a37260              ; ja: ->
37257
37257 a37257  jsr a42927              ; ---> getin
37260 a37260  sta a512                ; kode in puffer
37263         ldx #l,a512-1
37265         ldy #h,a512-1
37267         bne a37284              ; immer ->
```

```
37269 a37269  bpl a37274          ; 'input': ->
37271         jmp a37440          ; ---> fortsetzung 'read'
37274
37274 a37274  lda *a19            ; i/o-kanal
37276         bne a37281          ; nicht tastatur: ->
37278         jsr a37040          ; ---> '?' ausgeben
37281 a37281  jsr a37186          ; ---> '?', 'kursor nach rechts', eingabe
37284 a37284  stx *a59            ; progrzgr vor eingabepuffer setzen
37286         sty *a60
37288 a37288  jsr a1139           ; ---> chrget
37291         bit *a13            ; stringvariable?
37293         bpl a37344          ; nein: ->
37295         bit *a17            ; eingabeflag
37297         bvc a37308          ; nicht 'get': ->
37299         inx
37300         stx *a59            ; progrzgr inkrementieren
37302         lda #0
37304         sta *a7
37306         beq a37320
37308
37308 a37308  sta *a7
37310         cmp #'"
37312         beq a37321
37314         lda #':
37316         sta *a7
37318         lda #',
37320 a37320  clc
37321 a37321  sta *a8
37323         lda *a59            ; l,progrzgr
37325         ldy *a60            ; h,...
37327         adc #0              ; um carry erhöhen
37329         bcc a37332
37331         iny
37332 a37332  jsr a39802          ; ---> string übernehmen
37335         jsr a40390          ; ---> progrzgr hinter string setzen
37338         jsr a36529          ; ---> deskriptor in variablentabelle
37341         jmp a37352          ; --->
37344
37344 a37344  jsr a41855          ; ---> string-gk-wandlung
37347         lda *a14            ; integer-flag
37349         jsr a36505          ; ---> fac in variable übertragen
37352 a37352  jsr a1145           ; ---> chrgot
37355         beq a37416          ; trennzeichen: ->
37357         cmp #',
37359         beq a37416          ; komma: ->
37361         lda *a17            ; eingabeflag
37363         beq a37375          ; 'input': ->
37365         bmi a37371          ; 'read': ->
37367         ldx *a19            ; i/o-kanal
37369         bne a37379          ; nicht tastatur: ->
37371 a37371  ldx #22             ; 'type mismatch'
37373         bne a37381          ; immer ->
37375
37375 a37375  lda *a19            ; i/o-kanal
37377         beq a37384          ; tastatur: ->
37379 a37379  ldx #24             ; 'file data'
37381 a37381  jmp a34435          ; ---> fehlerausgang
```

```
37384 a37384  jsr a65359              ; ---> meldung ausgeben
37387         .by '?redo from start' 13 0
37405         lda a603               ; zeilenanfangsadresse
37408         ldy a604
37411         sta *a59               ; in programmzeiger bringen
37413         sty *a60
37415         rts
37416
37416 a37416  ldx #1
37418 a37418  lda *a59,x             ; programmzeiger
37420         sta *a67,x             ; in eingabezeiger bringen
37422         lda *a75,x             ; und ausgangswert wiederherstellen
37424         sta *a59,x
37426         dex
37427         bpl a37418
37429         jsr a1145              ; ---> chrgot
37432         beq a37482             ; trennzeichen:->
37434         jsr a38033             ; ---> chkcom
37437         jmp a37214             ; ---> nächste eingabe
37440
37440 a37440  jsr a36286             ; ---> offset zum trennzeichen nach y
37443         iny
37444         tax                    ; zeilenende?
37445         bne a37468             ; nein: ->
37447         ldx #13                ; 'out of data'
37449         iny
37450         jsr a1189              ; ---> lda (a59),y: h,linkadresse
37453         beq a37563             ; programmende: fehler ->
37455         iny
37456         jsr a1189              ; ---> lda (a59),y: l,zeilennummer
37459         sta *a63               ; l,data zeilennummer
37461         iny
37462         jsr a1189              ; ---> lda (a59),y: h,...
37465         iny
37466         sta *a64               ; h,data zeilennummer
37468 a37468  jsr a36275             ; ---> progrzgr um y erhöhen
37471         jsr a1145              ; ---> chrgot
37474         tax
37475         cpx #131               ; 'data'-token?
37477         bne a37440             ; nein: weitersuchen ->
37479         jmp a37288             ; ---> dateneingabe fortsetzen
37482
37482 a37482  lda *a67               ; eingabezeiger
37484         ldy *a68
37486         ldx *a17               ; eingabeflag
37488         bpl a37493             ; nicht 'read': ->
37490         jmp a36027             ; --->
37493
37493 a37493  ldy #0
37495         jsr a33109             ; ---> lda (a67),y: nächstes zeichen
37498         beq a37523             ; endezeichen: ->
37500         lda *a19               ; i/o-kanal
37502         bne a37523             ; nicht tastatur: ->
37504         jsr a65359             ; ---> meldung ausgeben
37507         .by '?extra ignored' 13 0
37523 a37523  rts
```

```
37524 ;------------------------------  N E X T
37524
37524 a37524  bne a37545              ; variablenname folgt: ->
37526         ldy #255
37528         bne a37550
37530
37530 a37530  ldy #18                 ; bytes im basic stack
37532         jsr a42866              ; ---> (a124,a125) um y erhöhen
37535         jsr a1145               ; ---> chrgot
37538         cmp #',
37540         bne a37651
37542         jsr a1139               ; ---> chrget
37545 a37545  jsr a38565              ; ---> variable suchen oder anlegen
37548         sta *a73                ; l,for-next variablenzeiger
37550 a37550  sty *a74                ; h,...
37552         ldy #129                ; 'for'-token
37554         sty *a2                 ; f suche im basic-stack speichern
37556         jsr a34929              ; ---> daten im stack suchen
37559         beq a37566              ; gefunden: ->
37561         ldx #10                 ; 'next without for'
37563 a37563  jmp a34435              ; ---> fehlerausgang
37566
37566 a37566  jsr a42857              ; ---> (a124,a125) := (a61,a62)
37569         lda *a61                ; (a,y) aus stepwert setzen
37571         clc
37572         adc #3
37574         ldy *a62
37576         bcc a37579
37578         iny
37579 a37579  jsr a41503              ; ---> step-wert nach fac
37582         ldy #8
37584         lda (a61),y             ; vorzeichen
37586         sta *a102               ; in vorzeichenbyte von fac
37588         ldy #1
37590         lda (a61),y
37592         pha
37593         tax
37594         iny
37595         lda (a61),y
37597         pha
37598         tay
37599         txa
37600         jsr a40603              ; ---> variablenwert zu fac addieren
37603         pla
37604         tay
37605         pla
37606         tax
37607         jsr a41561              ; ---> fac in basic stack übertragen
37610         lda *a61                ; (a,y) auf endwert setzen
37612         clc
37613         adc #9
37615         ldy *a62
37617         bcc a37620
37619         iny
37620 a37620  jsr a41696              ; ---> variablenwert mit endwert vergl
37623         ldy #8
37625         sec
37626         sbc (a61),y             ; endwert überschritten?
37628         beq a37530              ; ja: ->
```

```
37630          ldy #17
37632          lda (a61),y              ; schleifen-anfangsadresse
37634          sta *a59                 ; in programmzeiger
37636          dey
37637          lda (a61),y
37639          sta *a60
37641          dey
37642          lda (a61),y              ; anfangszeilennummer
37644          sta *a58                 ; als aktuelle zeilennummer
37646          dey                      ; speichern
37647          lda (a61),y
37649          sta *a57
37651 a37651   rts
37652
37652 ;------------------------------   F R M N U M
37652
37652 a37652   jsr a37676               ; ---> ausdruck auswerten
37655
37655 ;------------------------------   C H K N U M
37655
37655 a37655   clc                      ; prüfen ob numerisch
37656          bcc a37659
37658
37658 ;------------------------------   C H K S T R
37658
37658 a37658   sec                      ; prüfen ob string
37659 a37659   bit *a13                 ; string flag
37661          bmi a37666
37663          bcs a37668
37665 a37665   rts
37666
37666 a37666   bcs a37665
37668 a37668   ldx #22                  ; 'type mismatch'
37670          .by 44
37671 a37671   ldx #25                  ; 'formula too complex'
37673          jmp a34435               ; ---> fehlerausgang
37676
37676 ;------------------------------   F R M E V L
37676
37676 a37676   ldx *a59                 ; programmzeiger zurücksetzen
37678          bne a37682
37680          dec *a60
37682 a37682   dec *a59
37684          ldx #0
37686          .by 36
37687 a37687   pha
37688          txa
37689          pha
37690          tsx
37691          cpx #40                  ; platz im stack ausreichend?
37693          bcc a37671               ; nein: ->
37695          jsr a37908               ; ---> nächstes glied auswerten
37698          lda #0
37700          sta *a77                 ; operator-maske
37702 a37702   jsr a1145                ; ---> chrgot
```

```
37705 a37705  sec
37706         sbc #177              ; >, =, < ?
37708         bcc a37733
37710         cmp #3
37712         bcs a37733            ; nein: ->
37714         cmp #1
37716         rol a                 ; operator-maske:
37717         eor #1                ; >   =   <   >=  <=  <>   sonst
37719         eor *a77              ; 1   2   4   3   6   5     0
37721         cmp *a77
37723         bcc a37822
37725         sta *a77
37727         jsr a1139             ; ---> chrget
37730         jmp a37705            ; --->
37733
37733 a37733  ldx *a77              ; operator-maske = 0?
37735         bne a37781            ; nein: ->
37737         bcs a37865            ; +, -, *, /, , and, or ?
37739         adc #7
37741         bcc a37865            ; nein: ->
37743         adc *a13              ; string?
37745         bne a37750            ; nein: ->
37747         jmp a39898            ; ---> string-verkettung
37750
37750 a37750  adc #255              ; token wiederherstellen
37752         sta *a34              ; offset-zeiger erzeugen
37754         asl a
37755         adc *a34
37757         tay
37758 a37758  pla                   ; oberster wert im stack
37759         cmp a33875,y          ; < prio-flag der operation?
37762         bcs a37870            ; nein: ->
37764         jsr a37655            ; ---> fehler, wenn nicht numerisch
37767 a37767  pha
37768 a37768  jsr a37806            ; ---> routinen-aufruf
37771         pla
37772         ldy *a75
37774         bpl a37799
37776         tax
37777         beq a37868
37779         bne a37879
37781
37781 a37781  lsr *a13              ; stringflag rücksetzen
37783         txa                   ; operator-maske
37784         rol a
37785         ldx *a59              ; programmzeiger
37787         bne a37791            ; um 1 schritt zurücksetzen
37789         dec *a60
37791 a37791  dec *a59
37793         ldy #27               ; offset des prio-flags von <, =, >
37795         sta *a77
37797         bne a37758            ; immer ->
37799
37799 a37799  cmp a33875,y          ; vergleich mit prio-flag
37802         bcs a37879
37804         bcc a37767
```

```
37806 a37806  lda a33875+2,y          ; h,routinenadresse
37809         pha
37810         lda a33875+1,y          ; l,...
37813         pha
37814         jsr a37825              ; ---> operanden auf stack
37817         lda *a77                ; operator-maske
37819         jmp a37687              ; ---> zurück an den anfang
37822
37822 a37822  jmp a38049              ; ---> 'syntax error'
37825
37825 a37825  lda *a102               ; vorzeichen von fac
37827         ldx a33875,y            ; prio-flag
37830         tay
37831         clc
37832         pla                     ; rücksprungadresse
37833         adc #1
37835         sta *a34                ; in vektor (a34,a35) bringen
37837         pla
37838         adc #0
37840         sta *a35
37842         tya
37843         pha
37844         jsr a41632              ; ---> fac runden
37847         lda *a101               ; und auf stack legen
37849         pha
37850         lda *a100
37852         pha
37853         lda *a99
37855         pha
37856         lda *a98
37858         pha
37859         lda *a97
37861         pha
37862         jmp (a34)               ; ---> rücksprung
37865
37865 a37865  ldy #255
37867         pla
37868 a37868  beq a37905
37870 a37870  cmp #100
37872         beq a37877
37874         jsr a37655              ; ---> fehler, wenn nicht numerisch
37877 a37877  sty *a75
37879 a37879  pla                     ; arg vom stack holen
37880         lsr a
37881         sta *a18
37883         pla
37884         sta *a105
37886         pla
37887         sta *a106
37889         pla
37890         sta *a107
37892         pla
37893         sta *a108
37895         pla
37896         sta *a109
37898         pla
37899         sta *a110
37901         eor *a102
37903         sta *a111               ; vorzeichen von arg * fac
37905 a37905  lda *a97                ; exponent von fac als null-flag
37907         rts
```

```
37908 ;------------------------------  E V A L
37908
37908 a37908  jmp (a778)              ; ---> eval (a37911)
37911
37911 a37911  lda #0
37913         sta *a13                ; stringflag löschen
37915 a37915  jsr a1139               ; ---> chrget
37918         bcs a37923              ; keine ziffer: ->
37920 a37920  jmp a41855              ; ---> zahl nach fac
37923
37923 a37923  jsr a38714              ; ---> buchstabe?
37926         bcc a37931              ; nein: ->
37928         jmp a38061              ; ---> variable auswerten
37931
37931 a37931  cmp #255                ; pi?
37933         bne a37950              ; nein: ->
37935         lda #l,a37945
37937         ldy #h,a37945
37939         jsr a41505              ; ---> pi nach fac
37942         jmp a1139               ; ---> chrget
37945
37945 a37945  .by 130 73 15 218 161   ; pi
37950 ;
37950 a37950  cmp #'.
37952         beq a37920
37954         cmp #171                ; '-' token
37956         beq a38054
37958         cmp #170                ; '+' token
37960         beq a37915
37962         cmp #'"
37964         bne a37981
37966 a37966  lda *a59                ; programmzeiger
37968         ldy *a60
37970         adc #0                  ; um 1 erhöhen (carry = 1!)
37972         bcc a37975
37974         iny
37975 a37975  jsr a39796              ; ---> string übernehmen
37978         jmp a40390              ; ---> progrzgr hinter string setzen
37981
37981 a37981  cmp #168                ; 'not'-token?
37983         bne a38007              ; nein: ->
37985         ldy #24                 ; prio-flag offset
37987         bne a38056              ; immer ->
37989
37989 ;------------------------------  N O T
37989
37989 a37989  jsr a39046              ; ---> integer-gk wandlung
37992         lda *a101               ; integer in fac invertieren
37994         eor #255
37996         tay
37997         lda *a100
37999         eor #255
38001 a38001  jsr a39570              ; ---> fac vorbereiten
38004         jmp a41673              ; ---> integer-gk wandlung
38007
38007 a38007  cmp #165                ; 'fn'-token?
38009         bne a38014              ; nein: ->
38011         jmp a39646              ; ---> fn
38014
38014 a38014  cmp #180                ; funktionstoken?
38016         bcc a38021              ; nein: ->
38018         jmp a38297              ; ---> funktionsauswertung
```

```
38021 ;------------------------------  ( ... )  A U S W E R T E N
38021
38021 a38021  jsr a38030              ; ---> auf '(' prüfen
38024         jsr a37676              ; ---> ausdruck auswerten
38027 a38027  lda #')
38029         .by 44
38030 a38030  lda #'(
38032         .by 44
38033 a38033  lda #',
38035 a38035  ldy #0
38037         sta *a120               ; zeichenkode zwischenspeichern
38039         jsr a1189               ; ---> lda (a59),y: zeichen aus programm
38042         cmp *a120               ; vergleichen
38044         bne a38049              ; ungleich: fehler ->
38046         jmp a1139               ; ---> chrget
38049
38049 a38049  ldx #11                 ; 'syntax'
38051         jmp a34435              ; ---> fehlerausgang
38054
38054 a38054  ldy #21                 ; prio-flag offset f zeichenwechsel
38056 a38056  pla
38057         pla
38058         jmp a37768              ; ---> zurück in die auswertung
38061
38061 ;------------------------------  V A R I A B L E  A U S W E R T E N
38061
38061 a38061  jsr a38565              ; ---> variable suchen oder anlegen
38064         sta *a100               ; l,variablenzeiger
38066         sty *a101               ; h,...
38068         ldx *a69                ; variablenname
38070         ldy *a70
38072         lda *a13                ; string?
38074         beq a38145              ; nein: ->
38076         lda #0
38078         sta *a112               ; rundungsstelle von fac löschen
38080         cpx #'t                 ; variable ti$?
38082         bne a38120
38084         cpy #'I
38086         bne a38206              ; nein: ->
38088         lda *a100               ; zeigt variablenzeiger
38090         cmp #l,a1186            ; auf systemkonstante 0?
38092         bne a38206
38094         lda *a101
38096         cmp #h,a1186
38098         bne a38206              ; nein: ->
38100         jsr a38193              ; ---> rdtim: zeit nach fac
38103         sty *a94                ; = 0
38105         dey
38106         sty *a113               ; = 255
38108         ldy #6
38110         sty *a93
38112         ldy #36                 ; offset in umrechnungstabelle
38114         jsr a42234              ; ---> zeit in h, min, s umrechnen
38117         jmp a39792              ; ---> str$
38120
38120 a38120  cpx #'d                 ; variable ds$?
38122         bne a38206
38124         cpy #'S
38126         bne a38206              ; nein: ->
38128         jsr a38138              ; ---> ds$ einlesen, wenn nicht gültig
38131         lda *a122               ; l,adr(ds$)
38133         ldy *a123               ; h,...
38135         jmp a39796              ; ---> string übernehmen
```

```
38138 a38138  lda *a121             ; länge von ds$
38140         bne a38206            ; nicht null: ds$ gültig ->
38142         jmp a52431            ; ---> ds$ einlesen
38145
38145 a38145  bit *a14              ; integer?
38147         bpl a38164            ; nein: ->
38149         ldy #0
38151         jsr a1244             ; ---> lda (a100),y: h,integer
38154         tax
38155         iny
38156         jsr a1244             ; ---> lda (a100),y: l,integer
38159         tay
38160         txa
38161         jmp a38001            ; ---> integer-gk wandlung
38164
38164 a38164  lda *a101             ; zeigt variablenzeiger
38166         cmp #h,a1186          ; auf a1186 (konstante 0)?
38168         bne a38290
38170         lda *a100
38172         cmp #l,a1186
38174         bne a38290            ; nein: ->
38176         cpx #'t               ; variable ti?
38178         bne a38207
38180         cpy #'i
38182         bne a38290            ; nein: ->
38184         jsr a38193            ; ---> zeit nach fac
38187         tya
38188         ldx #160              ; exponent
38190         jmp a41684            ; ---> fac normalisieren
38193
38193 a38193  jsr a65502            ; ---> rdtim: zeit nach (a,x,y)
38196         stx *a100             ; und nach fac übertragen
38198         sty *a99
38200         sta *a101
38202         ldy #0
38204         sty *a98
38206 a38206  rts
38207
38207 a38207  cpx #'s               ; variable st?
38209         bne a38221
38211         cpy #'t
38213         bne a38290            ; nein: ->
38215         jsr a65463            ; ---> readst
38218         jmp a41665            ; ---> byte-gk wandlung
38221
38221 a38221  cpx #'d               ; variable ds?
38223         bne a38263
38225         cpy #'s
38227         bne a38290            ; nein: ->
38229         jsr a38138            ; ---> ds$ einlesen, wenn ungültig
38232         ldy #0
38234         lda #a122
38236         jsr a1172             ; ---> lda (a),y: 1. fehlerbyte
38239         and #15               ; unteres halbbyte isolieren
38241         asl a                 ; * 2
38242         sta *a15
38244         asl a
38245         asl a                 ; 1. fehlerbyte * 8
38246         adc *a15              ; + 1. fehlerbyte * 2
38248         sta *a15              ; ergibt zehnerstelle von ds
```

```
38250          iny
38251          lda #a122
38253          jsr a1172              ; ---> lda (a),y: 2. fehlerbyte
38256          and #15                ; unteres halbbyte isolieren
38258          adc *a15               ; + zehnerstelle ergibt ds
38260          jmp a41665             ; ---> byte-gk wandlung
38263
38263 a38263   cpx #'e                ; variable er oder el?
38265          bne a38290             ; nein: ->
38267          cpy #'r
38269          beq a38284             ; er: ->
38271          cpy #'l
38273          bne a38290             ; nicht el: ->
38275          lda a1265              ; l,fehlerzeilennummer
38278          ldy a1264              ; h,...
38281          jmp a39542             ; ---> integer-gk wandlung
38284
38284 a38284   lda a1263
38287          jmp a41665             ; ---> byte-gk wandlung
38290
38290 a38290   lda *a100              ; l,variablenzeiger
38292          ldy *a101              ; h,...
38294          jmp a41503             ; ---> variablenwert nach fac
38297
38297 ;------------------------------  F U N K T I O N  A U S W E R T E N
38297
38297 a38297   cmp #213               ; token > 212?
38299          bcs a38389             ; ja: keine funktion ->
38301          cmp #203               ; token < 203?
38303          bcc a38307             ; ja: ->
38305          sbc #1                 ; token 203 auslassen ('go')
38307 a38307   pha
38308          tax
38309          jsr a1139              ; ---> chrget
38312          cpx #211               ; 'instr'?
38314          beq a38324             ; ja: ->
38316          cpx #203               ; 'left$', 'mid$', 'right$'?
38318          bcs a38361
38320          cpx #200
38322          bcc a38361             ; nein: ->
38324 a38324   jsr a38030             ; ---> prüft ob '(' folgt
38327          jsr a37676             ; ---> ausdruck auswerten
38330          jsr a38033             ; ---> chkcom
38333          jsr a37658             ; ---> chkstr prüft ob string
38336          pla
38337          cmp #211               ; 'instr'?
38339          beq a38386             ; ja: ->
38341          tax
38342          lda *a101              ; h,zeiger auf string deskriptor
38344          pha
38345          lda *a100              ; l,...
38347          pha
38348          txa
38349          pha
38350          jsr a40324             ; ---> parameter holen
38353          pla
38354          tay
38355          txa
38356          pha
38357          tya
38358          jmp a38365             ; ---> funktionsroutine aufrufen
```

```
38361 a38361  jsr a38021              ; ---> (...) auswerten
38364         pla
38365 a38365  sec
38366         sbc #180                ; aus token tabellenoffset bilden
38368         asl a
38369         tay
38370         lda a33813+1,y          ; h,funktionsroutine
38373         sta *a86
38375         lda a33813,y            ; l,...
38378         sta *a85
38380         jsr a84                 ; ---> funktionsroutine
38383         jmp a37655              ; ---> fehler wenn nicht numerisch
38386
38386 a38386  jmp a45958              ; ---> instr
38389
38389 a38389  jmp a38049              ; ---> 'syntax error'
38392
38392 ;-------------------------------  O R
38392
38392 a38392  ldy #255
38394         .by 44
38395
38395 ;-------------------------------  A N D
38395
38395 a38395  ldy #0
38397         sty *a11
38399         jsr a39046              ; ---> gk-integer wandlung
38402         lda *a100
38404         eor *a11
38406         sta *a7
38408         lda *a101
38410         eor *a11
38412         sta *a8
38414         jsr a41601              ; ---> arg nach fac
38417         jsr a39046              ; ---> gk-integer wandlung
38420         lda *a101
38422         eor *a11
38424         and *a8
38426         eor *a11
38428         tay
38429         lda *a100
38431         eor *a11
38433         and *a7
38435         eor *a11
38437         jmp a38001              ; ---> integer-gk wandlung
38440
38440 ;-------------------------------  Z A H L E N V E R G L E I C H
38440
38440 a38440  jsr a37659              ; ---> variablentyp prüfen
38443         bcs a38464              ; string: ->
38445         lda *a110               ; vorzeichenbit
38447         ora #127                ; ins erste mantissenbyte
38449         and *a106               ; von fac bringen
38451         sta *a106
38453         lda #l,a105             ; l,arg
38455         ldy #h,a105             ; h,...
38457         jsr a41696              ; ---> arg mit fac vergleichen
38460         tax
38461         jmp a38515              ; ---> wahrheitswert nach fac
```

```
38464 ;------------------------------  S T R I N G V E R G L E I C H
38464
38464 a38464  lda #0
38466         sta *a13                ; stringflag rücksetzen
38468         dec *a77                ; operator-maske
38470         jsr a40014              ; ---> frestr
38473         sta *a97                ; stringlänge
38475         stx *a98                ; l,stringadresse
38477         sty *a99                ; h,...
38479         lda *a108               ; zeiger auf 2. string
38481         ldy *a109
38483         jsr a40018              ; ---> frestr
38486         stx *a108               ; l,adresse des 2. strings
38488         sty *a109               ; h,...
38490         tax
38491         sec
38492         sbc *a97                ; längenvergleich
38494         beq a38504              ; längen gleich: ->
38496         lda #1
38498         bcc a38504              ; 2. string kürzer: ->
38500         ldx *a97                ; länge des 1. strings
38502         lda #255
38504 a38504  sta *a102               ; vorzeichenbyte von fac
38506         ldy #255
38508         inx
38509 a38509  iny
38510         dex                     ; vergleich der beiden strings
38511         bne a38520              ; zeichen für zeichen
38513         ldx *a102
38515 a38515  bmi a38540
38517         clc
38518         bcc a38540
38520
38520 a38520  jsr a33157              ; ---> lda (a108),y
38523         pha
38524         jsr a33149              ; ---> lda (a98),y
38527         sta *a120
38529         pla
38530         cmp *a120
38532         beq a38509
38534         ldx #255                ; abschlu des vergleichs
38536         bcs a38540
38538         ldx #1
38540 a38540  inx
38541         txa
38542         rol a
38543         and *a18
38545         beq a38549
38547         lda #255
38549 a38549  jmp a41665              ; ---> byte-gk wandlung
38552
38552 ;------------------------------  V A R I A B L E N - V E R W A L T U N G
38552
38552 a38552  jsr a38033              ; ---> chkcom
38555
38555 ;------------------------------  D I M
38555
38555 a38555  tax
38556         jsr a38570              ; ---> variable dimensionieren
38559         jsr a1145               ; ---> chrgot
38562         bne a38552
38564         rts
```

```
38565 ;---------------------  V A R I A B L E  S U C H E N  /  A N L E G E N
38565
38565 a38565  ldx #0                  ; setzt dim-flag zurück
38567         jsr a1145               ; ---> chrgot
38570 a38570  stx *a12                ; dim-flag
38572 a38572  sta *a69                ; byte 1, variablenname
38574         jsr a1145               ; ---> chrgot
38577         jsr a38714              ; ---> buchstabe?
38580         bcs a38585              ; ja: ->
38582 a38582  jmp a38049              ; ---> 'syntax error'
38585
38585 a38585  ldx #0
38587         stx *a13                ; string-flag
38589         stx *a14                ; integer-flag
38591         jsr a1139               ; ---> chrget
38594         bcc a38601              ; ziffer: ->
38596         jsr a38714              ; ---> buchstabe?
38599         bcc a38612              ; nein: ->
38601 a38601  tax
38602 a38602  jsr a1139               ; ---> chrget
38605         bcc a38602              ; ziffer: ->
38607         jsr a38714              ; ---> buchstabe?
38610         bcs a38602              ; ja: ->
38612 a38612  cmp #'$                 ; stringvariable?
38614         bne a38622              ; nein: ->
38616         lda #255
38618         sta *a13                ; string-flag setzen
38620         bne a38638
38622
38622 a38622  cmp #'%                 ; integervariable?
38624         bne a38645              ; nein: ->
38626         lda *a16                ; integer zulässig?
38628         bne a38582              ; nein: syntax error ->
38630         lda #128
38632         sta *a14                ; integer-flag setzen
38634         ora *a69                ; bit 7 im 1. namensbyte setzen
38636         sta *a69
38638 a38638  txa
38639         ora #128                ; bit 7 im 2. namensbyte setzen
38641         tax
38642         jsr a1139               ; ---> chrget
38645 a38645  stx *a70                ; 2. byte variablenname
38647         sec
38648         ora *a16                ; integer-array-sperrflag
38650         sbc #'(                 ; feldvariable und zulässig?
38652         bne a38657              ; nein: ->
38654         jmp a39067              ; ---> feldvariable behandeln
38657
38657 a38657  ldy #0
38659         sty *a16                ; integer-array-sperre rücksetzen
38661         lda *a45                ; variablenanfang
38663         ldx *a46
38665 a38665  stx *a96                ; in suchzeiger übertragen
38667 a38667  sta *a95
38669         cpx *a48                ; feldvariablen-anfang erreicht?
38671         bne a38677
38673         cmp *a47
38675         beq a38724              ; ja: ->
```

```
38677 a38677  jsr a1233              ; ---> lda (a95),y
38680         sta *a120              ; variablennamen vergleichen
38682         lda *a69
38684         cmp *a120
38686         bne a38704
38688         iny
38689         jsr a1233              ; ---> lda (a95),y
38692         sta *a120
38694         lda *a70
38696         cmp *a120
38698         bne a38703             ; nicht gleich: ->
38700         jmp a38988             ; ---> variable gefunden
38703
38703 a38703  dey
38704 a38704  clc
38705         lda *a95               ; suchzeiger auf nächste
38707         adc #7                 ; variable setzen
38709         bcc a38667
38711         inx
38712         bne a38665             ; und weitersuchen ->
38714
38714 a38714  cmp #'a                ; buchstabe setzt carry,
38716         bcc a38723             ; sonst carry = 0
38718         sbc #91
38720         sec
38721         sbc #165
38723 a38723  rts
38724
38724 a38724  pla
38725         pha
38726         cmp #l,a38061+2        ; aufruf aus ausdrucksauswertung?
38728         bne a38772             ; nein: ->
38730 a38730  lda #l,a1186           ; (a,y) zeigt auf konstante 0
38732         ldy #h,a1186
38734         rts
38735
38735 a38735  cpy #'I
38737         beq a38730             ; ti$: ->
38739         cpy #'i
38741         bne a38792
38743         beq a38769             ; ti: ->
38745
38745 a38745  cpy #'S
38747         beq a38769             ; ds$: ->
38749         cpy #'s
38751         bne a38792
38753         beq a38769             ; ds: ->
38755
38755 a38755  cpy #'t
38757         bne a38792
38759         beq a38769             ; st: ->
38761
38761 a38761  cpy #'r
38763         beq a38769             ; er: ->
38765         cpy #'l
38767         bne a38792             ; nicht el: ->
38769 a38769  jmp a38049             ; ---> 'syntax error'
```

```
38772 a38772  lda *a69              ; 1. byte variablenname
38774         ldy *a70              ; 2. byte ...
38776         cmp #'t               ; auf reservierte namen prüfen
38778         beq a38735
38780         cmp #'s
38782         beq a38755
38784         cmp #'e
38786         beq a38761
38788         cmp #'d
38790         beq a38745
38792 a38792  lda *a47              ; feldvariablen-anfang
38794         ldy *a48
38796         sta *a95
38798         sty *a96
38800         lda *a49              ; feldvariablen-ende
38802         ldy *a50
38804         sta *a90
38806         sty *a91
38808         clc
38809         adc #7                ; feldvariablen-tabelle muss um
38811         bcc a38814            ; 7 bytes nach oben geschoben werden
38813         iny
38814 a38814  sta *a88
38816         sty *a89
38818         jsr a35008            ; ---> block verschieben
38821         lda *a88
38823         ldy *a89
38825         iny
38826         sta *a47              ; neuer feldvariablen-anfang
38828         sty *a48
38830         sta *a88              ; r-zeiger müssen bei den
38832         sty *a89              ; stringfeldern neu berechnet werden
38834 a38834  lda *a88
38836         ldx *a89
38838 a38838  cpx *a50              ; variablen-ende erreicht?
38840         bne a38848
38842         cmp *a49
38844         bne a38848
38846         beq a38968            ; ja: fertig ->
38848
38848 a38848  sta *a34              ; (a,x) zeigt auf feldkopf
38850         stx *a35
38852         ldy #0
38854         jsr a1200             ; ---> lda (a34),y: 1. namensbyte
38857         tax
38858         iny
38859         jsr a1200             ; ---> lda (a34),y: 2. namensbyte
38862         php
38863         iny
38864         jsr a1200             ; ---> lda (a34),y: feldlänge
38867         adc *a88              ; zu (a88,a89) addieren
38869         sta *a88
38871         iny
38872         jsr a1200             ; ---> lda (a34),y
38875         adc *a89
38877         sta *a89
38879         plp                   ; gk-variable?
38880         bpl a38834            ; ja: ->
38882         txa                   ; integer-variable?
38883         bmi a38834            ; ja: ->
```

```
38885          iny
38886          jsr a1200              ; ---> lda (a34),y: zahl der dimensionen
38889          ldy #0
38891          asl a                  ; * 2
38892          adc #5                 ; + 5
38894          adc *a34               ; zum zeiger addieren
38896          sta *a34               ; setzt den zeiger auf das erste
38898          bcc a38902             ; feldelement
38900          inc *a35
38902 a38902   ldx *a35
38904          cpx *a89               ; feldende erreicht?
38906          bne a38912
38908          cmp *a88
38910          beq a38838             ; ja: ->
38912 a38912   ldy #0
38914          jsr a1200              ; ---> lda (a34),y: länge des strings
38917          beq a38955             ; null: ->
38919          sta *a120
38921          iny
38922          jsr a1200              ; ---> lda (a34),y: l,stringadresse
38925          clc
38926          adc *a120              ; + länge
38928          sta *a90               ; = l,adresse des r-zeigers
38930          iny
38931          jsr a1200              ; ---> lda (a34),y: h,stringadresse
38934          adc #0                 ; + übertrag
38936          sta *a91               ; = h,adresse des r-zeigers
38938          ldy #0
38940          jsr a33161             ; ---> lda (a90),y: l,r-zeiger
38943          adc #7                 ; + 7
38945          sta (a90),y            ; = l,neuer r-zeiger
38947          iny
38948          jsr a33161             ; ---> lda (a90),y: h,r-zeiger
38951          adc #0                 ; + übertrag
38953          sta (a90),y            ; = h,neuer r-zeiger
38955 a38955   lda #3                 ; zeiger in stringfeld
38957          clc                    ; auf nächsten stringdeskriptor
38958          adc *a34               ; erhöhen
38960          sta *a34
38962          bcc a38902
38964          inc *a35
38966          bne a38902             ; und weitermachen ->
38968
38968 a38968   ldy #0
38970          lda *a69               ; 1. byte variablenname
38972          sta (a95),y            ; in variablenkopf bringen
38974          iny
38975          lda *a70               ; 2. byte variablenname
38977          sta (a95),y            ; in variablenkopf bringen
38979          lda #0
38981 a38981   iny
38982          sta (a95),y            ; alle variablenbytes
38984          cpy #6                 ; mit nullen füllen
38986          bne a38981
38988 a38988   lda *a95               ; (a,y) auf 1. variablenbyte setzen
38990          clc
38991          adc #2
38993          ldy *a96
38995          bcc a38998
38997          iny
38998 a38998   sta *a71                ; l,variablenzeiger
39000          sty *a72                ; h,...
39002          rts
```

```
39003 a39003  lda *a11              ; zahl der dimensionen
39005         asl a                 ; * 2
39006         adc #5                ; + 5 = länge des variablenkopfes
39008         adc *a95              ; zum zeiger auf feldkopf addieren
39010         ldy *a96
39012         bcc a39015
39014         iny
39015 a39015  sta *a88              ; ergibt zeiger auf erstes
39017         sty *a89              ; feldelement
39019         rts
39020
39020 a39020  .by 144 128 0 0 0     ; konstante 32768
39025
39025 a39025  jsr a39046            ; ---> gk-integer wandlung
39028         lda *a100             ; h,integer
39030         ldy *a101             ; l,...
39032         rts
39033
39033 a39033  jsr a1139             ; ---> chrget
39036         jsr a37676            ; ---> ausdruck auswerten
39039 a39039  jsr a37655            ; ---> fehler, wenn nicht numerisch
39042         lda *a102             ; vorzeichen von fac
39044         bmi a39059            ; negativ: fehler ->
39046 a39046  lda *a97              ; exponent von fac
39048         cmp #144              ; kleiner als 16?
39050         bcc a39064            ; ja: ->
39052         lda #l,a39020         ; konstante 32768
39054         ldy #h,a39020
39056         jsr a41696            ; ---> mit fac vergleichen
39059 a39059  bne a39064            ; ungleich: fehler ->
39061         jmp a39196            ; ---> gk-integer wandlung
39064
39064 a39064  jmp a41767            ; ---> 'illegal quantity error'
39067
39067 a39067  lda *a12              ; dim-flag
39069         ora *a14              ; integer-flag
39071         pha
39072         lda *a13              ; string-flag
39074         pha
39075         ldy #0
39077 a39077  tya
39078         pha
39079         lda *a70              ; 2. byte, variablenname
39081         pha
39082         lda *a69              ; 1. byte,...
39084         pha
39085         jsr a39033            ; ---> index auswerten
39088         pla
39089         sta *a69              ; variablennamen wiederherstellen
39091         pla
39092         sta *a70
39094         pla
39095         tay
39096         tsx
39097         lda a256+2,x          ; die beiden flags werden aus dem
39100         pha                   ; stack genommen und auf den stack
39101         lda a256+1,x          ; gelegt; an ihrem alten platz im
39104         pha                   ; stack wird der indexwert als
39105         lda *a100             ; integer abgelegt
39107         sta a256+2,x
39110         lda *a101
39112         sta a256+1,x
```

```
39115          iny
39116          sty *a11
39118          jsr a1145              ; ---> chrgot
39121          ldy *a11
39123          cmp #',                ; folgt komma?
39125          beq a39077             ; ja: nächsten index auswerten ->
39127          jsr a38027             ; ---> prüft ob ')' folgt
39130          pla
39131          sta *a13               ; flags wiederherstellen
39133          pla
39134          sta *a14
39136          and #127
39138          sta *a12
39140          ldx *a47               ; feldvariablen-anfang
39142          lda *a48
39144 a39144   stx *a95               ; in suchzeiger bringen
39146          sta *a96
39148          cmp *a50               ; variablen-ende erreicht?
39150          bne a39156
39152          cpx *a49
39154          beq a39226             ; ja: variable nicht gefunden ->
39156 a39156   ldy #0
39158          jsr a1233              ; ---> lda (a95),y: namen aus tabelle
39161          iny
39162          cmp *a69               ; mit gesuchtem namen vergleichen
39164          bne a39177
39166          jsr a1233              ; ---> lda (a95),y
39169          sta *a120
39171          lda *a70
39173          cmp *a120
39175          beq a39201             ; gleich: variable gefunden ->
39177 a39177   iny
39178          jsr a1233              ; ---> lda (a95),y: feldlänge aus header
39181          clc
39182          adc *a95               ; zum suchzeiger addieren
39184          tax
39185          iny
39186          jsr a1233              ; ---> lda (a95),y
39189          adc *a96
39191          bcc a39144             ; und weitersuchen ->
39193
39193 a39193   ldx #18                ; 'bad subscript'
39195          .by 44
39196 a39196   ldx #14                ; 'illegal quantity'
39198 a39198   jmp a34435             ; ---> fehlerausgang
39201
39201
39201 a39201   ldx #19                ; 'redim'd array'
39203          lda *a12               ; dim-flag mu^ null sein,
39205          bne a39198             ; sonst fehler ->
39207          jsr a39003             ; ---> zeiger auf erstes element setzen
39210          ldy #4
39212          jsr a1233              ; ---> lda (a95),y: dim aus header
39215          sta *a120
39217          lda *a11               ; und zahl der indices
39219          cmp *a120              ; gleich?
39221          bne a39193             ; nein: fehler ->
39223          jmp a39363             ; ---> feldelement aufsuchen
```

```
39226 a39226  jsr a39003            ; ---> suchzeiger hinter header setzen
39229         jsr a35107            ; ---> platz im speicher prüfen
39232         ldy #0
39234         sty *a114
39236         ldx #5                ; vorbesetzung f variablenlänge
39238         lda *a69              ; 1. byte variablenname
39240         sta (a95),y           ; in header bringen
39242         bpl a39245            ; keine integervariable: ->
39244         dex                   ; variablenlänge dekrementieren
39245 a39245  iny
39246         lda *a70              ; 2. byte variablenname
39248         sta (a95),y           ; in header bringen
39250         bpl a39254            ; keine stringvariable: ->
39252         dex                   ; variablenlänge um 2 vermindern
39253         dex
39254 a39254  stx *a113             ; var-länge: gk 5, str 3, int 2
39256         lda *a11              ; zahl der dimensionen
39258         iny
39259         iny
39260         iny
39261         sta (a95),y           ; in header bringen
39263 a39263  ldx #11               ; vorbelegung des dim-wertes
39265         lda #0
39267         bit *a12              ; aufruf durch 'dim'?
39269         bvc a39279            ; nein: ->
39271         pla                   ; dim-wert vom stack
39272         clc
39273         adc #1                ; + 1 (wegen wert 0)
39275         tax
39276         pla
39277         adc #0
39279 a39279  iny
39280         sta (a95),y           ; h,dim-wert in header
39282         iny
39283         txa
39284         sta (a95),y           ; l,...
39286         jsr a39471            ; ---> platz f dimension berechnen
39289         stx *a113             ; l,platzbedarf
39291         sta *a114             ; h,...
39293         ldy *a34              ; zeiger in header
39295         dec *a11              ; weitere dimension vorhanden?
39297         bne a39263            ; ja: ->
39299         adc *a89              ; feldlänge + feldanfangsadresse
39301         bcs a39406
39303         sta *a89              ; ergibt feld-endadresse
39305         tay
39306         txa
39307         adc *a88
39309         bcc a39314
39311         iny
39312         beq a39406            ; überlauf: out of memory ->
39314 a39314  jsr a35107            ; ---> platz im speicher prüfen
39317         sta *a49              ; l,variablenende
39319         sty *a50              ; h,...
39321         lda #0                ; alle feldbytes mit nullen füllen
39323         inc *a114
39325         ldy *a113
39327         beq a39334
39329 a39329  dey
39330         sta (a88),y
39332         bne a39329
```

```
39334 a39334  dec *a89
39336         dec *a114
39338         bne a39329
39340         inc *a89
39342         sec
39343         lda *a49              ; variablen-endadresse
39345         sbc *a95              ; - header-anfangsadresse
39347         ldy #2                ; = gesamte feldlänge
39349         sta (a95),y           ; l, feldlänge in header
39351         lda *a50
39353         iny
39354         sbc *a96
39356         sta (a95),y           ; h,...
39358         lda *a12              ; aufruf durch 'dim'?
39360         bne a39470            ; ja: fertig ->
39362         iny
39363 a39363  jsr a1233             ; ---> lda (a95),y: dim-wert
39366         sta *a11              ; aus header übernehmen
39368         lda #0
39370         sta *a113             ; multiplizier-register löschen
39372 a39372  sta *a114
39374         iny
39375         pla                   ; index vom stack
39376         tax
39377         sta *a100
39379         jsr a1233             ; ---> lda (a95),y: dimwert aus header
39382         sta *a120
39384         pla
39385         sta *a101
39387         cmp *a120             ; vergleichen
39389         bcc a39409            ; kleiner: ok ->
39391         bne a39403            ; grö^er: fehler ->
39393         iny
39394         jsr a1233             ; ---> lda (a95),y
39397         sta *a120
39399         cpx *a120
39401         bcc a39410
39403 a39403  jmp a39193            ; ---> 'bad subscript'
39406
39406 a39406  jmp a34433            ; ---> 'out of memory'
39409
39409 ;------------------------------  F E L D E L E M E N T   S U C H E N
39409
39409 a39409  iny
39410 a39410  lda *a114             ; multiplizierregister
39412         ora *a113
39414         clc
39415         beq a39427            ; leer: ->
39417         jsr a39471            ; ---> multiplizieren
39420         txa
39421         adc *a100
39423         tax
39424         tya
39425         ldy *a34
39427 a39427  adc *a101
39429         stx *a113
39431         dec *a11
39433         bne a39372
39435         sta *a114
39437         ldx #5                ; vorbesetzung f elementlänge
```

```
39439          lda *a69                ; 1. byte variablenname
39441          bpl a39444              ; keine integervariable: ->
39443          dex
39444 a39444   lda *a70                ; 2. byte variablenname
39446          bpl a39450              ; keine stringvariable: ->
39448          dex
39449          dex
39450 a39450   stx *a40                ; var-länge (gk 5, str 3, int 2)
39452          lda #0
39454          jsr a39482              ; ---> offset des feldelements berechn
39457          txa
39458          adc *a88                ; + adr des 1. feldelements
39460          sta *a71                ; = adr des gesuchten elements
39462          tya
39463          adc *a89
39465          sta *a72
39467          tay
39468          lda *a71
39470 a39470   rts
39471
39471 ;------------------------------  M U L T I P L I Z I E R R O U T I N E
39471
39471 a39471   sty *a34                ; y zwischenspeichern
39473          jsr a1233               ; ---> lda (a95),y
39476          sta *a40
39478          dey
39479          jsr a1233               ; ---> lda (a95),y
39482 a39482   sta *a41
39484          lda #16                 ; verschiebe-zähler
39486          sta *a93
39488          ldx #0
39490          ldy #0
39492 a39492   txa                     ; aus (y,x) und (a113,a114) wird je
39493          asl a                   ; ein 16-bit-schieberegister
39494          tax                     ; gebildet; dann wird (a40,a41)
39495          tya                     ; mit (a113,a114) multipliziert
39496          rol a                   ; und das ergebnis in (y,x) aufge-
39497          tay                     ; baut.
39498          bcs a39406              ; bei überlauf aus dem h-byte des
39500          asl *a113               ; ergebnisregisters folgt
39502          rol *a114               ; 'out of memory error'
39504          bcc a39517
39506          clc
39507          txa
39508          adc *a40
39510          tax
39511          tya
39512          adc *a41
39514          tay
39515          bcs a39406
39517 a39517   dec *a93
39519          bne a39492
39521          rts
39522
39522 ;------------------------------  F R E
39522
39522 a39522   lda *a13                ; string-flag
39524          beq a39529              ; keine stringvariable: ->
39526          jsr a40014              ; ---> frestr
39529 a39529   jsr a43348              ; ---> garbage collect
```

```
39532          sec
39533          lda *a51                ; stringbereich-anfang
39535          sbc *a49                ; - variablen-ende
39537          tay                     ; = freier speicherplatz
39538          lda *a52
39540          sbc *a50
39542 a39542   jsr a39570              ; ---> stringflag = 0, fac vorbereiten
39545          sec
39546          jmp a41678              ; ---> gk-zahl in fac erzeugen
39549
39549 ;------------------------------   P O S
39549
39549 a39549   sec
39550          jsr a65520              ; ---> plot: kursorspalte holen
39553 a39553   lda #0                  ; h-byte
39555          jmp a38001              ; ---> (a,y) als gk-zahl nach fac
39558
39558 ;------------------------------   D I R E K T M O D U S   P R U E F E N
39558
39558 a39558   bit *a129               ; run-modus?
39560          bmi a39470              ; ja: fertig ->
39562          ldx #21                 ; 'illegal direct'
39564          .by 44
39565 a39565   ldx #27                 ; 'undefined function'
39567          jmp a34435              ; ---> fehlerausgang
39570
39570 ;------------------------------   ( A , Y )   N A C H   F A C
39570
39570 a39570   ldx #0
39572          stx *a13                ; string-flag löschen
39574          sta *a98
39576          sty *a99
39578          ldx #144                ; exponent
39580          rts
39581
39581 ;------------------------------   D E F
39581
39581 a39581   jsr a39627              ; ---> fn-namen prüfen, zeiger setzen
39584          jsr a39558              ; ---> fehler wenn direktmodus
39587          jsr a38030              ; ---> prüfen ob '(' folgt
39590          lda #128                ; integervariable sperren
39592          sta *a16
39594          jsr a38565              ; ---> variable suchen oder anlegen
39597          jsr a37655              ; ---> numerischen ausdruck auswerten
39600          jsr a38027              ; ---> prüfen ob ')' folgt
39603          lda #178                ; '='-token
39605          jsr a38035              ; ---> syntax check
39608          pha                     ; 1. zeichen des fn-ausdrucks
39609          lda *a72                ; h,fn-variablenzeiger
39611          pha
39612          lda *a71                ; l,...
39614          pha
39615          lda *a60                ; h,programmzeiger
39617          pha
39618          lda *a59                ; l,...
39620          pha                     ; auf stack legen
39621          jsr a36272              ; ---> programmzeiger auf trennzeichen
39624          jmp a39742              ; ---> parameter in fn-namensvariable
```

```
39627 ;--------------------------------  F N
39627
39627 a39627  lda #165                  ; 'fn'-token
39629         jsr a38035                ; ---> syntax check
39632         ora #128                  ; integervariable sperren
39634         sta *a16
39636         jsr a38572                ; ---> variable suchen oder anlegen
39639         sta *a78                  ; l,fn-variablenzeiger
39641         sty *a79                  ; h,...
39643         jmp a37655                ; ---> prüfen ob numerisch
39646
39646 a39646  jsr a39627                ; ---> namen prüfen, zeiger setzen
39649         lda *a79                  ; zeiger auf fn-namensvariable
39651         pha                       ; auf stack legen
39652         lda *a78
39654         pha
39655         jsr a38021                ; ---> (...) auswerten
39658         jsr a37655                ; ---> prüfen ob numerisch
39661         pla
39662         sta *a78                  ; zeiger auf fn-namensvariable
39664         pla                       ; wiederherstellen
39665         sta *a79
39667         ldy #2
39669         jsr a33113                ; ---> lda (a78),y: l,fn-variablenzgr
39672         sta *a71
39674         tax
39675         iny
39676         jsr a33113                ; ---> lda (a78),y: h,...
39679         beq a39565                ; = 0: undefined function ->
39681         sta *a72
39683         iny
39684 a39684  jsr a33121                ; ---> lda (a71),y: wert d fn-variablen
39687         pha                       ; auf stack legen
39688         dey
39689         bpl a39684
39691         ldy *a72
39693         jsr a41561                ; ---> fac runden, in fn-var bringen
39696         lda *a60                  ; programmzeiger auf stack retten
39698         pha
39699         lda *a59
39701         pha
39702         jsr a33113                ; ---> lda (a78),y: zeiger auf fn-ausdr
39705         sta *a59                  ; in programmzeiger bringen
39707         iny
39708         jsr a33113                ; ---> lda (a78),y
39711         sta *a60
39713         lda *a72                  ; fn-variablenzeiger auf stack retten
39715         pha
39716         lda *a71
39718         pha
39719         jsr a37652                ; ---> fn-ausdruck auswerten
39722         pla
39723         sta *a78                  ; fn-variablenzeiger
39725         pla
39726         sta *a79
39728         jsr a1145                 ; ---> chrgot
39731         beq a39736                ; trennzeichen: ok ->
39733         jmp a38049                ; ---> 'syntax error'
```

```
39736 a39736  pla                     ; programmzeiger wiederherstellen
39737         sta *a59
39739         pla
39740         sta *a60
39742 a39742  ldy #0                  ; alten wert der fn-variable
39744         pla                     ; wiederherstellen
39745         sta (a78),y
39747         pla
39748         iny
39749         sta (a78),y
39751         pla
39752         iny
39753         sta (a78),y
39755         pla
39756         iny
39757         sta (a78),y
39759         pla
39760         iny
39761         sta (a78),y
39763         rts
39764
39764 ;---------------------   S T R I N G - P L A T Z R E S E R V I E R U N G
39764
39764 a39764  ldx *a100               ; zeiger auf string-deskriptor
39766         ldy *a101
39768         stx *a80                ; umspeichern
39770         sty *a81
39772 a39772  jsr a43270              ; ---> stringbereich reservieren
39775         stx *a98                ; l,stringadresse
39777         sty *a99                ; h,...
39779         sta *a97                ; länge des strings
39781         rts
39782
39782 ;------------------------------   S T R $
39782
39782 a39782  jsr a37655              ; ---> prüfen ob numerisch
39785         ldy #0
39787         jsr a42097              ; ---> gk in zahlenstring wandeln
39790         pla
39791         pla
39792 a39792  lda #l,a256-1           ; l,zahlenstring-puffer
39794         ldy #h,a256-1           ; h,...
39796
39796 ;------------------------------   S T R I N G - Ü B E R N A H M E
39796
39796 a39796  ldx #'"
39798         stx *a7
39800         stx *a8
39802 a39802  sta *a111               ; l,stringadresse
39804         sty *a112               ; h,...
39806         sta *a98
39808         sty *a99
39810         ldy #255
39812 a39812  iny
39813         jsr a1222               ; ---> lda (a111),y: nächstes zeichen
39816         beq a39830              ; endemarke: ->
39818         cmp *a7                 ; auf delimiter prüfen
39820         beq a39826
39822         cmp *a8
39824         bne a39812
```

```
39826 a39826  cmp #'"                 ; und auf gänsefu
39828         beq a39831
39830 a39830  clc
39831 a39831  sty *a97                ; stringlänge
39833         tya
39834         adc *a111               ; + string-anfangsadresse
39836         sta *a113               ; = string-endadresse + 1
39838         ldx *a112
39840         bcc a39843
39842         inx
39843 a39843  stx *a114
39845         tya
39846         jsr a39764              ; ---> stringbereich reservieren
39849         ldx *a111               ; l,stringadresse
39851         ldy *a112               ; h,...
39853         jsr a39980              ; ---> string in bereich kopieren
39856
39856 ;-----------------   D E S K R I P T O R   I N   S T R I N G - S T A C K
39856
39856 a39856  ldx *a22                ; string stack zeiger
39858         cpx #a34                ; stack voll?
39860         bne a39867              ; nein: ->
39862         ldx #25                 ; 'formula too complex'
39864         jmp a34435              ; ---> fehlerausgang
39867
39867 a39867  lda *a97                ; stringlänge
39869         sta *a0,x               ; in string-stack
39871         lda *a98                ; l,stringadresse
39873         sta *a0+1,x
39875         lda *a99                ; h,...
39877         sta *a0+2,x             ; in string-stack
39879         ldy #0
39881         stx *a100               ; l,adr des deskriptors im stack
39883         sty *a101               ; h,...
39885         sty *a112
39887         dey
39888         sty *a13                ; string-flag setzen
39890         stx *a23                ; zeiger auf letzten stack-eintrag
39892         inx
39893         inx
39894         inx
39895         stx *a22                ; zeiger auf neuen stack-eintrag
39897         rts
39898
39898 ;------------------------------   S T R I N G - V E R K E T T U N G
39898
39898 a39898  lda *a101               ; zeiger auf deskriptor des 1. strings
39900         pha                     ; auf stack legen
39901         lda *a100
39903         pha
39904         jsr a37908              ; ---> adr des 2. deskriptors holen
39907         jsr a37658              ; ---> prüfen ob string
39910         pla
39911         sta *a111               ; zeiger auf deskriptor des 1. strings
39913         pla
39914         sta *a112
39916         ldy #0
39918         jsr a1222               ; ---> lda (a111),y: länge des 1. strings
39921         sta *a120
39923         jsr a1244               ; ---> lda (a100),y: länge des 2. strings
```

```
39926          clc
39927          adc *a120              ; + länge des 1. strings
39929          bcc a39934             ; kein überlauf: ->
39931          jmp a52300             ; ---> 'string too long'
39934
39934 a39934   jsr a39764             ; ---> bereich f gesamtstring reservieren
39937          jsr a39963             ; ---> 1. string in bereich kopieren
39940          lda *a80               ; zeiger auf 2. deskriptor
39942          ldy *a81
39944          jsr a40018             ; ---> frestr
39947          jsr a39984             ; ---> 2. string hinter 1. string kopieren
39950          lda *a111              ; zeiger auf 1. deskriptor
39952          ldy *a112
39954          jsr a40018             ; ---> frestr
39957          jsr a39856             ; ---> deskriptor in stack
39960          jmp a37702             ; ---> zurück in die auswertung
39963
39963 ;------------------------------  S T R I N G - K O P I E R R O U T I N E
39963
39963 a39963   ldy #0
39965          jsr a1222              ; ---> lda (a111),y: stringlänge
39968          pha                    ; auf stack legen
39969          iny
39970          jsr a1222              ; ---> lda (a111),y: l,stringadresse
39973          tax
39974          iny
39975          jsr a1222              ; ---> lda (a111),y: h,...
39978          tay
39979          pla
39980 a39980   stx *a34               ; l,stringadresse
39982          sty *a35               ; h,...
39984 a39984   tay                   ; stringlänge
39985          beq a39998             ; = 0: fertig ->
39987          pha
39988 a39988   dey
39989          jsr a1200              ; ---> lda (a34),y: zeichen aus string
39992          sta (a53),y            ; in stringbereich kopieren
39994          tya                    ; stringende erreicht?
39995          bne a39988             ; nein: ->
39997          pla                    ; stringlänge
39998 a39998   clc
39999          adc *a53               ; + bereichsanfang
40001          sta *a53               ; = stringende
40003          bcc a40007
40005          inc *a54
40007 a40007   rts
40008
40008 ;------------------------------  F R E S T R
40008
40008 a40008   jsr a37676             ; ---> ausdruck auswerten
40011 a40011   jsr a37658             ; ---> prüfen ob string
40014 a40014   lda *a100              ; l,deskriptor zeiger
40016          ldy *a101              ; h,...
40018 a40018   sta *a34
40020          sty *a35
40022          jsr a40106             ; ---> deskriptor aus stack entfernen
40025          bne a40084             ; war nicht im stack: ->
40027          jsr a36764             ; ---> r-zeiger vorhanden?
40030          bcc a40084             ; nein: ->
40032          dey
```

```
40033          lda #255              ; string ungültig machen
40035          sta (a34),y           ; durch r-zeiger = länge, 255
40037          dey
40038          txa
40039          sta (a34),y
40041          pha                   ; stringlänge
40042          eor #255              ; invertieren
40044          sec                   ; + 1 ergibt negative stringlänge
40045          adc *a34              ; stringlänge von adresse
40047          ldy *a35              ; des r-zeigers subtrahieren
40049          bcs a40052
40051          dey
40052 a40052   sta *a34              ; l,stringadresse
40054          sty *a35              ; h,...
40056          tax
40057          pla
40058          cpy *a52              ; liegt string am unteren ende des
40060          bne a40122            ; stringspeichers?
40062          cpx *a51
40064          bne a40122            ; nein: ->
40066          pha
40067          sec
40068          adc *a51              ; stringbereich-anfangszeiger
40070          sta *a51              ; um stringlänge + 1
40072          bcc a40076            ; hinaufsetzen
40074          inc *a52
40076 a40076   inc *a51              ; und nochmal inkrementieren
40078          bne a40082            ; wegen r-zeiger
40080          inc *a52
40082 a40082   pla                   ; stringlänge
40083          rts
40084
40084 a40084   ldy #0
40086          jsr a1200             ; ---> lda (a34),y: stringlänge
40089          pha
40090          iny
40091          jsr a1200             ; ---> lda (a34),y: l,stringadresse
40094          tax
40095          iny
40096          jsr a1200             ; ---> lda (a34),y: h,...
40099          tay
40100          stx *a34              ; l,stringadresse
40102          sty *a35              ; h,...
40104          pla                   ; stringlänge
40105          rts
40106
40106 a40106   cpy *a24              ; deskriptor im string-stack?
40108          bne a40122
40110          cmp *a23
40112          bne a40122            ; nein: ->
40114          sta *a22              ; stack zeiger
40116          sbc #3                ; - 3
40118          sta *a23              ; = zeiger auf letzten string
40120          ldy #0                ; setzt zero-flag
40122 a40122   rts
```

```
40123 ;------------------------------  C H R $
40123
40123 a40123  jsr a40327              ; ---> getbyt: byte übernehmen
40126         txa
40127         pha                     ; und auf stack legen
40128         lda #1                  ; als stringlänge
40130         jsr a39772              ; ---> stringbereich reservieren
40133         pla                     ; byte vom stack
40134         ldy #0
40136         sta (a98),y             ; in stringbereich kopieren
40138 a40138  pla                     ; rücksprungadresse vom stack nehmen
40139         pla
40140         jmp a39856              ; ---> deskriptor in stack bringen
40143
40143 ;------------------------------  L E F T $
40143
40143 a40143  jsr a40262              ; ---> stringadr u parameter v stack
40146         pha                     ; left$-parameter merken
40147         jsr a33153              ; ---> lda (a80),y: stringlänge
40150         sta *a120
40152         pla                     ; left$-parameter
40153         cmp *a120               ; kleiner als stringlänge?
40155         tya                     ; = 0
40156 a40156  bcc a40163              ; ja: ->
40158         jsr a33153              ; ---> lda (a80),y: stringlänge
40161         tax                     ; als left$-parameter nehmen
40162         tya                     ; = 0
40163 a40163  pha                     ; als position des 1. elements
40164 a40164  txa                     ; left$-parameter
40165 a40165  pha
40166         jsr a39772              ; ---> stringbereich reservieren
40169         lda *a80                ; l,deskriptor-zeiger
40171         ldy *a81                ; h,...
40173         jsr a40018              ; ---> frestr
40176         pla                     ; länge des neuen strings
40177         tay
40178         pla                     ; nr. des 1. elements (bei left$: 0)
40179         clc
40180         adc *a34                ; + adresse des gegebenen strings
40182         sta *a34                ; = adresse des neuen strings
40184         bcc a40188
40186         inc *a35
40188 a40188  tya
40189         jsr a39984              ; ---> string in stringbereich kopieren
40192         jmp a39856              ; ---> deskriptor in stack bringen
40195
40195 ;------------------------------  R I G H T $
40195
40195 a40195  jsr a40262              ; ---> stringadr u parameter v stack
40198         pha                     ; right$-parameter
40199         jsr a33153              ; ---> lda (a80),y: stringlänge
40202         sta *a120
40204         pla                     ; right$-parameter
40205         clc                     ; - 1
40206         sbc *a120               ; - stringlänge
40208         eor #255                ; invertiert = nr. des 1. elements
40210         jmp a40156              ; ---> weiter wie bei left$
```

```
40213 ;------------------------------  M I D $
40213
40213 a40213  lda #255
40215         sta *a101
40217         jsr a1145               ; ---> chrgot
40220         cmp #')                 ; 2. parameter vorhanden?
40222         beq a40230              ; nein: ->
40224         jsr a38033              ; ---> chkcom
40227         jsr a40324              ; ---> getbyt: 2. parameter holen
40230 a40230  jsr a40262              ; ---> stringadr u 1. param holen
40233         beq a40318              ; 1. parameter = 0: fehler ->
40235         dex
40236         txa
40237         pha                     ; nr. des 1. elements im gegebenen str
40238         ldx #0
40240         pha
40241         jsr a33153              ; ---> lda (a80),y: länge des strings
40244         sta *a120
40246         pla                     ; nr. des 1. elements von mid$(...)
40247         clc
40248         sbc *a120               ; - länge des gegebenen strings
40250         bcs a40164              ; nicht negativ: ->
40252         eor #255                ; neue stringlänge berechnen
40254         cmp *a101
40256         bcc a40165              ; kleiner als 255: ->
40258         lda *a101               ; = 255
40260         bcs a40165              ; immer ->
40262
40262 a40262  jsr a38027              ; ---> prüfen ob ')' folgt
40265         pla                     ; h,letzte rücksprungadresse
40266         tay
40267         pla                     ; l,...
40268         sta *a85
40270         pla                     ; h,rücksprungadr von a38380 jsr a84
40271         pla                     ; l,...
40272         pla                     ; 1. parameter
40273         tax
40274         pla                     ; l,adresse des string-deskriptors
40275         sta *a80
40277         pla                     ; h,...
40278         sta *a81
40280         lda *a85                ; letzte rücksprungadresse
40282         pha                     ; auf stack zurücklegen
40283         tya
40284         pha
40285         ldy #0
40287         txa                     ; 1. parameter
40288         rts
40289
40289 ;------------------------------  L E N
40289
40289 a40289  jsr a40295              ; ---> frestr, string-flag rücksetzen
40292         jmp a39553              ; ---> byte nach fac
40295
40295 a40295  jsr a40011              ; ---> frestr
40298         ldx #0
40300         stx *a13                ; string-flag
40302         tay                     ; stringlänge
40303         rts
```

```
40304 ;------------------------------  A S C
40304
40304 a40304  jsr a40295              ; ---> frestr, string-flag rücksetzen
40307         beq a40315              ; stringlänge = 0: ->
40309         ldy #0
40311         jsr a1200               ; ---> lda (a34),y: 1. zeichen aus string
40314         tay
40315 a40315  jmp a39553              ; ---> byte nach fac
40318
40318 ;------------------------------
40318
40318 a40318  jmp a39196              ; ---> 'illegal quantity'
40321
40321 ;------------------------------  G E T B Y T
40321
40321 a40321  jsr a1139               ; ---> chrget
40324 a40324  jsr a37652              ; ---> numerischen ausdruck auswerten
40327 a40327  jsr a39039              ; ---> gk-integer, fehler wenn negativ
40330         ldx *a100               ; h-byte des ausdrucks
40332         bne a40318              ; nicht null: fehler ->
40334         ldx *a101               ; l-byte des ausdrucks = ergebnis
40336         jmp a1145               ; ---> chrgot
40339
40339 ;------------------------------  V A L
40339
40339 a40339  jsr a40295              ; ---> frestr, string-flag rücksetzen
40342         beq a40399              ; stringlänge = 0: ->
40344         ldx *a59                ; programmzeiger
40346         ldy *a60
40348         stx *a113               ; retten
40350         sty *a114
40352         ldx *a34                ; string-anfangsadresse
40354         stx *a59                ; in programmzeiger bringen
40356         clc
40357         adc *a34                ; stringlänge + anfangsadresse
40359         sta *a36                ; = string-endadresse
40361         ldx *a35
40363         stx *a60
40365         bcc a40368
40367         inx
40368 a40368  stx *a37
40370         ldy #0
40372         jsr a1211               ; ---> lda (a36),y: zeichen nach string
40375         pha                     ; merken
40376         tya
40377         sta (a36),y             ; und durch kode 0 ersetzen
40379         jsr a1145               ; ---> chrgot
40382         jsr a41855              ; ---> string-gk-wandlung, gk nach fac
40385         pla                     ; zeichen nach string
40386         ldy #0
40388         sta (a36),y             ; wiederherstellen
40390 a40390  ldx *a113               ; programmzeiger wiederherstellen
40392         ldy *a114
40394         stx *a59
40396         sty *a60
40398         rts
40399
40399 a40399  jmp a40747              ; ---> fac löschen
```

```
40402 ;--------------------------  G E T A D R ,  C H K C O M ,  G E T B Y T
40402
40402 a40402  jsr a37652            ; ---> numerischen ausdruck auswerten
40405         jsr a40420            ; ---> getadr
40408 a40408  jsr a38033            ; ---> chkcom
40411         jmp a40324            ; ---> getbyt
40414
40414 ;------------------------------  G E T A D R
40414
40414 a40414  jsr a38033            ; ---> chkcom
40417 a40417  jsr a37652            ; ---> numerischen ausdruck auswerten
40420 a40420  lda *a102             ; vorzeichen von fac
40422         bmi a40318            ; negativ: fehler ->
40424 a40424  lda *a97              ; exponent von fac
40426         cmp #145              ; gröer als 16?
40428         bcs a40318            ; ja: feher ->
40430         jsr a41767            ; ---> gk-integer-wandlung
40433         lda *a100             ; h,integer
40435         ldy *a101             ; l,...
40437         sty *a20              ; l,adresse
40439         sta *a21              ; h,...
40441         rts
40442
40442 ;------------------------------  P E E K
40442
40442 a40442  lda *a21              ; adresse auf stack retten
40444         pha
40445         lda *a20
40447         pha
40448         jsr a40420            ; ---> getadr
40451         ldy #0
40453         jsr a33117            ; ---> lda (a20),y: peek-wert
40456         tay
40457         pla                   ; adresse wiederherstellen
40458         sta *a20
40460         pla
40461         sta *a21
40463         jmp a39553            ; ---> byte nach fac
40466
40466 ;------------------------------  P O K E
40466
40466 a40466  jsr a40402            ; ---> getadr, chkcom, getbyt
40469         txa                   ; byte
40470         ldy #0
40472         sta (a20),y           ; an poke-adresse speichern
40474         rts
40475
40475 ;------------------------------  D E C
40475
40475 a40475  jsr a40295            ; ---> frestr, string-flag rücksetzen
40478         sta *a36              ; stringlänge
40480         ldy #0                ; zeiger in string initialisieren
40482         sty *a37              ; stellenzähler
40484         sty *a113
40486         sty *a114
40488 a40488  cpy *a36              ; string-ende erreicht?
40490         beq a40544            ; ja: ->
40492         jsr a1200             ; ---> lda (a34),y
40495         iny
```

```
40496          cmp #'                  ; leerstellen überlesen
40498          beq a40488
40500          inc *a37
40502          ldx *a37                ; stellenzähler
40504          cpx #5                  ; mehr als vier stellen: fehler ->
40506          beq a40551
40508          cmp #'0                 ; zulässige hex-ziffer?
40510          bcc a40551
40512          cmp #':
40514          bcc a40526
40516          cmp #'a
40518          bcc a40551
40520          cmp #'g
40522          bcs a40551              ; nein: fehler
40524          sbc #7
40526 a40526   sbc #47                 ; hex-ziffer um vier bits nach
40528          asl a                   ; links schieben
40529          asl a
40530          asl a
40531          asl a
40532          ldx #4
40534 a40534   asl a                   ; und in das ergebnisregister
40535          rol *a113               ; einrotieren
40537          rol *a114
40539          dex                     ; schon 4 bits einrotiert?
40540          bne a40534              ; nein: ->
40542          beq a40488              ; immer ->
40544
40544 a40544   ldy *a113                ; integer aus ergebnisregister
40546          lda *a114
40548          jmp a39542              ; ---> gk in fac erzeugen
40551
40551 a40551   jmp a39196              ; ---> 'illegal quantity'
40554
40554 ;-------------------------------  W A I T
40554
40554 a40554   jsr a40402              ; ---> getadr, chkcom, getbyt
40557          stx *a73                ; 1. byte
40559          ldx #0
40561          jsr a1145               ; ---> chrgot
40564          beq a40569              ; trennzeichen: ->
40566          jsr a40408              ; ---> chkcom, getbyt
40569 a40569   stx *a74                ; 2. byte
40571          ldy #0
40573 a40573   jsr a33117              ; ---> lda (a20),y: byte unter adresse
40576          eor *a74                ; exclusiv-or mit 2. byte
40578          and *a73                ; and mit 1. byte
40580          beq a40573              ; ergebnis = 0: warten ->
40582 a40582   rts
40583
40583 ;-------------------------------  F A C  :=  A R G - F A C
40583
40583 a40583   lda *a102                ; vorzeichen von fac
40585          eor #255                ; invertieren
40587          sta *a102
40589          eor *a110               ; mit vorzeichen von arg 'multiplizieren'
40591          sta *a111
40593          lda *a97                ; exponent von fac (als null-flag)
40595          jmp a40606              ; ---> fac := arg + fac
```

```
40598 ;------------------------------
40598
40598 a40598  jsr a40909              ; ---> exponenten von fac und arg gleichm
40601         bcc a40663              ; immer ->
40603
40603 ;------------------------------  F A C  :=  V A R I A B L E + F A C
40603
40603 a40603  jsr a41223              ; ---> konst von (a,y) in ram nach arg
40606
40606 ;------------------------------  F A C  :=  A R G + F A C
40606
40606 a40606  bne a40611              ; fac nicht gleich null: ->
40608         jmp a41601              ; ---> fac := arg
40611
40611 a40611  ldx *a112               ; rundungsstelle von fac
40613         stx *a86
40615         ldx #a105               ; offset-zeiger auf arg
40617         lda *a105               ; exponent von arg
40619 a40619  tay
40620         beq a40582              ; arg = 0: fertig ->
40622         sec
40623         sbc *a97                ; - exponent von fac
40625         beq a40663              ; exponenten gleich: ->
40627         bcc a40647              ; exponent von fac grö^er: ->
40629         sty *a97                ; exp von fac := exp von arg
40631         ldy *a110               ; vorzeichen von arg
40633         sty *a102               ; in fac übernehmen
40635         eor #255                ; differenz der exponenten invertieren
40637         adc #0                  ; + 1 (carry!)
40639         ldy #0
40641         sty *a86
40643         ldx #a97                ; offset-zeiger für fac
40645         bne a40651              ; immer ->
40647
40647 a40647  ldy #0
40649         sty *a112               ; rundungsstelle von fac
40651 a40651  cmp #249                ; exp-differenz > 7?
40653         bmi a40598              ; ja: egalisieren (bytes) ->
40655         tay
40656         lda *a112               ; rundungsstelle von fac
40658         lsr *a0+1,x
40660         jsr a40932              ; ---> egalisieren (bits)
40663 a40663  bit *a111               ; vorzeichen gleich?
40665         bpl a40754              ; ja: mantissen addieren ->
40667         ldy #a97                ; offset-zeiger für fac
40669         cpx #a105               ; ist x offset-zeiger für arg?
40671         beq a40675              ; ja: ->
40673         ldy #a105               ; y := offset-zeiger für arg
40675 a40675  sec                     ; beginn der mantissen-subtraktion
40676         eor #255
40678         adc *a86
40680         sta *a112               ; rundungsstelle
40682         lda a0+4,y
40685         sbc *a0+4,x
40687         sta *a101
40689         lda a0+3,y
40692         sbc *a0+3,x
40694         sta *a100
40696         lda a0+2,y
40699         sbc *a0+2,x
40701         sta *a99
```

```
40703          lda a0+1,y
40706          sbc *a0+1,x
40708          sta *a98
40710 a40710   bcs a40715              ; kein unterlauf: ->
40712          jsr a40827              ; ---> mantisse von fac negieren
40715 a40715   ldy #0                  ; fac linksbündig normalisieren
40717          tya
40718          clc
40719 a40719   ldx *a98                ; höchste stelle
40721          bne a40797              ; nicht null: ->
40723          ldx *a99                ; mantisse um 1 byte nach links
40725          stx *a98                ; schieben
40727          ldx *a100
40729          stx *a99
40731          ldx *a101
40733          stx *a100
40735          ldx *a112
40737          stx *a101
40739          sty *a112               ; rundungsstelle := 0
40741          adc #8                  ; es wurde um 8 bits verschoben
40743          cmp #32                 ; schon um 4 bytes verschoben?
40745          bne a40719              ; nein: ->
40747 a40747   lda #0                  ; alle mantissen-stellen sind null,
40749 a40749   sta *a97                ; also exp := 0 (als null-flag)
40751 a40751   sta *a102               ; vorzeichenbyte := 0
40753          rts
40754
40754 a40754   adc *a86                ; mantissen-addition
40756          sta *a112
40758          lda *a101
40760          adc *a109
40762          sta *a101
40764          lda *a100
40766          adc *a108
40768          sta *a100
40770          lda *a99
40772          adc *a107
40774          sta *a99
40776          lda *a98
40778          adc *a106
40780          sta *a98
40782          jmp a40810              ; ---> überlauf bearbeiten wenn nötig
40785
40785 a40785   adc #1                  ; fac so oft um 1 bit nach links
40787          asl *a112               ; schieben, bis das höchste bit der
40789          rol *a101               ; höchsten stelle gesetzt ist
40791          rol *a100
40793          rol *a99
40795          rol *a98
40797 a40797   bpl a40785
40799          sec                     ; zahl der verschiebungen
40800          sbc *a97                ; - exponent
40802          bcs a40747              ; positiv: fac = 0 ->
40804          eor #255                ; exponent - zahl der verschiebungen
40806          adc #1
40808          sta *a97                ; = neuer exponent
40810 a40810   bcc a40826              ; kein überlauf: ->
```

```
40812 a40812  inc *a97              ; exponent inkrementieren
40814         beq a40882            ; = 0: overflow ->
40816         ror *a98              ; mantisse um 1 bit nach rechts
40818         ror *a99              ; schieben (carry = 1!)
40820         ror *a100
40822         ror *a101
40824         ror *a112
40826 a40826  rts
40827
40827 a40827  lda *a102              ; mantisse von fac invertieren
40829         eor #255
40831         sta *a102
40833 a40833  lda *a98
40835         eor #255
40837         sta *a98
40839         lda *a99
40841         eor #255
40843         sta *a99
40845         lda *a100
40847         eor #255
40849         sta *a100
40851         lda *a101
40853         eor #255
40855         sta *a101
40857         lda *a112
40859         eor #255
40861         sta *a112
40863         inc *a112             ; und inkrementieren
40865         bne a40881
40867 a40867  inc *a101
40869         bne a40881
40871         inc *a100
40873         bne a40881
40875         inc *a99
40877         bne a40881
40879         inc *a98              ; bewirkt fac := - fac
40881 a40881  rts
40882
40882 a40882  ldx #15               ; 'overflow'
40884         jmp a34435            ; ---> fehlerausgang
40887
40887 ;-----------------  R E G I S T E R  R E C H T S V E R S C H I E B E N
40887
40887 a40887  ldx #a37              ; offset-zeiger f funktionsregister
40889 a40889  ldy *a0+4,x           ; rechtsverschieben um 1 byte
40891         sty *a112
40893         ldy *a0+3,x
40895         sty *a0+4,x
40897         ldy *a0+2,x
40899         sty *a0+3,x
40901         ldy *a0+1,x
40903         sty *a0+2,x
40905         ldy *a104
40907         sty *a0+1,x
40909 a40909  adc #8                ; bit-zähler um 8 erhöhen
40911         bmi a40889            ; noch negativ: ->
40913         beq a40889            ; oder null: ->
40915         sbc #8                ; bit-zähler um 8 zurücksetzen
40917         tay
40918         lda *a112             ; rundungsstelle
40920         bcs a40942            ; ergebnis = 0: fertig ->
```

```
40922 a40922  asl *a0+1,x             ; höchtes bit der mantisse
40924         bcc a40928              ; nicht gesetzt: ->
40926         inc *a0+1,x             ; höchste mantissenstelle inkrementieren
40928 a40928  ror *a0+1,x             ; mantisse um 1 bit nach rechts
40930         ror *a0+1,x             ; schieben
40932 a40932  ror *a0+2,x
40934         ror *a0+3,x
40936         ror *a0+4,x
40938         ror a
40939         iny                     ; bitzähler inkrementieren
40940         bne a40922              ; noch nicht 0: ->
40942 a40942  clc
40943         rts
40944
40944 ;------------------------------  L O G
40944
40944 a40944  .by 129 0 0 0 0         ;  1
40949 a40949  .by 3                   ;  polynomgrad
40950         .by 127 94 86 203 121   ;  0.434255942
40955         .by 128 19 155 11 100   ;  0.576584541
40960         .by 128 118 56 147 22   ;  0.961800759
40965         .by 130 56 170 59 32    ;  2.88539007
40970 a40970  .by 128 53 4 243 52     ;  0.707106781
40975 a40975  .by 129 53 4 243 52     ;  1.41421356
40980 a40980  .by 128 128 0 0 0       ; -0.5
40985 a40985  .by 128 49 114 23 248   ;  0.693147181 = ln 2
40990
40990 a40990  jsr a41648              ; ---> vorzeichen prüfen
40993         beq a40997              ; fac = 0: fehler ->
40995         bpl a41000              ; fac > 0: ok ->
40997 a40997  jmp a39196              ; ---> 'illegal quantity'
41000
41000 a41000  lda *a97                ; exponent-byte von fac
41002         sbc #127                ; - 128 (carry!)
41004         pha                     ; = exponent 'pur' auf stack legen
41005         lda #128                ; damit wird fac in den bereich
41007         sta *a97                ; 0.5 <= x < 1 transformiert
41009         lda #l,a40970
41011         ldy #h,a40970
41013         jsr a41062              ; ---> fac := fac + konst
41016         lda #l,a40975
41018         ldy #h,a40975
41020         jsr a41074              ; ---> fac := konst / fac
41023         lda #l,a40944
41025         ldy #h,a40944
41027         jsr a41068              ; ---> fac := konst - fac
41030         lda #l,a40949
41032         ldy #h,a40949
41034         jsr a42675              ; ---> polynom auswerten
41037         lda #l,a40980
41039         ldy #h,a40980
41041         jsr a41062              ; ---> fac := fac + konst
41044         pla                     ; exponent
41045         jsr a41994              ; ---> kennzahl berechnen u addieren
41048         lda #l,a40985
41050         ldy #h,a40985
41052 a41052  jsr a41180              ; ---> konstante aus rom nach arg
41055         jmp a41083              ; ---> fac := fac * arg
```

```
41058 ;------------------------------  F A C  :=  F A C  +  0 . 5
41058
41058 a41058  lda #l,a42403           ; adresse der konstanten 0.5
41060         ldy #h,a42403
41062 a41062  jsr a41180              ; ---> konstante aus rom nach arg
41065         jmp a40606              ; ---> fac := arg + fac
41068
41068 ;------------------------------  F A C  :=  K O N S T  -  F A C
41068
41068 a41068  jsr a41180              ; ---> konstante aus rom nach arg
41071         jmp a40583              ; ---> fac := arg - fac
41074
41074 ;------------------------------  F A C  :=  K O N S T  /  F A C
41074
41074 a41074  jsr a41180              ; ---> konstante aus rom nach arg
41077         jmp a41367              ; ---> fac := arg / fac
41080
41080 ;------------------------------  F A C  :=  V A R I A B L E  *  F A C
41080
41080 a41080  jsr a41223              ; ---> konst von (a,y) in ram nach arg
41083
41083 ;------------------------------  F A C  :=  A R G  *  F A C
41083
41083 a41083  bne a41088              ; fac nicht null: ->
41085         jmp a41179              ; ---> rts
41088
41088 a41088  jsr a41271              ; ---> exponent berechnen
41091         lda #0                  ; funktionsregister löschen
41093         sta *a38
41095         sta *a39
41097         sta *a40
41099         sta *a41
41101         lda *a112
41103         jsr a41129              ; ---> multiplizieren
41106         lda *a101
41108         jsr a41129              ; ---> multiplizieren
41111         lda *a100
41113         jsr a41129              ; ---> multiplizieren
41116         lda *a99
41118         jsr a41129              ; ---> multiplizieren
41121         lda *a98
41123         jsr a41129              ; ---> multiplizieren
41126         jmp a41484              ; ---> erg nach fac, normalisieren
41129
41129 a41129  bne a41134
41131         jmp a40887              ; ---> funktionsreg nach rechts schieben
41134
41134 ;------------------------------  M U L T I P L I K A T I O N
41134
41134 a41134  lsr a                   ; für jedes in a gesetzte bit wird
41135         ora #128                ; arg zum funktionsregister addiert,
41137 a41137  tay                     ; dann wird dieses um 1 bit nach
41138         bcc a41165              ; rechts verschoben; ist ein bit
41140         clc                     ; nicht gesetzt, so wird nur
41141         lda *a41                ; verschoben
41143         adc *a109
41145         sta *a41
41147         lda *a40
41149         adc *a108
41151         sta *a40
```

```
41153          lda *a39
41155          adc *a107
41157          sta *a39
41159          lda *a38
41161          adc *a106
41163          sta *a38
41165 a41165   ror *a38
41167          ror *a39
41169          ror *a40
41171          ror *a41
41173          ror *a112
41175          tya
41176          lsr a
41177          bne a41137
41179 a41179   rts
41180
41180 ;-------------------------   ( A , Y )   A U S   R O M   N A C H   A R G
41180
41180 a41180   sta *a34                ; l,quelladresse
41182          sty *a35                ; h,...
41184          ldy #4
41186          lda (a34),y             ; konstante nach arg kopieren
41188          sta *a109
41190          dey
41191          lda (a34),y
41193          sta *a108
41195          dey
41196          lda (a34),y
41198          sta *a107
41200          dey
41201          lda (a34),y
41203          sta *a110
41205          eor *a102
41207          sta *a111               ; vorzeichen von fac * arg
41209          lda *a110
41211          ora #128
41213          sta *a106
41215          dey
41216          lda (a34),y
41218          sta *a105
41220          lda *a97                ; exponent von fac (als null-flag)
41222          rts
41223
41223 ;-------------------------   ( A , Y )   A U S   R A M   N A C H   A R G
41223
41223 a41223   sta *a34                ; l,quelladresse
41225          sty *a35                ; h,...
41227          ldy #4
41229          jsr a1200               ; ---> lda (a34),y
41232          sta *a109               ; variableninhalt nach arg kopieren
41234          dey
41235          jsr a1200               ; ---> lda (a34),y
41238          sta *a108
41240          dey
41241          jsr a1200               ; ---> lda (a34),y
41244          sta *a107
41246          dey
41247          jsr a1200               ; ---> lda (a34),y
41250          sta *a110
41252          eor *a102
41254          sta *a111               ; vorzeichen von fac * arg
```

```
41256          lda *a110
41258          ora #128
41260          sta *a106
41262          dey
41263          jsr a1200               ; ---> lda (a34),y
41266          sta *a105
41268          lda *a97                ; exponent von fac (als null-flag)
41270          rts
41271
41271 ;--------------------------------  E X P O N E N T E N   A D D I E R E N
41271
41271 a41271   lda *a105               ; exponent von arg
41273 a41273   beq a41306              ; = 0: ->
41275          clc
41276          adc *a97                ; + exponent von fac
41278          bcc a41284              ; kein überlauf: ->
41280          bmi a41311              ; überlauf und erg > 127: overflow ->
41282          clc
41283          .by 44
41284 a41284   bpl a41306              ; kein überlauf u erg < 128: fertig ->
41286          adc #128
41288          sta *a97                ; neuer exponent
41290          bne a41295              ; nicht null: ->
41292          jmp a40751              ; ---> fac := 0
41295
41295 a41295   lda *a111               ; vorzeichen von fac * arg
41297          sta *a102               ; = vorzeichen von fac
41299          rts
41300
41300 ;--------------------------------
41300
41300 a41300   lda *a102               ; vorzeichen von fac
41302          eor #255                ; invertieren
41304          bmi a41311              ; ergebnis negativ: ->
41306 a41306   pla
41307          pla
41308          jmp a40747              ; ---> fac := 0
41311
41311 a41311   jmp a40882              ; ---> 'overflow'
41314
41314 ;--------------------------------  F A C  :=  F A C  *  1 0
41314
41314 a41314   jsr a41617              ; ---> arg := fac
41317          tax                     ; exponent von fac
41318          beq a41336              ; = 0: fac = 0 ->
41320          clc
41321          adc #2                  ; exp + 2 entspricht fac * 4
41323          bcs a41311              ; überlauf: fehler ->
41325 a41325   ldx #0
41327          stx *a111
41329          jsr a40619              ; ---> fac := fac + arg
41332          inc *a97                ; exp + 1 entspricht fac * 2
41334          beq a41311              ; überlauf: fehler ->
41336 a41336   rts
41337
41337 ;--------------------------------
41337
41337 a41337   .by 132 32 0 0 0        ; gk-konstante 10
41342 ;
41342 a41342   ldx #20                 ; 'division by zero'
41344          jmp a34435              ; ---> fehlerausgang
```

```
41347 ;--------------------------------  F A C := F A C / 1 0
41347
41347 a41347  jsr a41617              ; ---> fac runden, arg := fac
41350         lda #l,a41337
41352         ldy #h,a41337
41354         ldx #0
41356 a41356  stx *a111               ; vorzeichen von fac
41358         jsr a41505              ; ---> fac := arg / fac
41361         jmp a41367              ; --->
41364
41364 ;--------------------------------  F A C := A R G / F A C
41364
41364 a41364  jsr a41223              ; ---> arg aus (a,y) im ram laden
41367 a41367  beq a41342              ; fac = 0: ->
41369         jsr a41632              ; ---> fac runden
41372         lda #0                  ; vorzeichen des exponenten von
41374         sec                     ; fac wechseln
41375         sbc *a97
41377         sta *a97
41379         jsr a41271              ; ---> exp und vorzeichen ermitteln
41382         inc *a97                ; exponent erhöhen
41384         beq a41311              ; überlauf: fehler ->
41386         ldx #252                ; offsetzeiger (in zero page zyklisch!)
41388         lda #1
41390 a41390  ldy *a106               ; arg mit fac vergleichen
41392         cpy *a98
41394         bne a41412
41396         ldy *a107
41398         cpy *a99
41400         bne a41412
41402         ldy *a108
41404         cpy *a100
41406         bne a41412
41408         ldy *a109
41410         cpy *a101
41412 a41412  php                     ; vergleichsergebnis auf stack legen
41413         rol a
41414         bcc a41425
41416         inx
41417         sta *a41,x              ; ergebnis im funktionsreg aufbauen
41419         beq a41471              ; offset x = 0: ->
41421         bpl a41475              ; offset x > 0: ->
41423         lda #1
41425 a41425  plp                     ; vergleichsergebnis
41426         bcs a41442              ; arg >= fac: ->
41428 a41428  asl *a109               ; arg um 1 bit nach links schieben
41430         rol *a108
41432         rol *a107
41434         rol *a106
41436         bcs a41412              ; überlauf: ->
41438         bmi a41390              ; höchstes bit gesetzt: ->
41440         bpl a41412              ; immer ->
41442
41442 a41442  tay                     ; mantisse von fac von der
41443         lda *a109               ; mantisse von arg subtrahieren
41445         sbc *a101
41447         sta *a109
41449         lda *a108
41451         sbc *a100
41453         sta *a108
```

```
41455          lda *a107
41457          sbc *a99
41459          sta *a107
41461          lda *a106
41463          sbc *a98
41465          sta *a106
41467          tya
41468          jmp a41428              ; ---> zurück zum verschieben
41471
41471 a41471   lda #64
41473          bne a41425
41475
41475 a41475   asl a
41476          asl a
41477          asl a
41478          asl a
41479          asl a
41480          asl a
41481          sta *a112               ; rundungsstelle von fac
41483          plp
41484 a41484   lda *a38                ; ergebnis nach fac kopieren
41486          sta *a98
41488          lda *a39
41490          sta *a99
41492          lda *a40
41494          sta *a100
41496          lda *a41
41498          sta *a101
41500          jmp a40715              ; ---> fac normalisieren
41503
41503 ;------------------------------   ( A , Y )  N A C H  F A C
41503
41503 a41503   clc                     ; flag f 'laden aus ram'
41504          .by 36
41505 a41505   sec                     ; flag f 'laden aus rom'
41506          sta *a34                ; l,quelladresse
41508          sty *a35                ; h,
41510          ldy #4
41512          jsr a41760              ; ---> lda (a34),y
41515          sta *a101
41517          dey
41518          jsr a41760              ; ---> lda (a34),y
41521          sta *a100
41523          dey
41524          jsr a41760              ; ---> lda (a34),y
41527          sta *a99
41529          dey
41530          jsr a41760              ; ---> lda (a34),y
41533          sta *a102               ; vorzeichen
41535          ora #128
41537          sta *a98
41539          dey
41540          jsr a41760              ; ---> lda (a34),y
41543          sta *a97                ; exponent
41545          sty *a112               ; rundungsstelle := 0
41547          rts
```

```
41548 ;------------------------------  F A C  N A C H  M E M O R Y
41548
41548 a41548  ldx #a92                ; zielbereich a92,...,a96
41550         .by 44
41551 a41551  ldx #a87                ; zielbereich a87,...,a91
41553         ldy #0
41555         beq a41561
41557
41557 a41557  ldx *a73                ; l,startadr d zielbereichs
41559         ldy *a74                ; h,...
41561 a41561  jsr a41632              ; ---> fac runden
41564         stx *a34                ; l,zielbereichszeiger
41566         sty *a35                ; h,...
41568         ldy #4
41570         lda *a101
41572         sta (a34),y
41574         dey
41575         lda *a100
41577         sta (a34),y
41579         dey
41580         lda *a99
41582         sta (a34),y
41584         dey
41585         lda *a102
41587         ora #127
41589         and *a98
41591         sta (a34),y
41593         dey
41594         lda *a97
41596         sta (a34),y
41598         sty *a112               ; rundungsstelle := 0
41600         rts
41601
41601 ;------------------------------  A R G  N A C H  F A C
41601
41601 a41601  lda *a110
41603 a41603  sta *a102
41605         ldx #5
41607 a41607  lda *a104,x
41609         sta *a96,x
41611         dex
41612         bne a41607
41614         stx *a112
41616         rts
41617
41617 ;------------------------------  F A C  N A C H  A R G
41617
41617 a41617  jsr a41632              ; ---> fac runden
41620 a41620  ldx #6
41622 a41622  lda *a96,x
41624         sta *a104,x
41626         dex
41627         bne a41622
41629         stx *a112
41631 a41631  rts
```

```
41632 ;------------------------------  F A C  R U N D E N
41632
41632 a41632  lda *a97                ; fac = 0?
41634         beq a41631              ; ja: fertig ->
41636         asl *a112               ; rundungsstelle
41638         bcc a41631              ; < 128: fertig ->
41640 a41640  jsr a40867              ; ---> mantisse um 1 erhöhen
41643         bne a41631              ; ergebnis nicht null: fertig ->
41645         jmp a40812              ; ---> fac normalisieren
41648
41648 ;------------------------------  V O R Z E I C H E N  E R M I T T E L N
41648
41648 a41648  lda *a97                ; fac = 0?
41650         beq a41661              ; ja: ->
41652 a41652  lda *a102               ; vorzeichen
41654 a41654  rol a                   ; negativ setzt carry
41655         lda #255
41657         bcs a41661              ; negativ: a = 255 ->
41659         lda #1                  ; positiv: a = 1
41661 a41661  rts
41662
41662 ;------------------------------  S G N
41662
41662 a41662  jsr a41648              ; ---> vorzeichen prüfen
41665 a41665  sta *a98                ; höchste stelle := 1 / 0 / 255
41667         lda #0
41669         sta *a99                ; 2. stelle := 0
41671         ldx #136                ; exponent f byte-wert
41673 a41673  lda *a98                ; höchste stelle
41675         eor #255                ; invertieren
41677         rol a                   ; 0 und positiv setzt carry
41678 a41678  lda #0
41680         sta *a101               ; 4. stelle := 0
41682         sta *a100               ; 3. stelle := 0
41684 a41684  stx *a97                ; exponent
41686         sta *a112               ; rundungsstelle
41688         sta *a102               ; vorzeichen
41690         jmp a40710              ; ---> negieren wenn cc, normalisieren
41693
41693 ;------------------------------  A B S
41693
41693 a41693  lsr *a102               ; vorzeichen: bit 7 löschen
41695         rts
41696
41696 ;------------------------------  Z A H L E N V E R G L E I C H
41696
41696 a41696  sta *a36                ; l,adresse der vergleichszahl
41698         sty *a37                ; h,...
41700         ldy #0
41702         lda (a36),y             ; exponent
41704         iny
41705         tax
41706         beq a41648              ; exp = 0, also zahl = 0: ->
41708         lda (a36),y             ; höchste stelle der vergleichszahl
41710         eor *a102               ; vorzeichen von fac
41712         bmi a41652              ; vorzeichen verschieden: ->
41714         cpx *a97                ; exponenten gleich?
41716         bne a41751              ; nein: ->
41718         lda (a36),y             ; höchste stelle der vergleichszahl
41720         ora #128                ; vorzeichenbit setzen
```

```
41722          cmp *a98                ; = höchste stelle von fac?
41724          bne a41751              ; nein: ->
41726          iny
41727          lda (a36),y             ; 2. stelle der vergleichszahl
41729          cmp *a99                ; = 2. stelle von fac?
41731          bne a41751              ; nein: ->
41733          iny
41734          lda (a36),y             ; 3. stelle der vergleichszahl
41736          cmp *a100               ; = 3. stelle von fac?
41738          bne a41751              ; nein: ->
41740          iny
41741          lda #127
41743          cmp *a112               ; rundungsstelle < 128 setzt carry
41745          lda (a36),y             ; 4. stelle von vergleichszahl
41747          sbc *a101               ; - 4. stelle von fac
41749          beq a41798              ; = 0: beide zahlen gleich ->
41751 a41751   lda *a102               ; vorzeichen von fac
41753          bcc a41757              ; vergleichszahl kleiner: ->
41755          eor #255
41757 a41757   jmp a41654              ; ---> ergebnis nach a
41760
41760 ;-------------------------------
41760
41760 a41760   lda (a34),y             ; aus rom laden
41762          bcs a41798              ; cs: fertig ->
41764          jmp a1200               ; ---> lda (a34),y aus ram
41767
41767 ;-------------------------------  G K - I N T E G E R - W A N D L U N G
41767
41767 a41767   lda *a97                ; fac = 0?
41769          beq a41845              ; ja: ->
41771          sec
41772          sbc #160                ; exponent transformieren
41774          bit *a102               ; fac negativ?
41776          bpl a41787              ; nein: ->
41778          tax                     ; exponent - 160
41779          lda #255
41781          sta *a104
41783          jsr a40833              ; ---> fac negieren
41786          txa
41787 a41787   ldx #a97                ; offset-zeiger für fac
41789          cmp #249                ; exponent - 160 > - 8?
41791          bpl a41799              ; ja: ->
41793          jsr a40909              ; ---> fac nach rechts schieben
41796          sty *a104               ; = 0
41798 a41798   rts
41799
41799 a41799   tay                     ; exponent - 160
41800          lda *a102               ; vorzeichen von fac
41802          and #128                ; bit 7 isolieren
41804          lsr *a98                ; 1. stelle 1 bit nach rechts
41806          ora *a98                ; bit 7 einodern
41808          sta *a98                ; ergebnis als 1. stelle abspeichern
41810          jsr a40932              ; ---> fac bitweise nach rechts schieben
41813          sty *a104               ; = 0
41815          rts
```

```
41816 ;-------------------------------  I N T
41816
41816 a41816  lda *a97                ; exponent
41818         cmp #160                ; < 160?
41820         bcs a41854              ; nein: fac ist ganzzahlig ->
41822         jsr a41767              ; ---> gk in integer wandeln
41825         sty *a112               ; rundungsstelle := 0
41827         lda *a102               ; vorzeichen
41829         sty *a102               ; := 0
41831         eor #128                ; nicht negativ
41833         rol a                   ; setzt carry
41834         lda #160                ; exponent für integer
41836         sta *a97
41838         lda *a101               ; letzte stelle
41840         sta *a7                 ; für potenzier-routine merken
41842         jmp a40710              ; ---> fac normalisieren
41845
41845 a41845  sta *a98                ; mantisse von fac mit nullen füllen
41847         sta *a99
41849         sta *a100
41851         sta *a101
41853         tay                     ; y := 0
41854 a41854  rts
41855
41855 ;-------------------------------  S T R I N G - G K - W A N D L U N G
41855
41855 a41855  ldy #0                  ; a93,...,a103 löschen
41857         ldx #10
41859 a41859  sty *a93,x
41861         dex
41862         bpl a41859
41864         bcc a41881              ; ziffer in a: ->
41866         cmp #'-                 ; minuszeichen?
41868         bne a41874              ; nein: ->
41870         stx *a103               ; = 255
41872         beq a41878              ; immer ->
41874
41874 a41874  cmp #'+
41876         bne a41883
41878 a41878  jsr a1139               ; ---> chrget
41881 a41881  bcc a41974              ; ziffer: ->
41883 a41883  cmp #'.
41885         beq a41933
41887         cmp #'e                 ; aus exponential-darstellung
41889         bne a41939
41891         jsr a1139               ; ---> chrget
41894         bcc a41919              ; ziffer: ->
41896         cmp #171                ; '-' token
41898         beq a41914
41900         cmp #'-
41902         beq a41914
41904         cmp #170                ; '+' token
41906         beq a41916
41908         cmp #'+
41910         beq a41916
41912         bne a41921
```

```
41914 a41914  ror *a96              ; setzt bit 7
41916 a41916  jsr a1139             ; ---> chrget
41919 a41919  bcc a42013            ; ziffer: ->
41921 a41921  bit *a96              ; bit 7 gesetzt?
41923         bpl a41939            ; nein: ->
41925         lda #0
41927         sec
41928         sbc *a94
41930         jmp a41941            ; --->
41933
41933 a41933  ror *a95              ; aufruf durch dezimalpunkt
41935         bit *a95
41937         bvc a41878
41939 a41939  lda *a94
41941 a41941  sec
41942         sbc *a93
41944         sta *a94
41946         beq a41966
41948         bpl a41959
41950 a41950  jsr a41347            ; ---> fac := fac / 10
41953         inc *a94
41955         bne a41950
41957         beq a41966
41959
41959 a41959  jsr a41314            ; ---> fac := fac * 10
41962         dec *a94
41964         bne a41959
41966 a41966  lda *a103
41968         bmi a41971
41970         rts
41971
41971 a41971  jmp a42535            ; ---> vorzeichen von fac wechseln
41974
41974 a41974  pha                   ; aufruf durch mantissen-ziffer
41975         bit *a95
41977         bpl a41981
41979         inc *a93
41981 a41981  jsr a41314            ; ---> fac := fac * 10
41984         pla
41985         sec
41986         sbc #'0
41988         jsr a41994            ; ---> stelle zu fac addieren
41991         jmp a41878            ; ---> nächstes zeichen
41994
41994 a41994  pha
41995         jsr a41617            ; ---> arg := fac
41998         pla
41999         jsr a41665            ; ---> höchste stelle von fac := a
42002         lda *a110             ; vorzeichen von arg
42004         eor *a102             ; vorzeichen von fac
42006         sta *a111             ; vorzeichen von arg * fac
42008         ldx *a97              ; exponent von fac
42010         jmp a40606            ; ---> fac := fac + arg
42013
42013 a42013  lda *a94              ; aufruf durch exponent-ziffer
42015         cmp #10
42017         bcc a42028
42019         lda #100
42021         bit *a96
42023         bmi a42047
42025         jmp a40882            ; ---> 'overflow'
```

```
42028 a42028  asl a
42029         asl a
42030         clc
42031         adc *a94
42033         asl a
42034         clc
42035         ldy #0
42037         sta *a120
42039         jsr a1189              ; ---> lda (a59),y
42042         adc *a120
42044         sec
42045         sbc #'0
42047 a42047  sta *a94
42049         jmp a41916             ; ---> nächstes zeichen
42052
42052 ;-------------------------------  K O N S T A N T E N
42052
42052 a42052  .by 155 62 188 31 253   ;    99999999.9
42057 a42057  .by 158 110 107 39 253  ;   999999999
42062 a42062  .by 158 110 107 40 0    ;  1000000000
42067 ;
42067 ;-------------------------------  Z E I L E N N R  A U S G E B E N
42067
42067 a42067  jsr a65359              ; ---> ' in ' ausgeben
42070         .by ' in ' 0
42075 a42075  lda *a58                ; h,zeilennummer
42077         ldx *a57                ; l,...
42079
42079 ;--------------------------  I N T E G E R  ( A , X )  A U S G E B E N
42079
42079 a42079  sta *a98                ; höchste mantissenstelle von fac
42081         stx *a99                ; 2. stelle
42083         ldx #144                ; exponent für integer
42085         sec                     ; verhindert negierung der mantisse
42086         jsr a41678              ; ---> fac mit nullen auffüllen
42089         jsr a42097              ; ---> gk in string wandeln
42092         jmp a37000              ; ---> string ausgeben
42095
42095 ;-------------------------------  G K - S T R I N G - W A N D L U N G
42095
42095 a42095  ldy #1                  ; offsetzeiger in stringpuffer
42097 a42097  lda #'
42099         bit *a102               ; vorzeichen von fac negativ?
42101         bpl a42105              ; nein: ->
42103         lda #'-
42105 a42105  sta a256-1,y            ; 1. stelle des stringpuffers
42108         sta *a102               ; vorzeichen von fac
42110         sty *a113
42112         iny
42113         lda #'0
42115         ldx *a97                ; exponent von fac
42117         bne a42122              ; nicht null: ->
42119         jmp a42390              ; ---> '0' in puffer, abschlu^
42122
42122 a42122  lda #0
42124         cpx #128                ; exponent von fac
42126         beq a42130              ; 0.5 <= fac < 1: ->
42128         bcs a42139              ; fac >= 1: ->
```

```
42130 a42130  lda #l,a42062
42132         ldy #h,a42062
42134         jsr a41052            ; ---> fac := fac * konst
42137         lda #247              ; = -9
42139 a42139  sta *a93              ; = -9 für fac < 1, sonst = 0
42141 a42141  lda #l,a42057
42143         ldy #h,a42057
42145         jsr a41696            ; ---> fac mit konst vergleichen
42148         beq a42180            ; gleich: ->
42150         bpl a42170            ; fac grö^er: ->
42152 a42152  lda #l,a42052
42154         ldy #h,a42052
42156         jsr a41696            ; ---> fac mit konst vergleichen
42159         beq a42163            ; gleich: ->
42161         bpl a42177            ; fac grö^er: ->
42163 a42163  jsr a41314            ; ---> fac := fac * 10
42166         dec *a93
42168         bne a42152            ; immer ->
42170
42170 a42170  jsr a41347            ; ---> fac := fac / 10
42173         inc *a93
42175         bne a42141            ; immer ->
42177
42177 a42177  jsr a41058            ; ---> fac := fac + 0.5
42180 a42180  jsr a41767            ; ---> gk-integer-wandlung
42183         ldx #1
42185         lda *a93              ; fac wurde in den bereich 1e+08,...,
42187         clc                   ; 1e+09 transformiert; (a93) enthält
42188         adc #10               ; den korrigierenden exponenten
42190         bmi a42201            ; betrag der zahl < 0.01: ->
42192         cmp #11
42194         bcs a42202            ; betrag der zahl > 1e+09: ->
42196         adc #255
42198         tax
42199         lda #2
42201 a42201  sec
42202 a42202  sbc #2
42204         sta *a94              ; exp, wenn 0.01 <= betr < 1e+09
42206         stx *a93
42208         txa
42209         beq a42213            ; 0.1 <= betrag < 1: ->
42211         bpl a42232            ; 0.01 <= betrag < 0.1? nein: ->
42213 a42213  ldy *a113
42215         lda #'.
42217         iny
42218         sta a256-1,y
42221         txa
42222         beq a42230            ; 0.1 <= betrag < 1: ->
42224         lda #'0
42226         iny
42227         sta a256-1,y
42230 a42230  sty *a113
42232 a42232  ldy #0                ; zeiger in konstantentabelle
42234 a42234  ldx #128
42236 a42236  lda *a101             ; durch abwechselnde subtraktion und
42238         clc                   ; addition der entsprechenden stellen-
42239         adc a42408+3,y        ; werte aus der tabelle werden die
42242         sta *a101             ; einzelnen ziffern des zahlenstrings
42244         lda *a100             ; nacheinander berechnet
42246         adc a42408+2,y
42249         sta *a100
```

```
42251          lda *a99
42253          adc a42408+1,y
42256          sta *a99
42258          lda *a98
42260          adc a42408,y
42263          sta *a98
42265          inx
42266          bcs a42272
42268          bpl a42236
42270          bmi a42274
42272
42272 a42272   bmi a42236
42274 a42274   txa
42275          bcc a42281              ; komplement addiert? nein: ->
42277          eor #255                ; ergebnis bzgl 10 komplementieren
42279          adc #10
42281 a42281   adc #47                 ; ergibt den ziffernkode
42283          iny                     ; tabellenzeiger auf nächsten eintrag
42284          iny
42285          iny
42286          iny
42287          sty *a71
42289          ldy *a113
42291          iny
42292          tax
42293          and #127                ; ziffernkode
42295          sta a256-1,y            ; in puffer bringen
42298          dec *a93
42300          bne a42308              ; einerstelle noch nicht erreicht: ->
42302          lda #'.
42304          iny
42305          sta a256-1,y
42308 a42308   sty *a113
42310          ldy *a71
42312          txa
42313          eor #255
42315          and #128
42317          tax
42318          cpy #36
42320          beq a42326              ; ende bei gk-string-wandlung: ->
42322          cpy #60                 ; ende bei ti$-erzeugung erreicht?
42324          bne a42236              ; nein: ->
42326 a42326   ldy *a113
42328 a42328   lda a256-1,y            ; letzte von 0 verschiedene stelle
42331          dey                     ; suchen
42332          cmp #'0
42334          beq a42328
42336          cmp #'.
42338          beq a42341
42340          iny
42341 a42341   lda #'+
42343          ldx *a94                ; 0.01 <= betrag < 1e+09?
42345          beq a42393              ; ja: ->
42347          bpl a42357              ; positiver zehnerexponent: ->
42349          lda #0                  ; exponenten generieren
42351          sec
42352          sbc *a94
42354          tax
42355          lda #'-
```

```
42357 a42357  sta a256+1,y
42360         lda #'e
42362         sta a256,y
42365         txa                     ; stellen des exponenten
42366         ldx #47                 ; generieren
42368         sec
42369 a42369  inx
42370         sbc #10
42372         bcs a42369
42374         adc #58                 ; einerstelle
42376         sta a256+3,y
42379         txa                     ; zehnerstelle
42380         sta a256+2,y
42383         lda #0                  ; endemarke
42385         sta a256+4,y
42388         beq a42398              ; immer ->
42390
42390 a42390  sta a256-1,y
42393 a42393  lda #0
42395         sta a256,y
42398 a42398  lda #l,a256             ; startadresse des stringpuffers
42400         ldy #h,a256
42402         rts
42403
42403 a42403  .by 128 0 0 0 0         ;  0.5
42408
42408 a42408  .by 250 10 31 0         ; -100 000 000     konstanten für
42412         .by 0 152 150 128       ;   10 000 000     gk-string-wandlung
42416         .by 255 240 189 192     ;   -1 000 000
42420         .by 0 1 134 160         ;      100 000
42424         .by 255 255 216 240     ;      -10 000
42428         .by 0 0 3 232           ;        1 000
42432         .by 255 255 255 156     ;         -100
42436         .by 0 0 0 10            ;           10
42440         .by 255 255 255 255     ;           -1
42444
42444         .by 255 223 10 128      ;   -2 160 000     konstanten für
42448         .by 0 3 75 192          ;      216 000     erzeugung von ti$
42452         .by 255 255 115 96      ;      -36 000
42456         .by 0 0 14 16           ;        3 600
42460         .by 255 255 253 168     ;         -600
42464         .by 0 0 0 60            ;           60
42468
42468 ;-------------------------------  S Q R
42468
42468 a42468  jsr a41617              ; ---> fac runden, arg := fac
42471         lda #l,a42403
42473         ldy #h,a42403
42475         jsr a41505              ; ---> konstante 0.5 nach fac
42478
42478 ;-------------------------------  F A C  :=  A R G  H O C H  F A C
42478
42478 a42478  beq a42592              ; fac = 0: ->
42480         lda *a105               ; arg = 0?
42482         bne a42487              ; nein: ->
42484         jmp a40749              ; ---> fac := 0
```

```
42487 a42487  ldx #l,a78
42489         ldy #h,a78
42491         jsr a41561              ; ---> fac nach (x,y) kopieren
42494         lda *a110               ; vorzeichen von arg
42496         bpl a42513              ; positiv: ->
42498         jsr a41816              ; ---> int schneidet nachkommastellen ab
42501         lda #l,a78
42503         ldy #h,a78
42505         jsr a41696              ; ---> vergleicht fac mit int(fac)
42508         bne a42513              ; nicht gleich: ->
42510         tya                     ; = 4
42511         ldy *a7                 ; letzte stelle des exponenten (int!)
42513 a42513  jsr a41603              ; ---> fac := betrag von arg
42516         tya
42517         pha                     ; letzte exponentenstelle
42518         jsr a40990              ; ---> fac := ln(fac)
42521         lda #l,a78
42523         ldy #h,a78
42525         jsr a41080              ; ---> fac := fac * konstante
42528         jsr a42592              ; ---> fac := exp(fac)
42531         pla                     ; letzte exponentenstelle
42532         lsr a                   ; exponent geradzahlig?
42533         bcc a42545              ; ja: ->
42535
42535 ;-------------------------------  V O R Z E I C H E N W E C H S E L
42535
42535 a42535  lda *a97                ; fac = 0?
42537         beq a42545              ; ja: ->
42539         lda *a102               ; vorzeichen
42541         eor #255                ; invertieren
42543         sta *a102
42545 a42545  rts
42546
42546 ;-------------------------------  E X P
42546
42546 a42546  .by 129 56 170 59 41    ;  1.44269504
42551 a42551  .by 7                   ;  polynomgrad
42552         .by 113 52 88 62 86     ;  2.14987637 e-05
42557         .by 116 22 126 179 27   ;  1.43523140 e-04
42562         .by 119 47 238 227 133  ;  1.34226348 e-03
42567         .by 122 29 132 28 42    ;  9.61401701 e-03
42572         .by 124 99 89 88 10     ;  0.0555051269
42577         .by 126 117 253 231 198         ;  0.240226385
42582         .by 128 49 114 24 16    ;  0.693147186 = ln 2
42587         .by 129 0 0 0 0         ;  1
42592
42592 a42592  lda #l,a42546
42594         ldy #h,a42546
42596         jsr a41052              ; ---> fac := fac * konst
42599         lda *a112               ; rundungsstelle
42601         adc #80                 ; + 80
42603         bcc a42608              ; < 256: ->
42605         jsr a41640              ; ---> fac um 1 erhöhen
42608 a42608  sta *a86                ; rundungsstelle + 80
42610         jsr a41620              ; ---> arg := fac
42613         lda *a97                ; exponent von fac
42615         cmp #136
42617         bcc a42622              ; < 136: ->
42619 a42619  jsr a41300              ; ---> '+': overflow; '-': 0
```

```
42622 a42622  jsr a41816              ; ---> int entfernt nachkommastellen
42625         lda *a7                 ; ganzzahliger anteil
42627         clc
42628         adc #129
42630         beq a42619              ; fac = 127: ->
42632         sec
42633         sbc #1
42635         pha
42636         ldx #5                  ; arg mit fac vertauschen
42638 a42638  lda *a105,x
42640         ldy *a97,x
42642         sta *a97,x
42644         sty *a105,x
42646         dex
42647         bpl a42638
42649         lda *a86
42651         sta *a112               ; rundungsstelle
42653         jsr a40583              ; ---> fac := arg - fac
42656         jsr a42535              ; ---> fac := - fac
42659         lda #l,a42551
42661         ldy #h,a42551
42663         jsr a42697              ; ---> polynom auswerten
42666         lda #0
42668         sta *a111               ; vorzeichen von fac * arg
42670         pla
42671         jsr a41273              ; ---> exponenten addieren
42674         rts
42675
42675 ;------------------------------  P O L Y N O M A U S W E R T U N G
42675
42675 ;  reg1 = a87,...,a91
42675
42675 a42675  sta *a113               ; adresse des polynomgrads
42677         sty *a114
42679         jsr a41551              ; ---> fac runden, nach reg1 kopieren
42682         lda #a87
42684         jsr a41080              ; ---> fac := fac * reg1
42687         jsr a42701              ; ---> polynom auswerten
42690         lda #l,a87
42692         ldy #h,a87
42694         jmp a41080              ; ---> fac := fac * reg1
42697
42697 ;------------------------------  H O R N E R - A L G O R I T H M U S
42697
42697 ;  reg2 = a92,...,a96
42697
42697 a42697  sta *a113               ; adresse des polynomgrads
42699         sty *a114
42701 a42701  jsr a41548              ; ---> fac nach reg2 kopieren
42704         lda (a113),y            ; polynomgrad
42706         sta *a103               ; als zähler einrichten
42708         ldy *a113               ; zeiger auf anfang der
42710         iny                     ; koeffiziententabelle setzen
42711         tya
42712         bne a42716
42714         inc *a114
42716 a42716  sta *a113
42718         ldy *a114
```

```
42720 a42720  jsr a41052            ; ---> fac := fac * koeffizient
42723         lda *a113             ; zeiger auf nächsten
42725         ldy *a114             ; koeffizienten erhöhen
42727         clc
42728         adc #5
42730         bcc a42733
42732         iny
42733 a42733  sta *a113
42735         sty *a114
42737         jsr a41062            ; ---> fac := fac + koeffizient
42740         lda #l,a92
42742         ldy #h,a92
42744         dec *a103             ; zähler dekrementieren
42746         bne a42720            ; noch nicht null: ->
42748         rts
42749
42749 ;------------------------------  R N D
42749
42749 a42749  .by 152 53 68 122 0   ;  zufallszahl 1
42754 a42754  .by 104 40 177 70 0   ;  zufallszahl 2
42759
42759 a42759  jsr a41648            ; ---> vorzeichen von fac
42762         bmi a42810            ; negativ: ->
42764         bne a42789            ; positiv: ->
42766         lda a65280            ; bytes aus timer lesen
42769         sta *a98              ; und vermischt nach fac bringen
42771         lda a65281
42774         sta *a100
42776         lda a65282
42779         sta *a99
42781         lda a65283
42784         sta *a101
42786         jmp a42826            ; --->
42789
42789 a42789  lda #l,a1283          ; adresse der letzten zufallszahl
42791         ldy #h,a1283
42793         jsr a41505            ; ---> zufallszahl nach fac
42796         lda #l,a42749
42798         ldy #h,a42749
42800         jsr a41052            ; ---> fac := fac * konst
42803         lda #l,a42754
42805         ldy #h,a42754
42807         jsr a41062            ; ---> fac := fac + konst
42810 a42810  ldx *a101             ; stellen von fac vertauschen
42812         lda *a98
42814         sta *a101
42816         stx *a98
42818         ldx *a99
42820         lda *a100
42822         sta *a99
42824         stx *a100
42826 a42826  lda #0
42828         sta *a102             ; vorzeichen von fac
42830         lda *a97              ; exponent von fac
42832         sta *a112             ; rundungsstelle
42834         lda #128
42836         sta *a97              ; exponent
42838         jsr a40715            ; ---> fac normalisieren
42841         ldx #l,a1283
42843         ldy #h,a1283
42845 a42845  jmp a41561            ; ---> fac runden und abspeichern
```

```
42848 ;-------------------  B A S I C - S T A C K  H I L F S R O U T I N E N
42848
42848 a42848  lda *a124             ; stack-zeiger umspeichern
42850         sta *a61
42852         lda *a125
42854         sta *a62
42856         rts
42857
42857 a42857  lda *a61
42859         sta *a124
42861         lda *a62
42863         sta *a125
42865         rts
42866
42866 a42866  tya                   ; stack-zeiger um y erhöhen
42867         clc
42868         adc *a124
42870         sta *a124
42872         bcc a42876
42874         inc *a125
42876 a42876  rts
42877
42877 ;-------,-----------------------  K E R N A L - A U F R U F E
42877
42877 a42877  tax                   ; fehlerkode
42878         bne a42882
42880         ldx #30               ; 'break'
42882 a42882  jmp a34435            ; ---> fehlerausgang
42885
42885 a42885  jsr a65472            ; ---> open
42888         bcs a42877            ; fehler: ->
42890         rts
42891
42891 a42891  jsr a65490            ; ---> bsout
42894         bcs a42877            ; fehler: ->
42896         rts
42897
42897 a42897  jsr a65487            ; ---> basin
42900         bcs a42877            ; fehler: ->
42902         rts
42903
42903 a42903  pha
42904         jsr a65481            ; ---> ckout
42907         jsr a43256            ; ---> falls bus, ds$ löschen
42910         tax
42911         pla
42912         bcc a42917            ; ok: ->
42914         txa
42915         bcs a42877            ; fehler: ->
42917
42917 a42917  rts
42918
42918 a42918  jsr a65478            ; ---> chkin
42921         jsr a43256            ; ---> falls bus, ds$ löschen
42924         bcs a42877            ; fehler: ->
42926         rts
42927
42927 a42927  jsr a65508            ; ---> getin
42930         bcs a42877            ; fehler: ->
42932         rts
```

```
42933 ;--------------------------------  S Y S
42933
42933 a42933  jsr a40417              ; ---> adresse holen
42936         lda #h,a42959-1         ; rücksprungadresse auf stack
42938         pha                     ; (ginge auch einfacher mit 'jsr'!)
42939         lda #l,a42959-1
42941         pha
42942         lda a2037               ; cpu-status und
42945         pha
42946         lda a2034               ; register a, x, y setzen
42949         ldx a2035
42952         ldy a2036
42955         plp
42956         jmp (a20)               ; ---> aufruf des maschinenprogramms
42959
42959 a42959  php                     ; cpu-status
42960         sta a2034               ; und register a, x, y
42963         stx a2035               ; abspeichern
42966         sty a2036
42969         pla
42970         sta a2037
42973         rts
42974
42974 ;--------------------------------  S A V E
42974
42974 a42974  jsr a43115              ; ---> parameter übernehmen
42977 a42977  ldx *a45                ; l,programmende
42979         ldy *a46                ; h,...
42981         lda #a43                ; zeiger auf programmanfangsadresse
42983         jsr a65496              ; ---> save
42986         jsr a43256              ; ---> falls bus, ds$ löschen
42989         bcs a42877              ; fehler: ->
42991         rts
42992
42992 ;--------------------------------  V E R I F Y
42992
42992 a42992  lda #1
42994         .by 44
42995
42995 ;--------------------------------  L O A D
42995
42995 a42995  lda #0
42997         sta *a10                ; verify-flag
42999         jsr a43115              ; ---> parameter übernehmen
43002 a43002  lda *a10                ; load/verify-flag
43004         ldx *a43                ; l,programmanfang
43006         ldy *a44                ; h,...
43008         jsr a65493              ; ---> load
43011         php
43012         jsr a43256              ; ---> falls bus, ds$ löschen
43015         plp
43016         bcs a43112              ; fehler: ->
43018         lda *a10                ; load?
43020         beq a43044              ; ja: ->
43022         ldx #28                 ; 'verify'
43024         jsr a65463              ; ---> readst
43027         and #16                 ; verify-error-bit
43029         bne a43053              ; gesetzt: ->
43031         bit *a129               ; direktmodus?
43033         bmi a43043              ; nein: ->
43035         jsr a65359              ; ---> 'ok' ausgeben
43038         .by 13 'ok' 13 0
43043 a43043  rts
```

```
43044 a43044  jsr a65463              ; ---> readst
43047         and #%10111111         ; eoi loeschen
43049         beq a43056             ; kein weiteres bit gesetzt: ->
43051         ldx #29                ; 'load'
43053 a43053  jmp a34435             ; ---> fehlerausgang
43056
43056 a43056  bit *a129              ; direktmodus?
43058         bmi a43076             ; nein: ->
43060         stx *a45               ; variablenanfang := ladeendadresse
43062         sty *a46
43064         jsr a34415             ; ---> 'ready.' ausgeben
43067         jsr a34840             ; ---> linkadressen berechnen
43070         jsr a35475             ; ---> progrzgr rücksetzen, clr
43073         jmp a34575             ; ---> ready-schleife
43076
43076 a43076  jsr a35569             ; ---> progzgr rücksetzen
43079         jsr a34840             ; ---> linkadressen berechnen
43082         jmp a35541             ; ---> restore, clr stack, progr starten
43085
43085 ;------------------------------  O P E N
43085
43085 a43085  jsr a43184             ; ---> parameter übernehmen
43088         clc
43089         jsr a42885             ; ---> open
43092         jsr a43256             ; ---> falls bus, ds$ löschen
43095         bcs a43112             ; fehler: ->
43097         rts
43098
43098 ;------------------------------  C L O S E
43098
43098 a43098  jsr a43184             ; ---> parameter übernehmen
43101         lda *a73
43103         clc
43104         jsr a65475             ; ---> close
43107         jsr a43256             ; ---> falls bus, ds$ löschen
43110         bcc a43043             ; ok: ->
43112 a43112  jmp a42877             ; ---> fehlerausgang
43115
43115 a43115  lda #0
43117         jsr a65469             ; ---> setnam: filenamen löschen
43120         ldx #1                 ; fa: vorbelegung kassette
43122         ldy #0                 ; sa: vorbelegung 'lesen'
43124         jsr a65466             ; ---> setlfs
43127         jsr a43165             ; ---> chrgot, abbruch bei trennzeichen
43130         jsr a43246             ; ---> filenamen übernehmen
43133         jsr a43165             ; ---> chrgot, abbruch bei trennzeichen
43136         jsr a43159             ; ---> chkcom, getbyt
43139         ldy #0                 ; sa: vorbelegung
43141         stx *a73               ; geräteadresse
43143         jsr a65466             ; ---> setlfs
43146         jsr a43165             ; ---> chrgot, abbruch bei trennzeichen
43149         jsr a43159             ; ---> chkcom, getbyt
43152         txa                    ; sekundäradresse
43153         tay
43154         ldx *a73               ; geräteadresse
43156         jmp a65466             ; ---> setlfs
43159
43159 a43159  jsr a43173             ; ---> chkcom, chrgot, fehler bei ende
43162         jmp a40324             ; ---> getbyt: byte übernehmen
```

```
43165 a43165  jsr a1145              ; ---> chrgot
43168         bne a43172             ; kein trennzeichen: ->
43170         pla
43171         pla
43172 a43172  rts
43173
43173 a43173  jsr a38033             ; ---> chkcom
43176 a43176  jsr a1145              ; ---> chrgot
43179         bne a43172             ; kein trennzeichen: ->
43181         jmp a38049             ; ---> 'syntax error'
43184
43184 a43184  lda #0
43186         jsr a65469             ; ---> setnam: filenamen löschen
43189         jsr a43176             ; ---> chrgot, fehler bei ende
43192         jsr a40324             ; ---> getbyt: byte übernehmen
43195         stx *a73
43197         txa                    ; log adresse
43198         ldx #1                 ; vorbelegung geräteadresse
43200         ldy #0                 ; vorbelegung sekundäradresse
43202         jsr a65466             ; ---> setlfs
43205         jsr a43165             ; ---> chrgot, abbruch bei trennzeichen
43208         jsr a43159             ; ---> chkcom, getbyt
43211         stx *a74               ; geräteadresse
43213         ldy #0                 ; vorbelegung sekundäradresse
43215         lda *a73               ; log adresse
43217         cpx #3                 ; geräteadresse < 3?
43219         bcc a43222             ; ja: ->
43221         dey                    ; sekundäradresse := 255
43222 a43222  jsr a65466             ; ---> setlfs
43225         jsr a43165             ; ---> chrgot, abbruch bei trennzeichen
43228         jsr a43159             ; ---> chkcom, getbyt
43231         txa
43232         tay
43233         ldx *a74
43235         lda *a73
43237         jsr a65466             ; ---> setlfs
43240         jsr a43165             ; ---> chrgot, abbruch bei trennzeichen
43243         jsr a43173             ; ---> chkcom, chrgot, fehler bei ende
43246 a43246  jsr a40008             ; ---> filenamen übernehmen
43249         ldx *a34               ; l,stringadresse
43251         ldy *a35               ; h,...
43253         jmp a65469             ; ---> setnam
43256
43256 ;------------------------------  D I S K  S T A T U S  L ö S C H E N
43256
43256 a43256  php
43257         pha
43258         lda *a174              ; fa: geräteadresse
43260         cmp #4                 ; bus?
43262         bcc a43267             ; nein: ->
43264         jsr a52567             ; ---> ds$ löschen
43267 a43267  pla
43268         plp
43269         rts
```

```
43270 ;---------------------  S T R I N G - P L A T Z R E S E R V I E R U N G
43270
43270 a43270  lsr *a15                ; flag f garbage collect löschen
43272 a43272  tax                     ; stringlänge
43273         beq a43331              ; = 0: ->
43275         pha                     ; auf stack legen
43276         lda *a51                ; stringbereich-anfang
43278         sec
43279         sbc #2                  ; - 2 (für r-zeiger)
43281         ldy *a52
43283         bcs a43286
43285         dey
43286 a43286  sta *a34                ; adresse des r-zeigers
43288         sty *a35
43290         txa                     ; stringlänge
43291         eor #255                ; negieren
43293         sec
43294         adc *a34                ; und addieren
43296         bcs a43299
43298         dey
43299 a43299  cpy *a50                ; ergebnis < variablenende?
43301         bcc a43332
43303         bne a43309
43305         cmp *a49
43307         bcc a43332              ; ja: ->
43309 a43309  sta *a53                ; string-transportzeiger
43311         sty *a54
43313         ldy #1                  ; r-zeiger für ungültigen string
43315         lda #255                ; einsetzen
43317         sta (a34),y
43319         dey
43320         pla                     ; stringlänge
43321         sta (a34),y
43323         ldx *a53
43325         ldy *a54
43327         stx *a51                ; neuer stringbereichs-anfang
43329         sty *a52
43331 a43331  rts
43332
43332 a43332  lda *a15                ; garbage collect schon durchgeführt?
43334         bmi a43345              ; ja: out of memory ->
43336         jsr a43348              ; ---> garbage collect
43339         sec                     ; garbage collect flag setzen
43340         ror *a15
43342         pla                     ; stringlänge
43343         bne a43272              ; immer ->
43345
43345 a43345  jmp a34433              ; ---> 'out of memory'
43348
43348 ;-------------------------------  G A R B A G E   C O L L E C T
43348
43348 a43348  ldx *a22                ; string stack zeiger
43350 a43350  cpx #a25                ; stack leer?
43352         beq a43370              ; ja: ->
43354         jsr a43607              ; ---> r-zeiger-adr nach a92,a93
43357         beq a43350              ; stringlänge = 0: ->
43359         txa                     ; strings, deren deskriptoren
43360         ldy #0                  ; im string stack stehen, mit
43362         sta (a92),y             ; r-zeigern für gültige strings
43364         tya                     ; versehen
43365         iny
43366         sta (a92),y
43368         bne a43350              ; immer ->
```

```
43370 a43370  ldy #0
43372         sty *a88              ; flag
43374         ldx *a55              ; string-obergrenze
43376         ldy *a56
43378         stx *a95              ; zielbereichszeiger
43380         stx *a78              ; quellbereichszeiger
43382         stx *a53              ; transportzeiger
43384         sty *a96
43386         sty *a79
43388         sty *a54
43390         txa
43391 a43391  jsr a43498            ; ---> zeiger nachstellen
43394         bne a43408
43396 a43396  dey
43397         jsr a33125            ; ---> lda (a78),y: stringlänge
43400         jsr a43577            ; ---> vom q-zeiger subtrahieren
43403         sec
43404         ror *a88              ; flag setzen
43406         bne a43391            ; immer ->
43408
43408 a43408  bit *a88              ; flag gesetzt?
43410         bpl a43478            ; nein: ->
43412         ldx #0
43414         stx *a88              ; flag löschen
43416         lda #2                ; ??? nicht sinnvoll!
43418 a43418  ldy #1                ; r-zeiger verschieben
43420         jsr a33125            ; ---> lda (a78),y
43423         sta (a95),y
43425         dey
43426         jsr a33125            ; ---> lda (a78),y
43429         sta (a95),y
43431         jsr a1200             ; ---> lda (a34),y
43434         tax                   ; stringlänge
43435         jsr a43592            ; ---> vom z-zeiger subtrahieren
43438         sta *a53              ; ergebnis in transportzeiger
43440         sty *a54
43442         txa                   ; stringlänge
43443         jsr a43577            ; ---> vom q-zeiger subtrahieren
43446         txa
43447         tay                   ; stringlänge ist offset
43448 a43448  dey                   ; string verschieben
43449         jsr a33125            ; ---> lda (a78),y
43452         sta (a95),y
43454         dex
43455         bne a43448
43457         ldy #2
43459 a43459  lda a95-1,y           ; z-zeiger
43462         sta (a34),y           ; in deskriptor kopieren
43464         dey
43465         bne a43459
43467         lda *a78              ; q-zeiger
43469         ldy *a79
43471         jsr a43498            ; ---> zeiger nachstellen
43474         beq a43396            ; ungültiger string: ->
43476         bne a43418            ; immer ->
```

```
43478 a43478  ldy #0
43480         jsr a1200             ; ---> lda (a34),y
43483         tax                   ; stringlänge
43484         jsr a43592            ; ---> vom z-zeiger subtrahieren
43487         sta *a53              ; ergebnis in transportzeiger
43489         sty *a54
43491         txa                   ; stringlänge
43492         jsr a43577            ; ---> vom q-zeiger subtrahieren
43495         jmp a43391            ; --->
43498
43498 a43498  cpy *a52              ; q-zeiger
43500         bcc a43544
43502         bne a43510
43504         cmp *a51
43506         beq a43544
43508         bcc a43544            ; kleiner als string-untergrenze: ->
43510 a43510  bit *a88              ; flag
43512         bmi a43519            ; gesetzt: ->
43514         lda #2
43516         jsr a43592            ; ---> vom z-zeiger subtrahieren
43519 a43519  lda #2
43521         jsr a43577            ; ---> vom q-zeiger subtrahieren
43524         ldy #1
43526         jsr a33125            ; ---> lda (a78),y: r-zeiger
43529         cmp #255              ; string ungültig?
43531         bne a43534            ; nein: ->
43533         rts
43534
43534 a43534  jsr a33125            ; ---> lda (a78),y: r-zeiger
43537         sta a34,y             ; = deskriptor-zeiger
43540         dey
43541         bpl a43534
43543         rts
43544
43544 a43544  ldx *a22              ; string-stack-zeiger
43546 a43546  cpx #a25              ; stack leer?
43548         beq a43566            ; ja: fertig ->
43550         jsr a43607            ; ---> r-zeiger-adr nach a92,a93
43553         beq a43546            ; stringlänge = 0: ->
43555         ldy #0                ; string ungültig machen:
43557         sta (a92),y           ; stringlänge in r-zeiger
43559         iny
43560         lda #255
43562         sta (a92),y           ; 255 in r-zeiger
43564         bne a43546            ; immer ->
43566
43566 a43566  pla                   ; rücksprungadresse vom stack entfernen
43567         pla
43568         lda *a53              ; transportzeiger
43570         ldy *a54
43572         sta *a51              ; ergibt neue string-untergrenze
43574         sty *a52
43576         rts
43577
43577 a43577  eor #255              ; a vom q-zeiger subtrahieren
43579         sec
43580         adc *a78
43582         ldy *a79
43584         bcs a43587
43586         dey
43587 a43587  sta *a78
43589         sty *a79
43591         rts
```

```
43592 a43592  eor #255              ; a vom z-zeiger subtrahieren
43594         sec
43595         adc *a95
43597         ldy *a96
43599         bcs a43602
43601         dey
43602 a43602  sta *a95
43604         sty *a96
43606         rts
43607
43607 a43607  dex
43608         lda *a0,x             ; h,stringadresse aus stack
43610         sta *a93
43612         dex
43613         lda *a0,x             ; l,...
43615         sta *a92
43617         dex
43618         lda *a0,x             ; stringlänge
43620         pha
43621         clc
43622         adc *a92              ; + stringadresse
43624         sta *a92              ; = r-zeiger-adresse
43626         bcc a43630
43628         inc *a93
43630 a43630  pla
43631         rts
43632
43632 ;------------------------------  C O S
43632
43632 a43632  lda #l,a43756
43634         ldy #h,a43756
43636         jsr a41062            ; ---> fac := fac + pi/2
43639
43639 ;------------------------------  S I N
43639
43639 a43639  jsr a41617            ; ---> arg := fac
43642         lda #l,a43761
43644         ldy #h,a43761
43646         ldx *a110             ; vorzeichen von arg
43648         jsr a41356            ; ---> fac := arg / konst
43651         jsr a41617            ; ---> arg := fac
43654         jsr a41816            ; ---> int
43657         lda #0
43659         sta *a111             ; vorzeichen von fac * arg
43661         jsr a40583            ; ---> fac := arg - fac
43664         lda #l,a43766
43666         ldy #h,a43766
43668         jsr a41068            ; ---> fac := konst - fac
43671         lda *a102             ; vorzeichen von fac
43673         pha
43674         bpl a43689            ; positiv: ->
43676         jsr a41058            ; ---> fac := fac + 0.5
43679         lda *a102             ; vorzeichen von fac
43681         bmi a43692            ; negativ: ->
43683         lda *a18              ; vorzeichen-flag
43685         eor #255              ; invertieren
43687         sta *a18
43689 a43689  jsr a42535            ; ---> fac := - fac
```

```
43692 a43692  lda #l,a43766
43694         ldy #h,a43766
43696         jsr a41062              ; ---> fac := fac + konst
43699         pla
43700         bpl a43705
43702         jsr a42535              ; ---> fac := - fac
43705 a43705  lda #l,a43771
43707         ldy #h,a43771
43709         jmp a42675              ; ---> polynom auswerten
43712
43712 ;------------------------------   T A N
43712
43712 a43712  jsr a41551              ; ---> fac nach reg2 kopieren
43715         lda #0
43717         sta *a18                ; vorzeichen-flag auf '+' setzen
43719         jsr a43639              ; ---> sin
43722         ldx #l,a78
43724         ldy #h,a78
43726         jsr a42845              ; ---> fac runden und nach reg1 kopieren
43729         lda #l,a87
43731         ldy #h,a87
43733         jsr a41505              ; ---> reg2 nach fac kopieren
43736         lda #0
43738         sta *a102               ; vorzeichen von fac
43740         lda *a18                ; vorzeichen-flag
43742         jsr a43752              ; ---> fac := cos(fac)
43745         lda #l,a78
43747         ldy #h,a78
43749         jmp a41364              ; ---> fac := reg1 / fac
43752
43752 a43752  pha
43753         jmp a43689              ; ---> cos
43756
43756 a43756  .by 129 73 15 218 162   ;  1.570079633 = pi/2
43761 a43761  .by 131 73 15 218 162   ;  6.28318531  = 2*pi
43766 a43766  .by 127 0 0 0 0         ;  0.25
43771 a43771  .by 5                   ;  polynomgrad
43772         .by 132 230 26 45 27    ;-14.3813907
43777         .by 134 40 7 251 248    ; 42.0077971
43782         .by 135 153 104 137 1   ;-76.7041703
43787         .by 135 35 53 223 225   ; 81.6052237
43792         .by 134 165 93 231 40   ;-41.3417021
43797         .by 131 73 15 218 162   ;  6.28318531 = 2*pi
43802
43802 ;------------------------------   A T N
43802
43802 a43802  lda *a102               ; vorzeichen von fac
43804         pha
43805         bpl a43810              ; positiv: ->
43807         jsr a42535              ; ---> fac := - fac
43810 a43810  lda *a97                ; exponent von fac
43812         pha
43813         cmp #129                ; fac < 1?
43815         bcc a43824              ; ja: ->
43817         lda #l,a40944
43819         ldy #h,a40944
43821         jsr a41074              ; ---> fac := 1 / fac
```

```
43824 a43824  lda #l,a43850
43826         ldy #h,a43850
43828         jsr a42675              ; ---> polynom auswerten
43831         pla
43832         cmp #129                ; war fac < 1?
43834         bcc a43843              ; ja: ->
43836         lda #l,a43756
43838         ldy #h,a43756
43840         jsr a41068              ; ---> fac := konst - fac
43843 a43843  pla                     ; vorzeichen des arguments
43844         bpl a43849              ; positiv: ->
43846         jmp a42535              ; ---> fac := - fac
43849
43849 a43849  rts
43850
43850 a43850  .by 11                  ;  polynomgrad
43851         .by 118 179 131 189 211         ; -6.84793912 e-04
43856         .by 121 30 244 166 245  ;  4.85094216 e-03
43861         .by 123 131 252 176 16  ; -0.0161117018
43866         .by 124 12 31 103 202   ;  0.034209638
43871         .by 124 222 83 203 193  ; -0.0542791328
43876         .by 125 20 100 112 76   ;  0.0724571965
43881         .by 125 183 234 81 122  ; -0.0898023954
43886         .by 125 99 48 136 126   ;  0.110932413
43891         .by 126 146 68 153 58   ; -0.142839808
43896         .by 126 76 204 145 199  ;  0.19999912
43901         .by 127 170 170 170 19  ; -0.333333316
43906         .by 129 0 0 0 0         ;  1
43911
43911 ;-------------------------------   R E N U M B E R
43911
43911 a43911  .by 137 138 141 167 140 214 215 213     ; renumber tokens
43919
43919 ; vorbesetzung: neunummer 10, schrittweite 10, altnummer 0
43919
43919 a43919  jsr a46814               ; ---> 'direct mode only'
43922         lda #0                  ; vorbesetzung anfangszeile
43924         ldx #10                 ; und schrittweite
43926         stx *a3                 ; neue anfangszeilennummer (neunummer)
43928         sta *a4
43930         stx *a5                 ; schrittweite
43932         sta *a6
43934         sta *a90                ; alte anfangszeilennummer (altnummer)
43936         sta *a91
43938         jsr a1145               ; ---> chrgot
43941         beq a44045              ; trennzeichen: ->
43943
43943 ; übernahme der parameter: neunummer, schrittweite, altnummer
43943
43943         jsr a36414              ; ---> zeilennummer holen
43946         lda *a8                 ; zeilennummer angegeben?
43948         beq a43958              ; nein: ->
43950         lda *a20                ; zeilennummer
43952         ldx *a21
43954         sta *a3                 ; = neunummer
43956         stx *a4
43958 a43958  jsr a1145               ; ---> chrgot
43961         beq a44045              ; trennzeichen: ->
43963         jsr a38033              ; ---> chkcom
```

```
43966          jsr a36414              ; ---> zeilennummer holen
43969          lda *a8                 ; zeilennummer angegeben?
43971          beq a43989              ; nein: ->
43973          lda *a20                ; zeilennummer
43975          ldx *a21
43977          sta *a5                 ; = schrittweite
43979          stx *a6
43981          bne a43989
43983          tax
43984          bne a43989              ; schrittweite nicht null: ->
43986 a43986   jmp a39196              ; ---> 'illegal quantity'
43989
43989 a43989   jsr a1145               ; ---> chrgot
43992          beq a44045              ; trennzeichen: ->
43994          jsr a38033              ; ---> chkcom
43997          jsr a36414              ; ---> zeilennummer holen
44000          lda *a20                ; zeilennummer
44002          ldx *a21
44004          sta *a90                ; = altnummer
44006          stx *a91
44008          jsr a35389              ; ---> zeilenadresse ermitteln
44011          lda *a95
44013          ldx *a96
44015          sta *a88                ; adresse der 1. renumber-zeile
44017          stx *a89
44019          lda *a3                 ; neunummer
44021          ldx *a4
44023          sta *a20
44025          stx *a21
44027          jsr a35389              ; ---> zeilenadresse ermitteln
44030          lda *a96                ; neuadresse
44032          sec
44033          sbc *a89                ; < altadresse?
44035          bcc a43986              ; d.h.: gibt es zeilen mit nummern
44037          bne a44045              ; in dem bereich, der durch
44039          lda *a95                ; renumber belegt wird?
44041          sbc *a88
44043          bcc a43986              ; ja: fehler ->
44045
44045 ; zielzeilennummern im programm (z.b. nach goto) korrigieren
44045
44045 a44045   jsr a35569              ; ---> programmzeiger rücksetzen
44048 a44048   jsr a44422              ; ---> l,linkadresse holen
44051          jsr a44422              ; ---> h,...
44054          bne a44117              ; programm nicht zu ende: ->
44056
44056 ; alte zeilennummern in neue umwandeln
44056
44056          jsr a44377              ; ---> neunummer nach fac, stxtpt
44059 a44059   jsr a44422              ; ---> l,linkadresse holen
44062          jsr a44422              ; ---> h,...
44065          bne a44070              ; programm nicht zu ende: ->
44067          jmp a44723              ; ---> abschlu, ready
44070
44070 a44070   jsr a44422              ; ---> l,zeilennummer
44073          sta *a20
44075          iny
44076          jsr a1189               ; ---> lda (a59),y: h,zeilennummer
```

```
44079          sec                    ; zeilennummer
44080          sbc *a91               ; < altnummer?
44082          bcc a44109
44084          bne a44092
44086          lda *a20
44088          sbc *a90
44090          bcc a44109             ; ja: keine änderung ->
44092 a44092   lda *a98               ; neunummer
44094          sta (a59),y            ; in zeilenkopf bringen
44096          dey
44097          lda *a99
44099          sta (a59),y
44101          jsr a44422             ; ---> nächstes zeichen
44104          jsr a44403             ; ---> nächste neunummer, nächste zeile
44107          beq a44059             ; immer ->
44109
44109 a44109   jsr a44422             ; ---> nächstes zeichen
44112          jsr a44416             ; ---> nächste zeile
44115          beq a44059             ; immer ->
44117
44117 a44117   jsr a44422             ; ---> nächstes zeichen
44120          jsr a44422             ; ---> nächstes zeichen
44123 a44123   jsr a44422             ; ---> nächstes zeichen
44126 a44126   cmp #'"
44128          bne a44141
44130 a44130   jsr a44422             ; ---> nächstes zeichen
44133          beq a44048             ; zeilenende: ->
44135          cmp #'"                ; text überlesen
44137          bne a44130
44139          beq a44123
44141
44141 ; zielzeilennummern im programm suchen und korrigieren
44141
44141 a44141   tax                    ; zeichen aus zeile
44142          beq a44048             ; zeilenende: ->
44144          bpl a44123             ; kein token: ->
44146          ldx #8
44148 a44148   cmp a43911-1,x         ; token in tabelle?
44151          beq a44169             ; ja: ->
44153          dex
44154          bne a44148
44156          cmp #203               ; 'go'-token?
44158          bne a44123             ; nein: ->
44160          jsr a1139              ; ---> chrget
44163          beq a44048             ; trennzeichen: ->
44165          cmp #164               ; 'to'-token?
44167          bne a44123             ; nein: ->
44169
44169 ; zielzeilennummern korrigieren
44169
44169 a44169   lda *a59               ; programmzeiger
44171          sta a601               ; merken
44174          lda *a60
44176          sta a602
44179          jsr a1139              ; ---> chrget
44182          bcs a44126             ; keine ziffer: weitersuchen ->
44184          jsr a36414             ; ---> zeilennummer holen
44187          jsr a44270             ; ---> neue zielzeilennummer erzeugen
```

```
44190          lda a601                ; programmzeiger
44193          sta *a59                ; wiederherstellen
44195          lda a602
44198          sta *a60
44200          ldy #0
44202          ldx #0
44204 a44204   lda a256+1,x            ; zahlenstring (ohne vorzeichen)
44207          beq a44237              ; ende erreicht: ->
44209          pha
44210          jsr a1139               ; ---> chrget
44213          bcc a44229              ; ziffer: keine verschiebung nötig ->
44215          jsr a44354              ; ---> verschiebung vorbereiten
44218          inc *a108               ; schreibzeiger erhöhen
44220          jsr a44475              ; ---> progr 1 byte nach hinten schieben
44223          inc *a45                ; variablenanfang nachführen
44225          bne a44229
44227          inc *a46
44229 a44229   pla                     ; ziffer aus neuer nummer
44230          ldy #0
44232          sta (a59),y             ; in programm schreiben
44234          inx                     ; zeiger in nummernstring erhöhen
44235          bne a44204              ; immer ->
44237
44237 a44237   jsr a1139               ; ---> chrget
44240          bcs a44263              ; keine ziffer: ->
44242 a44242   jsr a44354              ; ---> verschiebung vorbereiten
44245          dec *a108               ; schreibzeiger erniedrigen
44247          jsr a44450              ; ---> progr 1 byte vorziehen
44250          lda *a45                ; variablenanfang nachführen
44252          bne a44256
44254          dec *a46
44256 a44256   dec *a45
44258          jsr a1145               ; ---> chrgot
44261          bcc a44242              ; ziffer: nochmal vorziehen ->
44263 a44263   cmp #',
44265          beq a44169              ; weitere zeilennummer folgt: ->
44267          jmp a44126              ; ---> weiter im programmtext
44270
44270 ; neue zielzeilennummer erzeugen
44270
44270 a44270   jsr a44377              ; ---> neunummer nach fac, stxtpt
44273 a44273   jsr a44422              ; ---> l,linkadresse holen
44276          bne a44291              ; kein programmende: ->
44278          jsr a44422              ; ---> h,linkadresse holen
44281          bne a44294              ; kein programmende: ->
44283          lda #255                ; programmende: zeile nicht vorhanden
44285          sta *a98                ; zeilennummer 65535
44287          sta *a99
44289          bmi a44333              ; immer ->
44291
44291 a44291   jsr a44422              ; ---> h,linkadresse holen
44294 a44294   jsr a44422              ; ---> l,zeilennummer holen
44297          sta *a88
44299          cmp *a20                ; = l,alte zielzeilennummer?
44301          bne a44342              ; nein: nächste zeile ->
44303          jsr a44422              ; ---> h,zeilennummer
44306          sta *a89
44308          cmp *a21                ; = h,alte zielzeilennummer?
44310          bne a44347              ; nein: nächste zeile ->
```

```
44312          sec
44313          sbc *a91                ; zeile im renumber-bereich?
44315          bcc a44325
44317          bne a44333
44319          lda *a20
44321          sbc *a90
44323          bcs a44333              ; ja: neue nummer schon in fac ->
44325 a44325   lda *a20                ; alte zielzeilennummer
44327          sta *a99                ; nach fac
44329          lda *a21
44331          sta *a98
44333 a44333   ldx #144                ; exponent
44335          sec                     ; verhindert negierung
44336          jsr a41678              ; ---> fac aufbereiten
44339          jmp a42095              ; ---> gk-string-wandlung
44342
44342 a44342   jsr a44422              ; ---> h,zeilennummer
44345          sta *a89
44347 a44347   jsr a44388              ; ---> nächste neunummer, nächste zeile
44350          beq a44273              ; immer ->
44352 ;
44352          .by 217 234
44354 ;
44354 ; verschieben vorbereiten
44354
44354 a44354   lda *a59                ; programmzeiger
44356          sta *a34                ; = anfang des verschiebebereichs
44358          lda *a60
44360          sta *a35
44362          lda *a45                ; variablenanfang
44364          sta *a36                ; = ende des verschiebebereichs
44366          lda *a46
44368          sta *a37
44370          ldy #0
44372          sty *a11                ; lesezeiger
44374          sty *a108               ; schreibzeiger
44376          rts
44377
44377 ; neue anfangszeilennummer nach fac, programmzeiger rücksetzen
44377
44377 a44377   lda *a3                 ; neunummer
44379          sta *a99                ; nach fac
44381          lda *a4
44383          sta *a98
44385          jmp a35569              ; ---> programmzeiger rücksetzen
44388
44388 ; neue zeilennummer im renumber-bereich um schrittweite erhöhen
44388
44388 a44388   lda *a89                ; zeile im renumber-bereich?
44390          sec
44391          sbc *a91
44393          bcc a44416
44395          bne a44403
44397          lda *a88
44399          sbc *a90
44401          bcc a44416              ; nein: ->
```

```
44403 a44403  lda *a99                ; zeilennummer in fac
44405         clc
44406         adc *a5                 ; + schrittweite
44408         sta *a99                ; = nächste zeilennummer in fac
44410         lda *a98
44412         adc *a6
44414         sta *a98
44416 a44416  jsr a44422              ; ---> nächstes zeichen
44419         bne a44416              ; noch nicht zeilenende: ->
44421         rts
44422
44422 ; nächstes zeichen aus programm lesen
44422
44422 a44422  ldy #0
44424         inc *a59                ; programmzeiger erhöhen
44426         bne a44430
44428         inc *a60
44430 a44430  jmp a1189               ; ---> lda (a59),y
44433
44433 ; verschiebezeiger mit randzeiger vergleichen
44433
44433 a44433  lda *a34
44435         cmp *a36
44437         bne a44443
44439         lda *a35
44441         cmp *a37
44443 a44443  rts
44444
44444 ; programmblock nach vorn ziehen
44444
44444 a44444  inc *a34                ; anfangszeiger erhöhen
44446         bne a44450
44448         inc *a35
44450 a44450  ldy *a11                ; lesezeiger
44452         iny
44453         jsr a1200               ; ---> lda (a34),y
44456         ldy *a108               ; schreibzeiger
44458         iny
44459         sta (a34),y
44461         jsr a44433              ; ---> ende des blocks erreicht?
44464         bne a44444              ; nein: weiter verschieben ->
44466         rts
44467
44467 ; programmblock nach hinten schieben
44467
44467 a44467  lda *a36                ; endezeiger erniedrigen
44469         bne a44473
44471         dec *a37
44473 a44473  dec *a36
44475 a44475  ldy *a11                ; lesezeiger
44477         jsr a1211               ; ---> lda (a36),y
44480         ldy *a108               ; schreibzeiger
44482         sta (a36),y
44484         jsr a44433              ; ---> anfang des blocks erreicht?
44487         bne a44467              ; nein: weiter verschieben ->
44489         rts
```

```
44490 ;-------------------------------  F O R
44490
44490 a44490  lda #128                ; integervariable sperren
44492         sta *a16
44494         jsr a36476              ; ---> let: variable belegen
44497         lda #129                ; 'for'-token
44499         sta *a2                 ; für stack-suche speichern
44501         jsr a34929              ; ---> suche im basic-stack
44504         beq a44514
44506         ldy #18
44508         jsr a35077              ; ---> platz im basic-stack prüfen
44511         jsr a42848              ; ---> (a61,a62) := (a124,a125)
44514 a44514  jsr a42857              ; ---> (a124,a125) := (a61,a62)
44517         jsr a36286              ; ---> offset bis trennzeichen nach y
44520         tya                     ; offset
44521         ldy #17
44523         clc
44524         adc *a59                ; + programmzeiger
44526         sta (a124),y            ; = anfang der for-next-schleife
44528         lda *a60                ; auf basic-stack legen
44530         adc #0
44532         dey
44533         sta (a124),y
44535         lda *a58                ; zeilennummer
44537         dey
44538         sta (a124),y            ; auf basic-stack legen
44540         lda *a57
44542         dey
44543         sta (a124),y
44545         lda #164                ; 'to'-token
44547         jsr a38035              ; ---> syntax check
44550         jsr a37655              ; ---> prüfen ob variable numerisch
44553         jsr a37652              ; ---> to-ausdruck auswerten
44556         lda *a102               ; vorzeichen von fac
44558         ora #127                ; in hüchstes bit
44560         and *a98                ; der mantisse von fac
44562         sta *a98                ; übertragen
44564         ldx #4
44566         ldy #13
44568 a44568  lda *a97,x              ; to-wert auf basic-stack legen
44570         sta (a124),y
44572         dex
44573         dey
44574         bpl a44568
44576         lda #l,a40944           ; ersatzwert 1 für step
44578         ldy #h,a40944
44580         jsr a41505              ; ---> nach fac kopieren
44583         jsr a1145               ; ---> chrgot
44586         cmp #169                ; 'step'-token?
44588         bne a44596              ; nein: ->
44590         jsr a1139               ; ---> chrget
44593         jsr a37652              ; ---> step-ausdruck auswerten
44596 a44596  jsr a41648              ; ---> vorzeichen von fac
44599         pha
44600         jsr a41632              ; ---> fac runden
44603         pla
44604         ldy #8
44606         ldx #5
```

```
44608 a44608  sta (a124),y          ; vorzeichen und fac auf basic-stack
44610         lda *a96,x            ; legen
44612         dey
44613         dex
44614         bpl a44608
44616         lda *a74              ; for-next-variablenzeiger
44618         sta (a124),y          ; auf basic-stack legen
44620         lda *a73
44622         dey
44623         sta (a124),y
44625         lda #129              ; 'for'-token
44627         dey
44628         sta (a124),y          ; auf basic-stack legen
44630         rts
44631
44631 ;-----------------------------
44631
44631 a44631  jmp a38049            ; ---> 'syntax error'
44634
44634 ;-----------------------------  D E L E T E
44634
44634 a44634  jsr a46814            ; ---> 'direct mode only'
44637         jsr a1145             ; ---> chrgot
44640         beq a44631            ; kein zeilenbereich: fehler ->
44642         jsr a44746            ; ---> zeilenbereich übernehmen
44645         lda *a95              ; anfangsadresse des löschbereichs
44647         ldx *a96
44649         sta *a36              ; umspeichern
44651         stx *a37
44653         jsr a35389            ; ---> endzeilenadresse holen
44656         bcc a44679            ; zeile nicht vorhanden: ->
44658         ldy #1
44660         jsr a1233             ; ---> lda (a95),y: h,linkadresse
44663         dey
44664         tax
44665         bne a44672
44667         jsr a1233             ; ---> lda (a95),y: l,...
44670         beq a44679            ; programmende: ->
44672 a44672  jsr a1233             ; ---> lda (a95),y
44675         sta *a95              ; ende des löschbereichs
44677         stx *a96
44679 a44679  lda *a36              ; anfangsadresse
44681         sec
44682         sbc *a95              ; kleiner als endadresse?
44684         tax
44685         lda *a37
44687         sbc *a96
44689         tay
44690         bcs a44723            ; nein: fertig ->
44692         txa
44693         clc
44694         adc *a45              ; variablenanfangszeiger
44696         sta *a45              ; um länge des löschbereichs
44698         tya                   ; erniedrigen
44699         adc *a46
44701         sta *a46
44703         ldy #0
44705 a44705  jsr a1233             ; ---> lda (a95),y
44708         sta (a36),y           ; oberen programmabschnitt nach
44710         iny                   ; unten kopieren
44711         bne a44705
```

```
44713          inc *a96
44715          inc *a37
44717          lda *a46
44719          cmp *a37
44721          bcs a44705
44723 a44723   jsr a34840          ; ---> linkadressen berechnen
44726          lda *a34            ; programmende
44728          ldx *a35
44730          clc
44731          adc #2              ; + 2
44733          sta *a45
44735          bcc a44738
44737          inx
44738 a44738   stx *a46            ; = variablenanfang
44740          jsr a35475          ; ---> restore, clr
44743          jmp a34430          ; ---> ready
44746
44746 ;----------------------------  Z E I L E N B E R E I C H S A N A L Y S E
44746
44746 a44746   beq a44754          ; trennzeichen: ->
44748          bcc a44754          ; ziffer: ->
44750          cmp #171            ; '-' token?
44752          bne a44788          ; nein: fehler ->
44754 a44754   jsr a36414          ; ---> zeilennummer übernehmen
44757          jsr a35389          ; ---> zeilenadresse holen
44760          jsr a1145           ; ---> chrgot
44763          beq a44777          ; trennzeichen: ->
44765          cmp #171            ; '-' token?
44767          bne a44788          ; nein: fehler ->
44769          jsr a1139           ; ---> chrget
44772          jsr a36414          ; ---> zeilennummer übernehmen
44775          bne a44788          ; kein trennzeichen: fehler ->
44777 a44777   lda *a8             ; zeilennummer angegeben?
44779          bne a44787          ; ja: ->
44781          lda #255            ; sonst
44783          sta *a20            ; zeilennummer 65535 übergeben
44785          sta *a21
44787 a44787   rts
44788
44788 a44788   jmp a38049          ; ---> 'syntax error'
44791
44791 ;------------------------------  U S I N G
44791
44791 a44791   ldx #255            ; vorbelegung für
44793          stx a736            ; feldende
44796          jsr a1139           ; ---> chrget
44799          jsr a37676          ; ---> formatstring übernehmen
44802          jsr a37658          ; ---> prüfen ob string
44805          lda *a100           ; deskriptor-zeiger des formatstrings
44807          pha                 ; auf stack legen
44808          lda *a101
44810          pha
44811          ldy #2
44813 a44813   jsr a1244           ; ---> lda (a100),y: adr des formatstr
44816          dey
44817          sta a61,y           ; in formatstringzeiger
44820          bne a44813
44822          jsr a1244           ; ---> lda (a100),y
44825          sta a735            ; länge des formatstrings
44828          tay
44829          beq a44842          ; stringlänge = 0: fehler ->
```

```
44831 a44831  dey
44832         jsr a33137          ; ---> lda (a61),y: zchn aus formatstr
44835         cmp #'#
44837         beq a44845          ; '#' gefunden: ->
44839         tya
44840         bne a44831          ; sonst:
44842 a44842  jmp a38049          ; ---> 'syntax error'
44845
44845 a44845  lda #';             ; mu auf formatstring folgen
44847 a44847  jsr a38035          ; ---> syntax check
44850         sty *a118           ; := 0 (vom syntax check)
44852         sty a717            ; offset in zahlenstring
44855         jsr a37676          ; ---> ausgabeausdruck auswerten
44858         bit *a13            ; string?
44860         bpl a44919          ; nein: ->
44862         jsr a45424          ; ---> frestr, bereich initialisieren
44865         jsr a45751          ; ---> formatstring auswerten
44868         ldx a725
44871         beq a44894          ; weder '=' noch '' im formatstring: ->
44873         ldx #0
44875         sec
44876         lda a731            ; format-zähler
44879         sbc *a119           ; - länge des ausgabestrings
44881         bcc a44894          ; ausgabestring länger: ->
44883         ldx #'=
44885         cpx a725            ; zentrieren?
44888         bne a44893          ; nein: ->
44890         lsr a               ; (a) halbieren
44891         adc #0              ; und aufrunden
44893 a44893  tax                 ; zahl der vorlauf-blanks
44894 a44894  ldy #0              ; offset in ausgabestring
44896 a44896  txa                 ; noch blanks zu drucken?
44897         beq a44904          ; nein: ->
44899         dex                 ; blank-zähler erniedrigen
44900 a44900  lda #'              ; blank
44902         bne a44912          ; ausgeben ->
44904
44904 a44904  cpy *a119            ; ausgabestring zu ende?
44906         bcs a44900          ; ja: ->
44908         jsr a1200           ; ---> lda (a34),y: zchn aus ausgabestring
44911         iny                 ; offset in ausgabestring erhöhen
44912 a44912  jsr a45744          ; ---> ausgeben, formatzähler erniedrigen
44915         bne a44896          ; formatzähler noch nicht null: ->
44917         beq a44955          ; sonst: fertig ->
44919
44919 a44919  jsr a42095          ; ---> gk-string-wandlung
44922         ldy #255
44924 a44924  iny                 ; länge des zahlenstrings
44925         lda a256,y          ; feststellen
44928         bne a44924
44930         tya                 ; länge
44931         jsr a39772          ; ---> platz für string reservieren
44934         ldy #0
44936 a44936  lda a256,y          ; zahlenstring in stringbereich
44939         beq a44946
44941         sta (a98),y         ; kopieren
44943         iny
44944         bne a44936
```

```
44946 a44946  jsr a39856              ; ---> string-deskr in string-stack
44949         jsr a45424              ; ---> frestr, bereich initialisieren
44952         jsr a44987              ; ---> zahlenstring formatiert ausgeben
44955 a44955  jsr a1145               ; ---> chrgot
44958         cmp #',                 ; weitere ausgabe gewünscht?
44960         beq a44847              ; ja: ->
44962         sec
44963         ror *a118               ; flag für ausgabe-abbruch setzen
44965         jsr a45751              ; ---> restl zchn aus formatstr ausgeben
44968         pla                     ; h, zeiger auf deskr des formatstrings
44969         tay
44970         pla                     ; l,...
44971         jsr a40018              ; ---> frestr
44974         jsr a1145               ; ---> chrgot
44977         cmp #';
44979         beq a44984              ; wenn ';' folgt, kein zeilenvorschub ->
44981         jmp a36926              ; ---> cr ausgeben
44984
44984 a44984  jmp a1139               ; ---> chrget
44987
44987 ; z a h l e n s t r i n g   f o r m a t i e r e n
44987
44987 a44987  lda a1255               ; ' ' oder pudef-ersatz
44990         sta a733                ; als füllzeichen speichern
44993         lda #255                ; verhindert vorzeichen-ausgabe
44995 a44995  sta a732                ; vorzeichen-flag
44998         jmp a45003              ; --->
45001
45001 a45001  stx *a130               ; offset 1. nachkommastelle (vorb 255)
45003 a45003  cpy *a119               ; ende des zahlenstrings?
45005         beq a45058              ; ja: ->
45007         lda a256,y              ; nächstes zeichen aus zahlenstring
45010         iny                     ; offset erhöhen
45011         cmp #'                  ; blank?
45013         beq a45003              ; ja: überlesen ->
45015         cmp #'-                 ; minuszeichen?
45017         beq a44995              ; ja: in vorzeichenflag speichern ->
45019         cmp #'.                 ; dezimalpunkt?
45021         beq a45001              ; ja: offset 1. nachkommastelle merken ->
45023         cmp #'e                 ; exponent?
45025         beq a45044              ; ja: ->
45027         sta a256,x              ; ziffer wieder abspeichern
45030         stx a718                ; ende des zahlenstrings
45033         inx                     ; schreib-offset erhöhen
45034         bit *a130               ; schon dezimalpunkt gefunden?
45036         bpl a45003              ; ja: ->
45038         inc a724                ; vorkomma-stellenzähler erhöhen
45041         jmp a45003              ; ---> nächstes zeichen
45044
45044 a45044  lda a256,y              ; nächstes zeichen
45047         cmp #'-                 ; minus?
45049         bne a45054              ; nein: positiver exponent ->
45051         ror a722                ; exponent-negativ-flag
45054 a45054  iny                     ; offset in zahlenstring erhöhen
45055         sty a723                ; offset der 1. exponent-ziffer
45058 a45058  lda *a130               ; dezimalpunkt gefunden?
45060         bpl a45064              ; ja: ->
45062         stx *a130               ; 1. nachkommastelle hinter mantisse
```

```
45064 a45064  jsr a45751          ; ---> formatstring auswerten
45067         lda a726            ; vorkommastellen im formatstring
45070         cmp #255            ; nur '$' ohne vorkommastellen?
45072         beq a45115          ; ja: sternchen ausgeben ->
45074         lda a729            ; exponent im formatstring?
45077         beq a45142          ; nein: ->
45079         lda a723            ; exponent im zahlenstring?
45082         bne a45102          ; ja: ->
45084         ldx a718            ; ende des zahlenstrings
45087         jsr a45381          ; ---> '01' als exponenten speichern
45090         dec a256+2,x        ; macht '00' aus '01'
45093         inx                 ; offset in zahlenstring erhöhen
45094         stx a723            ; und als exponent-anfang merken
45097         jsr a45516          ; ---> exp-darstellung normalisieren
45100         beq a45139          ; zahl = 0: ->
45102 a45102  ldy a728            ; vorzeichen im formatstring?
45105         bne a45130          ; ja: ->
45107         ldy a732            ; minuszeichen vor dem zahlenstring?
45110         bmi a45130          ; nein: ->
45112         lda a726            ; vorkommastellen im formatstring?
45115 a45115  beq a45223          ; nein: sternchen ausgeben ->
45117         dec a726            ; vorkommastellen dekrementieren
45120         bne a45127          ; nicht null: ->
45122         lda a727            ; nachkommastellen im formatstring?
45125         beq a45223          ; nein: sternchen ausgeben ->
45127 a45127  inc a721
45130 a45130  jsr a45247          ; ---> zahl formatieren
45133         jsr a45450          ; ---> runden nach formatanweisung
45136         jsr a45247          ; ---> zahl formatieren
45139 a45139  jmp a45549          ; ---> zahl mit zusätzen ausgeben
45142
45142 a45142  ldy a723            ; exponent im zahlenstring?
45145         beq a45169          ; nein: ->
45147         sta *a119           ; := 0
45149         sec
45150         ror a730            ; flag 'exponent entfernen' setzen
45153         ldy *a130           ; offset der 1. nachkommastelle
45155         lda a722            ; vorzeichen des exponenten
45158         bpl a45166          ; '+': ->
45160         jsr a45304          ; ---> exp auf 0 erhöhen
45163         jmp a45178          ; --->
45166
45166 a45166  jsr a45273          ; ---> exp auf 0 erniedrigen
45169 a45169  ldy *a130           ; 1. nachkommastelle
45171         beq a45178          ; keine vorkommastellen: ->
45173         jsr a45520          ; ---> 1. vorkommastelle = 0?
45176         beq a45184          ; ja: ->
45178 a45178  jsr a45450          ; ---> zahl nach formatanw runden
45181         jmp a45187          ; --->
45184
45184 a45184  dec a724            ; zahl der vorkommastellen erniedrigen
45187 a45187  sec
45188         lda a726            ; vorkommastellen im formatstring
45191         sbc a724            ; - vorkommastellen in der zahl
45194         bcc a45223          ; ergebnis negativ: sternchen ausg ->
45196         sta a721            ; differenz merken
45199         ldy a728            ; vorzeichen im formatstring?
45202         bne a45231          ; ja: ->
45204         ldy a732            ; vorzeichen im zahlenstring?
45207         bmi a45231          ; nein: ->
```

```
45209          tay                    ; vorkommastellen-differenz
45210          beq a45223             ; = 0: sternchen ausgeben ->
45212          dey                    ; = 1?
45213          bne a45234             ; nein: ->
45215          lda a727               ; nachkommastellen im formatstring
45218          ora a724               ; oder vorkommastellen in der zahl
45221          bne a45139             ; vorhanden: zur ausgabe ->
45223 a45223   lda #'*
45225 a45225   jsr a45744             ; ---> ausgeben, formatzähler erniedrigen
45228          bne a45225             ; zähler noch nicht null: ->
45230          rts
45231
45231 a45231   tay                    ; vorkommastellen-differenz
45232          beq a45139             ; = 0: zur ausgabe ->
45234 a45234   lda a724               ; vorkommastellen in der zahl
45237          bne a45139             ; vorhanden: zur ausgabe ->
45239          dec a721               ; differenz dekrementieren
45242          inc *a118              ;
45244          jmp a45139             ; ---> zur ausgabe
45247
45247 ; e x p - d a r s t e l l u n g   e r z e u g e n
45247
45247 a45247   sec
45248          lda a726               ; vorkommastellen im formatstring
45251          sbc a724               ; - vorkommastellen im zahlenstring
45254          beq a45313             ; = 0: fertig ->
45256          ldy *a130              ; offset erste nachkommastelle
45258          bcc a45282             ; mehr vorkommastellen im zahlenstr: ->
45260          sta *a119              ; differenz in zähler
45262 a45262   cpy a718               ; 1. nachkommastelle
45265          beq a45269             ; = letzte stelle des zahlenstrings: ->
45267          bcs a45270             ; hinter der letzten stelle: ->
45269 a45269   iny
45270 a45270   inc a724               ; zahl der vorkommastellen erhöhen
45273 a45273   jsr a45326             ; ---> exponenten erniedrigen
45276          dec *a119              ; differenzzähler erniedrigen
45278          bne a45262             ; noch nicht null: weitermachen ->
45280          beq a45311             ; sonst: fertig ->
45282
45282 a45282   eor #255               ; differenz negieren
45284          adc #1
45286          sta *a119              ; und in zähler bringen
45288 a45288   cpy a717               ; 1. nachkommastelle
45291          beq a45300             ; = 1. stelle des zahlenstrings: ->
45293          dey
45294          dec a724               ; zahl der vorkommastellen erniedrigen
45297          jmp a45302             ; --->
45300
45300 a45300   inc *a118              ;
45302 a45302   lda #128               ; vorgabe exp-vorzeichen
45304 a45304   jsr a45328             ; ---> exponenten erhöhen
45307          dec *a119              ; differenzzähler erniedrigen
45309          bne a45288             ; noch nicht null: ->
45311 a45311   sty *a130              ; offset erste nachkommastelle
45313 a45313   rts
45314
45314 a45314   bne a45373             ; kein überlauf in exp-ziffer: ->
45316          eor #9                 ; '9' in '0', '0' in '9' umwandeln
45318          sta a256,x             ; ergebnis in zahlenstring
45321          dex                    ; offset in zahlenstring erniedrigen
45322          cpx a723               ; anfang des exponenten erreicht?
45325          rts
```

```
45326 a45326  lda #0                  ; vorgabe exp-vorzeichen
45328 a45328  ldx a723                ; offset exponent-zehnerstelle
45331         inx                     ; auf einerstelle erhöhen
45332         bit a730                ; soll exponent entfernt werden?
45335         bmi a45353              ; ja: ->
45337         eor a722                ; exp-vorzeichen in zahlenstring
45340         beq a45353              ; wie vorgabe: ->
45342 a45342  jsr a45395              ; ---> exponent-ziffer erhöhen
45345         jsr a45314              ; ---> übertrag wenn nötig
45348         bcs a45342              ; wenn dies die einerstelle war: ->
45350         jmp a40882              ; ---> 'overflow'
45353
45353 a45353  lda a256,x              ; exponent-ziffer
45356         dec a256,x              ; erniedrigen
45359         cmp #'0                 ; war sie 0?
45361         jsr a45314              ; ---> übertrag wenn nötig
45364         bcs a45353              ; wenn dies die einerstelle war: ->
45366         bit a730                ; soll exponent entfernt werden?
45369         bpl a45376              ; nein: ->
45371         sty *a130               ; offset erste nachkommastelle
45373 a45373  pla                     ; rücksprung auf nächsthöherer ebene
45374         pla
45375         rts
45376
45376 a45376  lda a722                ; vorzeichen des exponenten
45379         eor #128                ; wechseln
45381 a45381  sta a722                ; und wieder abspeichern
45384         lda #'0                 ; exponent '01' in zahlenstring
45386         sta a256+1,x
45389         lda #'1
45391         sta a256+2,x
45394         rts
45395
45395 a45395  lda a256,x              ; exponent-ziffer
45398         inc a256,x              ; erhöhen
45401         cmp #'9                 ; war sie = 9?
45403         rts
45404
45404 ; n ä c h s t e s   z e i c h e n   a u s   f o r m a t s t r i n g
45404
45404 a45404  clc
45405         iny                     ; zeiger in formatstring erhöhen
45406         beq a45413              ; überlauf: ->
45408         cpy a735                ; ende des formatstrings?
45411         bcc a45417              ; nein: ->
45413 a45413  ldy *a118               ; flag für ausgabe-abbruch
45415         bne a45373              ; gesetzt: abbruch ->
45417 a45417  jsr a33137              ; ---> lda (a61),y: zchn aus formatstr
45420         inc a731                ; format-zähler erhöhen
45423         rts
45424
45424 ; p a r a m e t e r   i n i t i a l i s i e r e n
45424
45424 a45424  jsr a40014              ; ---> frestr
45427         sta *a119               ; länge des ausgabestrings
45429         ldx #10
45431         lda #0
45433 a45433  sta a721,x              ; parameterbereich löschen
45436         dex
45437         bpl a45433
```

```
45439          stx a720                ; trennkomma-flag
45442          stx *a130               ; offset 1. nachkommastelle
45444          stx a719                ; dollar-flag
45447          tax                     ; = 0
45448          tay                     ; = 0
45449          rts
45450
45450 ; z a h l   n a c h   f o r m a t a n w e i s u n g   r u n d e n
45450
45450 a45450   clc
45451          lda *a130               ; offset der 1. nachkommastelle
45453          adc a727                ; + nachkommastellen im format
45456          bcs a45515              ; überlauf: fertig ->
45458          sec
45459          sbc *a118               ; - zeichen vor der zahl
45461          bcc a45515              ; ergebnis negativ: fertig ->
45463          cmp a718                ; mit offset zahlende vergleichen
45466          beq a45470              ; gleich: ->
45468          bcs a45515              ; grö^er: fertig ->
45470 a45470   cmp a717                ; mit offset zahlanfang vergleichen
45473          bcc a45515              ; kleiner: fertig ->
45475          tax                     ; offset der rundungsstelle
45476          lda a256,x              ; ziffer aus zahlenstring
45479          cmp #'5                 ; kleiner als 5?
45481          bcc a45515              ; ja: fertig ->
45483 a45483   cpx a717                ; erste stelle der zahl?
45486          beq a45498              ; ja: ->
45488          dex                     ; offset erniedrigen
45489          jsr a45395              ; ---> ziffer inkrementieren
45492          stx a718                ; offset =: ende der zahl
45495          beq a45483              ; übertrag entstanden: ->
45497          rts
45498
45498 a45498   lda #'1                 ; auf 1 aufrunden
45500          sta a256,x
45503          inx                     ; offset erhöhen
45504          stx *a130               ; als offset 1. nachkommastelle sp
45506          dec *a118               ;
45508          bpl a45515
45510          inc *a118               ;
45512          inc a724                ; vorkommastellen der zahl
45515 a45515   rts
45516
45516 ; e x p - d a r s t e l l u n g   n o r m a l i s i e r e n
45516
45516 a45516   ldy *a130               ; offset der 1. nachkommastelle
45518          beq a45543              ; keine vorkommastellen: ->
45520 a45520   ldy a717                ; offset des anfangs der zahl
45523 a45523   lda a256,y              ; ziffer aus zahl
45526          cmp #'0                 ; = '0'?
45528          rts
45529
45529 a45529   inc *a130               ; offset der 1. nachkommastelle erhöhen
45531          jsr a45326              ; ---> exponenten erniedrigen
45534          inc a717                ; offset anfang der zahl erhöhen
45537          cpy a718                ; ende der zahl erreicht?
45540          beq a45515              ; ja: fertig ->
45542          iny                     ; offset in zahl erhöhen
45543 a45543   jsr a45523              ; ---> ziffer aus zahl = 0?
45546          beq a45529              ; ja: ->
45548          rts
```

```
45549 ; f o r m a t i e r t e   z a h l   a u s g e b e n
45549
45549 a45549  lda a719              ; dollar-flag
45552         bmi a45556            ; schon '$' ausgegeben: ->
45554         inc *a118             ;
45556 a45556  ldx a717              ; offset anfang der zahl
45559         dex                   ; erniedrigen
45560         ldy a734              ; feldanfang
45563 a45563  jsr a33137            ; ---> lda (a61),y: zchn aus formatstr
45566         iny                   ; offset in formatstring erhöhen
45567         cmp #',               ; trennkomma?
45569         bne a45588            ; nein: ->
45571         bit a720              ; schon ziffer ausgegeben?
45574         bmi a45582            ; nein: füllzeichen ausgeben ->
45576         lda a1256             ; trennkomma oder pudef-ersatz
45579         jmp a45686            ; ---> ausgeben, formatzähler erniedrigen
45582
45582 a45582  lda a733              ; ' ' oder pudef-ersatz
45585         jmp a45686            ; ---> ausgeben, formatzähler erniedrigen
45588
45588 a45588  cmp #'.               ; dezimalpunkt im formatstring?
45590         bne a45598            ; nein: ->
45592         lda a1257             ; '.' oder pudef-ersatz
45595         jmp a45686            ; ---> ausgeben, formatzähler erniedrigen
45598
45598 a45598  cmp #'+               ; pluszeichen im formatstring?
45600         beq a45661            ; ja: ->
45602         cmp #'-               ; minuszeichen im formatstring?
45604         beq a45656            ; ja: ->
45606         cmp #'^               ; exponent-symbol im formatstring?
45608         bne a45709            ; nein: ->
45610         lda #'e
45612         jsr a45744            ; ---> ausgeben, formatzähler erniedrigen
45615         ldy a723              ; offset des exponenten in der zahl
45618         jsr a45523            ; ---> 1. exponent-ziffer = 0?
45621         bne a45629            ; nein: ->
45623         iny
45624         jsr a45523            ; ---> 2. exponent-ziffer = 0?
45627         beq a45636            ; ja: ->
45629 a45629  lda #'-
45631         bit a722              ; exponent-vorzeichen
45634         bmi a45638            ; '-': ->
45636 a45636  lda #'+
45638 a45638  jsr a45744            ; ---> ausgeben, formatzähler erniedrigen
45641         ldx a723              ; offset 1. exponent-ziffer
45644         lda a256,x            ; 1. exponent-ziffer
45647         jsr a45744            ; ---> ausgeben, formatzähler erniedrigen
45650         ldy a736              ; feldende
45653         jmp a45676            ; --->
45656
45656 a45656  lda a732              ; vorzeichen-ausgabe
45659         bmi a45582            ; gesperrt: füllzeichen ausgeben ->
45661 a45661  lda a732              ; vorzeichen
45664         jmp a45686            ; ---> ausgeben, formatzähler erniedrigen
45667
45667 a45667  lda *a118             ;
45669         bne a45692
45671         cpx a718              ; ende der zahl?
45674         beq a45681            ; ja: ersatzziffer 0 ausgeben ->
45676 a45676  inx                   ; offset erhöhen
45677         lda a256,x            ; nächste ziffer aus der zahl
45680         .by 44
```

```
45681 a45681  lda #'0
45683 a45683  lsr a720              ; trennkomma-flag setzen
45686 a45686  jsr a45744            ; ---> ausgeben, formatzähler erniedrigen
45689         bne a45563            ; kein endezeichen: ->
45691         rts
45692
45692 a45692  dec *a118             ;
45694         lda a719              ; dollar-flag
45697         bmi a45681            ; '$' schon ausgegeben: ->
45699         sec
45700         ror a719              ; dollar-flag setzen
45703         lda a1258             ; '$' oder pudef-ersatz
45706         jmp a45683            ; ---> ausgeben, formatzähler erniedrigen
45709
45709 a45709  lda a721              ; vorkommastellen-differenz
45712         beq a45667            ; = 0: ->
45714         dec a721              ; differenz erniedrigen
45717 a45717  beq a45722            ; jetzt = 0: ->
45719         jmp a45582            ; ---> füllzeichen ausgeben
45722
45722 a45722  lda a728              ; vorzeichen im formatstring?
45725         bmi a45717            ; ja: füllzeichen ausgeben ->
45727 a45727  jsr a33137            ; ---> lda (a61),y: zchn aus formatstr
45730         cmp #',               ; trennkomma?
45732         bne a45656            ; nein: ->
45734         lda a733              ; ' ' oder pudef-ersatz
45737         jsr a45744            ; ---> ausgeben, formatzähler erniedrigen
45740         iny                   ; offset in formatstring erhöhen
45741         jmp a45727            ; --->
45744
45744 a45744  jsr a37042            ; ---> zeichen ausgeben
45747         dec a731              ; format-zähler erniedrigen
45750         rts
45751
45751 ; f o r m a t s t r i n g   a u s w e r t e n
45751
45751 a45751  ldy a736              ; feldende
45754 a45754  jsr a45404            ; ---> nächstes zchn aus formatstr
45757         jsr a45932            ; ---> zeichen aus (+,-,.,=,,#)?
45760         bne a45782            ; nein: ->
45762         sty a734              ; feldanfang
45765         bcc a45793            ; '#': ->
45767         tax
45768 a45768  jsr a45404            ; ---> nächstes zchn aus formatstr
45771         bcs a45778            ; formatstring zu ende: ->
45773         jsr a45940            ; ---> zeichen aus (.,=,,#)?
45776         beq a45788            ; ja: ->
45778 a45778  ldy a734              ; feldanfang
45781         txa                   ; zeichen aus formatstring
45782 a45782  jsr a37042            ; ---> ausgeben
45785         jmp a45754            ; ---> zurück zum schleifenanfang
45788
45788 a45788  bcs a45768            ; nicht '#': ->
45790         ldy a734              ; feldanfang
45793 a45793  ldx *a118             ; abbruch-flag gesetzt?
45795         bne a45919            ; ja: fertig ->
45797         stx a731              ; format-zähler := 0
45800         dey
45801 a45801  dec a731              ; format-zähler erniedrigen
```

```
45804 a45804  jsr a45404          ; ---> nächstes zchn aus formatstr
45807         bcs a45925          ; formatstring zu ende: ->
45809         cmp #','            ; trennkomma?
45811         beq a45804          ; ja: ->
45813         jsr a45891          ; ---> '+','-'? wenn ja, übernehmen
45816         bcc a45801          ; ja: ->
45818         cmp #'.             ; dezimalpunkt?
45820         bne a45830          ; nein: ->
45822         inx
45823         cpx #2              ; zweiter dezimalpunkt?
45825         bcc a45804          ; nein: ->
45827 a45827  jmp a38049          ; ---> 'syntax error'
45830
45830 a45830  jsr a45944          ; ---> zeichen aus (=,,#)?
45833         bne a45846          ; nein: ->
45835         bcc a45840          ; '#': ->
45837         sta a725            ; zeichen merken
45840 a45840  inc a726,x          ; vor- bzw. nachkommastellen zählen
45843         jmp a45804          ; ---> nächstes zeichen
45846
45846 a45846  cmp #'$              ; dollar-zeichen?
45848         bne a45865          ; nein: ->
45850         bit a719            ; dollar-flag
45853         bpl a45840          ; schon ein '$' gefunden: ->
45855         clc
45856         ror a719            ; dollar-flag setzen
45859         dec a726            ; vorkommastellenzahl erniedrigen
45862         jmp a45840          ; --->
45865
45865 a45865  cmp #'^              ; exponent-symbol?
45867         bne a45891          ; nein: ->
45869         ldx #2
45871 a45871  jsr a45404          ; ---> nächstes zchn aus formatstr
45874         bcs a45827          ; formatstring zu ende: fehler ->
45876         cmp #'^             ; insgesamt 4 mal '^'?
45878         bne a45827          ; nein: fehler ->
45880         dex
45881         bpl a45871
45883         inc a729            ; exponent-flag setzen
45886         jsr a45404          ; ---> nächstes zchn aus formatstr
45889         bcs a45925          ; formatstring zu ende: ->
45891 a45891  cmp #'+              ; pluszeichen?
45893         bne a45920          ; nein: ->
45895         lda a732            ; vorzeichen-flag
45898         bpl a45905          ; vorzeichen im zahlenstring: ->
45900         lda #'+
45902         sta a732            ; vorzeichen-flag durch '+' ersetzen
45905 a45905  lda a728            ; schon vorzeichen im formatstring?
45908         bne a45827          ; ja: fehler ->
45910         ror a728            ; vorzeichen-im-format-flag setzen
45913         sty a736            ; feldende
45916         inc a731            ; format-zähler erhöhen
45919 a45919  rts
45920
45920 a45920  cmp #'-              ; minuszeichen?
45922         beq a45905          ; ja: ->
45924         sec
45925 a45925  sty a736            ; feldende
45928         dec a736            ; vermindern
45931         rts
```

```
45932 ; f o r m a t z e i c h e n   t e s t e n
45932
45932 a45932  cmp #'+
45934         beq a45957
45936         cmp #'-
45938         beq a45957
45940 a45940  cmp #'.                 ; dezimalpunkt
45942         beq a45957
45944 a45944  cmp #'=                 ; zentriersymbol
45946         beq a45957
45948         cmp #'^                 ; exponentsymbol
45950         beq a45957
45952         cmp #'#                 ; ziffernsymbol
45954         bne a45957
45956         clc
45957 a45957  rts
45958
45958 ;------------------------------   I N S T R
45958
45958 a45958  lda *a100               ; 1. deskriptor-zeiger umspeichern
45960         sta a1259
45963         lda *a101
45965         sta a1260
45968         jsr a37676              ; ---> ausdruck auswerten
45971         jsr a37658              ; ---> prüfen ob string
45974         lda *a100               ; 2. deskriptor-zeiger umspeichern
45976         sta a1261
45979         lda *a101
45981         sta a1262
45984         ldx #1                  ; vorbesetzung für suchanfang
45986         stx *a101
45988         jsr a1145               ; ---> chrgot
45991         cmp #')
45993         beq a45998              ; kein parameter: ->
45995         jsr a40408              ; ---> parameter übernehmen
45998 a45998  jsr a38027              ; ---> prüfen ob ')' folgt
46001         ldx *a101               ; parameterwert
46003         bne a46008              ; nicht 0: ->
46005         jmp a39196              ; ---> 'illegal quantity'
46008
46008 a46008  dex
46009         stx *a97                ; zeiger in suchstring
46011         ldx #3
46013 a46013  lda a1259,x             ; deskriptor-zeiger umspeichern
46016         sta *a87,x
46018         dex
46019         bpl a46013
46021         ldy #2
46023 a46023  jsr a33141              ; ---> lda (a87),y: 1. deskriptor
46026         sta a91,y
46029         jsr a33145              ; ---> lda (a89),y: 2. deskriptor
46032         sta a94,y
46035         dey
46036         bpl a46023
46038         lda *a94                ; länge des gesuchten strings
46040         beq a46097              ; = 0: fertig ->
```

```
46042 a46042  lda #0
46044         sta *a98              ; zeiger in gesuchten string
46046         clc
46047         lda *a94              ; länge des gesuchten strings
46049         adc *a97              ; + zeiger in suchstring
46051         bcs a46097            ; > 255: fertig ->
46053         cmp *a91              ; > länge des suchstrings?
46055         bcc a46059
46057         bne a46097            ; ja: fertig ->
46059 a46059  ldy *a98              ; zeiger in gesuchten string
46061         cpy *a94              ; = länge des gesuchten strings?
46063         beq a46092            ; ja: ->
46065         tya
46066         clc
46067         adc *a97              ; + zeiger in suchstring
46069         tay
46070         jsr a33129            ; ---> lda (a92),y: zchn aus suchstring
46073         sta *a120             ; merken
46075         ldy *a98              ; zeiger in gesuchten string
46077         jsr a33133            ; ---> lda (a95),y: zchn aus ges string
46080         cmp *a120             ; = zeichen aus suchstring?
46082         beq a46088            ; ja: ->
46084         inc *a97              ; zeiger in suchstring erhöhen
46086         bne a46042            ; immer ->
46088
46088 a46088  inc *a98              ; zeiger in gesuchten string erhöhen
46090         bne a46059            ; immer ->
46092
46092 a46092  inc *a97              ; zeiger in suchstring
46094         lda *a97
46096         .by 44
46097 a46097  lda #0
46099         pha
46100         lda a1261             ; 2. deskriptor-zeiger
46103         ldy a1262
46106         jsr a40018            ; ---> frestr
46109         lda a1259             ; 1. deskriptor-zeiger
46112         ldy a1260
46115         jsr a40018            ; ---> frestr
46118         pla
46119         tay                   ; zeiger in suchstring
46120         jmp a39553            ; ---> nach fac bringen
46123
46123 ;-------------------------------  T R A P
46123
46123 a46123  jsr a39558            ; ---> 'illegal direct'
46126         jsr a1145             ; ---> chrgot
46129         beq a46138            ; trennzeichen: abschalten ->
46131         jsr a40417            ; ---> getadr: adresse holen
46134         sty a1266             ; l,on-error-zeilennummer
46137         .by 44
46138 a46138  lda #255
46140         sta a1267             ; h,...
46143         rts
```

```
46144 ;------------------------------  R E S U M E
46144
46144 a46144  jsr a39558              ; ---> 'illegal direct'
46147         ldx a1265               ; on-error-flag
46150         inx
46151         beq a46265              ; nicht gesetzt: ->
46153         jsr a1145               ; ---> chrgot
46156         beq a46229              ; trennzeichen: ->
46158         bcc a46218              ; ziffer: ->
46160         cmp #130                ; 'next'-token?
46162         bne a46262              ; nein: ->
46164         jsr a46229              ; ---> fehleradresse in programmzeiger
46167         ldy #0
46169         jsr a1189               ; ---> lda (a59),y: zeichen aus zeile
46172         bne a46212              ; nicht zeilenende: ->
46174         iny
46175         jsr a1189               ; ---> lda (a59),y
46178         bne a46189
46180         iny
46181         jsr a1189               ; ---> lda (a59),y
46184         bne a46189              ; nicht programmende: ->
46186         jmp a34430              ; ---> ready
46189
46189 a46189  ldy #3
46191         jsr a1189               ; ---> lda (a59),y: l,zeilennummer
46194         sta *a57
46196         iny
46197         jsr a1189               ; ---> lda (a59),y: h,...
46200         sta *a58
46202         tya                     ; offset
46203         clc
46204         adc *a59                ; zum programmzeiger addieren
46206         sta *a59
46208         bcc a46212
46210         inc *a60
46212 a46212  jsr a1139               ; ---> chrget
46215         jmp a36272              ; ---> data: fortsetzung nach trennzchn
46218
46218 a46218  jsr a40417              ; ---> getadr: adresse holen
46221         sta *a21                ; ? überflüssig!
46223         jsr a46244              ; ---> fehlerstatus rücksetzen
46226         jmp a36201              ; ---> zeile suchen
46229
46229 a46229  ldx #1
46231 a46231  lda a1264,x             ; zeilennummer
46234         sta *a57,x              ; und
46236         lda a1269,x             ; programmadresse
46239         sta *a59,x              ; wiederherstellen
46241         dex
46242         bpl a46231
46244 a46244  ldx #255
46246         stx a1263               ; fehlerkode
46249         stx a1264               ; l,fehlerzeilennummer
46252         stx a1265               ; h,...
46255         ldx a1268               ; h,on-error-zeilennummer
46258         stx a1267               ; wiederherstellen
46261         rts
46262
46262 a46262  jmp a38049              ; ---> 'syntax error'
46265
46265 a46265  ldx #31                 ; 'can't resume'
46267         jmp a34435              ; ---> fehlerausgang
```

```
46270 ;------------------------------  E R R $
46270
46270 a46270  jsr a40327            ; ---> getbyt: fehlerkode übernehmen
46273         dex
46274         txa
46275         cmp #36               ; hoechster fehlerkode
46277         bcs a46331            ; überschritten: fehler ->
46279         jsr a34387            ; ---> zeiger auf meldung (x) setzen
46282         ldy #255
46284         ldx #0
46286 a46286  inx                   ; zähler für länge der meldung
46287 a46287  iny                   ; zeiger in meldung
46288         lda (a36),y
46290         bmi a46298
46292         cmp #'                ; steuerkode?
46294         bcc a46287            ; ja: nicht mitzählen ->
46296         bcs a46286
46298
46298 a46298  txa                   ; länge der meldung ohne steuerkodes
46299         jsr a39772            ; ---> platz für string reservieren
46302         ldx #0
46304         ldy #255
46306 a46306  iny
46307         lda (a36),y           ; zeichen aus meldung
46309         cmp #'
46311         bcc a46306            ; steuerkodes übergehen
46313         jsr a46334            ; ---> x und y vertauschen
46316         pha
46317         and #127
46319         sta (a98),y           ; zeichen in stringbereich kopieren
46321         jsr a46334            ; ---> x und y vertauschen
46324         inx
46325         pla
46326         bpl a46306            ; nicht letztes zeichen: ->
46328         jmp a40138            ; ---> string in stack
46331
46331 a46331  jmp a39196            ; ---> 'illegal quantity'
46334
46334 a46334  pha                   ; x und y vertauschen
46335         txa
46336         pha
46337         tya
46338         tax
46339         pla
46340         tay
46341         pla
46342         rts
46343
46343 ;------------------------------  H E X $
46343
46343 a46343  jsr a37655            ; ---> prüfen ob numerisch
46346         lda *a20              ; adresse retten
46348         pha
46349         lda *a21
46351         pha
46352         jsr a40420            ; ---> getadr: adresse holen
46355         lda #4                ; länge der hex-adresse
46357         jsr a39772            ; ---> platz für string reservieren
```

```
46360          ldy #0
46362          lda *a21                ; h-byte der adresse
46364          jsr a46381              ; ---> als 2 hex-ziffern in string
46367          lda *a20                ; l-byte der adresse
46369          jsr a46381              ; ---> als 2 hex-ziffern in string
46372          pla                     ; alte adresse wiederherstellen
46373          sta *a21
46375          pla
46376          sta *a20
46378          jmp a40138              ; ---> string in stack
46381
46381 a46381   pha                     ; byte merken
46382          lsr a                   ; oberes halbbyte nach unten schieben
46383          lsr a
46384          lsr a
46385          lsr a
46386          jsr a46390              ; ---> in ascii wandeln und in string
46389          pla                     ; byte erinnern
46390 a46390   and #15                 ; unteres halbbyte isolieren
46392          cmp #10                 ; kleiner als 10?
46394          bcc a46398              ; ja: ->
46396          adc #6                  ; in bereich a,...,f transformieren
46398 a46398   adc #48                 ; ascii-kode vervollständigen
46400          sta (a98),y             ; und in stringbereich speichern
46402          iny
46403          rts
46404
46404 ;-------------------------------   P U D E F
46404
46404 a46404   jsr a40008              ; ---> frmevl, chkstr, frestr
46407          tay                     ; länge
46408          dey
46409          cpy #4                  ; gröer als 4?
46411          bcs a46331              ; ja: fehler ->
46413 a46413   jsr a1200               ; ---> lda (a34),y: zeichen aus string
46416          sta a1255,y             ; in pudef-bereich kopieren
46419          dey
46420          bpl a46413
46422          rts
46423
46423 ;-------------------------------   D O
46423
46423 a46423   ldy #1
46425 a46425   lda a59,y               ; programmzeiger
46428          sta a1272,y             ; und
46431          lda a57,y               ; zeilennummer
46434          sta a1274,y             ; umspeichern
46437          dey
46438          bpl a46425
46440          jsr a1145               ; ---> chrgot
46443          beq a46473              ; trennzeichen: ->
46445          cmp #252                ; 'until'-token?
46447          beq a46466              ; ja: ->
46449          cmp #253                ; 'while'-token?
46451          bne a46516              ; nein: fehler ->
46453          jsr a46668              ; ---> chrget, frmevl
46456          lda *a97                ; bedingung erfüllt?
46458          bne a46473              ; ja: ->
46460 a46460   jsr a1145               ; ---> chrgot
46463          jmp a46522              ; ---> loop verlassen
```

```
46466 a46466  jsr a46668              ; ---> chrget, frmevl
46469         lda *a97                ; bedingung erfüllt?
46471         bne a46460              ; ja: ->
46473 a46473  ldy #5
46475         jsr a35077              ; ---> platz im basic-stack prüfen
46478         dey
46479         lda a1273               ; h,programmzeiger
46482         sta (a124),y
46484         dey
46485         lda a1272               ; l,...
46488         sta (a124),y
46490         dey
46491         lda a1275               ; h,zeilennummer
46494         sta (a124),y
46496         dey
46497         lda a1274               ; l,...
46500         sta (a124),y
46502         dey
46503         lda #235                ; 'do'-token
46505         sta (a124),y            ; auf basic-stack legen
46507         rts
46508
46508 ;-------------------------------  E X I T
46508
46508 a46508  jsr a46612              ; ---> 'do' im basic-stack suchen
46511         jsr a1145               ; ---> chrgot
46514         beq a46522              ; trennzeichen: ->
46516 a46516  jmp a38049              ; ---> 'syntax error'
46519
46519 a46519  jsr a1139               ; ---> chrget
46522 a46522  beq a46553              ; trennzeichen: ->
46524         cmp #236                ; 'loop'-token?
46526         beq a46592              ; ja: ->
46528         cmp #'"                 ; text?
46530         beq a46542              ; ja: ->
46532         cmp #235                ; 'do'-token?
46534         bne a46519              ; nein: ->
46536         jsr a46519              ; ---> rekursion
46539         jmp a46460              ; ---> chrgot, loop verlassen
46542
46542 a46542  jsr a1139               ; ---> chrget
46545         beq a46553              ; trennzeichen: ->
46547         cmp #'"                 ; text-ende?
46549         bne a46542              ; nein: ->
46551         beq a46519              ; ja: ->
46553
46553 a46553  cmp #':
46555         beq a46519              ; kein zeilenende: ->
46557         bit *a129               ; direktmodus?
46559         bpl a46629              ; ja: fehler ->
46561         ldy #2
46563         jsr a1189               ; ---> lda (a59),y: h,linkadresse
46566         beq a46629              ; programmende: fehler ->
46568         iny
46569         jsr a1189               ; ---> lda (a59),y: l,zeilennummer
46572         sta *a57
46574         iny
46575         jsr a1189               ; ---> lda (a59),y: h,...
46578         sta *a58
46580         tya
```

```
46581          clc
46582          adc *a59                ; programmzeiger auf das ende des
46584          sta *a59                ; zeilenkopfes setzen
46586          bcc a46519
46588          inc *a60
46590          bne a46519              ; immer ->
46592
46592 a46592   jmp a36272              ; ---> data: fortsetzung nach trennzeichen
46595
46595 ;--------------------------------  L O O P
46595
46595 a46595   beq a46642              ; trennzeichen folgt: ->
46597          cmp #253                ; 'while'-token?
46599          beq a46637              ; ja: ->
46601          cmp #252                ; 'until'-token?
46603          bne a46516              ; ja: ->
46605          jsr a46668              ; ---> chrget, frmevl
46608          lda *a97                ; bedingung erfüllt?
46610          beq a46642              ; nein: ->
46612 a46612   lda #235                ; 'do'-token
46614          sta *a2                 ; für stack-suche speichern
46616          jsr a34929              ; ---> suche im basic-stack
46619          bne a46632              ; nicht gefunden: fehler ->
46621          jsr a42857              ; ---> (a124,a125) := (a61,a62)
46624          ldy #5
46626          jmp a42866              ; ---> (a124,a125) um y erhöhen
46629
46629 a46629   ldx #32                 ; 'loop not found'
46631          .by 44
46632 a46632   ldx #33                 ; 'loop without do'
46634          jmp a34435              ; ---> fehlerausgang
46637
46637 a46637   jsr a46668              ; ---> chrget, frmevl
46640          beq a46612              ; bedingung nicht erfüllt: ->
46642 a46642   jsr a46612              ; ---> 'do' im basic-stack suchen
46645          dey
46646          lda (a61),y             ; aus dem basic stack übernehmen:
46648          sta *a60                ; h,programmzeiger
46650          dey
46651          lda (a61),y
46653          sta *a59                ; l,...
46655          dey
46656          lda (a61),y
46658          jsr a52607              ; ---> h,zeilennummer, ggf. direktmodus
46661          lda (a61),y
46663          sta *a57                ; l,...
46665          jmp a46423              ; ---> do
46668
46668 a46668   jsr a1139               ; ---> chrget
46671          jmp a37676              ; ---> ausdruck auswerten
46674
46674 ;--------------------------------  T R O N
46674
46674 a46674   lda #255
46676          .by 44
46677
46677 ;--------------------------------  T R O F F
46677
46677 a46677   lda #0
46679          sta a747                ; trace-flag
46682          rts
```

```
46683 ;------------------------------  M I D $ ( ... )  =  A $
46683
46683 a46683  jsr a38030              ; ---> prüfen ob '(' folgt
46686         jsr a38565              ; ---> variable suchen oder anlegen
46689         sta *a73                ; l,variablenzeiger
46691         sty *a74                ; h,...
46693         jsr a37658              ; ---> prüfen ob string
46696         jsr a40408              ; ---> chkcom, getbyt
46699         dex
46700         stx *a119               ; 1. parameter
46702         cmp #')
46704         beq a46710              ; kein 2. parameter: ->
46706         jsr a40408              ; ---> chkcom, getbyt
46709         .by 44
46710 a46710  ldx #255
46712         stx *a120               ; 2. parameter
46714         jsr a38027              ; ---> prüfen ob ')' folgt
46717         lda #178                ; '=' token
46719         jsr a38035              ; ---> syntax check
46722         jsr a37676              ; ---> ausdruck auswerten
46725         jsr a37658              ; ---> prüfen ob string
46728         ldy #2
46730 a46730  lda #a73
46732         jsr a1172               ; ---> lda (a),y: deskriptor des
46735         sta a91,y               ; zielstrings nach a91,a92,a93
46738         jsr a1244               ; ---> lda (a100),y: deskriptor des
46741         sta a94,y               ; einbaustrings nach a94,a95,a96
46744         dey
46745         bpl a46730
46747         sec
46748         lda *a95                ; l,adresse des einbaustrings
46750         sbc *a119               ; - 1. parameter
46752         sta *a95
46754         bcs a46758
46756         dec *a96
46758 a46758  lda *a120               ; 2. parameter
46760         cmp *a94                ; < länge des einbaustrings?
46762         bcc a46766              ; ja: ->
46764         lda *a94                ; sonst durch länge ersetzen
46766 a46766  tax
46767         beq a46791              ; ergebnis = 0: fertig ->
46769         clc
46770         adc *a119               ; + 1. parameter
46772         bcs a46794              ; > 255: fehler ->
46774         cmp *a91                ; > länge des zielstrings?
46776         bcc a46780
46778         bne a46794              ; ja: fehler ->
46780 a46780  ldy *a119               ; 1. parameter
46782 a46782  jsr a33133              ; ---> lda (a95),y: zchn aus einbaustring
46785         sta (a92),y             ; in zielstring kopieren
46787         iny
46788         dex
46789         bne a46782
46791 a46791  jmp a40014              ; ---> frestr
46794
46794 a46794  jmp a39196              ; ---> 'illegal quantity'
```

```
46797 ;------------------------------  A U T O
46797
46797 a46797  jsr a46814              ; ---> 'direct mode only'
46800         jsr a36414              ; ---> zeilennummer übernehmen
46803         lda *a20                ; und als
46805         sta *a115               ; auto-schrittweite speichern
46807         lda *a21
46809         sta *a116
46811         jmp a34430              ; ---> ready
46814
46814 a46814  bit *a129               ; direktmodus?
46816         bmi a46819              ; nein: fehler ->
46818         rts
46819
46819 a46819  ldx #34                 ; 'direct mode only'
46821         jmp a34435              ; ---> fehlerausgang
46824
46824 ;------------------------------  H E L P
46824
46824 a46824  ldx a1263               ; fehlerkode
46827         inx
46828         beq a46857              ; nicht definiert: ->
46830         lda a1264               ; l,fehlerzeilennummer
46833         ldy a1265               ; h,...
46836         sta *a20
46838         sty *a21
46840         jsr a35389              ; ---> zeile suchen, adresse holen
46843         bcc a46857              ; zeile nicht vorhanden: ->
46845         ror *a83                ; flag für fehlerlisting setzen
46847         jsr a36926              ; ---> cr ausgeben
46850         ldx *a20                ; l,fehlerzeilennummer
46852         lda *a21                ; h,...
46854         jsr a35648              ; ---> zeile listen
46857 a46857  jmp a36926              ; ---> cr ausgeben
46860
46860 ; die folgende subroutine wird bei gesetztem flag a83, bit 7,
46860 ; von der list-routine bei jedem ausgegebenen zeichen gerufen
46860
46860 a46860  ldx *a96                ; fehlerstelle erreicht?
46862         tya
46863         clc
46864         adc *a95
46866         bcc a46869
46868         inx
46869 a46869  cpx a1270
46872         bne a46888
46874         cmp a1269
46877         bcc a46888
46879         beq a46888              ; nein: ->
46881         lsr *a83                ; flag für fehlerlisting löschen
46883         lda #130                ; 'blinkflag setzen'
46885         jmp a37042              ; ---> ausgeben
46888
46888 a46888  rts
```

```
46889 ;------------------------------  K E Y   A N Z E I G E N
46889
46889 a46889  bne a47015              ; nummer folgt: ->
46891         ldx #0
46893         ldy #0
46895 a46895  inx
46896         lda a1375-1,x
46899         beq a46984
46901         sta *a119               ; key textlänge
46903         stx *a118               ; key nummer (1,...,8)
46905         ldx #5
46907 a46907  lda a52590,x            ; 'key 0,'
46910         dex
46911         bne a46915
46913         ora *a118               ; konkrete nummer einsetzen
46915 a46915  jsr a65490              ; ---> bsout
46918         txa
46919         bpl a46907
46921         ldx #7
46923 a46923  lda a1383,y             ; zeichen aus text
46926         iny
46927         pha
46928         stx *a128
46930         ldx #4
46932 a46932  cmp a47155+6,x          ; einer der kodes 13, 141, 34, 27?
46935         beq a46989              ; ja: ->
46937         dex
46938         bne a46932
46940         ldx *a128
46942         cpx #8
46944         bcc a46953
46946         bne a46958
46948         lda #'+
46950         jsr a65490              ; ---> bsout
46953 a46953  lda #'"
46955         jsr a65490              ; ---> bsout
46958 a46958  pla
46959         jsr a65490              ; ---> bsout
46962         ldx #9
46964 a46964  dec *a119               ; textlänge dekrementieren
46966         bne a46923
46968         cpx #9
46970         bcc a46977
46972         lda #'"
46974         jsr a65490              ; ---> bsout
46977 a46977  lda #141                ; 'shift return'
46979         jsr a65490              ; ---> bsout
46982         ldx *a118               ; key nummer
46984 a46984  cpx #8                  ; kleiner als 8?
46986         bne a46895              ; ja: ->
46988         rts
46989
46989 a46989  ldx *a128
46991 a46991  lda a47155-3,x          ; '"+chr$('
46994         jsr a65490              ; ---> bsout
46997         dex
46998         cpx #3
47000         bcs a46991
47002         pla
47003         jsr a52596              ; ---> kode als bytewert ausgeben
```

```
47006          lda #')
47008          jsr a65490              ; ---> bsout
47011          ldx #8
47013          bne a46964
47015
47015 ;--------------------------------  K E Y   D E F I N I E R E N
47015
47015 a47015   jsr a40324              ; ---> getbyt: nummer holen
47018          dex
47019          cpx #8
47021          bcc a47026
47023          jmp a39196              ; ---> 'illegal quantity'
47026
47026 a47026   stx *a118               ; key nummer - 1
47028          jsr a38033              ; ---> chkcom
47031          jsr a40008              ; ---> frmevl, chkstr, frestr
47034          jsr a47042              ; ---> text in textpuffer übertragen
47037          bcc a47153              ; ok: ->
47039          jmp a34433              ; ---> 'out of memory'
47042
47042 a47042   sta *a119               ; text länge
47044          ldx #8
47046          jsr a47166              ; ---> a := offset des textes (x)
47049          sta a717                ; ende des letzten textes + 1
47052          ldx *a118               ; nummer der funktionstaste
47054          inx
47055          jsr a47166              ; ---> a := offset des textes (x)
47058          sta a718                ; ende des aktuellen textes + 1
47061          ldx *a118               ; nummer der funktionstaste
47063          lda *a119               ; länge des neuen textes
47065          sec
47066          sbc a1375,x             ; - länge des alten textes
47069          beq a47124              ; gleiche länge: ->
47071          bcc a47102              ; neuer text kürzer: ->
47073          clc
47074          adc a717                ; differenz zum textende addieren
47077          bcs a47154              ; überlauf: fehler ->
47079          cmp #129                ; ergebnis > 128?
47081          bcs a47154              ; ja: fehler ->
47083          tax                     ; neues ende + 1
47084          ldy a717                ; altes ende + 1
47087 a47087   cpy a718                ; anfang des verschiebereichs erreicht?
47090          beq a47124              ; ja: ->
47092          dey                     ; quellzeiger erniedrigen
47093          dex                     ; zielzeiger erniedrigen
47094          lda a1383,y             ; zeichen aus quellbereich
47097          sta a1383,x             ; in zielbereich kopieren
47100          bcs a47087              ; immer ->
47102
47102 a47102   adc a718                ; negative differenz + altes ende
47105          tax                     ; = ende des neuen textes
47106          ldy a718                ; ende des alten textes
47109 a47109   cpy a717                ; altes ende des letzten textes erreicht?
47112          bcs a47124              ; ja: ->
47114          lda a1383,y             ; zeichen aus quellbereich
47117          sta a1383,x             ; in zielbereich kopieren
47120          iny                     ; quellzeiger erhöhen
47121          inx                     ; zielzeiger erhöhen
47122          bcc a47109              ; immer ->
```

```
47124 a47124  ldx *a118               ; nummer der funktionstaste
47126         jsr a47166              ; ---> a := offset des textes (x)
47129         tax
47130         ldy *a118               ; nummer der funktionstaste
47132         lda *a119               ; länge des neuen textes
47134         sta a1375,y             ; in längentabelle speichern
47137         ldy #0
47139 a47139  jsr a1200               ; ---> lda (a34),y: zeichen aus string
47142         dec *a119               ; länge erniedrigen
47144         bmi a47153              ; ende erreicht: fertig ->
47146         sta a1383,x             ; zeichen in textpuffer kopieren
47149         inx                     ; offset in puffer erhöhen
47150         iny                     ; offset in string erhöhen
47151         bne a47139              ; immer ->
47153
47153 a47153  clc
47154 a47154  rts
47155
47155 a47155  .by '($rhc+"' 13 141 34 27
47166
47166 a47166  lda #0                   ; anfangswert des offsets
47168         clc
47169 a47169  dex                     ; nummer der funktionstaste erniedrigen
47170         bmi a47154              ; ende erreicht: ->
47172         adc a1375,x             ; länge des textes (x) addieren
47175         bcc a47169              ; immer ->
47177
47177 ;-------------------------------  S O U N D
47177
47177 a47177  jsr a40324              ; ---> getbyt: sound typ holen
47180         dex                     ; sound typ
47181         cpx #3                  ; größer als 2?
47183         bcs a47285              ; ja: illegal quantity ->
47185         stx *a128               ; sound typ merken
47187         jsr a40414              ; ---> chkcom, getadr: tonhöhe holen
47190         cmp #4                  ; größer als 1023?
47192         bcs a47285              ; ja: illegal quantity ->
47194         sty *a126               ; l,tonhöhe
47196         sta *a127               ; h,...
47198         jsr a40414              ; ---> chkcom, getadr: tondauer holen
47201         ldx *a128               ; soundtyp
47203         cpx #2                  ; = 2?
47205         bne a47208              ; nein: ->
47207         dex                     ; auf 1 setzen
47208 a47208  pha                     ; h,tondauer auf stack legen
47209         cpy #0                  ; tondauer = 0?
47211         bne a47220
47213         cmp #0
47215         bne a47220              ; nein: ->
47217         iny                     ; auf 1 erhöhen
47218         bne a47235              ; immer ->
47220
47220 a47220  tya
47221         pha
47222 a47222  jsr a36032              ; ---> stoptaste prüfen
47225         lda a1278,x             ; ausgabe des laufenden sounds
47228         ora a1276,x             ; im interrupt beendet?
47231         bne a47222              ; nein: warten ->
47233         pla
47234         tay
```

```
47235 a47235  tya                     ; tondauer
47236         eor #255                ; negieren
47238         clc
47239         adc #1
47241         sei                     ; ergebnis
47242         sta a1276,x             ; für soundtyp 1 in a1276,a1278
47245         pla                     ; für soundtyp 2 in a1277,a1279
47246         eor #255                ; speichern: die beiden zähler werden
47248         adc #0                  ; im interrupt inkrementiert bis 0,
47250         sta a1278,x             ; dann wird der sound abgeschaltet
47253         lda *a126               ; 1,tonhöhe
47255         sta a65294,x            ; nach a65294 bzw. a65295
47258         lda a47288,x
47261         tax
47262         lda a65296,x
47265         and #%11111100
47267         ora *a127               ; h,tonhöhe
47269         sta a65296,x            ; nach a65298 bzw. a65296 (bits 0,1)
47272         ldx *a128               ; soundtyp (0, 1, 2)
47274         lda a47290,x            ; bit 4 bzw. 5 bzw. 6
47277         ora a65297              ; in a65297 setzen
47280         sta a65297
47283         cli
47284         rts
47285
47285 a47285  jmp a39196              ; ---> 'illegal quantity'
47288
47288 a47288  .by 2 0
47290 a47290  .by 16 32 64
47293 ;
47293 ;-------------------------------  V O L
47293
47293 a47293  jsr a40324              ; ---> frmevl, getbyt
47296         cpx #9                  ; gröer als 8?
47298         bcs a47285              ; ja: fehler ->
47300         stx *a128
47302         lda a65297
47305         and #%11110000
47307         ora *a128               ; eingegebener wert
47309         sta a65297              ; vol einstellen
47312         rts
47313
47313 ;-------------------------------  P A I N T
47313
47313 a47313  jsr a50102              ; ---> farb-parameter holen
47316         ldx #4                  ; zeiger in koordinatentabelle
47318         jsr a50137              ; ---> koordinaten holen
47321         jsr a50043              ; ---> und in tabelle speichern
47324         jsr a50085              ; ---> rand-parameter holen
47327         cpx #2                  ; gröer als 1?
47329         bcc a47334              ; nein: ->
47331         jmp a39196              ; ---> 'illegal quantity'
47334
47334 a47334  txa                     ; rand-parameter
47335         lsr a
47336         ror a
47337         sta *a139               ; ergibt rand-flag (bit 7)
47339         bpl a47345              ; nicht gesetzt: ->
47341         lda *a132               ; farb-parameter
47343         beq a47352              ; = 0: fertig ->
```

```
47345 a47345  jsr a49651          ; ---> punkt (x,y) gesetzt?
47348         bcs a47352          ; punkt nicht im bereich: ->
47350         bne a47353          ; punkt nicht gesetzt: ->
47352 a47352  rts
47353
47353 a47353  jsr a43348          ; ---> garbage collect
47356         lda *a49            ; variablenende
47358         sta *a34            ; =: zeiger in koordinaten-stack
47360         lda *a50
47362         sta *a35
47364         sec
47365         lda *a51            ; string-anfang
47367         sbc #3              ; - 3
47369         sta *a25            ; =: maximaler stack-bereich
47371         lda *a52
47373         sbc #0
47375         sta *a26
47377
47377 ;                           x - s c h l e i f e
47377
47377 a47377  ldx #0
47379         stx *a137           ; flag für eintrag in stack 'links'
47381         stx *a138           ; flag für eintrag in stack 'rechts'
47383 a47383  ldx a687            ; y-koordinate
47386         bne a47391
47388         dec a688            ; vermindern
47391 a47391  dec a687
47394         jsr a49651          ; ---> punkt (x,y) gesetzt?
47397         bcs a47401          ; punkt nicht im graphikbereich: ->
47399         bne a47383          ; punkt nicht gesetzt: weiter nach oben ->
47401 a47401  inc a687            ; bis grenze erreicht, dann
47404         bne a47409          ; wieder einen schritt nach unten
47406         inc a688
47409
47409 ;                           y - s c h l e i f e
47409
47409 a47409  jsr a49603          ; ---> punkt (x,y) setzen/löschen
47412         ldx a685            ; x-koordinate um 1 erniedrigen
47415         bne a47420
47417         dec a686
47420 a47420  dec a685
47423         lda *a137           ; flag für 'stack-eintrag links'
47425         jsr a47519          ; ---> stack-eintrag, flag setzen
47428         sta *a137           ; flag wieder abspeichern
47430         clc
47431         lda a685            ; x-koordinate um 2 erhöhen
47434         adc #2
47436         sta a685
47439         bcc a47444
47441         inc a686
47444 a47444  lda *a138           ; flag für 'stack-eintrag rechts'
47446         jsr a47519          ; ---> stack-eintrag, flag setzen
47449         sta *a138           ; flag wieder abspeichern
47451         ldx a685            ; ursprüngliche x-koordinate
47454         bne a47459          ; wiederherstellen
47456         dec a686
47459 a47459  dec a685
47462         inc a687            ; y-koordinate erhöhen
47465         bne a47470
47467         inc a688
```

```
47470 a47470  jsr a49651              ; ---> punkt (x,y) gesetzt?
47473         bcs a47477              ; punkt nicht im graphik-bereich ->
47475         bne a47409              ; punkt nicht gesetzt: neue y-schleife ->
47477 a47477  ldx #3
47479         ldy #0
47481         lda *a35                ; zeiger in koordinaten-stack
47483         cmp *a50                ; stack leer?
47485         bne a47493
47487         lda *a34
47489         cmp *a49
47491         beq a47516              ; ja: fertig ->
47493 a47493  lda *a34                ; zeiger in koordinaten-stack
47495         bne a47499              ; erniedrigen
47497         dec *a35
47499 a47499  dec *a34
47501         jsr a1200               ; ---> lda (a34),y
47504         sta a685,x              ; nächste koordinaten (x,y) aus stack
47507         dex                     ; in koordinatentabelle bringen
47508         bpl a47493
47510         jsr a36032              ; ---> abbruch wenn stoptaste gedrückt
47513         jmp a47377              ; ---> neue x-schleife
47516
47516 a47516  jmp a50043              ; ---> anfangskoordinaten wiederherstellen
47519
47519 a47519  pha                     ; flag merken
47520         jsr a49651              ; ---> punkt (x,y) gesetzt?
47523         bcs a47568              ; punkt nicht im graphik-bereich: ->
47525         beq a47568              ; punkt gesetzt: ->
47527         pla                     ; flag
47528         bne a47571              ; bereits gesetzt: ->
47530         tax                     ; = 0
47531         tay                     ; = 0
47532         lda *a35                ; stack-zeiger
47534         cmp *a26                ; schon am oberen stack-ende?
47536         bcc a47549
47538         bne a47546
47540         lda *a34
47542         cmp *a25
47544         bcc a47549              ; nein: ->
47546 a47546  jmp a34433              ; ---> 'out of memory'
47549
47549 a47549  lda a685,x              ; koordinaten aus tabelle
47552         sta (a34),y             ; auf stack ablegen
47554         inc *a34                ; stack-zeiger erhöhen
47556         bne a47560
47558         inc *a35
47560 a47560  inx
47561         cpx #4
47563         bne a47549
47565         lda #128                ; flagwert für 'stack-eintrag'
47567         rts
47568
47568 a47568  pla
47569         lda #0                  ; flagwert für 'kein stack-eintrag'
47571 a47571  rts
```

```
47572 ;-------------------------------  C H A R
47572
47572 a47572  jsr a50105              ; ---> farb-parameter holen
47575         jsr a40408              ; ---> chkcom, getbyt holt spalte
47578         cpx #40                 ; zulässiger spaltenwert?
47580         bcs a47592              ; nein: fehler ->
47582         stx a730                ; spalte abspeichern
47585         jsr a40408              ; ---> chkcom, getbyt holt zeile
47588         cpx #25                 ; zulässiger zeilenwert?
47590         bcc a47595              ; ja: ->
47592 a47592  jmp a39196              ; ---> 'illegal quantity'
47595
47595 a47595  stx a731                ; zeile abspeichern
47598         jsr a38033              ; ---> chkcom
47601         jsr a40008              ; ---> textstring übernehmen
47604         sta a746                ; stringlänge abspeichern
47607         tya                     ; h,adresse des deskriptors
47608         pha
47609         txa                     ; l,...
47610         pha
47611         jsr a50085              ; ---> getbyt holt rvs-parameter
47614         txa
47615         ror a
47616         ror a697                ; bit 7 = 0: normal, = 1: revers
47619         pla
47620         sta *a34                ; l,zeiger auf string deskriptor
47622         pla
47623         sta *a35                ; h,...
47625         lda *a131               ; graphik-modus eingeschaltet?
47627         bne a47656              ; ja: ->
47629         ldx a731                ; zeile
47632         ldy a730                ; spalte
47635         clc
47636         jsr a65520              ; ---> plot setzt kursor
47639         ldy #0                  ; offset in string
47641 a47641  cpy a746                ; stringende erreicht?
47644         beq a47655              ; ja: ->
47646         jsr a1200               ; ---> lda (a34),y
47649         jsr a65356              ; ---> print
47652         iny                     ; offset erhöhen
47653         bne a47641              ; immer ->
47655
47655 a47655  rts
47656
47656 a47656  jsr a51135              ; ---> prüfen ob graphic eingeschaltet
47659         lda *a134               ; farbe 1 (zeichen-hintergrund)
47661         pha
47662         lda *a132               ; farb-parameter
47664         pha
47665         bit *a131               ; graphik-modus
47667         bpl a47684              ; nicht multicolor: ->
47669         pla
47670         beq a47690              ; farb-parameter = 0: ->
47672         lsr a
47673         beq a47690
47675         ldx *a133               ; farbe 2
47677         bcc a47692
47679         ldx a65302              ; farbe 3
47682         bcs a47692
```

```
47684 a47684  ldx a65301              ; farbe 0 (bildfeld-grundfarbe)
47687         pla
47688         beq a47692              ; farb-parameter = 0: ->
47690 a47690  ldx *a134               ; farbe 1 (zeichen-hintergrund)
47692 a47692  stx *a134
47694         ldx a731                ; zeile
47697         ldy #0                  ; offset in string
47699         sty a732                ; initialisieren
47702 a47702  ldy a732                ; offset in string
47705         inc a732                ; offset erhöhen
47708         jsr a1200               ; ---> lda (a34),y: zeichen aus string
47711         dec a746                ; stringlänge erniedrigen
47714         bmi a47739              ; string zu ende: ->
47716         ldy a730                ; spalte
47719         jsr a47743              ; ---> zeichen in bitspeicher bringen
47722         inc a730                ; spalte erhöhen
47725         cpy #39                 ; letzte spalte?
47727         bcc a47702              ; nein: ->
47729         ldy #0
47731         sty a730                ; spalte := 0
47734         inx                     ; zeile erhöhen
47735         cpx #24                 ; letzte zeile?
47737         bcc a47702              ; nein: ->
47739 a47739  pla
47740         sta *a134               ; farbe 1 wiederherstellen
47742         rts
47743
47743 a47743  pha                     ; zeichen aus string
47744         jsr a49690              ; ---> farbe speichern
47747         jsr a49809              ; ---> zeiger in bitspeicher setzen
47750         lda #0                  ; aus dem zeichenkode wird
47752         sta *a126               ; zunächst der bildschirmkode
47754         pla                     ; erzeugt, dieser wird um 3 bits
47755         pha                     ; nach links geshiftet, also mit
47756         asl a                   ; 8 multipliziert
47757         rol *a126
47759         asl a
47760         asl a                   ; (a)    := l,ergebnis
47761         rol *a126               ; (a126) := h,...
47763         sta *a36                ; l,zeiger in zeichen-bittabelle
47765         lda *a126
47767         adc a740                ; h,anfang der tabelle 1 (aus reset)
47770         sta *a37                ; h,zeiger in zeichen-bittabelle
47772         tya                     ; spalte merken
47773         pha
47774         ldy #7                  ; 8 bitstrings je zeichen
47776 a47776  lda a697                ; rvs-flag
47779         asl a                   ; in carry schieben
47780         lda (a36),y             ; bitstring aus tabelle
47782         bcc a47786              ; nicht revers: ->
47784         eor #255                ; bitstring invertieren
47786 a47786  bit *a131               ; multicolor?
47788         bpl a47833              ; nein: ->
47790         and #%10101010
47792         sta *a126               ; bitstring
47794         lda *a132               ; farb-parameter
47796         bne a47813              ; gesetzt: ->
47798         lda *a126               ; bitstring
47800         bcs a47809              ; rvs-flag gesetzt: ->
```

```
47802          lsr a
47803          eor *a126
47805          eor #%10101010
47807          bne a47833
47809
47809 a47809   ora #%01010101
47811          bne a47833
47813
47813 a47813   cmp #2
47815          bne a47821
47817          lda *a126
47819          bcs a47833
47821
47821 a47821   bcc a47830
47823          lda *a126
47825          lsr a
47826          eor *a126
47828          bcc a47833
47830
47830 a47830   lda *a126
47832          lsr a
47833 a47833   sta (a140),y          ; bitstring in bitspeicher schreiben
47835          dey                  ; zeichen fertig?
47836          bpl a47776           ; nein: ->
47838          pla
47839          tay                  ; spalte
47840          pla                  ; zeichen aus string
47841          rts
47842
47842 ;------------------------------  B O X
47842
47842 a47842   jsr a50102           ; ---> farb-parameter holen
47845          ldx #a716-a685
47847          jsr a50164           ; ---> chkcom, 1. koordinatenpaar holen
47850          ldx #a728-a685
47852          jsr a50137           ; ---> 2. koordinatenpaar holen
47855          jsr a50063           ; ---> winkel holen
47858          sty a720             ; l,winkel (in grad)
47861          sta a721             ; h,...
47864          jsr a50085           ; ---> rand/fläche-parameter holen
47867          cpx #2               ; zulässiger wert 0 oder 1?
47869          bcc a47874           ; ja: ->
47871          jmp a39196           ; ---> 'illegal quantity'
47874
47874 a47874   stx a744              ; rand/fläche
47877          txa
47878          pha
47879          jsr a48052           ; ---> drehung ausführen
47882          pla
47883          bne a47913           ; fläche: ->
47885          beq a47890           ; rand: ->
47887
47887 a47887   jsr a48182           ; ---> neue richtung
47890 a47890   jsr a49370           ; ---> strecke zeichnen
47893          lda a714             ; alle seiten gezeichnet?
47896          bne a47887           ; nein: ->
47898 a47898   ldx #4
47900 a47900   lda a727,x           ; ausgangskoordinaten wiederherstellen
47903          sta a684,x
47906          dex
47907          bne a47900
47909          stx a744
47912          rts
```

```
47913 a47913  ldx #0
47915         lda a709              ; quadrant - 1
47918         lsr a
47919         bcc a47923            ; gerade: ->
47921         ldx #2
47923 a47923  lda a732,x            ; x- bzw. y-koordinatendifferenz
47926         sta a726
47929         lda a733,x
47932         sta a727
47935         lda #0
47937         ldx #3
47939 a47939  sta a722,x            ; bereich a722,...,a725 löschen
47942         dex
47943         bpl a47939
47945 a47945  ldx #7
47947 a47947  lda a685,x            ; koordinaten auf stack legen
47950         pha
47951         dex
47952         bpl a47947
47954         jsr a49370            ; ---> strecke zeichnen
47957         ldx #0
47959 a47959  pla                   ; koordinaten wiederherstellen
47960         sta a685,x
47963         inx
47964         cpx #8
47966         bne a47959
47968 a47968  lda a726              ; koordinatendifferenz vermindern
47971         bne a47978
47973         dec a727
47976         bmi a47898            ; unterlauf: fertig ->
47978 a47978  dec a726
47981         ldx #a722-a685
47983         ldy #a712-a685        ; offset des cosinus
47985         lda a709              ; quadrant - 1
47988         lsr a
47989         bcc a47993            ; gerade: ->
47991         ldy #a710-a685        ; sonst offset des sinus
47993 a47993  lda #0
47995 a47995  lsr a
47996         pha
47997         jsr a49910            ; ---> (a,y) := ko(y) + ko(x)
48000         sta a685,x
48003         tya
48004         sta a686,x
48007         pla
48008         bcc a48012
48010         ora #%10100000        ; flags für koordinatenerhöhung
48012 a48012  inx
48013         inx
48014         ldy #a710-a685        ; offset des sinus
48016         lsr a709              ; quadrant - 1
48019         bcc a48023            ; gerade: ->
48021         ldy #a712-a685        ; sonst offset des cosinus
48023 a48023  rol a709              ; quadrant wiederherstellen
48026         cpx #a724-a685
48028         beq a47995            ; andere koordinate auch ->
48030         ldx #6
48032         asl a                 ; flags für koordinatenerhöhung
48033         beq a47968            ; = 0: ->
```

```
48035 a48035  bcc a48045              ; flag nicht gesetzt: ->
48037         inc a685,x              ; sonst koordinate erhöhen
48040         bne a48045
48042         inc a686,x
48045 a48045  asl a                   ; nächstes flag ins carry
48046         dex
48047         dex
48048         bpl a48035              ; und weitermachen
48050         bmi a47945              ; bis alle koordinaten bearbeitet
48052
48052 ;-------------------------------  D R E H U N G   A U S F Ü H R E N
48052
48052 a48052  ldy #a720-a685          ; offset des drehwinkels
48054         jsr a48214              ; ---> sinus und cosinus berechnen
48057         ldx #a716-a685
48059         ldy #a728-a685
48061 a48061  tya
48062         pha
48063         jsr a49954              ; ---> (a,y) := abs(ko(y)-ko(x)), p=sgn
48066         sta a689,x
48069         sta a693,x
48072         sta a701,x
48075         tya
48076         sta a690,x
48079         sta a694,x
48082         sta a702,x
48085         pla
48086         tay
48087         jsr a49910              ; ---> (a,y) := ko(y) + ko(x)
48090         sta a685,x
48093         tya
48094         sta a686,x
48097         ldy #a730-a685
48099         inx
48100         inx
48101         cpx #a718-a685
48103         beq a48061
48105         lda #%10010000          ; sin/cos-flags (0: sin, 1: cos)
48107         jsr a48341              ; ---> koord mit sin/cos multiplizieren
48110         lda a709
48113         and #3
48115         sta a709
48118         tax
48119         lda a48152,x            ; vorzeichenstring
48122         jsr a48182              ; ---> bildpunkt des 1. punktes berechnen
48125         jsr a50043              ; ---> koordinaten speichern
48128         lda a714                ; plusminus-flags
48131         jsr a48182              ; ---> bildpunkt des 2. punktes berechnen
48134         ldx a709                ; quadrant - 1
48137         lda a48152,x            ; bitstrings aus tabelle
48140         and #%11110000
48142         sta a715                ; als plusminus-flags speichern
48145         lda a48156,x
48148         sta a714
48151         rts
48152
48152 a48152  .by %10111110 %11100100 %01000001 %00011011
48156 a48156  .by %01000001 %00011011 %10111110 %11100100
```

```
48160 ;--------------------------------
48160
48160          .by 'fred b' 13          ; geheimnisvolle namen
48167          .by 'terry r' 13
48175          .by 'mike i' 13
48182
48182 ;-------  B I L D P U N K T  B E I  D R E H U N G  B E R E C H N E N
48182
48182 a48182   jsr a48389               ; ---> verdoppelte koordinaten berechnen
48185          ldx #4
48187 a48187   lda a686,x
48190          asl a                    ; höchstes bit abtrennen
48191          ror a686,x               ; koordinaten durch 2 dividieren
48194          ror a685,x
48197          bcc a48207               ; niederwertigstes bit war null: ->
48199          inc a685,x               ; aufrunden
48202          bne a48207
48204          inc a686,x
48207 a48207   inx
48208          inx
48209          cpx #6
48211          beq a48187
48213          rts
48214
48214 ;-------------------------------  S I N  &  C O S  B E R E C H N E N
48214
48214 a48214   jsr a49944               ; ---> (a,y) := winkel
48217 a48217   ldx #0
48219 a48219   inx                      ; (a,y) durch 90 dividieren
48220          sec
48221          sbc #90
48223          bcs a48219
48225          dey
48226          bpl a48219
48228          stx a709                 ; quadrant des winkels
48231          pha                      ; negativer winkel im 4. quadr
48232          adc #90                  ; positiver winkel im 1. quadr
48234          jsr a48246               ; ---> sinus berechnen
48237          pla                      ; negativer winkel im 4. quadr
48238          clc
48239          eor #255                 ; vorzeichen wechseln
48241          adc #1
48243          dec a709                 ; quadr dekr zur cosinus-berechnung
48246 a48246   ldx #255
48248 a48248   inx                      ; winkel durch 10 dividieren
48249          sec
48250          sbc #10
48252          bcs a48248
48254          adc #10
48256          sta *a142                ; rest
48258          txa                      ; zehnerwert
48259          asl a                    ; verdoppeln
48260          tax                      ; ergibt zeiger in sinus-tabelle
48261          lda a50355+1,x           ; l,sinus
48264          ldy a50355,x             ; h,...
48267 a48267   clc
48268          dec *a142                ; einerwert vermindern
48270          bmi a48284               ; negativ: fertig ->
```

```
48272          adc a50375+1,x          ; wert aus interpolationstabelle
48275          pha                     ; so oft addieren, wie einerwert
48276          tya                     ; des winkels angibt
48277          adc a50375,x
48280          tay
48281          pla
48282          bcc a48267              ; immer ->
48284
48284 a48284   pha                     ; l,sinus
48285          ldx #0
48287          lda a709                ; nummer des quadranten
48290          lsr a
48291          bcs a48295              ; ungerade (gerade): x = 0 ->
48293          ldx #2                  ; sonst: x = 2
48295 a48295   pla                     ; ergebnis:
48296          sta a710,x              ; (a710,a711) := abs(sin(w))
48299          tya                     ; (a712,a713) := abs(cos(w))
48300          sta a711,x
48303          rts
48304
48304 ;----------------------------   ( A , Y )  :=  K O ( X ) * S I N / C O S
48304
48304 a48304   ldy #a710-a685          ; cc: sinus
48306          bcc a48310
48308          ldy #a712-a685          ; cs: cosinus
48310 a48310   lda a709                ; quadrant - 1
48313          adc #2                  ; vorzeichen bestimmen
48315          lsr a
48316          lsr a                   ; '+' setzt carry als vorzeichenflag
48317          php
48318          jsr a49944              ; ---> (a,y) := sin/cos
48321          cpy #255                ; sin/cos kleiner als 1?
48323          bcc a48332              ; ja: ->
48325          txa
48326          tay                     ; y := x
48327          jsr a49944              ; ---> (a,y) := ko(x)
48330          bcs a48335              ; immer ->
48332
48332 a48332   jsr a49975              ; ---> (a,y) := (a,y) * ko(x)
48335 a48335   plp                     ; vorzeichen
48336          bcs a48365              ; '+': ->
48338          jmp a49959              ; ---> (a,y) := - (a,y)
48341
48341 ;-  4  K O O R D I N A T E N  M I T  S I N / C O S  M U L T I P L
48341
48341 a48341   sta a714                ; sin/cos-flags
48344          ldx #a720-a685
48346 a48346   asl a714                ; sin/cos-flag ins carry schieben
48349          jsr a48304              ; ---> (a,y) := ko(x) * sin/cos
48352          sta a685,x              ; ko(x) := (a,y)
48355          tya
48356          sta a686,x
48359          inx
48360          inx
48361          cpx #a728-a685          ; schon 4 koordinaten multipliziert?
48363          bcc a48346              ; nein: ->
48365 a48365   rts
```

```
48366 ;-----------------  K O O R D I N A T E N - T R A N S F O R M A T I O N
48366
48366 a48366  ldy #a728-a685
48368 a48368  jsr a48214              ; ---> sinus und cosinus berechnen
48371         ldx #7
48373 a48373  lda a732,x              ; koordinaten umkopieren
48376         sta a720,x
48379         dex
48380         bpl a48373
48382         lda #%01010000          ; sin/cos-flags (0: sin, 1: cos)
48384         jsr a48341              ; ---> koord mit sin/cos multiplizieren
48387         lda #%00010000
48389 a48389  sta a714                ; sin/cos-flags
48392         ldy #a716-a685
48394         ldx #a720-a685
48396 a48396  asl a715                ; plusminus-flag ins carry
48399         rol a714
48402         jsr a49908              ; ---> (a,y) := ko(y) +- ko(x) (cc: +)
48405         inx
48406         inx
48407         asl a715                ; plusminus-flag ins carry
48410         rol a714
48413         jsr a49904              ; ---> (a,y) := (a,y) +- ko(x) (cc: +)
48416         pha                     ; (a,y) auf stack legen
48417         tya
48418         pha
48419         ldy #a718-a685
48421         inx
48422         inx
48423         cpx #a724-a685          ; zweimal durchlaufen?
48425         beq a48396              ; nein: ->
48427         ldx #3
48429 a48429  pla                     ; ergebnisse vom stack nehmen
48430         sta a689,x              ; und speichern
48433         dex
48434         bpl a48429
48436         rts
48437
48437 ;-------------------------------  G S H A P E
48437
48437 a48437  jsr a51135              ; ---> prüfen ob graphic eingeschaltet
48440         jsr a40008              ; ---> frmevl, chkstr, frestr
48443         sta a719                ; stringlänge
48446         stx *a36                ; l,stringadresse
48448         sty *a37                ; h,...
48450         ldx #a689-a685
48452         jsr a50137              ; ---> koordinaten holen
48455         jsr a50085              ; ---> getbyt
48458         cpx #5                  ; parameter gröer als 4?
48460         bcc a48465              ; nein: ->
48462         jmp a39196              ; ---> 'illegal quantity'
48465
48465 a48465  stx a720                ; ausgabeparameter merken
48468         ldx #3
48470         ldy a719                ; stringlänge
48473         cpy #5                  ; gröer als 4?
48475         bcs a48478              ; ja: ->
48477         rts
```

```
48478 a48478  dey
48479         jsr a1211              ; ---> lda (a36),y: die letzten vier
48482         sta a725,x             ; zeichen aus dem string sind breite
48485         dex                    ; und höhe des shape-bereichs
48486         bpl a48478
48488         stx a721               ; = 255: zeiger in string initialisieren
48491         jsr a50043             ; ---> koordinaten umspeichern
48494         lda a725               ; breite des bereichs
48497         sta a729
48500         lda a726
48503         sta a730
48506 a48506  lda #8
48508         sta a741               ; bitzähler initialisieren
48511         inc a721               ; zeiger in string erhöhen
48514         ldy a721
48517         jsr a1211              ; ---> lda (a36),y: zeichen aus string
48520         sta a723               ; merken
48523 a48523  jsr a49651             ; ---> punkt (x,y) testen
48526         sta a722               ; bits 0 und 1 sind die punktbits
48529         asl a723               ; zeichen aus string nach links shiften
48532         rol a
48533         dec a741               ; bitzähler erniedrigen
48536         bit *a131              ; graphik-modus
48538         bpl a48547             ; nicht multicolor: ->
48540         asl a723               ; zeichen aus string nach links shiften
48543         rol a
48544         dec a741               ; bitzähler erniedrigen
48547 a48547  ldx a720               ; ausgabeparameter p
48550         cpx #3
48552         bcc a48566             ; kleiner als 3: ->
48554         beq a48561             ; = 3: ->
48556         eor a722               ; p = 4
48559         bcs a48578
48561
48561 a48561  and a722               ; p = 3
48564         bcs a48578
48566
48566 a48566  cpx #1
48568         bcc a48578             ; p = 0: ->
48570         beq a48576             ; p = 1: ->
48572         ora a722               ; p = 2
48575         .by 44
48576 a48576  eor #255               ; p = 1
48578 a48578  and #3                 ; bits 0 und 1 isolieren
48580         bit *a131              ; graphik-modus
48582         bmi a48586             ; multicolor: ->
48584         and #1                 ; bit 0 isolieren
48586 a48586  sta *a132              ; als farb-parameter speichern
48588         jsr a49603             ; ---> punkt (x,y) setzen oder löschen
48591         inc a685               ; koordinate x erhöhen
48594         bne a48599
48596         inc a686
48599 a48599  sec
48600         lda a729               ; breite des shape-bereichs
48603         bit *a131              ; bei multicolor
48605         bpl a48610
48607         sbc #2                 ; um 2,
48609         .by 44
```

```
48610 a48610  sbc #1                  ; sonst um 1 vermindern
48612         sta a729
48615         lda a730
48618         sbc #0
48620         sta a730
48623         bcs a48670              ; kein unterlauf: ->
48625         ldx #1
48627 a48627  lda a725,x              ; ausgangswert der breite
48630         sta a729,x              ; wiederherstellen
48633         lda a689,x              ; koordinaten
48636         sta a685,x              ; wiederherstellen
48639         dex
48640         bpl a48627
48642         inc a687                ; y-koordinate erhöhen
48645         bne a48650
48647         inc a688
48650 a48650  sec
48651         lda a727                ; höhe des bereichs
48654         sbc #1                  ; um 1 vermindern
48656         sta a727
48659         lda a728
48662         sbc #0
48664         sta a728
48667         bcs a48678              ; kein unterlauf: ->
48669         rts
48670
48670 a48670  lda a741                ; bitzähler
48673         beq a48678              ; = 0: ->
48675         jmp a48523              ; ---> nächster x-wert
48678
48678 a48678  jmp a48506              ; ---> nächstes zeichen
48681
48681 ;-------------------------------  S S H A P E
48681
48681 a48681  jsr a51135              ; ---> prüfen ob graphic eingeschaltet
48684         jsr a38565              ; ---> variable suchen oder anlegen
48687         sta a731                ; l,zeiger auf deskriptor
48690         sty a732                ; h,...
48693         bit *a13                ; string?
48695         bmi a48700              ; ja: ->
48697         jmp a37668              ; ---> 'type mismatch'
48700
48700 a48700  ldx #a725-a685
48702         jsr a50164              ; ---> anfangskoordinaten holen
48705         ldx #a689-a685
48707         jsr a50137              ; ---> endkoordinaten holen
48710         ldx #a727-a685
48712         ldy #a691-a685
48714         lda #2
48716         sta *a142
48718 a48718  jsr a49954              ; ---> (a,y) := abs(ko(y)-ko(x)), p=sgn
48721         tax
48722         tya
48723         pha
48724         ldy *a142
48726         jsr a50050              ; ---> ko(y) := ko(y+4)
48729         bcc a48743              ; koordinatendifferenz negativ: ->
48731         lda a725,y              ; eingegebene koordinate übernehmen
48734         sta a685,y
48737         lda a726,y
48740         sta a686,y
```

```
48743 a48743  txa                    ; betrag der differenz
48744         sta a725,y             ; als höhe bzw. breite des
48747         sta a734,y             ; shape-bereichs speichern
48750         pla
48751         sta a726,y
48754         sta a735,y
48757         ldx #a725-a685         ; offsets für x-koordinaten einrichten
48759         ldy #a689-a685
48761         dec *a142
48763         dec *a142
48765         beq a48718             ; nach erstem durchlauf ->
48767         ldy #255
48769         sty a721               ; zeiger in string initialisieren
48772         lda a685               ; x-koordinate
48775         sta a729
48778         lda a686
48781         sta a730
48784         tya                    ; = 255
48785         jsr a39772             ; ---> platz für string reservieren
48788 a48788  jsr a49764             ; ---> zeiger in bitspeicher setzen
48791         lda (a140),y           ; byte aus bitspeicher
48793         bcc a48809             ; punkt im graphik-bereich: ->
48795         lda a685               ; l,x-koordinate
48798         bit *a131              ; multicolor?
48800         bpl a48804             ; nein: ->
48802         sec                    ; bit 1 einrotieren
48803         rol a
48804 a48804  and #7                 ; bits 0,1,2 isolieren
48806         tax                    ; ergebnis als bitoffset für x,
48807         lda #0                 ; bitwert 0 übernehmen
48809 a48809  bit *a131              ; multicolor?
48811         bpl a48814             ; nein: ->
48813         dex                    ; bitoffset erniedrigen
48814 a48814  stx a733               ; bitoffset merken
48817 a48817  asl a                  ; bitwert
48818         dex                    ; ins carry rotieren
48819         bpl a48817
48821         ror a                  ; zurück ins bit 7
48822         sta *a142              ; und abspeichern
48824         lda #8                 ; inkrement für x
48826         bit *a131              ; multicolor?
48828         bpl a48831             ; nein: ->
48830         lsr a                  ; sonst halbieren
48831 a48831  clc
48832         adc a685               ; zu x-koordinate addieren
48835         sta a685
48838         bcc a48843
48840         inc a686
48843 a48843  jsr a49764             ; ---> zeiger in bitspeicher setzen
48846         lda #0
48848         bcs a48852             ; punkt nicht im grahik bereich: ->
48850         lda (a140),y           ; byte aus bitspeicher
48852 a48852  sta *a143              ; merken
48854         ldx a733               ; bitoffset vom letzten punkt
48857 a48857  lsr a                  ; byte aus bitspeicher nach rechts
48858         inx                    ; shiften und x erhöhen
48859         cpx #8
48861         bne a48857             ; x noch nicht 8: ->
48863         ora *a142              ; ergebnis vom vorigen punkt einodern
```

```
48865          inc a721              ; zeiger in string erhöhen
48868          ldy a721
48871          cpy #252              ; noch vier stellen frei?
48873          bcc a48878            ; ja: ->
48875          jmp a52300            ; ---> 'string too long'
48878
48878 a48878   sta (a98),y           ; byte in string bringen
48880          ldx a733              ; bitoffset
48883          lda a725              ; breite des shape-bereichs
48886          sec
48887          bit *a131             ; multicolor?
48889          bpl a48894            ; nein: ->
48891          sbc #4                ; breite bei multicolor um 4,
48893          .by 44
48894 a48894   sbc #8                ; sonst um 8 vermindern
48896          sta a725
48899          lda *a143             ; letztes byte aus bitspeicher
48901          bcs a48817            ; kein unterlauf: ->
48903          dec a726              ; h,breite des shape-bereichs vermindern
48906          bpl a48817            ; kein unterlauf: ->
48908          ldx a727              ; l,höhe des shape-bereichs
48911          bne a48979            ; nicht null: ->
48913          dec a728              ; h,höhe des shape-bereichs vermindern
48916          bpl a48979            ; kein unterlauf: ->
48918          bit *a131             ; graphik-modus
48920          bpl a48928            ; nicht multicolor: ->
48922          asl a734              ; breite verdoppeln
48925          rol a735
48928 a48928   ldx #0
48930 a48930   lda a734,x            ; breite und höhe als letzte 4 bytes
48933          iny
48934          sta (a98),y           ; in den string schreiben
48936          inx
48937          cpx #4
48939          bne a48930
48941          iny
48942          sty a734              ; stringlänge
48945          lda *a98              ; l,stringadresse
48947          sta a735
48950          lda *a99              ; h,...
48952          sta a736
48955          lda #l,a734           ; adresse des quelldeskriptors
48957          sta *a100             ; übergeben
48959          lda #h,a734
48961          sta *a101
48963          lda a731              ; zeiger auf zieldeskriptor
48966          sta *a73              ; übergeben
48968          lda a732
48971          sta *a74
48973          jsr a36672            ; ---> string in stringbereich kopieren
48976          jmp a50043            ; ---> ko(0) := ko(4)
48979
48979 a48979   dec a727              ; l,höhe des shape-bereichs vermindern
48982          inc a687              ; y-koordinate erhöhen
48985          bne a48990
48987          inc a688
48990 a48990   lda a729              ; ausgangswert der x-koordinate
48993          sta a685              ; wiederherstellen
48996          lda a730
48999          sta a686
```

```
49002          lda a734              ; ausgangswert der breite des shape-
49005          sta a725              ; bereichs wiederherstellen
49008          lda a735
49011          sta a726
49014          jmp a48788            ; ---> weitermachen
49017
49017 ;------------------------------  R G R
49017
49017 a49017   lda *a131             ; graphik-modus
49019          clc                   ; a = 0, 32, 96, 160, 224
49020          rol a
49021          rol a
49022          rol a
49023          adc #0
49025          tay                   ; y  =  0, 1, 2, 3, 4
49026          jmp a39553            ; ---> (y) als gk-zahl nach fac
49029
49029 ;------------------------------  R C L R
49029
49029 a49029   sec
49030          .by 36
49031
49031 ;------------------------------  R L U M
49031
49031 a49031   clc
49032          php
49033          jsr a40327            ; ---> getbyt (0,...,4)
49036          lda a65305            ; farbe 4
49039          and #127              ; bit 7 löschen
49041          cpx #4                ; eingabe 4?
49043          beq a49070            ; ja: ->
49045          bcs a49086            ; gröer als 4: fehler ->
49047          lda a65301            ; farbe 0
49050          and #127              ; bit 7 löschen
49052          dex                   ; eingabe 0?
49053          bmi a49070            ; ja: ->
49055          lda *a134             ; farbe 1
49057          dex                   ; eingabe 1?
49058          bmi a49070            ; ja: ->
49060          lda *a133             ; farbe 2
49062          dex                   ; eingabe 2?
49063          bmi a49070            ; ja: ->
49065          lda a65302            ; farbe 3
49068          and #127              ; bit 7 löschen
49070 a49070   plp                   ; carry wiederherstellen
49071          bcs a49078            ; aufruf mit cs: ->
49073          lsr a                 ; 'lum' = bits 4,5,6
49074          lsr a                 ; werden nach unten geschoben
49075          lsr a
49076          lsr a
49077          clc                   ; nichts addieren
49078 a49078   adc #0                 ; bei aufruf mit cs wird 1 addiert
49080          and #15               ; oberes halbbyte löschen
49082          tay
49083          jmp a39553            ; ---> (y) als gk-zahl nach fac
49086
49086 a49086   jmp a39196            ; ---> 'illegal quantity'
```

```
49089 ;-------------------------------  J O Y
49089
49089 a49089  jsr a40327              ; ---> getbyt (1, 2)
49092         dex
49093         cpx #2                  ; zulässige eingabe?
49095         bcs a49086              ; nein: fehler ->
49097         lda a49147,x            ; bitmuster zur abfrage des
49100         tax                     ; joystick-ports
49101         sei
49102 a49102  stx a65288              ; port aktivieren
49105         lda a65288              ; und lesen
49108         stx a65288              ; daten stabil?
49111         cmp a65288
49114         bne a49102              ; nein: warten ->
49116         cli
49117         eor #255                ; datenbyte invertieren
49119         tay                     ; und merken
49120         and #15                 ; richtungsbits isolieren
49122         tax
49123         lda a49136,x            ; richtungswert aus tabelle         1
49126         cpy #15                 ; feuer-knopf gedrückt?          8     2
49128         bcc a49132              ; nein: ->                     7         3
49130         ora #128                ; bit 7 setzen                   6     4
49132 a49132  tay                     ;                                   5
49133         jmp a39553              ; ---> (y) als gk-zahl nach fac
49136
49136 a49136  .by 0 1 5 0 7 8 6 0 3 2 4
49147 a49147  .by 250 253
49149 ;
49149 ;-------------------------------  R D O T
49149
49149 a49149  jsr a40327              ; ---> getbyt (0, 1, 2)
49152         cpx #2                  ; kleiner als 2?
49154         bcc a49169              ; ja: koordinate ausgeben ->
49156         bne a49086              ; gröer als 2: fehler ->
49158         jsr a49651              ; ---> punkt gesetzt?
49161         tay                     ; gesetzt: > 0, nicht gesetzt: 0
49162         bcc a49166              ; punkt im graphikbereich: ->
49164         ldy #0                  ; sonst 0 ausgeben
49166 a49166  jmp a39553              ; ---> (y) als gk-zahl nach fac
49169
49169 a49169  txa                     ; eingabe
49170         asl a                   ; verdoppeln
49171         tax
49172         lda a685,x              ; eingabe 0: x-koordinate
49175         tay                     ;         1: y-koordinate
49176         lda a686,x
49179         jmp a38001              ; ---> (a,y) als gk-zahl nach fac
49182
49182 ;-------------------------------  C I R C L E
49182
49182 a49182  jsr a50102              ; ---> farb-parameter holen
49185         ldx #a716-a685
49187         jsr a50137              ; ---> mittelpunktskoordinaten holen
49190         jsr a50063              ; ---> x-halbmesser holen
49193         sty a720                ; l,x-halbmesser
49196         sta a721                ; h,...
49199         jsr a50063              ; ---> y-halbmesser holen
49202         sty a722                ; l,y-halbmesser
49205         sta a723                ; h,...
```

```
49208          php
49209          ldx #a720-a685
49211          jsr a49875              ; ---> scale
49214          plp
49215          bcs a49234              ; letzte eingabe nicht leer: ->
49217          lda a720                ; x-halbmesser
49220          sta a722                ; =: y-halbmesser
49223          lda a721
49226          bit *a131               ; graphik-modus
49228          bpl a49234              ; nicht multicolor: ->
49230          asl a722                ; sonst halbmesser verdoppeln
49233          rol a
49234 a49234   sta a723
49237          jsr a50063              ; ---> anfangswinkel des bogens holen
49240          sty a728                ; l,anfangswinkel
49243          sta a729                ; h,...
49246          jsr a50063              ; ---> endwinkel des bogens holen
49249          sty a730                ; l,endwinkel
49252          sta a731                ; h,...
49255          jsr a50063              ; ---> drehwinkel holen
49258          sta *a128               ; a und y vertauschen
49260          tya
49261          ldy *a128
49263          jsr a48217              ; ---> sinus und cosinus berechnen
49266          ldx #a730-a685
49268          ldy #a728-a685
49270          jsr a49925              ; ---> (a,y) := ko(y) - ko(x)
49273          bcc a49289              ; anfangswinkel kleiner als endwinkel: ->
49275          lda #l,360
49277          ldy #h,360
49279          jsr a49913              ; ---> (a,y) := ko(x) + (a,y)
49282          sta a685,x              ; endwinkel um 360 grad erhöhen
49285          tya
49286          sta a686,x
49289 a49289   ldx #3
49291 a49291   lda a720,x              ; halbmesserwerte umspeichern
49294          sta a724,x
49297          dex
49298          bpl a49291
49300          lda #%10010000          ; sin/cos-flags (0: sin, 1: cos)
49302          jsr a48341              ; ---> koord mit sin/cos multiplizieren
49305          ldx #7
49307 a49307   lda a720,x
49310          sta a732,x
49313          dex
49314          bpl a49307
49316          jsr a48366              ; ---> drehung ausführen
49319          jsr a50043              ; ---> (x,y) wiederherstellen
49322          ldx #2                  ; ersatzwert
49324          jsr a50087              ; ---> getbyt
49327          stx *a233               ; schrittweite in grad
49329          clc
49330 a49330   lda *a233                ; schrittweite
49332          bne a49337              ; nicht null: ->
49334          jmp a39196              ; ---> 'illegal quantity'
49337
49337 a49337   adc a728                ; schrittweite zum anfangswert
49340          sta a728                ; des bogens addieren
49343          bcc a49348
49345          inc a729
```

```
49348 a49348  ldx #a730-a685          ; offset des bogenendes
49350         ldy #a728-a685          ; offset der aktuellen bogenkoordinate
49352         jsr a49925              ; ---> (a,y) := ko(y) - ko(x)
49355         bcs a49365              ; differenz positiv: ->
49357         jsr a48366              ; ---> drehung ausführen
49360         jsr a49370              ; ---> strecke zeichnen
49363         bcc a49330              ; immer ->
49365
49365 a49365  ldy #a730-a685
49367         jsr a48368              ; ---> drehung ausführen
49370
49370 ;-------------------------------  S T R E C K E  Z E I C H N E N
49370
49370 a49370  ldx #a687-a685          ; offset von y1
49372         ldy #a691-a685          ; offset von y2
49374 a49374  lda #0
49376         sta a697,x              ; addend := 0
49379         sta a698,x
49382         jsr a49954              ; ---> (a,y) := ko(y) - ko(x)
49385         bpl a49395              ; nicht negativ: ->
49387         dec a697,x              ; addend := 65535
49390         dec a698,x
49393         bne a49406
49395
49395 a49395  cmp #0                  ; differenz = 0?
49397         bne a49403
49399         cpy #0
49401         beq a49406              ; ja: ->
49403 a49403  inc a697,x              ; addend := 1
49406 a49406  sta a693,x              ; koordinatendifferenz abspeichern
49409         asl a                   ; verdoppeln
49410         sta a701,x              ; ergebnis abspeichern
49413         tya
49414         sta a694,x
49417         rol a
49418         sta a702,x
49421         dex
49422         dex
49423         ldy #a689-a685
49425         cpx #0
49427         beq a49374              ; nach dem ersten durchlauf ->
49429         ldx #a695-a685
49431         ldy #a693-a685
49433         jsr a49925              ; ---> (a,y) := ko(y) - ko(x)
49436         lda #0                  ; nicht-negatives ergebnis setzt carry
49438         rol a
49439         rol a
49440         sta a707                ; = 0 wenn negativ, sonst = 2
49443         eor #2
49445         sta a708                ; = 2 bzw. 0
49448         clc
49449         lda #16
49451         adc a707
49454         tay                     ; = 16 bzw. 18
49455         pha
49456         eor #2
49458         tax                     ; = 18 bzw. 16
49459         jsr a49925              ; ---> (a,y) := ko(y) - ko(x)
49462         sta a685,x
49465         tya
```

```
49466          sta a686,x
49469          pla
49470          tay                      ; = 16 bzw. 18
49471          clc
49472          lda #8
49474          adc a708
49477          tax                      ; = 10 bzw. 8
49478          jsr a49925               ; ---> (a,y) := ko(y) - ko(x)
49481          sta a705
49484          sty a706
49487 .a49487  jsr a49573               ; ---> punkt (x,y) setzen
49490          ldy a708
49493          sec
49494          lda a693,y               ; koordinatendifferenz vermindern
49497          sbc #1
49499          sta a693,y
49502          bcs a49515
49504          lda a694,y
49507          sbc #0
49509          sta a694,y
49512          bcs a49515               ; kein unterlauf: ->
49514          rts
49515
49515 a49515   ldx a707
49518          lda a706
49521          bmi a49529
49523          jsr a49556               ; ---> inkrement addieren
49526          ldx a708
49529 a49529   clc
49530          lda a705
49533          adc a701,x
49536          sta a705
49539          lda a706
49542          adc a702,x
49545          sta a706
49548          ldx a708
49551          jsr a49556               ; ---> inkrement addieren
49554          beq a49487               ; immer ->
49556
49556 a49556   ldy #2
49558          clc
49559 a49559   lda a685,x
49562          adc a697,x
49565          sta a685,x
49568          inx
49569          dey
49570          bne a49559
49572          rts
49573
49573 ;-------------------------------
49573
49573 a49573   lda a744
49576          ora a743
49579          beq a49603
49581          inc a685
49584          bne a49589
49586          inc a686
49589 a49589   jsr a49603               ; ---> punkt (x,y) setzen
49592          ldx a685
49595          bne a49600
49597          dec a686
49600 a49600   dec a685
```

```
49603 ;------------------------------  P U N K T   S E T Z E N
49603
49603 a49603  jsr a49837              ; ---> zeile und spalte nach (x,y)
49606         bcs a49644              ; graphik-bereich überschritten: ->
49608         jsr a49690              ; ---> farbe und helligkeit speichern
49611         jsr a49769              ; ---> zeiger in bitspeicher setzen
49614         sta a745                ; bitwert des offsets
49617         lda (a140),y            ; byte aus bit-speicher
49619         ora a745                ; bitwert einodern
49622         bit *a131               ; graphik-modus
49624         bpl a49645              ; nicht multicolor: ->
49626         pha
49627         ldx *a132               ; farb-parameter
49629         lda a745                ; bitwert des offsets
49632         and a50351,x            ; und maske des farbparameters
49635         sta a745                ; ergibt farbmaske
49638         pla                     ; byte für bitspeicher
49639 a49639  eor a745                ; farbmaske
49642 a49642  sta (a140),y            ; byte wieder abspeichern
49644 a49644  rts
49645
49645 a49645  ldx *a132               ; farb-parameter
49647         bne a49642              ; nicht null: ->
49649         beq a49639              ; sonst ->
49651
49651 ;------------------------------  P U N K T   T E S T E N
49651
49651 a49651  jsr a49764              ; ---> zeiger in bitspeicher setzen
49654         bcs a49689              ; graphikbereich überschritten: ->
49656         sta a745                ; bitwert des punktes (x,y)
49659         lda (a140),y            ; byte aus bitspeicher
49661         and a745                ; punktbit(s) isolieren
49664 a49664  rol a                   ; nach links rotieren
49665         dex                     ; bis die bits an den stellen 0 und 1
49666         bpl a49664              ; stehen
49668         rol a
49669         bit *a139               ; flag für vergleich gesetzt?
49671         bmi a49679              ; nein: ->
49673         and #3                  ; bits isolieren
49675         cmp *a132               ; und mit farb-parameter vergleichen
49677         clc
49678         rts
49679
49679 a49679  clc
49680         and #3                  ; bits isolieren
49682         beq a49687              ; beide null: ->
49684         ldx #0
49686         rts
49687
49687 a49687  ldx #255
49689 a49689  rts
49690
49690 ;---------------  F A R B E   U N D   H E L L I G K E I T   S E T Z E N
49690
49690 a49690  lda a55298,x            ; l,zeilenadresse (x)
49693         sta *a140
49695         lda a55323,x            ; h,...
49698         and #3
49700         pha
49701         ora #h,a7168
49703         sta *a141
```

```
49705          jsr a49720              ; ---> farbkode erzeugen
49708          sta (a140),y            ; und speichern
49710          pla
49711          ora #h,a6144
49713          sta *a141
49715          jsr a49742              ; ---> helligkeitskode erzeugen
49718          sta (a140),y            ; und speichern
49720 a49720   lda *a134               ; farb-parameter 1 (zeichen)
49722          asl a                   ; farbkode in oberes halbbyte schieben
49723          asl a
49724          asl a
49725          asl a
49726          sta *a126               ; ergebnis merken
49728          lda a65301              ; farb-parameter 0 (bildhintergrund)
49731          bit *a131               ; graphik-modus
49733          bpl a49737              ; nicht multicolor: ->
49735          lda *a133               ; farb-parameter 0 bei multicolor
49737 a49737   and #15                 ; farbkode isolieren
49739          ora *a126               ; zeichenfarbe einodern
49741          rts
49742
49742 a49742   lda *a134               ; farb-parameter 1 (zeichen)
49744          lsr a                   ; helligkeitskode ins untere
49745          lsr a                   ; halbbyte schieben
49746          lsr a
49747          lsr a
49748          sta *a126               ; ergebnis merken
49750          lda a65301              ; farb-parameter 0 (bildhintergrund)
49753          bit *a131               ; graphik-modus
49755          bpl a49759              ; nicht multicolor: ->
49757          lda *a133               ; farb-parameter 0 bei multicolor
49759 a49759   and #240                ; helligkeitskode isolieren
49761          ora *a126               ; zeichenhelligkeit einodern
49763          rts
49764
49764 ;-------------  Z E I G E R   I N   B I T S P E I C H E R   S E T Z E N
49764
49764 a49764   jsr a49837              ; ---> zeile und spalte nach (x,y)
49767          bcs a49800              ; graphik-bereich überschritten: ->
49769 a49769   jsr a49809              ; ---> byteadresse nach (a140,a141)
49772          lda a687                ; l,y-koordinate
49775          and #7                  ; bits 0,1,2
49777          tay                     ; ergeben y-offset
49778          lda a685                ; l,x-koordinate
49781          bit *a131               ; graphik-modus
49783          php
49784          bpl a49787              ; nicht multicolor: ->
49786          asl a                   ; l,x-koordinate verdoppeln
49787 a49787   and #7                  ; bits 0,1,2
49789          tax                     ; ergeben x-offset
49790          lda a49801,x            ; bitwert des x-offsets
49793          plp                     ; graphik-modus
49794          bpl a49800              ; nicht multicolor: ->
49796          inx                     ; sonst
49797          ora a49801,x            ; auch noch das nächste bit setzen
49800 a49800   rts
49801
49801 a49801   .by 128 64 32 16 8 4 2 1        ; bitwerte
```

```
49809 ;------------------------------  B Y T E Z E I G E R  S E T Z E N
49809
49809 a49809  tya                     ; spalte
49810         clc
49811         adc a55298,x            ; + 1,zeilenanfangsadresse
49814         sta *a140               ; =: l,bytezeiger
49816         lda a55323,x            ; h,zeilenanfangsadresse
49819         and #3                  ; bits 0 und 1 isolieren
49821         adc #0                  ; carry addieren
49823         asl *a140               ; die ganze adresse
49825         rol a                   ; mit 8 multiplizieren
49826         asl *a140
49828         rol a
49829         asl *a140
49831         rol a                   ; h-byte der adresse
49832         ora #h,a8192            ; + basisadresse des bitspeichers
49834         sta *a141               ; =: h,bytezeiger
49836         rts
49837
49837 ;---------------------  Z E I L E ,  S P A L T E  N A C H  ( X , Y )
49837
49837 a49837  lda a686                ; h,x-koordinate
49840         lsr a                   ; gröer als 511?
49841         bne a49873              ; ja: nicht im graphik-bereich ->
49843         lda a685                ; l,x-koordinate
49846         ror a                   ; durch 4 dividieren
49847         lsr a
49848         bit *a131               ; graphik-modus
49850         bmi a49853              ; multicolor: ->
49852         lsr a                   ; nochmal durch 2 dividieren
49853 a49853  tay                     ; ergebnis
49854         cpy #40                 ; gröer als 39?
49856         bcs a49873              ; ja: keine zulässige spaltennummer ->
49858         lda a688                ; h,y-koordinate
49861         bne a49873              ; nicht null: nicht im graphik-bereich ->
49863         lda a687                ; l,y-koordinate
49866         lsr a                   ; durch 8 dividieren
49867         lsr a
49868         lsr a
49869         tax
49870         cmp *a136               ; mit graphik-zeilenzahl vergleichen
49872         rts
49873
49873 a49873  sec
49874         rts
49875
49875 ;------------------------------  S C A L E  A U S F Ü H R E N
49875
49875 a49875  lda a742                ; scale-flag
49878         beq a49903              ; nicht gesetzt: ->
49880         lda *a135               ; spaltenzahl
49882         jsr a49887              ; ---> x-koordinate umrechnen
49885         lda *a136               ; zeilenzahl
49887 a49887  asl a                   ; verdoppeln
49888         tay                     ; * 256, weil h-byte
49889         lda #0                  ; l-byte
49891         jsr a49975              ; ---> (a,y) := (a,y) * ko(x) / 65536
49894         sta a685,x              ; neue koordinatenwerte speichern,
49897         tya                     ; die x-koordinate ist mit 40/256 (bei
49898         inx                     ; multicolor mit 20/256), die y-koordi-
49899         sta a685,x              ; nate mit 25/256 multipliziert worden,
49902         inx                     ; dadurch haben beide koordinaten nun
49903 a49903  rts                     ; den maximalwert 1023
```

```
49904 ;----------------------------  ( A , Y )  :=  ( A , Y )  + -  K O ( X )
49904
49904 a49904  bcc a49913             ; cc: addieren ->
49906         bcs a49928             ; cs: subtrahieren ->
49908
49908 ;----------------------------  ( A , Y )  :=  K O ( Y )  + -  K O ( X )
49908
49908 a49908  bcs a49925             ; cs: subtrahieren ->
49910 a49910  jsr a49944             ; ---> (a,y) := ko(y)
49913 a49913  clc
49914         adc a685,x             ; ko(x) addieren
49917         pha
49918         tya
49919         adc a686,x
49922         tay
49923         pla
49924         rts
49925
49925 ;----------------------------  ( A , Y )  :=  K O ( Y )  -  K O ( X )
49925
49925 a49925  jsr a49944             ; ---> (a,y) := ko(y)
49928 a49928  sec
49929         sbc a685,x             ; ko(x) subtrahieren
49932         sta *a87
49934         tya
49935         sbc a686,x
49938         tay
49939         php
49940         lda *a87
49942         plp
49943         rts
49944
49944 ;----------------------------    ( A , Y )  :=  K O ( Y )
49944
49944 a49944  lda a685,y             ; l,ko(y)
49947         pha
49948         lda a686,y             ; h,ko(y)
49951         tay
49952         pla
49953         rts
49954
49954 ;----------------------------    ( A , Y )  :=  ABS ( KO(Y) - KO(X) )
49954
49954 a49954  jsr a49925             ; ---> (a,y) := ko(y) - ko(x)
49957 a49957  bpl a49974             ; nicht negativ: fertig ->
49959 a49959  php                    ; vorzeichen merken
49960         clc                    ; (a,y) negieren
49961         eor #255
49963         adc #1
49965         pha
49966         tya
49967         eor #255
49969         adc #0
49971         tay
49972         pla
49973         plp                    ; vorzeichen erinnern
49974 a49974  rts
```

```
49975 ;------------------------  ( A , Y )  :=  ( A , Y ) * K O ( X ) / 65536
49975
49975 a49975  sty *a142              ; h,multiplikator
49977         sta *a143              ; l,...
49979         lda a685,x             ; l,koordinate
49982         ldy a686,x             ; h,...
49985         php                    ; vorzeichen merken
49986         jsr a49957             ; ---> negieren, falls negativ
49989         sta a685,x
49992         tya
49993         sta a686,x
49996         lda #0                 ; l,ergebnis löschen
49998         sta a751               ; h,...
50001         ldy #16                ; bitzähler
50003 a50003  lsr *a142              ; multiplikator nach rechts shiften
50005         ror *a143
50007         bcc a50024             ; bit = 0: ->
50009         clc
50010         adc a685,x             ; koordinate zum ergebnis addieren
50013         pha
50014         lda a751
50017         adc a686,x
50020         sta a751
50023         pla
50024 a50024  lsr a751               ; ergebnis nach rechts shiften
50027         ror a
50028         dey                    ; bitzähler erniedrigen
50029         bne a50003             ; noch nicht fertig: ->
50031         adc #0                 ; letztes bit addieren (aufrunden)
50033         ldy a751
50036         bcc a50039
50038         iny
50039 a50039  plp                    ; vorzeichen erinnern
50040         jmp a49957             ; ---> negieren, falls negativ
50043
50043 ;-------------------  K O ( 0 ) := K O ( 4 ) ,  K O ( 2 ) := K O ( 6 )
50043
50043 a50043  ldy #0
50045         jsr a50050             ; ---> ko(y) := ko(y+4)
50048         ldy #2
50050 a50050  lda a689,y
50053         sta a685,y
50056         lda a690,y
50059         sta a686,y
50062         rts
50063
50063 ;------------------------------  A D R E S S E  Ü B E R N E H M E N
50063
50063 a50063  jsr a1145              ; ---> chrgot
50066         beq a50080             ; trennzeichen: ->
50068         jsr a38033             ; ---> chkcom
50071         cmp #',                ; noch ein komma?
50073         beq a50080             ; ja: ->
50075         jsr a40417             ; ---> frmevl, getadr
50078         sec                    ; flag für 'eingabe'
50079         rts
50080
50080 a50080  lda #0                 ; ersatzwert 0 wenn keine eingabe
50082         tay
50083         clc                    ; flag wür 'ersatzwert'
50084 a50084  rts
```

```
50085 ;------------------------------  B Y T E  Ü B E R N E H M E N
50085
50085 a50085  ldx #0                  ; ersatzwert 0, wenn keine eingabe
50087 a50087  jsr a1145               ; ---> chrgot
50090         beq a50084              ; trennzeichen: fertig ->
50092         jsr a38033              ; ---> chkcom
50095         cmp #',                 ; noch ein komma?
50097         beq a50084              ; ja: fertig ->
50099         jmp a40324              ; ---> frmevl, getbyt
50102
50102 ;-------------------  F A R B - P A R A M E T E R  Ü B E R N E H M E N
50102
50102 a50102  jsr a51135              ; ---> prüfen ob graphic eingeschaltet
50105 a50105  ldx #1                  ; ersatzwert 1, wenn keine eingabe
50107         jsr a1145               ; ---> chrgot
50110 a50110  beq a50131              ; trennzeichen: ->
50112         cmp #',
50114         beq a50131              ; komma: ->
50116         jsr a40324              ; ---> frmevl, getbyt
50119         cpx #4                  ; gröer als 3?
50121         bcs a50134              ; ja: fehler ->
50123         cpx #2                  ; gröer als 1?
50125         bit *a131               ; graphik-modus
50127         bmi a50131              ; multicolor: ok ->
50129         bcs a50134              ; sonst fehler wenn gröer als 1 ->
50131 a50131  stx *a132               ; als farb-parameter speichern
50133         rts
50134
50134 a50134  jmp a39196              ; ---> 'illegal quantity'
50137
50137 ;-------------------------  K O O R D I N A T E N  Ü B E R N E H M E N
50137
50137 a50137  jsr a1145               ; ---> chrgot
50140         beq a50149              ; trennzeichen: ->
50142         jsr a38033              ; ---> chkcom
50145         cmp #',                 ; noch ein komma?
50147         bne a50167              ; nein: ->
50149 a50149  ldy #0
50151 a50151  lda a685,y              ; ersatzwerte ko(0), wenn keine eingabe
50154         sta a685,x
50157         inx
50158         iny
50159         cpy #4
50161         bne a50151
50163         rts
50164
50164 a50164  jsr a38033              ; ---> chkcom
50167 a50167  stx a752                ; koordinaten-offset
50170         jsr a50319              ; ---> 1. koordinate holen
50173         jsr a1145               ; ---> chrgot
50176         cmp #',                 ; rechtwinklige koordinaten?
50178         beq a50266              ; ja: ->
50180         cmp #';                 ; polarkoordinaten?
50182         beq a50187              ; ja: ->
50184         jmp a38049              ; ---> 'syntax error'
```

```
50187 ;------------------------------  P O L A R K O O R D I N A T E N
50187
50187 a50187  jsr a1139               ; ---> chrget
50190         jsr a40417              ; ---> frmevl, getadr
50193         sta *a128               ; a und y vertauschen
50195         tya
50196         ldy *a128
50198         jsr a48217              ; ---> sinus und cosinus berechnen
50201         ldx a752                ; koordinaten-offset
50204         lda a685,x              ; koordinaten umspeichern
50207         sta a687,x
50210         lda a686,x
50213         sta a688,x
50216         jsr a49875              ; ---> scale ausführen
50219         lda #%00001110          ; sin/cos-flags (0: sin, 1: cos)
50221         sta a753                ; merken
50224         clc
50225         ldx a752                ; koordinatenoffset
50228 a50228  jsr a48304              ; ---> (a,y) := ko(x) * cos/sin
50231         sta a685,x              ; ergebnis speichern
50234         tya
50235         sta a686,x
50238         ldy #a685-a685
50240         lsr a753                ; nächstes flag ins carry
50243         bcc a50247              ; nicht gesetzt:
50245         ldy #a687-a685
50247 a50247  jsr a49908              ; ---> (a,y) := ko(y) +- ko(x) (cc: +)
50250         sta a685,x              ; rechtwinklige koordinaten speichern
50253         tya
50254         sta a686,x
50257         inx
50258         inx
50259         lsr a753
50262         bne a50228              ; nach dem ersten durchlauf ->
50264         clc
50265         rts
50266
50266 ;-------------------  R E C H T W I N K L I G E  K O O R D I N A T E N
50266
50266 a50266  jsr a1139               ; ---> chrget
50269         inc a752                ; koordinaten-offset um 2 erhöhen
50272         inc a752
50275         jsr a50319              ; ---> koordinate holen
50278         ldx a752                ; koordinaten-offset
50281         dex
50282         dex
50283         jsr a49875              ; ---> scale ausführen
50286         ldy #a687-a685
50288         ldx a752
50291         inx
50292         inx
50293 a50293  dex
50294         dex
50295         lsr a753                ; relative koordinate?
50298         bcc a50310              ; nein: ->
50300         jsr a49910              ; ---> (a,y) := ko(y) + ko(x)
50303         sta a685,x              ; neue koordinate speichern
50306         tya
50307         sta a686,x
50310 a50310  ldy #a685-a685
50312         cpx a752
50315         beq a50293              ; nach dem ersten durchgang ->
50317         clc
50318         rts
```

```
50319 ;------------------------------  K O O R D I N A T E   H O L E N
50319
50319 a50319  jsr a1145               ; ---> chrgot
50322         cmp #170                ; '+' token?
50324         beq a50331              ; ja: ->
50326         cmp #171                ; '-' token?
50328         beq a50331              ; ja: ->
50330         clc
50331 a50331  rol a753                ; flag für 'relative koordinate'
50334         jsr a37652              ; ---> frmnum
50337         jsr a40424              ; ---> getadr+4
50340         ldx a752                ; koordinaten-offset
50343         sta a686,x              ; koordinatenwert speichern
50346         tya
50347         sta a685,x
50350         rts
50351
50351 ;------------------------------  K O N S T A N T E N
50351
50351 a50351  .by %11111111 %10101010 %01010101 0     ; konst f multicolor
50355
50355 ;                                 s i n u s - t a b e l l e
50355
50355 a50355  .by 0 0                 ;     0  =  65536 * 0.0000    sin  0 grad
50357         .by 44 113              ; 11377             0.1736       10
50359         .by 87 141              ; 22413             0.3420       20
50361         .by 128 0               ; 32768             0.5000       30
50363         .by 164 143             ; 42127             0.6428       40
50365         .by 196 25              ; 50201             0.7660       50
50367         .by 221 178             ; 56754             0.8660       60
50369         .by 240 144             ; 61584             0.9397       70
50371         .by 252 28              ; 64540             0.9848       80
50373         .by 255 255             ; 65535             1.0000       90
50375
50375 a50375  .by 4 114               ;  1138      interpolationstabelle
50377         .by 4 80                ;  1104
50379         .by 4 11                ;  1035
50381         .by 3 168               ;   936
50383         .by 3 40                ;   808
50385         .by 2 144               ;   656
50387         .by 1 227               ;   483
50389         .by 1 40                ;   296
50391         .by 0 99                ;    99
50393
50393 ;------------------------------  D R A W
50393
50393 a50393  jsr a51135              ; ---> prüfen ob graphic eingeschaltet
50396         jsr a1145               ; ---> chrgot
50399         beq a50419              ; trennzeichen: fertig ->
50401         ldx #1
50403         cmp #164                ; 'to'-token
50405         jsr a50110              ; ---> farbparameter holen, wenn angegeben
50408 a50408  jsr a1145               ; ---> chrgot
50411         cmp #',
50413         beq a50420              ; komma: ->
50415         cmp #164                ; 'to'-token?
50417         beq a50420              ; ja: ->
50419 a50419  rts
```

```
50420 a50420  pha
50421         jsr a1139               ; ---> chrget
50424         ldx #a689-a685
50426         jsr a50167              ; ---> endkoordinaten holen
50429         pla
50430         bpl a50438              ; nicht 'to': ->
50432         jsr a49370              ; ---> strecke zeichnen
50435         jmp a50408              ; ---> befehl weiterbearbeiten
50438
50438 a50438  jsr a50043              ; ---> anfangskoordinate := endkoordinate
50441         jsr a49573              ; ---> punkt setzen
50444         jmp a50408              ; ---> befehl weiterbearbeiten
50447
50447 ;-------------------------------  L O C A T E
50447
50447 a50447  jsr a51135              ; ---> prüfen ob graphic eingeschaltet
50450         ldx #a689-685
50452         jsr a50167              ; ---> koordinaten holen
50455         jmp a50043              ; ---> und als ko(0) speichern
50458
50458 ;-------------------------------  C O L O R
50458
50458 a50458  jsr a40324              ; ---> farb-parameter holen
50461         cpx #5                  ; gröer als 4?
50463         bcs a50532              ; ja: fehler ->
50465         stx *a126               ; farbparameter
50467         jsr a40408              ; ---> farbe holen
50470         dex
50471         cpx #16                 ; farbkode gröer als 16?
50473         bcs a50532              ; ja: fehler ->
50475         stx *a127               ; farbkode
50477         ldx #7                  ; ersatzwert
50479         jsr a50087              ; ---> helligkeit holen
50482         cpx #8                  ; gröer als 7?
50484         bcs a50532              ; ja: fehler ->
50486         txa                     ; helligkeit
50487         asl a                   ; ins obere halbbyte schieben
50488         asl a
50489         asl a
50490         asl a
50491         ora *a127               ; farbkode einodern
50493         ldx *a126               ; farbparameter
50495         cpx #1
50497         beq a50506              ; = 1: ->
50499         bcs a50513              ; gröer als 1: ->
50501         sta a65301              ; = 0: bildhintergrund
50504         bne a50531              ; immer ->
50506
50506 a50506  sta *a134               ; = 1: farbe für graphik-punkte
50508         sta a1339               ; und für normale print-ausgabe
50511         beq a50531              ; immer ->
50513
50513 a50513  cpx #3
50515         beq a50523              ; = 3: ->
50517         bcs a50528              ; = 4: ->
50519         sta *a133               ; = 2: 1. farbparameter für multicolor
50521         bne a50531              ; immer ->
50523
50523 a50523  sta a65302              ; = 3: 2. farbparameter für multicolor
50526         beq a50531              ; immer ->
```

```
50528 a50528  sta a65305              ; = 4: bildrahmen-farbe
50531 a50531  rts
50532
50532 a50532  jmp a39196              ; ---> 'illegal quantity'
50535
50535 ;-------------------------------  S C N C L R
50535
50535 a50535  lda *a131               ; graphik-modus
50537         bne a50544              ; eingeschaltet: ->
50539 a50539  lda #147                ; clr
50541         jmp a65490              ; ---> bsout
50544
50544 a50544  and #64                 ; mit text?
50546         beq a50559              ; nein: ->
50548         jsr a50539              ; ---> bildschirm löschen
50551         ldx #20                 ; zeile
50553         ldy #0                  ; spalte
50555         clc
50556         jsr a65520              ; ---> plot: kursor setzen
50559 a50559  lda #0
50561         ldy #h,a8192
50563         ldx #32                 ; anzahl der pages
50565         jsr a50599              ; ---> bitspeicher löschen
50568         jsr a49720              ; ---> (a) := farbe 1/0 (zeichen/hintergr)
50571         ldy #h,a7168
50573         ldx #4                  ; anzahl der pages
50575         jsr a50599              ; ---> farbspeicher löschen
50578         jsr a49742              ; ---> (a) := helligkeit 1/0
50581         ldy #h,a6144
50583         ldx #4
50585         jsr a50599              ; ---> helligkeitsspeicher löschen
50588         lda #0
50590         ldx #3
50592 a50592  sta a685,x              ; koordinaten (x,y) := (0,0)
50595         dex
50596         bpl a50592
50598         rts
50599
50599 a50599  sty *a141               ; h,zeiger in speicher
50601         ldy #0
50603         sty *a140               ; l,...
50605 a50605  sta (a140),y            ; löschroutine
50607         dey
50608         bne a50605
50610         inc *a141
50612         dex
50613         bne a50605
50615         rts
50616
50616 ;-------------------------------  S C A L E
50616
50616 a50616  jsr a40324              ; ---> scale-byte holen
50619         cpx #2
50621         bcs a50532              ; gröer als 1: fehler ->
50623         stx a742                ; scale-flag
50626         rts
```

```
50627 ;------------------------------  G R A P H I C
50627
50627 a50627  cmp #156                ; folgt 'clr'-token?
50629         bne a50641              ; nein: ->
50631         jsr a51000              ; ---> normale speicheraufteilung
50634         jsr a1139               ; ---> chrget
50637         lda #0
50639         beq a50651              ; immer ->
50641
50641 a50641  jsr a40324              ; ---> graphik-modus holen
50644         cpx #5                  ; gröer als 4?
50646         bcs a50669              ; ja: fehler ->
50648         lda a50743,x            ; graphic-modus
50651 a50651  cmp *a131               ; unverändert?
50653         beq a50730              ; ja: ->
50655         sta *a131               ; neuen graphic-modus speichern
50657         tax                     ; = 0?
50658         bne a50672              ; nein: ->
50660         jsr a51145              ; ---> graphic-modus abschalten
50663         lda #40                 ; spalten
50665         ldx #25                 ; zeilen
50667         bne a50726              ; immer ->
50669
50669 a50669  jmp a39196              ; ---> 'illegal quantity'
50672
50672 a50672  jsr a50748              ; ---> graphic-speicher einrichten
50675         lda a65286
50678         ora #%00100000          ; bitspeicher-freigabe-bit
50680         sta a65286              ; setzen
50683         lda a65287
50686         and #%11101111          ; multicolor-freigabe-bit rücksetzen
50688         bit *a131               ; graphic-modus
50690         bpl a50694              ; nicht multicolor: ->
50692         ora #%00010000          ; multicolor-freigabe-bit setzen
50694 a50694  sta a65287
50697         lda a65298
50700         and #%11000011
50702         ora #%00001000
50704         sta a65298
50707         lda a65300
50710         and #%00000011
50712         ora #h,a6144            ; bit-speicherbereich einschalten
50714         sta a65300
50717         lda #40                 ; spaltenzahl
50719         ldx #25                 ; zeilenzahl
50721         bit *a131               ; graphic-modus
50723         bpl a50726              ; nicht multicolor: ->
50725         lsr a                   ; sonst spaltenzahl halbieren
50726 a50726  sta *a135               ; graphic-spalten für scale
50728         stx *a136               ; graphic-zeilen
50730 a50730  jsr a50085              ; ---> scnclr-flag holen
50733         txa
50734         lsr a
50735         bne a50669              ; gröer als 1: fehler ->
50737         bcc a50742              ; = 0: fertig ->
50739         jmp a50535              ; ---> scnclr
50742
50742 a50742  rts
```

```
50743 a50743  .by 0                   ;   0 normal-modus
50744         .by %00100000           ;  32 hochauflösende graphik
50745         .by %01100000           ;  96 hochauflösende graphik mit text
50746         .by %10100000           ; 160 multicolor-graphik
50747         .by %11100000           ; 224 multicolor-graphik mit text
50748
50748 a50748  lda *a117                ; graphic-speicher-flag
50750         beq a50753              ; = 0: ->
50752         rts
50753
50753 a50753  lda *a56                 ; basic-obergrenze
50755         cmp #h,16384
50757         bcs a50811              ; nicht unter 16384: ->
50759         jsr a43348              ; ---> garbage collect
50762         jsr a51307              ; ---> (x,y) := länge des stringbereichs
50765         txa
50766         clc
50767         adc *a49                ; + variablenende
50769         tya
50770         adc *a50
50772         cmp #h,a6144            ; > 6 k?
50774         bcs a50808              ; ja: fehler ->
50776         dec *a117
50778         lda #0
50780         sta *a34
50782         lda #h,a6144
50784         sta *a35
50786         jsr a51184              ; ---> strings verschieben
50789         lda *a34                ; neue stringbereich-untergrenze
50791         sta *a51                ; in string-anfangszeiger
50793         lda *a35
50795         sta *a52
50797         lda #0
50799         sta *a55                ; neue basic-obergrenze
50801         lda #h,a6144            ; = 6144
50803         sta *a56
50805         jmp a51237              ; ---> r-zeiger korrigieren
50808
50808 a50808  jmp a34433              ; ---> 'out of memory'
50811
50811 a50811  jsr a43348              ; ---> garbage collect
50814         ldy *a49                ; l,variablenende
50816         sty *a95                ; l,transportzeiger
50818         lda *a50                ; h,variablenende
50820         clc
50821         adc #h,12288            ; + 12 k
50823         bcs a50808              ; überlauf: fehler ->
50825         sta *a96                ; h,transportzeiger
50827         cmp *a52                ; kleiner als string-untergrenze?
50829         bcc a50837
50831         bne a50808
50833         cpy *a51
50835         bcs a50808              ; nein: fehler ->
50837 a50837  dec *a117               ; flag für graphic-speicher setzen
50839         lda #l,12288
50841         sta *a78
50843         lda #h,12288
50845         sta *a79
50847         jsr a51319              ; ---> r-zeiger der strings korrigieren
```

```
50850          lda *a95              ; zielbereich-ende
50852          sta *a34              ; nach (a34,a35)
50854          lda *a96
50856          sta *a35
50858          ldx *a49              ; variablen-ende (= quellbereichs-ende)
50860          stx *a36              ; nach (a36,a37)
50862          lda *a50
50864          sta *a37
50866          sec
50867          sbc #h,a4096          ; programmanfang subtrahieren
50869          tay                   ; ergibt länge des blocks
50870          jsr a51192            ; ---> block nach oben schieben
50873          clc
50874          lda *a50              ; variablenendezeiger
50876          adc #h,12288          ; um 12 k
50878          sta *a50              ; hinaufsetzen
50880          lda *a48              ; array-anfangszeiger
50882          adc #h,12288          ; um 12 k
50884          sta *a48              ; hinaufsetzen
50886          lda *a46              ; variablen-anfangszeiger
50888          adc #h,12288          ; um 12 k
50890          sta *a46              ; hinaufsetzen
50892          lda *a44              ; programm-anfangszeiger
50894          adc #h,12288          ; um 12 k
50896          sta *a44              ; hinaufsetzen
50898          lda *a66              ; data-zeiger
50900          adc #h,12288          ; um 12 k
50902          sta *a66              ; hinaufsetzen
50904 a50904   jsr a34840            ; ---> linkadressen erzeugen
50907          jsr a34891            ; ---> variablenanfang setzen
50910          bit *a129             ; direktmodus?
50912          bpl a50959            ; ja: fertig ->
50914          ldx #h,12288
50916          bit *a117
50918          bmi a50922
50920          ldx #208              ; = -h,12288
50922 a50922   txa
50923          clc
50924          adc *a60              ; programmzeiger
50926          sta *a60
50928          txa
50929          clc
50930          adc a604              ; adresse für trap und cont
50933          sta a604
50936          txa
50937          clc
50938          adc a1270             ; adresse für trap
50941          sta a1270             ; um 12 k erhöhen
50944          jsr a42848            ; ---> (a61,a62) := (a124,a125)
50947 a50947   lda *a61              ; basic-stack-zeiger
50949          cmp #l,a1968
50951          bne a50960
50953          lda *a62
50955          cmp #h,a1968
50957          bne a50960            ; basic-stack nicht leer: ->
50959 a50959   rts
```

```
50960 a50960  ldy #0                ; korrektur des basic-stacks
50962         lda (a61),y
50964         cmp #129              ; 'for'-token?
50966         bne a50982            ; nein: ->
50968         ldy #2                ; h,variablenzeiger
50970         jsr a51117            ; ---> adresse korrigieren
50973         ldy #16               ; h,startadresse der schleife
50975         jsr a51117            ; ---> adresse korrigieren
50978         lda #18               ; länge des for-datensatzes im stack
50980         bne a50989
50982
50982 a50982  ldy #4                ; h,rücksprungadresse
50984         jsr a51117            ; ---> adresse korrigieren
50987         lda #5                ; länge des gosub-datensatzes im stack
50989 a50989  clc
50990         adc *a61              ; stack-zeiger auf nächsten
50992         sta *a61              ; eintrag setzen
50994         bcc a50947
50996         inc *a62
50998         bne a50947            ; und weitermachen ->
51000
51000 ;-------------------  S P E I C H E R A U F T E I L U N G   N O R M A L
51000
51000 a51000  lda *a117             ; speicheraufteilung bereits normal?
51002         bne a51005            ; nein: ->
51004         rts
51005
51005 a51005  ldy #0
51007         sty *a117             ; speicher-flag rücksetzen
51009         lda *a56              ; basic-obergrenze
51011         bmi a51049            ; nicht unter 32768: ->
51013         jsr a43348            ; ---> garbage collect
51016         jsr a51307            ; ---> (x,y) := länge des stringbereichs
51019         lda a1331             ; memtop
51022         sta *a34              ; nach (a34,a35)
51024         lda a1332
51027         sta *a35
51029         jsr a51184            ; ---> programm und variablen verschieben
51032         ldx #1
51034 a51034  lda a1331,x           ; memtop
51037         sta *a55,x            ; =: basic-obergrenze-zeiger
51039         lda *a34,x            ; untergrenze des verschobenen blocks
51041         sta *a51,x            ; =: stringbereich-anfang
51043         dex
51044         bpl a51034
51046         jmp a51237            ; ---> r-zeiger korrigieren
51049
51049 a51049  ldy #0
51051         sty *a117             ; speicher-flag rücksetzen
51053         sty *a34              ; l,zielzeiger
51055         sty *a36              ; l,quellzeiger
51057         lda #h,4096
51059         sta *a35              ; h,zielzeiger
51061         lda #h,16384
51063         sta *a37              ; h,quellzeiger
51065 a51065  jsr a1211             ; ---> lda (a36),y
51068         sta (a34),y           ; programm und variablen herunterkopieren
51070         iny
51071         bne a51065
```

```
51073          inc *a35
51075          inc *a37
51077          lda *a50              ; variablenende erreicht?
51079          cmp *a37
51081          bcs a51065            ; nein: ->
51083          lda *a50              ; variablen-endezeiger
51085          sec
51086          sbc #h,12288          ; um 12 k erniedrigen
51088          sta *a50
51090          lda *a44              ; programm-anfangszeiger
51092          sbc #h,12288          ; um 12 k erniedrigen
51094          sta *a44
51096          lda *a46              ; variablen-anfangszeiger
51098          sbc #h,12288          ; um 12 k erniedrigen
51100          sta *a46
51102          lda *a48              ; array-anfangszeiger
51104          sbc #h,12288          ; um 12 k erniedrigen
51106          sta *a48
51108          lda *a66              ; data-zeiger
51110          sbc #h,12288          ; um 12 k erniedrigen
51112          sta *a66
51114          jmp a50904            ; ---> linkadressen, basic stack usw.
51117
51117 a51117   lda (a61),y           ; h,adresse aus basic-stack
51119          bit *a117             ; normale speicheraufteilung?
51121          bne a51129            ; nein: ->
51123          sec
51124          sbc #h,12288          ; adresse um 12 k erniedrigen
51126          sta (a61),y
51128          rts
51129
51129 a51129   clc
51130          adc #h,12288          ; adresse um 12 k erhöhen
51132          sta (a61),y
51134          rts
51135
51135 ;--------------------------------  G R A P H I C   P R Ü F E N
51135
51135 a51135   lda *a117             ; graphik-speicher eingerichtet?
51137          beq a51140            ; nein: ->
51139          rts
51140
51140 a51140   ldx #35               ; 'no graphics area'
51142          jmp a34435            ; ---> fehlerausgang
51145
51145 ;--------------------------------  G R A P H I C   A B S C H A L T E N
51145
51145 a51145   lda a65286
51148          and #%11011111        ; bitspeicher-freigabebit löschen
51150          sta a65286
51153          lda a65287
51156          and #%11101111        ; multicolor-freigabebit löschen
51158          sta a65287
51161          lda a65300
51164          and #%00000111
51166          ora #h,a2048          ; normalen bildschirmspeicherbereich
51168          sta a65300            ; einschalten
51171          lda a65298
51174          ora #%00000100
51176          sta a65298
51179          lda #0
51181          sta *a131             ; graphic-modus
51183          rts
```

```
51184 ;---------  S P E I C H E R A U F T E I L U N G  F Ü R  G R A P H I C
51184
51184 a51184  lda *a55              ; basic-obergrenze
51186         sta *a36              ; nach (a36,a37) als quellzeiger
51188         lda *a56
51190         sta *a37
51192 a51192  txa                   ; verschiebelänge
51193         eor #255              ; negieren
51195         sta *a78              ; und nach (a78,a79)
51197         tya
51198         eor #255
51200         sta *a79
51202         ldy #0
51204 a51204  inc *a78
51206         bne a51212
51208         inc *a79
51210         beq a51236            ; verschiebelänge 0: fertig ->
51212 a51212  lda *a34
51214         bne a51218
51216         dec *a35
51218 a51218  dec *a34
51220         lda *a36
51222         bne a51226
51224         dec *a37
51226 a51226  dec *a36
51228         jsr a1211             ; ---> lda (a36),y: zchn aus quellbereich
51231         sta (a34),y           ; in zielbereich kopieren
51233         jmp a51204            ; --->
51236
51236 a51236  rts
51237
51237 a51237  lda *a55              ; basic-obergrenze
51239         ldy *a56
51241         sta *a34              ; nach (a34,a35)
51243         sty *a35
51245 a51245  sec
51246         lda *a51              ; stringbereich-untergrenze
51248         sbc *a34
51250         lda *a52
51252         sbc *a35
51254         bcs a51236            ; erreicht: fertig ->
51256         sec
51257         lda *a34              ; (a34,a35) auf r-zeiger setzen
51259         sbc #2
51261         sta *a34
51263         bcs a51267
51265         dec *a35
51267 a51267  ldy #1
51269 a51269  jsr a1200             ; ---> lda (a34),y: r-zeiger
51272         sta a36,y             ; nach a36,a37
51275         dey
51276         bpl a51269
51278         iny
51279         jsr a1211             ; ---> lda (a36),y
51282         sta *a128             ; stringlänge aus deskriptor
51284         lda *a34              ; r-zeiger-adresse
51286         sec
51287         sbc *a128             ; - stringlänge
51289         sta *a34              ; = string-anfangsadresse
51291         bcs a51295
51293         dec *a35
```

```
51295 a51295  ldy #2
51297 a51297  lda a34-1,y             ; string-anfangsadresse
51300         sta (a36),y             ; in deskriptor bringen
51302         dey
51303         bne a51297
51305         beq a51245
51307
51307 ;------   ( X , Y )  :=  L Ä N G E   D E S   S T R I N G B E R E I C H S
51307
51307 a51307  sec
51308         lda *a55                ; l,basic-obergrenze
51310         sbc *a51                ; - l,stringbereich-anfang
51312         tax                     ; = l,länge des stringbereichs
51313         lda *a56                ; h,basic-obergrenze
51315         sbc *a52                ; - h,stringbereich-anfang
51317         tay                     ; = h,länge des stringbereichs
51318         rts
51319
51319 ;-----------------------------  R - Z E I G E R   K O R R I G I E R E N
51319
51319 a51319  lda *a55                ; basic-obergrenze
51321         sta *a34                ; nach (a34,a35)
51323         lda *a56
51325         sta *a35
51327 a51327  sec
51328         lda *a51                ; string-untergrenze
51330         sbc *a34                ; - (a34,a35)
51332         lda *a52
51334         sbc *a35
51336         bcs a51387              ; = 0: fertig ->
51338         sec
51339         lda *a34                ; a34,a35 auf r-zeiger setzen
51341         sbc #2
51343         sta *a34
51345         bcs a51349
51347         dec *a35
51349 a51349  clc
51350         ldy #0
51352 a51352  jsr a1200               ; ---> lda (a34),y: r-zeiger
51355         sta a36,y               ; nach (a36,a37)
51358         adc a78,y               ; um (a78,a79) erhöhen
51361         sta (a34),y             ; und wieder abspeichern
51363         iny
51364         cpy #1                  ; ??? damit wird nur das 1-byte
51366         bne a51352              ; ??? erfat!!!
51368         dey
51369         jsr a1211               ; ---> lda (a36),y: länge des strings
51372         sta *a128
51374         lda *a34                ; adresse des deskriptors
51376         sec
51377         sbc *a128               ; - länge des strings
51379         sta *a34                ; = anfangsadresse des strings
51381         bcs a51327
51383         dec *a35
51385         bcc a51327              ; immer ->
51387
51387 a51387  rts
```

```
51388 ;------------------------------  D I R E C T O R Y
51388
51388 a51388  jsr a51999              ; ---> kommando-analyse
51391         and #230                ; syntax ok?
51393         bne a51518              ; nein: fehler ->
51395         ldy #0
51397         jsr a51775              ; ---> kommando erzeugen
51400         lda #0                  ; la
51402         ldx a631                ; fa
51405         ldy #96                 ; sa
51407         jsr a65466              ; ---> setlfs
51410         sec
51411         jsr a65472              ; ---> open: kommando senden
51414         bcc a51425              ; ok: ->
51416         pha
51417         jsr a51509              ; ---> clrch, close
51420         pla
51421         tax
51422         jmp a34435              ; ---> fehlerausgang
51425
51425 a51425  ldx #0
51427         jsr a65478              ; ---> chkin
51430         ldy #3                  ; programm- und zeilenlink überlesen
51432 a51432  sty a748                ; zeichenzähler
51435 a51435  jsr a65487              ; ---> basin
51438         sta a749                ; 1. zeichen merken
51441         jsr a65463              ; ---> readst: status gesetzt?
51444         bne a51509              ; ja: abbruch ->
51446         jsr a65487              ; ---> basin
51449         sta a750                ; 2. zeichen merken
51452         jsr a65463              ; ---> readst
51455         bne a51509              ; clrch, close
51457         dec a748                ; schon 6 zeichen gelesen?
51460         bne a51435              ; nein: weiterlesen ->
51462         ldx a749                ; l,blockzahl
51465         lda a750                ; h,...
51468         jsr a42079              ; ---> (a,x) ausgeben
51471         lda #'
51473         jsr a65490              ; ---> bsout
51476 a51476  jsr a65487              ; ---> basin
51479         pha
51480         jsr a65463              ; ---> readst: status gesetzt?
51483         bne a51508              ; ja: abbruch ->
51485         pla                     ; gelesenes zeichen
51486         beq a51494              ; zeilenende: ->
51488         jsr a65490              ; ---> bsout
51491         jmp a51476              ; ---> weiterlesen
51494
51494 a51494  lda #13                 ; cr-kode
51496         jsr a65490              ; ---> bsout
51499         jsr a65505              ; ---> stoptaste gedrückt?
51502         beq a51509              ; ja: abbruch ->
51504         ldy #2                  ; zeilenlink überlesen
51506         bne a51432              ; immer ->
51508
51508 a51508  pla
51509 a51509  jsr a65484              ; ---> clrch
51512         lda #0
51514         clc
51515         jmp a65475              ; ---> close
51518
51518 a51518  jmp a38049              ; ---> 'syntax error'
```

```
51521 ;------------------------------  D S A V E
51521
51521 a51521  lda #%01100110          ; filename, fa, lw, replace
51523         jsr a52001              ; ---> kommando-analyse
51526         jsr a52405              ; ---> fehler wenn kein filename
51529         ldy #4                  ; offset in tabelle
51531         jsr a51775              ; ---> kommando erzeugen
51534         jmp a42977              ; ---> save-routine
51537
51537 ;------------------------------  D L O A D
51537
51537 a51537  lda #%11100110          ; filename, fa, lw
51539         jsr a52001              ; ---> kommando-analyse
51542 .       jsr a52405              ; ---> fehler wenn kein filename
51545         lda #0
51547         sta a632                ; sekundäradresse
51550         sta *a10                ; verify-flag
51552         ldy #5                  ; offset in tabelle
51554         jsr a51775              ; ---> kommando erzeugen
51557         jmp a43002              ; ---> load-routine
51560
51560 ;------------------------------  H E A D E R
51560
51560 a51560  jsr a51999              ; ---> kommando-analyse
51563         jsr a52399              ; ---> syntax-prüfung
51566         and #%00010001          ; diskname und laufwerknummer
51568         cmp #%00010001          ; beide angegeben?
51570         beq a51575              ; ja: ->
51572 a51572  jmp a38049              ; ---> 'syntax error'
51575
51575 a51575  jsr a65511              ; ---> clall
51578         jsr a52523              ; ---> 'are you sure?'
51581         bne a51606              ; nein: ->
51583         ldy #9                  ; offset in tabelle
51585         jsr a51775              ; ---> kommando erzeugen und senden
51588         jsr a52431              ; ---> disk status übernehmen
51591         bit *a129               ; direktmodus?
51593         bmi a51606              ; nein: ->
51595         ldy #0
51597         lda #a122
51599         jsr a1172               ; ---> lda (a),y: erste stelle von ds
51602         cmp #'2                 ; grö^er als 2?
51604         bcs a51607              ; ja: formatierungsfehler ->
51606 a51606  rts
51607
51607 a51607  ldx #36                 ; 'bad disk'
51609         jmp a34435              ; ---> fehlerausgang
51612
51612 ;------------------------------  S C R A T C H
51612
51612 a51612  jsr a51999              ; ---> kommando-analyse
51615         jsr a52399              ; ---> syntax-prüfung
51618         jsr a52523              ; ---> 'are you sure?'
51621         bne a51606              ; nein: ->
51623         ldy #15                 ; offset in tabelle
51625         jsr a51775              ; ---> kommando erzeugen
51628         jsr a52431              ; ---> disk status übernehmen
51631         bit *a129               ; direktmodus?
51633         bmi a51606              ; nein: ->
51635         lda #13                 ; cr-kode
51637         jsr a65490              ; ---> bsout
51640         ldy #0
```

```
51642 a51642  lda #a122              ; adresse von ds$
51644         jsr a1172              ; ---> lda (a),y: zeichen aus meldung
51647         beq a51655             ; endezeichen: ->
51649         jsr a65490             ; ---> bsout
51652         iny
51653         bne a51642             ; immer
51655
51655 a51655  lda #13                ; cr-kode
51657         jmp a65490             ; ---> bsout
51660
51660 ;-------------------------------  C O L L E C T
51660
51660 a51660  jsr a51999             ; ---> kommando-analyse
51663         and #%11100111         ; nur gerätenummer und laufwerk?
51665         bne a51572             ; nein: fehler ->
51667         jsr a65511             ; ---> clall
51670         ldy #20                ; offset in tabelle
51672         bne a51775             ; kommando erzeugen und senden ->
51674
51674 ;-------------------------------  C O P Y
51674
51674 a51674  jsr a51999             ; ---> kommando-analyse
51677         and #%00110000         ; genau zwei laufwerknummern
51679         cmp #%00110000         ; angegeben?
51681         bne a51689             ; nein: ->
51683         lda *a130              ; syntax-flag
51685         and #%11000111         ; filenamen angegeben?
51687         beq a51696             ; nein: ->
51689 a51689  lda *a130
51691         jsr a52416             ; ---> syntax-prüfung
51694         lda *a130
51696 a51696  ldy #23                ; offset in tabelle
51698         bne a51775             ; kommando erzeugen und senden ->
51700
51700 ;-------------------------------  R E N A M E
51700
51700 a51700  lda #%11100100         ; 2 filenamen, fa, lw
51702         jsr a52001             ; ---> kommando-analyse
51705         jsr a52422             ; ---> auf 2 filenamen prüfen
51708         ldy #30                ; offset in tabelle
51710         bne a51775             ; kommando erzeugen und senden ->
51712
51712 ;-------------------------------  B A C K U P
51712
51712 a51712  lda #%11000111         ; fa, 2 laufwerke
51714         jsr a52001             ; ---> kommando-analyse
51717         and #%00110000         ; 2 laufwerke angegeben?
51719         cmp #%00110000
51721         beq a51726             ; ja: ->
51723         jmp a38049             ; ---> 'syntax error'
51726
51726 a51726  jsr a65511             ; ---> clall
51729         ldy #37                ; offset in tabelle
51731         jmp a51775             ; ---> kommando erzeugen und senden
51734
51734 ;-------------------------------  K O M M A N D O   S E N D E N
51734
51734 a51734  pha                    ; endekode in tabelle
51735         lda a605               ; länge des filenamens
51738         ldx #l,a636            ; l,adresse des filenamens
51740         ldy #h,a636            ; h,...
51742         jsr a65469             ; ---> setnam
```

```
51745          lda a630                ; la: log adresse
51748          ldx a631                ; fa: geräteadresse
51751          ldy a632                ; sa: sekundäradresse
51754          jsr a65466              ; ---> setlfs
51757          pla                     ; endekode
51758          beq a51774              ; = 0: fertig ->
51760          ldx a605                ; länge des kommandos
51763          sec
51764          jsr a65472              ; ---> open, kommando senden
51767          lda a630                ; la
51770          sec
51771          jmp a65475              ; ---> close
51774
51774 a51774   rts
51775
51775 ;------------------------------  K O M M A N D O   E R Z E U G E N
51775
51775 a51775   jsr a52567              ; ---> disk-status löschen
51778          ldx #0
51780          stx a605                ; zähler für kommandolänge
51783 a51783   lda a51957,y
51786          beq a51734              ; endekode 0: ->
51788          cmp #128
51790          beq a51734              ; endekode 128: ->
51792          tax                     ; kode aus tabelle
51793          dex
51794          beq a51826              ; 1: laufwerknummer 1, falls vorhanden
51796          dex
51797          beq a51832              ; 2: laufwerknummer 1
51799          dex
51800          beq a51838              ; 3: laufwerknummer 2
51802          dex
51803          beq a51845              ; 4: replace-kode
51805          dex
51806          beq a51853              ; 5: filenamen 1, falls vorhanden
51808          dex
51809          beq a51863              ; 6: filenamen 1
51811          dex
51812          beq a51877              ; 7: filenamen 2, falls vorhanden
51814          dex
51815          beq a51887              ; 8: filenamen 2
51817          dex
51818          beq a51922              ; 9: id, falls vorhanden
51820
51820 a51820   jsr a51947              ; ---> in kommandostring einbauen
51823 a51823   iny
51824          bne a51783              ; immer ->
51826
51826 a51826   lda *a130               ; syntax-flag
51828          and #16                 ; laufwerknummer angegeben?
51830          beq a51823              ; nein: ->
51832 a51832   lda a623                ; laufwerknummer 1
51835          jmp a51841              ; --->
51838
51838 a51838   lda a627
51841 a51841   ora #'0                 ; ascii-kode erzeugen
51843          bne a51820              ; immer ->
51845
51845 a51845   lda #'ß                 ; zeichen für 'replace'
51847          bit *a130               ; syntax-flag
51849          bmi a51820              ; replace gefordert: ->
51851          bpl a51823              ; immer ->
```

```
51853 a51853  lda a622              ; länge des filenamens 1
51856         beq a51823            ; kein filename: ->
51858         lda #':
51860         jsr a51947            ; ---> in kommandostring einbauen
51863 a51863  tya
51864         pha
51865         lda a624              ; l,filenamensadresse 1
51868         ldy a625              ; h,...
51871         ldx a622              ; länge des filenamens 1
51874         jmp a51898            ; ---> filenamen in kommando einbauen
51877
51877 a51877  lda a626              ; länge des filenamens 2
51880         beq a51823            ; kein filename: ->
51882         lda #':
51884         jsr a51947            ; ---> in kommandostring einbauen
51887 a51887  tya
51888         pha
51889         lda a628              ; l,adresse des filenamens 2
51892         ldy a629              ; h,...
51895         ldx a626              ; länge des filenamens 2
51898 a51898  sta *a34              ; l,adresse
51900         sty *a35              ; h,...
51902         stx *a128             ; länge
51904         ldy #0
51906 a51906  jsr a1200             ; ---> lda (a34),y: zeichen aus namen
51909         jsr a51947            ; ---> in kommandostring einbauen
51912         iny
51913         cpy *a128             ; ende des namens erreicht?
51915         bne a51906            ; nein: ->
51917         pla
51918         tay
51919         jmp a51823            ; --->
51922
51922 a51922  lda a633              ; id byte 1
51925         beq a51823            ; nicht vorhanden: ->
51927         lda #',
51929         jsr a51947            ; ---> in kommandostring einbauen
51932         lda a633              ; id byte 1
51935         jsr a51947            ; ---> in kommandostring einbauen
51938         lda a634              ; id byte 2
51941         jsr a51947            ; ---> in kommandostring einbauen
51944         jmp a51823            ; --->
51947
51947 a51947  ldx a605              ; zeiger in kommandostring
51950         sta a636,x            ; zeichen in puffer
51953         inc a605              ; zeiger erhöhen
51956         rts
51957
51957 ;------------------------------  K O M M A N D O T A B E L L E
51957
51957 a51957  .by '$' 7 5 0         ;         directory
51961         .by 4 2 ':' 6 0       ;         dsave, dload, dopen
51966         .by 'n' 2 ':' 6 9 128 ;         header
51972         .by 's' 2 ':' 6 128   ;         scratch
51977         .by 'v' 2 128         ;         collect
51980         .by 'c' 3 7 '=' 2 5 128       ; copy
51987         .by 'r' 2 ':' 8 '=' 6 128     ; rename
51994         .by 'd' 3 '=' 2 128   ;         backup
```

```
51999 ; bedeutung der kodes:    1      laufwerknummer 1 optional
51999 ;                         2      laufwerknummer 1
51999 ;                         3      laufwerknummer 2
51999 ;                         4      replace optional
51999 ;                         5      filename 1 optional
51999 ;                         6      filename 1
51999 ;                         7      filename 2 optional
51999 ;                         8      filename 2
51999 ;                         9      id optional
51999 ;                       128      kommando an floppy senden
51999
51999 ;------------------------------  D B - K O M M A N D O - A N A L Y S E
51999
51999 ; syntax-flag  a130     bit 0:   filename 1
51999 ;                          1:   filename 2
51999 ;                          2:   la (log adresse)
51999 ;                          3:   fa (geräteadresse)
51999 ;                          4:   laufwerk 1
51999 ;                          5:   laufwerk 2
51999 ;                          6:
51999 ;                          7:   replace
51999
51999 a51999  lda #0
52001 a52001  pha                     ; syntax-maske
52002         lda #0
52004         sta *a130               ; disk basic syntax flag
52006         ldx #30
52008 a52008  sta a606,x              ; disk basic arbeitsbereich löschen
52011         dex
52012         bne a52008
52014         ldx #8
52016         stx a631                ; fa: geräteadresse
52019         ldx #111
52021         stx a632                ; sa: sekundäradresse
52024         ldx #0
52026         stx a630                ; la: log adresse
52029         jsr a1145               ; ---> chrgot
52032         bne a52041              ; kein trennzeichen: ->
52034 a52034  pla                     ; syntax-maske
52035         jsr a52394              ; ---> syntax prüfen
52038         lda *a130               ; syntax-flag
52040         rts
52041
52041 a52041  cmp #'d                 ; drive: laufwerknummer
52043         beq a52074
52045         cmp #145                ; 'on'-token
52047         beq a52127
52049         cmp #'u                 ; unit: gerätenummer
52051         beq a52068
52053         cmp #'i                 ; id: identifikationsmerkmal
52055         beq a52099
52057         cmp #'"                 ; text folgt: ->
52059         beq a52133
52061         cmp #'(                 ; variable folgt: ->
52063         beq a52133
52065 a52065  jmp a38049              ; ---> 'syntax error'
52068
52068 a52068  jsr a52312              ; ---> gerätenummer auswerten
52071         jmp a52173              ; ---> syntax-flag setzen
```

```
52074 a52074  lda #16                 ; laufwerk-bit 1
52076         jsr a52394              ; ---> fehler, wenn schon gesetzt
52079         jsr a52375              ; ---> frmevl, getbyt
52082         cpx #2                  ; grö^er als 1?
52084         bcs a52096              ; ja: fehler ->
52086         stx a623                ; laufwerk 1
52089         stx a627                ; laufwerk 2
52092         lda #16                 ; laufwerk-bit 1
52094         bne a52173              ; setzen ->
52096
52096 a52096  jmp a52297              ; ---> 'illegal quantity'
52099
52099 a52099  lda a635                ; id-flag gesetzt?
52102         bne a52065              ; ja: fehler ->
52104         jsr a1139               ; ---> chrget
52107         sta a633                ; id byte 1
52110         jsr a1139               ; ---> chrget
52113         sta a634                ; id byte 2
52116         lda #255
52118         sta a635                ; id-flag setzen
52121         jsr a1139               ; ---> chrget
52124         jmp a52177              ; --->
52127
52127 a52127  jsr a52305              ; ---> 'u' auswerten
52130         jmp a52173              ; ---> syntax-flag setzen
52133
52133 a52133  lda #1                  ; filename-bit 1
52135         jsr a52329              ; ---> filenamen übernehmen
52138         sta a622                ; länge des filenamens
52141         sta a605
52144         lda #l,a606             ; adresse des filenamens im puffer
52146         sta a624                ; l,adresse
52149         sta *a36
52151         lda #h,a606
52153         sta a625                ; h,...
52156         sta *a37
52158         ldy #0
52160 a52160  jsr a1200               ; ---> lda (a34),y: filenamen
52163         sta (a36),y             ; in puffer kopieren
52165         iny
52166         cpy a605                ; name zu ende?
52169         bcc a52160              ; nein: ->
52171         lda #1                  ; filename-bit 1
52173 a52173  ora *a130               ; in syntax-flag odern
52175         sta *a130
52177 a52177  jsr a1145               ; ---> chrgot
52180         bne a52185              ; kein trennzeichen: ->
52182 a52182  jmp a52034              ; ---> abschlu^
52185
52185 a52185  cmp #',
52187         bne a52195
52189         jsr a1139               ; ---> chrget
52192         jmp a52041              ; ---> analyse fortsetzen
52195
52195 a52195  cmp #145                ; 'on'-token
52197         beq a52127
52199         cmp #164                ; 'to'-token?
52201         bne a52294              ; nein: fehler ->
```

```
52203 a52203  jsr a1139             ; ---> chrget
52206         cmp #'d               ; drive: laufwerknummer
52208         beq a52226
52210         cmp #145              ; 'on'-token
52212         beq a52245
52214         cmp #'u               ; unit: gerätenummer
52216         beq a52251
52218         cmp #'"               ; text folgt: ->
52220         beq a52257
52222         cmp #'(               ; variable folgt: ->
52224         beq a52257
52226 a52226  lda #32               ; laufwerk-bit 2
52228         jsr a52394            ; ---> fehler, wenn schon gesetzt
52231         jsr a52375            ; ---> frmevl, getbyt
52234         cpx #2                ; grö^er als 1?
52236         bcs a52297            ; ja: fehler ->
52238         stx a627
52241         lda #32               ; laufwerk-bit 2
52243         bne a52273            ; setzen ->
52245
52245 a52245  jsr a52305            ; ---> 'u' übernehmen
52248         jmp a52273            ; ---> syntax-flag setzen
52251
52251 a52251  jsr a52312            ; ---> gerätenummer übernehmen
52254         jmp a52273            ; ---> syntax-flag setzen
52257
52257 a52257  lda #2                ; filename-bit 2
52259         jsr a52329            ; ---> filenamen übernehmen
52262         sta a626              ; länge des namens
52265         stx a628              ; l,adresse des namens
52268         sty a629              ; h,...
52271         lda #2                ; filename-bit 2
52273 a52273  ora *a130             ; in syntax-flag odern
52275         sta *a130
52277         jsr a1145             ; ---> chrgot
52280         beq a52182            ; trennzeichen: ->
52282         cmp #',
52284         beq a52203
52286         cmp #145              ; 'on'-token
52288         beq a52245
52290         cmp #'u               ; unit: gerätenummer
52292         beq a52251
52294 a52294  ldx #11               ; 'syntax'
52296         .by 44
52297 a52297  ldx #14               ; 'illegal quantity'
52299         .by 44
52300 a52300  ldx #23               ; 'string too long'
52302         jmp a34435            ; ---> fehlerausgang
52305
52305 a52305  jsr a1139             ; ---> chrget
52308         cmp #'u               ; unit?
52310         bne a52294            ; nein: fehler ->
52312 a52312  jsr a52375            ; ---> frmevl, getbyt
52315         cpx #32               ; grö^er als 31?
52317         bcs a52297            ; ja: fehler ->
52319         cpx #3                ; kleiner als 3?
52321         bcc a52297            ; ja: fehler ->
52323         stx a631
52326         lda #8
52328         rts
```

```
52329 a52329  jsr a52394          ; ---> fehler, wenn schon gesetzt
52332         jsr a40008          ; ---> frmevl, chkstr, frestr
52335         tax                 ; stringlänge
52336         beq a52297          ; = 0: fehler ->
52338         ldy #0
52340         jsr a1200           ; ---> lda (a34),y
52343         cmp #'ß             ; 'replace'?
52345         bne a52365          ; nein: ->
52347         lda #128            ; replace-bit
52349         jsr a52394          ; ---> fehler, wenn schon gesetzt
52352         lda *a130           ; replace-bit setzen
52354         ora #128
52356         sta *a130
52358         dex                 ; stringlänge erniedrigen
52359         inc *a34            ; string-anfangsadresse erhöhen
52361         bne a52365
52363         inc *a35
52365 a52365  txa                 ; stringlänge
52366         cmp #17             ; grö^er als 16?
52368         bcs a52300          ; ja: fehler ->
52370         ldx *a34            ; l,stringanfangsadresse
52372         ldy *a35            ; h,...
52374         rts
52375
52375 a52375  jsr a1139           ; ---> chrget
52378         beq a52294          ; trennzeichen: fehler ->
52380         bcc a52391
52382         jsr a38030          ; ---> prüfen ob '(' folgt
52385         jsr a40324          ; ---> frmevl, getbyt
52388         jmp a38027          ; ---> prüfen ob ')' folgt
52391
52391 a52391  jmp a40324          ; ---> frmevl, getbyt
52394
52394 a52394  and *a130           ; syntax flag
52396         bne a52294          ; syntax error ->
52398         rts
52399
52399 a52399  and #%11100110      ; 2. filename, 2. laufwerk, la?
52401         beq a52405          ; nein: ->
52403 a52403  bne a52294          ; fehler ->
52405
52405 a52405  lda *a130
52407         and #1              ; filename angegeben?
52409         cmp #1
52411         bne a52403          ; nein: fehler ->
52413         lda *a130
52415         rts
52416
52416 a52416  and #%11000100      ; la oder replace?
52418         bne a52403          ; ja: fehler ->
52420         lda *a130
52422 a52422  and #3              ; genau 2 filenamen angegeben?
52424         cmp #3
52426         bne a52403          ; nein: fehler ->
52428         lda *a130           ; syntax-flag
52430         rts
```

```
52431 a52431  lda *a121              ; länge von ds$
52433         bne a52452             ; ds$ gültig: ->
52435         lda #40
52437         sta *a121
52439         jsr a43270             ; ---> platz für string reservieren
52442         stx *a122              ; l,adresse von ds$
52444         sty *a123              ; h,...
52446         ldy #40
52448         lda #13
52450         sta (a122),y           ; wozu??? geht in r-zeiger!
52452 a52452  ldx a631               ; fa: geräteadresse
52455         bne a52462             ; nicht null: ->
52457         ldx #8                 ; ersatzwert 8
52459         stx a631               ; für geräteadresse
52462 a52462  lda #0                 ; la: logische adresse
52464         ldy #111               ; sa: sekundäradresse kanal 15
52466         jsr a65466             ; ---> setlfs
52469         lda #0                 ; filenamen löschen
52471         jsr a65469             ; ---> setnam
52474         jsr a65472             ; ---> open
52477         ldx #0
52479         jsr a65478             ; ---> chkin
52482         bcs a52511             ; fehler: ->
52484         ldy #255
52486 a52486  iny
52487         jsr a65487             ; ---> basin
52490         cmp #13                ; cr-code?
52492         beq a52498             ; ja: fertig ->
52494         sta (a122),y           ; eingelesenes zeichen in ds$ kopieren
52496         bne a52486             ; immer ->
52498
52498 a52498  lda #0                 ; endemarke
52500         sta (a122),y           ; schlie^t ds$ ab
52502         jsr a65484             ; ---> clrch
52505         lda #0
52507         sec
52508         jmp a65475             ; ---> close kanal 15
52511
52511 a52511  pha                    ; fehlerkode
52512         jsr a52498             ; ---> ds$ abschlie^en
52515         jsr a52567             ; ---> disk-status löschen
52518         pla
52519         tax
52520         jmp a34435             ; ---> fehlerausgang
52523
52523 ;------------------------------  A R E   Y O U   S U R E ?
52523
52523 a52523  bit *a129              ; direktmodus?
52525         bmi a52564             ; nein: keine meldung ->
52527         jsr a65359             ; ---> meldung ausgeben
52530         .by 'are you sure?' 0
52544         jsr a65484             ; ---> clrch
52547         jsr a65487             ; ---> basin
52550         pha                    ; 1. eingegebenes zeichen merken
52551 a52551  cmp #13                ; cr-kode?
52553         beq a52560             ; ja: ->
52555         jsr a65487             ; ---> basin
52558         bne a52551             ; immer ->
```

```
52560 a52560  pla                     ; 1. eingegebenes zeichen
52561         cmp #'y
52563         rts
52564
52564 a52564  lda #0                  ; simuliert 'y'
52566         rts
52567
52567 ;-------------------------------  C L E A R  D I S K  S T A T U S
52567
52567 a52567  tya
52568         pha
52569         lda *a121               ; länge von ds$
52571         beq a52583              ; = 0: ->
52573         ldy #40                 ; string ds$ ungültig machen
52575         tya
52576         sta (a122),y
52578         iny
52579         lda #255
52581         sta (a122),y
52583 a52583  lda #0
52585         sta *a121               ; länge von ds$
52587         pla
52588         tay
52589         rts
52590
52590 ;-------------------------------  R E S T E  V O N  K E Y
52590
52590 a52590  .by ',0 yek'            ; 'key 0,' rückwärts
52596
52596 a52596  tax
52597         tya
52598         pha
52599         lda #0
52601         jsr a42079              ; ---> (a,x) ausgeben
52604         pla
52605         tay
52606         rts
52607
52607 ;-------------------------------  R U C K S A C K  A U S  L O O P
52607
52607 a52607  sta *a58                ; h,zeilennummer
52609         dey
52610         tax                     ; = 255?
52611         inx
52612         bne a52616              ; nein: ->
52614         stx *a129               ; direktmodus setzen
52616 a52616  rts
52617
52617 ;-------------------------------  A U T O R E N L I S T E
52617
52617 a52617  .by 216 27 20 12 7 123 1 215    ; t.ryan
52625         .by 216 17 7 16 29 123 23       ; b.herd
52632         .by 216 7 16 5 26 26 22 123 31  ; j.cooper
52641         .by 216 27 16 2 26 23 123 19 71 216     ; f.bowen
52651 ;
52651 a52651  ldy #33
52653 a52653  lda a52617,y            ; verschlüsselte namen aus tabelle
52656         eor #%01010101          ; entschlüsseln
52658         jsr a65490              ; ---> bsout
52661         dey
52662         bpl a52653
52664         rts
```

```
52665 ;------------------------------
52665
52665         .by 0
52666
52666 ; der bereich 52666...52735 enthält den kode 255
52666
52666 ;------------------------------   I N T E R R U P T
52666
52736 a52736  tsx
52737         lda a256+4,x
52740         and #16                 ; hardware-interrupt?
52742         bne a52747              ; nein: ->
52744         jmp (a788)              ; ---> (a52750)
52747
52747 a52747  jmp (a790)              ; ---> (a62540) monitor break entry
52750
52750 a52750  lda a65289              ; interrupt-flags
52753         and #2                  ; raster-interrupt?
52755         beq a52760              ; nein: ->
52757         jsr a52832              ; ---> graphic bearbeiten
52760 a52760  bit a2008               ; rs-232 in betrieb?
52763         bpl a52779              ; nein: ->
52765         lda a64769              ; rs-232-status
52768         sta a2004
52771         bpl a52779              ; kein rs-232-interrupt: ->
52773         jsr a60053              ; ---> datenbyte empfangen
52776         jsr a59995              ; --->
52779 a52779  jsr a58340              ; ---> timer setzen im kassettenbetrieb
52782         lda a65289
52785         and #2
52787         beq a52829
52789         sta a65289              ; interrupt rücksetzen
52792         bit a65291
52795         lda #204                ; nächster raster-interrupt
52797         bvc a52826              ; interrupt war bei raster 161
52799         jmp (a786)              ; standard interrupt vektor (a52802)
52802
52802 a52802  jsr a53183              ; ---> recorder bedienen, uhr takten
52805         jsr a52941              ; ---> sound bearbeiten
52808         lda *a251               ; aktuellen modul
52810         pha                     ; merken
52811         lda #0                  ; rom
52813         sta *a251               ; einschalten
52815         php
52816         cli
52817         jsr a56081              ; ---> scnkey: tastaturabfrage
52820         plp
52821         pla
52822         sta *a251               ; aktuellen modul wiederherstellen
52824         lda #161                ; nächster raster-interrupt
52826 a52826  sta a65291              ; raster-interrupt vorbereiten
52829 a52829  jmp a64702              ; ---> modul einschalten, rti
52832
52832 a52832  lda a65308              ; h,raster
52835         and #1                  ; gesetzt?
52837         bne a52896              ; ja: raster nicht sichtbar ->
52839         lda a65309              ; l,raster
52842         cmp #163                ; noch im graphik-bereich?
52844         bcs a52892              ; nein: ->
52846         bit *a131               ; graphic mit text?
52848         bvc a52932              ; nein: fertig ->
```

```
52850          lda #h,a2048            ; zeichen-speicher einschalten
52852          sta a65300
52855          lda a65286
52858          and #%11011111          ; bitspeicher-freigabebit rücksetzen
52860          tay                     ; byte merken
52861          lda a65287
52864          and #%11101111          ; multicolor-freigabebit rücksetzen
52866          tax                     ; byte merken
52867          lda a65298
52870          ora a2042               ; = 4 (vom reset)
52873          pha
52874 a52874   lda a65309              ; l,raster
52877          cmp #163                ; noch im graphikbereich?
52879          bcc a52874              ; ja: warten ->
52881          pla
52882          sta a65298              ; graphic-modus abschalten
52885          sty a65286
52888          stx a65287
52891          rts
52892
52892 a52892   cmp #204                ; l,raster noch im textbereich?
52894          bcc a52932              ; ja: fertig ->
52896 a52896   ldx *a131               ; graphic-modus
52898          beq a52932              ; ausgeschaltet: ->
52900          bpl a52910              ; kein multicolor: ->
52902          lda a65287
52905          ora #%00010000          ; multicolor-freigabebit
52907          sta a65287              ; setzen
52910 a52910   lda a65286
52913          ora #%00100000          ; bitspeicher-freigabebit
52915          sta a65286              ; setzen
52918          lda a65298
52921          and #%11111011
52923          sta a65298
52926          lda a2043               ; = h,a6144 (vom reset)
52929          sta a65300              ; bitspeicher einschalten
52932 a52932   rts
52933
52933 ;-------------------------------
52933
52933          .by 234 234 234 234 234 234 234 234
52941
52941 ;-------------------------------  S O U N D  B E A R B E I T E N
52941
52941 a52941   ldx #1
52943 a52943   lda a1276,x             ; zähler (x) abgelaufen?
52946          ora a1278,x
52949          beq a52970              ; ja: ->
52951          inc a1276,x             ; zähler (x) inkrementieren
52954          bne a52970
52956          inc a1278,x
52959          bne a52970              ; noch nicht am ende: ->
52961          lda a52974,x
52964          and a65297              ; sound (x) abschalten
52967          sta a65297
52970 a52970   dex
52971          bpl a52943
52973          rts
52974
52974 a52974   .by %11101111 %10011111
```

```
52976 ;------------------------------  U H R   T A K T
52976
52976 a52976  inc *a165               ; uhrzeit inkrementieren
52978         bne a52986
52980         inc *a164
52982         bne a52986
52984         inc *a163
52986 a52986  sec                     ; 24 h erreicht?
52987         lda *a165
52989         sbc #1
52991         lda *a164
52993         sbc #26
52995         lda *a163
52997         sbc #79
52999         bcc a53009              ; nein: ->
53001         ldx #0                  ; uhr auf 0 zurücksetzen
53003         stx *a163
53005         stx *a164
53007         stx *a165
53009 a53009  lda #127
53011         jsr a56176              ; ---> tastatur-dekoder abfragen
53014         sta *a238               ; ergebnis merken
53016         lda #127
53018         jsr a56176              ; ---> tastatur-dekoder abfragen
53021         cmp *a238               ; gleiches ergebnis?
53023         bne a53009              ; nein: abfrage wiederholen ->
53025         ora #127
53027         sta *a145               ; stkey: 127 = stoptaste gedrückt
53029         rts                     ;        255 = ... nicht gedrückt
53030
53030 ;------------------------------  U H R   L E S E N
53030
53030 a53030  sei
53031         lda *a165
53033         ldx *a164
53035         ldy *a163
53037
53037 ;------------------------------  U H R   S T E L L E N
53037
53037 a53037  sei
53038         sta *a165
53040         stx *a164
53042         sty *a163
53044         cli
53045         rts
53046
53046 ;------------------------------  M O N I T O R - M E L D U N G E N
53046
53046 a53046  .by 13 'monitor' 141
53055         .by 13 'breaK'
53061         .by 13 '   pc  sr ac xr yr sp' 13 ';' 160
53086         .by 'a' 160
53088         .by ' erroR'
53094 ;
53094 a53094  lda a53046,x            ; zeichen aus text
53097         php
53098         and #127
53100         jsr a65490              ; ---> bsout
53103         inx
53104         plp
53105         bpl a53094              ; nicht letztes zeichen: ->
53107         rts
```

```
53108 ;-----------  A B S C H L U S S  D E R  Z E I L E N Ü B E R N A H M E
53108
53108 a53108  lda #13
53110         ldx *a152               ; eingabe-kanal
53112         cpx #3                  ; bildschirm?
53114         beq a53122              ; ja: ->
53116         ldx *a153               ; ausgabe-kanal
53118         cpx #3                  ; bildschirm?
53120         beq a53125              ; ja: ->
53122 a53122  jsr a56393              ; ---> print
53125 a53125  lda #13
53127         jmp a55728              ; ---> register wiederherstellen
53130
53130 ;------------------------------
53130
53130 a53130  lda a275,x              ; farbkodes im ram
53133         bit a2041               ; flag für kodes im rom gesetzt?
53136         bpl a53141              ; nein: ->
53138         lda a57667,x            ; farbkodes im rom
53141 a53141  rts
53142
53142 a53142  bit a2040               ; flag für laden aus ram gesetzt?
53145         bmi a53150              ; ja: ->
53147         lda (a161),y            ; laden aus rom
53149         rts
53150
53150 a53150  lda #a161                ; ladeadresse
53152         sta a2015               ; einsetzen
53155         jmp a2009               ; ---> lda (a161),y
53158
53158 a53158  lda #9                   ; teil von reset
53160         sta a64800
53163         ora #128
53165         sta a64800
53168         jmp a64542              ; ---> modul-initialisierung
53171
53171 a53171  php                     ; diese routine wird ins ram kopiert
53172         sei                     ; ab a2009
53173         sta a65343              ; ram einschalten
53176         lda (a0),y              ; adresse wird vorher eingesetzt
53178         sta a65342              ; rom einschalten
53181         plp
53182         rts
53183
53183 ;-------------------  K A S S E T T E  U N D  U H R  B E D I E N E N
53183
53183 a53183  lda a64784
53186         and #4                  ; recorder play-taste gedrückt?
53188         bne a53217              ; nein: ->
53190         bit a2044               ; läuft motor?
53193         bmi a53201              ; ja: ->
53195         lda *a1
53197         and #%11110111          ; motor einschalten
53199         sta *a1
53201 a53201  dec a2045               ; zähler für uhr-takt:
53204         bpl a53214              ; in 5 interrupts wird die uhr
53206         lda #4                  ; um 6 takte weitergeschaltet
53208         sta a2045
53211         jsr a52976              ; ---> uhr weiterschalten
53214 a53214  jmp a52976              ; ---> uhr weiterschalten
```

```
53217 a53217  sta a2044              ; motor-flag rücksetzen
53220         jsr a58288             ; ---> motor ausschalten
53223         jmp a53201             ; ---> uhr weiterschalten
53226
53226 ;-----------------  R U C K S A C K   V O N   I E C - A B S C H L U S S
53226
53226 a53226  inx
53227         stx a65220
53230         stx a65216
53233         lda #128
53235         sta a65297
53238         rts
53239
53239 ;------------------------------
53239
53239         .by 255 255 255 255 255 255 255 255 255
53248
53248 ;--------------------------  Z E I C H E N S Ä T Z E   F Ü R   C H A R
53248
53248 a53248  .by %00111100          ;                ß
53249         .by %01100110
53250         .by %01101110
53251         .by %01101110
53252         .by %01100000
53253         .by %01100010
53254         .by %00111100
53255         .by %00000000
53256
53256         .by 24 60 102 126 102 102 102 0        ; A
53264         .by 124 102 102 124 102 102 124 0      ; B
53272         .by 60 102 96 96 96 102 60 0    ;        C
53280         .by 120 108 102 102 102 108 120 0      ; D
53288         .by 126 96 96 120 96 96 126 0   ;        E
53296         .by 126 96 96 120 96 96 96 0    ;        F
53304         .by 60 102 96 110 102 102 60 0  ;        G
53312         .by 102 102 102 126 102 102 102 0      ; H
53320         .by 60 24 24 24 24 24 60 0      ;        I
53328         .by 30 12 12 12 12 108 56 0     ;        J
53336         .by 102 108 120 112 120 108 102 0      ; K
53344         .by 96 96 96 96 96 96 126 0     ;        L
53352         .by 99 119 127 107 99 99 99 0   ;        M
53360         .by 102 118 126 126 110 102 102 0      ; N
53368         .by 60 102 102 102 102 102 60 0        ; O
53376         .by 124 102 102 124 96 96 96 0  ;        P
53384         .by 60 102 102 102 102 60 14 0  ;        Q
53392         .by 124 102 102 124 120 108 102 0      ; R
53400         .by 60 102 96 60 6 102 60 0     ;        S
53408         .by 126 24 24 24 24 24 24 0     ;        T
53416         .by 102 102 102 102 102 102 60 0       ; U
53424         .by 102 102 102 102 102 60 24 0        ; V
53432         .by 99 99 99 107 127 119 99 0   ;        W
53440         .by 102 102 60 24 60 102 102 0  ;        X
53448         .by 102 102 102 60 24 24 24 0   ;        Y
53456         .by 126 6 12 24 48 96 126 0     ;        Z
53464         .by 60 48 48 48 48 48 60 0      ;        ä
53472         .by 12 18 48 124 48 98 252 0    ;        ö
53480         .by 60 12 12 12 12 12 60 0      ;        ü
53488         .by 0 24 60 126 24 24 24 24     ;        ^
53496         .by 0 16 48 127 127 48 16 0     ;        _
```

```
53504 a53504  .by 0 0 0 0 0 0 0 0     ;                blank
53512         .by 24 24 24 24 0 0 24 0        ;        !
53520         .by 102 102 102 0 0 0 0 0       ;        "
53528         .by 102 102 255 102 255 102 102 0       ; #
53536         .by 24 62 96 60 6 124 24 0      ;        $
53544         .by 98 102 12 24 48 102 70 0    ;        %
53552         .by 60 102 60 56 103 102 63 0   ;        &
53560         .by 6 12 24 0 0 0 0 0   ;                '
53568         .by 12 24 48 48 48 24 12 0      ;        (
53576         .by 48 24 12 12 12 24 48 0      ;        )
53584         .by 0 102 60 255 60 102 0 0     ;        *
53592         .by 0 24 24 126 24 24 0 0       ;        +
53600         .by 0 0 0 0 0 24 24 48  ;                ,
53608         .by 0 0 0 126 0 0 0 0   ;                -
53616         .by 0 0 0 0 0 24 24 0   ;                .
53624         .by 0 3 6 12 24 48 96 0         ;        /
53632         .by 60 102 110 118 102 102 60 0         ; 0
53640         .by 24 24 56 24 24 24 126 0     ;        1
53648         .by 60 102 6 12 48 96 126 0     ;        2
53656         .by 60 102 6 28 6 102 60 0      ;        3
53664         .by 6 14 30 102 127 6 6 0       ;        4
53672         .by 126 96 124 6 6 102 60 0     ;        5
53680         .by 60 102 96 124 102 102 60 0  ;        6
53688         .by 126 102 12 24 24 24 24 0    ;        7
53696         .by 60 102 102 60 102 102 60 0  ;        8
53704         .by 60 102 102 62 6 102 60 0    ;        9
53712         .by 0 0 24 0 0 24 0 0   ;                :
53720         .by 0 0 24 0 0 24 24 48         ;        ;
53728         .by 14 24 48 96 48 24 14 0      ;        <
53736         .by 0 0 126 0 126 0 0 0         ;        =
53744         .by 112 24 12 6 12 24 112 0     ;        >
53752         .by 60 102 6 12 24 0 24 0       ;        ?
53760
53760 a53760  .by 0 0 0 255 255 0 0 0         ;        192
53768         .by 8 28 62 127 127 28 62 0     ;        193
53776         .by 24 24 24 24 24 24 24 24     ;        194
53784         .by 0 0 0 255 255 0 0 0         ;        195
53792         .by 0 0 255 255 0 0 0 0         ;        196
53800         .by 0 255 255 0 0 0 0 0         ;        197
53808         .by 0 0 0 0 255 255 0 0         ;        198
53816         .by 48 48 48 48 48 48 48 48     ;        199
53824         .by 12 12 12 12 12 12 12 12     ;        200
53832         .by 0 0 0 224 240 56 24 24      ;        201
53840         .by 24 24 28 15 7 0 0 0         ;        202
53848         .by 24 24 56 240 224 0 0 0      ;        203
53856         .by 192 192 192 192 192 192 255 255     ; 204
53864         .by 192 224 112 56 28 14 7 3    ;        205
53872         .by 3 7 14 28 56 112 224 192    ;        206
53880         .by 255 255 192 192 192 192 192 192     ; 207
53888         .by 255 255 3 3 3 3 3 3         ;        208
53896         .by 0 60 126 126 126 126 60 0   ;        209
53904         .by 0 0 0 0 0 255 255 0         ;        210
53912         .by 54 127 127 127 62 28 8 0    ;        211
53920         .by 96 96 96 96 96 96 96 96     ;        212
53928         .by 0 0 0 7 15 28 24 24         ;        213
53936         .by 195 231 126 60 60 126 231 195       ; 214
53944         .by 0 60 126 102 102 126 60 0   ;        215
53952         .by 24 24 102 102 24 24 60 0    ;        216
53960         .by 6 6 6 6 6 6 6 6     ;                217
53968         .by 8 28 62 127 62 28 8 0       ;        218
```

```
53976           .by 24 24 24 255 255 24 24 24   ;          219
53984           .by 192 192 48 48 192 192 48 48         ; 220
53992           .by 24 24 24 24 24 24 24 24     ;          221
54000           .by 0 0 3 62 118 54 54 0        ;          222
54008           .by 255 127 63 31 15 7 3 1      ;          223
54016
54016 a54016    .by 0 0 0 0 0 0 0 0     ;                  160
54024           .by 240 240 240 240 240 240 240 240     ; 161
54032           .by 0 0 0 0 255 255 255 255     ;          162
54040           .by 255 0 0 0 0 0 0 0   ;                  163
54048           .by 0 0 0 0 0 0 0 255   ;                  164
54056           .by 192 192 192 192 192 192 192 192     ; 165
54064           .by 204 204 51 51 204 204 51 51         ; 166
54072           .by 3 3 3 3 3 3 3 3     ;                  167
54080           .by 0 0 0 0 204 204 51 51       ;          168
54088           .by 255 254 252 248 240 224 192 128     ; 169
54096           .by 3 3 3 3 3 3 3 3     ;                  170
54104           .by 24 24 24 31 31 24 24 24     ;          171
54112           .by 0 0 0 0 15 15 15 15         ;          172
54120           .by 24 24 24 31 31 0 0 0        ;          173
54128           .by 0 0 0 248 248 24 24 24      ;          174
54136           .by 0 0 0 0 0 0 255 255         ;          175
54144           .by 0 0 0 31 31 24 24 24        ;          176
54152           .by 24 24 24 255 255 0 0 0      ;          177
54160           .by 0 0 0 255 255 24 24 24      ;          178
54168           .by 24 24 24 248 248 24 24 24   ;          179
54176           .by 192 192 192 192 192 192 192 192     ; 180
54184           .by 224 224 224 224 224 224 224 224     ; 181
54192           .by 7 7 7 7 7 7 7 7     ;                  182
54200           .by 255 255 0 0 0 0 0 0         ;          183
54208           .by 255 255 255 0 0 0 0 0       ;          184
54216           .by 0 0 0 0 0 255 255 255       ;          185
54224           .by 3 3 3 3 3 3 255 255         ;          186
54232           .by 0 0 0 0 240 240 240 240     ;          187
54240           .by 15 15 15 15 0 0 0 0         ;          188
54248           .by 24 24 24 248 248 0 0 0      ;          189
54256           .by 240 240 240 240 0 0 0 0     ;          190
54264           .by 240 240 240 240 15 15 15 15         ; 191
54272
54272 a54272    .by 60 102 110 110 96 98 60 0   ;          ß
54280           .by 0 0 60 6 62 102 62 0        ;          a
54288           .by 96 96 124 102 102 102 124 0         ; b
54296           .by 0 0 60 102 96 102 60 0      ;          c
54304           .by 6 6 62 102 102 102 62 0     ;          d
54312           .by 0 0 60 102 126 96 62 0      ;          e
54320           .by 28 54 48 120 48 48 48 0     ;          f
54328           .by 0 0 62 102 102 62 6 124     ;          g
54336           .by 96 96 124 102 102 102 102 0         ; h
54344           .by 24 0 24 24 24 24 24 0       ;          i
54352           .by 6 0 6 6 6 6 102 60  ;                  j
54360           .by 96 96 102 108 120 124 102 0         ; k
54368           .by 56 24 24 24 24 24 60 0      ;          l
54376           .by 0 0 107 127 127 99 99 0     ;          m
54384           .by 0 0 124 102 102 102 102 0   ;          n
54392           .by 0 0 60 102 102 102 60 0     ;          o
54400           .by 0 0 124 102 102 124 96 96   ;          p
54408           .by 0 0 62 102 102 62 6 6       ;          q
54416           .by 0 0 124 102 96 96 96 0      ;          r
54424           .by 0 0 60 96 60 6 124 0        ;          s
54432           .by 48 48 252 48 48 54 28 0     ;          t
```

```
54440          .by 0 0 102 102 102 102 60 0    ;          u
54448          .by 0 0 102 102 102 60 24 0     ;          v
54456          .by 0 0 99 107 127 54 34 0      ;          w
54464          .by 0 0 102 60 24 60 102 0      ;          x
54472          .by 0 0 102 102 102 62 6 124    ;          y
54480          .by 0 0 126 12 24 48 126 0      ;          z
54488          .by 60 48 48 48 48 48 60 0      ;          ä
54496          .by 12 18 48 124 48 98 252 0    ;          ö
54504          .by 60 12 12 12 12 12 60 0      ;          ü
54512          .by 0 24 60 126 24 24 24 24     ;          ^
54520          .by 0 16 48 127 127 48 16 0     ;          _
54528
54528 a54528   .by 0 0 0 0 0 0 0 0     ;                  blank
54536          .by 24 24 24 24 0 0 24 0        ;          !
54544          .by 102 102 102 0 0 0 0 0       ;          "
54552          .by 102 102 255 102 255 102 102 0      ; #
54560          .by 24 62 96 60 6 124 24 0      ;          $
54568          .by 98 102 12 24 48 102 70 0    ;          %
54576          .by 60 102 60 56 103 102 63 0   ;          &
54584          .by 6 12 24 0 0 0 0 0   ;                  '
54592          .by 12 24 48 48 48 24 12 0      ;          (
54600          .by 48 24 12 12 12 24 48 0      ;          )
54608          .by 0 102 60 255 60 102 0 0     ;          *
54616          .by 0 24 24 126 24 24 0 0       ;          +
54624          .by 0 0 0 0 0 24 24 48  ;                  ,
54632          .by 0 0 0 126 0 0 0 0   ;                  -
54640          .by 0 0 0 0 0 0 24 24 0   ;                .
54648          .by 0 3 6 12 24 48 96 0         ;          /
54656          .by 60 102 110 118 102 102 60 0        ; 0
54664          .by 24 24 56 24 24 24 126 0     ;          1
54672          .by 60 102 6 12 48 96 126 0     ;          2
54680          .by 60 102 6 28 6 102 60 0      ;          3
54688          .by 6 14 30 102 127 6 6 0       ;          4
54696          .by 126 96 124 6 6 102 60 0     ;          5
54704          .by 60 102 96 124 102 102 60 0  ;          6
54712          .by 126 102 12 24 24 24 24 0    ;          7
54720          .by 60 102 102 60 102 102 60 0  ;          8
54728          .by 60 102 102 62 6 102 60 0    ;          9
54736          .by 0 0 24 0 0 24 0 0   ;                  :
54744          .by 0 0 24 0 0 24 24 48         ;          ;
54752          .by 14 24 48 96 48 24 14 0      ;          <
54760          .by 0 0 126 0 126 0 0 0         ;          =
54768          .by 112 24 12 6 12 24 112 0     ;          >
54776          .by 60 102 6 12 24 0 24 0       ;          ?
54784
54784 a54784   .by 0 0 0 255 255 0 0 0         ;          192
54792          .by 24 60 102 126 102 102 102 0        ; A
54800          .by 124 102 102 124 102 102 124 0      ; B
54808          .by 60 102 96 96 96 102 60 0    ;          C
54816          .by 120 108 102 102 102 108 120 0      ; D
54824          .by 126 96 96 120 96 96 126 0   ;          E
54832          .by 126 96 96 120 96 96 96 0    ;          F
54840          .by 60 102 96 110 102 102 60 0  ;          G
54848          .by 102 102 102 126 102 102 102 0      ; H
54856          .by 60 24 24 24 24 24 60 0      ;          I
54864          .by 30 12 12 12 12 108 56 0     ;          J
54872          .by 102 108 120 112 120 108 102 0      ; K
54880          .by 96 96 96 96 96 96 126 0     ;          L
54888          .by 99 119 127 107 99 99 99 0   ;          M
54896          .by 102 118 126 126 110 102 102 0      ; N
```

```
54904          .by 60 102 102 102 102 102 60 0          ; O
54912          .by 124 102 102 124 96 96 96 0  ;          P
54920          .by 60 102 102 102 102 60 14 0  ;          Q
54928          .by 124 102 102 124 120 108 102 0        ; R
54936          .by 60 102 96 60 6 102 60 0      ;          S
54944          .by 126 24 24 24 24 24 24 0      ;          T
54952          .by 102 102 102 102 102 102 60 0         ; U
54960          .by 102 102 102 102 102 60 24 0          ; V
54968          .by 99 99 99 107 127 119 99 0    ;          W
54976          .by 102 102 60 24 60 102 102 0   ;          X
54984          .by 102 102 102 60 24 24 24 0    ;          Y
54992          .by 126 6 12 24 48 96 126 0      ;          Z
55000          .by 24 24 24 255 255 24 24 24    ;          219
55008          .by 192 192 48 48 192 192 48 48          ; 220
55016          .by 24 24 24 24 24 24 24 24      ;          221
55024          .by 51 51 204 204 51 51 204 204          ; 222
55032          .by 51 153 204 102 51 153 204 102        ; 223
55040
55040 a55040   .by 0 0 0 0 0 0 0 0      ;                  blank
55048          .by 240 240 240 240 240 240 240 240      ; 161
55056          .by 0 0 0 0 255 255 255 255      ;          162
55064          .by 255 0 0 0 0 0 0 0    ;                 163
55072          .by 0 0 0 0 0 0 0 255    ;                 164
55080          .by 192 192 192 192 192 192 192 192      ; 165
55088          .by 204 204 51 51 204 204 51 51          ; 166
55096          .by 3 3 3 3 3 3 3 3      ;                 167
55104          .by 0 0 0 0 204 204 51 51        ;          168
55112          .by 204 153 51 102 204 153 51 102        ; 169
55120          .by 3 3 3 3 3 3 3 3      ;                 170
55128          .by 24 24 24 31 31 24 24 24      ;          171
55136          .by 0 0 0 0 15 15 15 15          ;          172
55144          .by 24 24 24 31 31 0 0 0         ;          173
55152          .by 0 0 0 248 248 24 24 24       ;          174
55160          .by 0 0 0 0 0 0 255 255          ;          175
55168          .by 0 0 0 31 31 24 24 24         ;          176
55176          .by 24 24 24 255 255 0 0 0       ;          177
55184          .by 0 0 0 255 255 24 24 24       ;          178
55192          .by 24 24 24 248 248 24 24 24    ;          179
55200          .by 192 192 192 192 192 192 192 192      ; 180
55208          .by 224 224 224 224 224 224 224 224      ; 181
55216          .by 7 7 7 7 7 7 7 7      ;                 182
55224          .by 255 255 0 0 0 0 0 0          ;          183
55232          .by 255 255 255 0 0 0 0 0        ;          184
55240          .by 0 0 0 0 0 255 255 255        ;          185
55248          .by 1 3 6 108 120 112 96 0       ;          186
55256          .by 0 0 0 0 240 240 240 240      ;          187
55264          .by 15 15 15 15 0 0 0 0          ;          188
55272          .by 24 24 24 248 248 0 0 0       ;          189
55280          .by 240 240 240 240 0 0 0 0      ;          190
55288          .by 240 240 240 240 15 15 15 15          ; 191
55296
55296 ;--------------------------------
55296
55296          .by 4 24
55298
55298 ;--------------------------------  Z E I L E N A D R E S S E N
55298
55298 a55298   .by 0 40 80 120 160 200 240 24 64 104    ; l-bytes
55308          .by 144 184 224 8 48 88 128 168 208 248
55318          .by 32 72 112 152 192
55323
55323 a55323   .by 12 12 12 12 12 12 12 13 13 13        ; h-bytes
55333          .by 13 13 13 14 14 14 14 14 14 14
55343          .by 15 15 15 15 15
```

```
55348 ;------------------------------  S C R E E N
55348
55348 a55348  ldx #40                 ; spaltenzahl
55350         ldy #25                 ; zeilenzahl
55352         rts
55353
55353 ;------------------------------  P L O T
55353
55353 a55353  bcs a55369
55355         stx *a205               ; kursorzeile
55357         stx *a196               ; lstp: letzter zeilenwert
55359         sty *a202               ; kursorspalte
55361         sty *a197               ; lsxp: letzter spaltenwert
55363         jsr a56944              ; ---> volles bildfenster
55366         jsr a55464              ; ---> setpnt: kursorzeiger setzen
55369 a55369  ldx *a205               ; kursorzeile
55371         ldy *a202               ; kursorspalte
55373         rts
55374
55374 ;------------------------------  E D I T O R  R E S E T
55374
55374 a55374  lda #12
55376         sta a1342               ; wird anscheinend nicht abgefragt
55379         lda #3
55381         sta *a153               ; ausgabekanal = bildschirm
55383         lda #0
55385         sta *a152               ; eingabekanal = tastatur
55387         sta a1351               ; shift-C= freigeben
55390         sta *a131               ; textmodus einschalten
55392         sta *a239               ; ndx: tastenpuffer-index
55394         sta *a240               ; print-warteflag
55396         lda #l,a56186
55398         sta a1349               ; l,keylog-vektor
55401         lda #h,a56186
55403         sta a1350               ; h,...
55406         lda #10
55408         sta a1343               ; maximalwert für tastenpuffer-index
55411         sta a1354
55414         sta a1346               ; delay: anlauf-verzögerung
55417         lda #128
55419         sta a1344               ; rptflg: flag für tastenwiederholung
55422         lda #16                 ; schwarz
55424         sta a1339               ; zeichenfarbe
55427         lda #4
55429         sta a1345               ; kount: normalverzögerung
55432 a55432  jsr a56944              ; ---> volles bildfenster
55435
55435 ;------------------------------  C L E A R  S C R E E N
55435
55435 a55435  jsr a55450              ; ---> home
55438 a55438  jsr a55466              ; ---> kursorzeiger auf zeile (x)
55441         jsr a56055              ; ---> zeile (x) löschen
55444         cpx a2021               ; endzeile erreicht?
55447         inx
55448         bcc a55438              ; nein: ->
```

```
55450 ;------------------------------  H O M E
55450
55450 a55450  ldx a2022               ; sctop: oberste zeile
55453         stx *a205               ; kursorzeile
55455         stx *a196               ; lstp: letzter zeilenwert
55457 a55457  ldy a2023               ; linker rand
55460         sty *a202               ; kursorspalte
55462         sty *a197               ; lsxp: letzter spaltenwert
55464 a55464  ldx *a205               ; kursorzeile
55466 a55466  lda a55298,x            ; l-adressentabelle
55469         sta *a200               ; l,pnt: zeilenanfangsadresse
55471         lda a55323,x            ; h-adressentabelle
55474         sta *a201               ; h,pnt: zeilenanfangsadresse
55476 a55476  lda *a200               ; l,pnt: zeilenanfangsadresse
55478         sta *a234               ; l,cpnt: adresse im color ram
55480         lda *a201               ; h,pnt: zeilenanfangsadresse
55482         and #3
55484         ora #h,a2048
55486         sta *a235               ; h,cpnt: adresse im color ram
55488         rts
55489
55489 ;---------------  K O D E  A U S  T A S T E N P U F F E R  H O L E N
55489
55489 a55489  ldy a1373               ; kyndx: funktionstextlänge
55492         beq a55508              ; = 0: ->
55494         ldy a1374               ; keyidx: zeiger in textpuffer
55497         lda a1383,y             ; zeichen aus text
55500         dec a1373               ; zeichenzähler erniedrigen
55503         inc a1374               ; zeiger erhöhen
55506         cli
55507         rts
55508
55508 a55508  ldy a1319               ; erster kode im tastenpuffer
55511         ldx #0
55513 a55513  lda a1319+1,x           ; pufferinhalt nachschieben
55516         sta a1319,x
55519         inx
55520         cpx *a239               ; ndx: alle kodes verschoben?
55522         bne a55513              ; nein: ->
55524         dec *a239               ; ndx: index erniedrigen
55526         tya                     ; erster kode
55527         cli
55528         clc
55529         rts
55530
55530 ;-------------------------  E I N G A B E  V O M  B I L D S C H I R M
55530
55530 a55530  jsr a56393              ; ---> print
55533 a55533  jsr a55476              ; ---> zeiger in color-ram setzen
55536         ldy *a202               ; kursorspalte
55538         lda (a234),y            ; kode aus color-ram
55540         pha                     ; retten
55541         lda a1339               ; zeichenfarbe
55544         sta (a234),y            ; in color-ram
55546         tya                     ; kursorspalte
55547         clc
55548         adc *a200               ; + l,pnt: zeilenanfangsadresse
55550         sta a65293              ; = l,kursoradresse
55553         lda *a201               ; h,pnt: zeilenanfangsadresse
55555         adc #0                  ; + übertrag
55557         sbc #11                 ; - 11
55559         sta a65292              ; = h,kursoradresse
```

```
55562 a55562  lda *a239               ; ndx: tastenpuffer-index
55564         ora a1373               ; kyndx: funktionstextlänge
55567         beq a55562              ; beide = 0: warten ->
55569         pla                     ; alten kode wieder
55570         sta (a234),y            ; in color-ram bringen
55572         lda #255                ; kursor abschalten
55574         sta a65292
55577         sta a65293
55580         jsr a55489              ; ---> kode aus tastenpuffer holen
55583         cmp #131                ; 'run'-taste?
55585         bne a55603              ; nein: ->
55587         ldx #9
55589         sei
55590         stx *a239               ; ndx: tastenpuffer-index
55592 a55592  lda a57642-1,x          ; text für 'run'-taste
55595         sta a1319-1,x           ; in tastenpuffer kopieren
55598         dex
55599         bne a55592
55601 a55601  beq a55533              ; zurück zum schleifenanfang ->
55603
55603 a55603  cmp #13                 ; 'return'?
55605         bne a55530              ; nein: ->
55607         sta *a199               ; crsw: flag für zeilenübernahme
55609         jsr a57237              ; ---> kursor an log. zeilenende
55612         stx a1353               ; bildschirmzeile
55615         jsr a57223              ; ---> kursor an zeilenanfang
55618         lda #0
55620         sta *a203               ; qtsw: quote-modus
55622         ldy a2023               ; linker rand
55625         lda *a196               ; lstp: letzter zeilenwert
55627         bmi a55648              ; flag für 'ganze zeile' gesetzt: ->
55629         cmp *a205               ; kursorzeile
55631         bcc a55648              ; lstp < kursorzeile: ->
55633         ldy *a197               ; lsxp: letzter spaltenwert
55635         cmp a1353               ; lstp = log. endzeile?
55638         bne a55644              ; nein: ->
55640         cpy *a195               ; letzter spaltenwert = zeilenende?
55642         beq a55646              ; ja: ->
55644 a55644  bcs a55663              ; letzter spaltenwert > zeilenende: ->
55646 a55646  sta *a205               ; kursorzeile
55648 a55648  sty *a202               ; kursorspalte
55650         jmp a55671              ; ---> zeile übernehmen
55653
55653 ;-------------  Z E I C H E N   V O M   B I L D S C H I R M   H O L E N
55653
55653 a55653  tya
55654         pha
55655         txa
55656         pha
55657         lda *a199               ; crsw: flag für zeilenübernahme
55659         beq a55601              ; = 0: zeileneingabe starten
55661         bpl a55671              ; < 128: zeilenübernahme starten
55663 a55663  lda #0                  ; zeilenübernahme abschlieen
55665         sta *a199               ; crsw: flag für zeilenübernahme
55667         jmp a53108              ; ---> abschlu
55670
55670         nop
```

```
55671 a55671  jsr a55464              ; ---> kursorzeiger setzen
55674         jsr a57135              ; ---> zeichen vom bildschirm lesen
55677         sta *a206               ; und merken
55679         and #63
55681         asl *a206
55683         bit *a206
55685         bpl a55689              ; bit 6 gesetzt: ->
55687         ora #128
55689 a55689  bcc a55695              ; bit 7 gelöscht: ->
55691         ldx *a203               ; qtsw: quote-modus
55693         bne a55699              ; gesetzt: ->
55695 a55695  bvs a55699              ; bit 5 gesetzt: ->
55697         ora #64
55699 a55699  jsr a55738              ; ---> quote-modus setzen falls '"'
55702         ldy *a205               ; kursorzeile
55704         cpy a1353               ; < logische endzeile?
55707         bcc a55719              ; ja: ->
55709         ldy *a202               ; kursorspalte
55711         cpy *a195               ; < indx: zeilenende?
55713         bcc a55719              ; ja: ->
55715         ror *a199               ; crsw: flag für zeilenübernahme
55717         bmi a55722              ; zeilenübernahme abschlieen ->
55719
55719 a55719  jsr a57279              ; ---> kursor nach rechts
55722 a55722  cmp #222                ; kode für pi?
55724         bne a55728              ; nein: ->
55726         lda #255                ; ersetzen durch 255
55728 a55728  sta *a206               ; übernommenes zeichen
55730         pla
55731         tax
55732         pla
55733         tay
55734         lda *a206               ; übernommenes zeichen
55736         clc
55737         rts
55738
55738 ;------------------------------  Q U O T E - M O D U S
55738
55738 a55738  cmp #'"
55740         bne a55750
55742         lda *a203               ; qtsw: quote-modus
55744         eor #1
55746         sta *a203               ; qtsw: quote-modus
55748         lda #'"
55750 a55750  rts
55751
55751 ;------------------------------  A B S C H L U S S  V O N  P R I N T
55751
55751 a55751  lda *a206               ; ausgegebenes zeichen
55753         sta a2027               ; letztes ausgegebenes zeichen
55756         pla
55757         tay
55758         lda *a207               ; insert-zähler
55760         beq a55764              ; kein insert: ->
55762         lsr *a203               ; qtsw: quote-modus rücksetzen
55764 a55764  pla
55765         tax
55766         pla
55767         clc
55768         rts
```

```
55769 a55769  ora #64
55771 a55771  ldx *a194              ; rvs-flag
55773         beq a55777             ; nicht gesetzt: ->
55775 a55775  ora #128               ; bit 7 setzen
55777 a55777  ldx *a207              ; insert-zähler
55779         beq a55783             ; kein insert: ->
55781         dec *a207              ; insert-zähler erniedrigen
55783 a55783  bit a2026              ; auto-insert eingeschaltet?
55786         bpl a55797             ; nein: ->
55788         pha
55789         jsr a56782             ; ---> zeichen einfügen
55792         ldx #0
55794         stx *a207              ; insert-zähler rücksetzen
55796         pla
55797 a55797  jsr a57345             ; ---> zeichen auf bildschirm ausgeben
55800 a55800  cpy a2024              ; rechter rand erreicht?
55803         bcc a55817             ; nein: ->
55805         ldx *a205              ; kursorzeile
55807         cpx a2021              ; = unterste bildschirmzeile?
55810         bcc a55817             ; nein: ->
55812         bit a2025              ; scroll abgeschaltet?
55815         bmi a55840             ; ja: fertig ->
55817 a55817  jsr a55464             ; ---> kursorzeiger setzen
55820         jsr a57279             ; ---> kursor nach rechts
55823         bcc a55840
55825         jsr a57145             ; ---> getbit: zeilentabelle
55828         bcs a55839
55830         sec
55831         bit a2025              ; scroll abgeschaltet?
55834         bvs a55840
55836         jsr a55902             ; ---> leerzeile einfügen
55839 a55839  clc
55840 a55840  rts
55841
55841 ;------------------------------  S C R O L L   A U F
55841
55841 a55841  ldx *a205              ; kursorzeile
55843         cpx a2021              ; < unterste bildschirmzeile?
55846         bcc a55864             ; ja: ->
55848         bit a2025              ; scroll abgeschaltet?
55851         bpl a55860             ; nein: ->
55853         lda a2022              ; oberste bildschirmzeile
55856         sta *a205              ; = kursorzeile
55858         bcs a55866             ; immer ->
55860
55860 a55860  jsr a55945             ; ---> scroll auf
55863         clc
55864 a55864  inc *a205              ; kursorzeile
55866 a55866  jmp a55464             ; ---> kursorzeiger setzen
55869
55869 ;-----------------  B I L D S C H I R M Z E I L E   U M K O P I E R E N
55869
55869 a55869  lda a55298,x           ; l,adresse der zeile (x)
55872         sta *a169              ; l,zeiger in color-ram
55874         sta *a192              ; l,zeiger in zeichen-ram
55876         lda a55323,x           ; h,adresse der zeile (x)
55879         sta *a193              ; h,zeiger in zeichen-ram
55881         and #3
55883         ora #h,a2048
55885         sta *a170              ; h,zeiger in color-ram
```

```
55887 a55887  lda (a192),y          ; kode aus zeichen-ram
55889         sta (a200),y          ; in kursorzeile umspeichern
55891         lda (a169),y          ; kode aus color-ram
55893         sta (a234),y          ; in kursorzeile umspeichern
55895         cpy a2024             ; rechter rand erreicht?
55898         iny
55899         bcc a55887            ; nein: ->
55901         rts
55902
55902 ;------------------------------  L E E R Z E I L E  E I N F Ü G E N
55902
55902 a55902  ldx *a196             ; lstp: letzter zeilenwert
55904         bmi a55912            ; flag für 'ganze zeile' gesetzt: ->
55906         cpx *a205             ; lstp < kursorzeile?
55908         bcc a55912            ; ja: ->
55910         inc *a196             ; letzten zeilenwert erhöhen
55912 a55912  ldx a2021             ; zeilenzeiger auf unterste zeile setzen
55915 a55915  jsr a55466            ; ---> kursorzeiger auf zeile (x)
55918         ldy a2023             ; linker rand
55921         cpx *a205             ; kursorzeile erreicht?
55923         beq a55939            ; ja: fertig ->
55925         dex
55926         jsr a57147            ; ---> getbit: zeilentabelle
55929         inx
55930         jsr a57160            ; ---> clrbit: zeilentabelle
55933         dex
55934         jsr a55869            ; ---> zeile (x) nach (x+1) kopieren
55937         bcs a55915            ; immer ->
55939
55939 a55939  jsr a56055            ; ---> zeile (x) löschen
55942         jmp a57177            ; ---> setbit: zeilentabelle
55945
55945 a55945  ldx a2022             ; oberste bildschirmzeile
55948 a55948  inx
55949         jsr a57147            ; ---> getbit: zeilentabelle
55952         bcc a55966
55954         cpx a2021             ; unterste bildschirmzeile
55957         bcc a55948
55959         ldx a2022             ; oberste bildschirmzeile
55962         inx
55963         jsr a57162            ; ---> clrbit: zeilentabelle
55966 a55966  dec *a205             ; kursorzeile
55968         bit *a196             ; lstp: letzter zeilenwert
55970         bmi a55974
55972         dec *a196             ; lstp: letzter zeilenwert
55974 a55974  ldx a2022             ; oberste bildschirmzeile
55977         cpx *a254
55979         bcs a55983
55981         dec *a254
55983 a55983  jsr a56005            ; ---> ab zeile (x) aufwärts scrollen
55986         ldx a2022             ; oberste bildschirmzeile
55989         jsr a57147            ; ---> getbit: zeilentabelle
55992         php
55993         jsr a57162            ; ---> clrbit: zeilentabelle
55996         plp
55997         bcc a56004
55999         bit a2028             ; log-scroll-flag
56002         bmi a55945            ; gesetzt: ->
56004 a56004  rts
```

```
56005 a56005  jsr a55466              ; ---> kursorzeiger auf zeile (x)
56008         ldy a2023               ; linker rand
56011         cpx a2021               ; unterste bildschirmzeile
56014         bcs a56030              ; erreicht: ->
56016         inx
56017         jsr a57147              ; ---> getbit: zeilentabelle
56020         dex
56021         jsr a57160              ; ---> clrbit: zeilentabelle
56024         inx
56025         jsr a55869              ; ---> zeile (x) auf zeile (x-1) kopieren
56028         bcs a56005              ; immer ->
56030
56030 a56030  jsr a56055              ; ---> zeile (x) löschen
56033         lda #%01111111
56035         jsr a56176              ; ---> dekoder abfragen
56038         cmp #%11011111          ; C= - taste gedrückt?
56040         bne a56051              ; nein: ->
56042         ldy #0                  ; warteschleife ca. 0.5 s
56044 a56044  nop
56045         dex
56046         bne a56044
56048         dey
56049         bne a56044
56051 a56051  rts
56052
56052         nop
56053         nop
56054         nop
56055
56055 a56055  ldy a2023               ; linker rand
56058         jsr a57162              ; ---> clrbit: zeilentabelle
56061 a56061  jsr a55466              ; ---> kursorzeiger auf zeile (x)
56064         dey
56065 a56065  iny
56066         lda #'
56068         sta (a200),y
56070         lda a1339               ; color
56073         sta (a234),y            ; in color-ram
56075         cpy a2024               ; rechter rand erreicht?
56078         bne a56065              ; nein: ->
56080         rts
56081
56081 ;------------------------------   T A S T A T U R A B F R A G E
56081
56081 a56081  lda #0
56083         sta a1347               ; shflg: shift flag
56086         ldy #64
56088         sty *a198               ; offset in kode-tabelle
56090         jsr a56176              ; ---> dekoder abfragen
56093         tax
56094         cpx #255                ; taste gedrückt?
56096         bne a56101              ; ja: ->
56098         jmp a56321              ; --->
56101
56101 a56101  ldy #0                  ; offset
56103         lda #l,a57382           ; adresse der kodetabelle 1
56105         sta *a236               ; in zeiger bringen
56107         lda #h,a57382
56109         sta *a237
56111         lda #%11111110          ; für dekoder-eingang
```

```
56113 a56113  ldx #8                  ; bitzähler
56115         pha
56116 a56116  pla
56117         pha
56118         jsr a56176              ; ---> dekoder abfragen
56121         sta *a238               ; ergebnis merken
56123         pla
56124         pha
56125         jsr a56176              ; ---> dekoder nochmal abfragen
56128         cmp *a238               ; gleiches ergebnis?
56130         bne a56116              ; nein: abfrage wiederholen ->
56132 a56132  lsr a                   ; bit ins carry schieben
56133         bcs a56157              ; bit gesetzt ->
56135         pha                     ; a retten
56136         lda (a236),y            ; kode aus tabelle
56138         cmp #5                  ; gröer als 4?
56140         bcs a56154              ; ja: ->
56142         cmp #3                  ; stop-kode?
56144         beq a56154              ; ja: ->
56146         ora a1347               ; kode ins shift-flag odern
56149         sta a1347               ; 1 shift, 2 C=, 4 ctrl
56152         bpl a56156
56154
56154 a56154  sty *a198               ; offset in kodetabelle
56156 a56156  pla                     ; a wiederherstellen
56157 a56157  iny                     ; offset erhöhen
56158         cpy #65
56160         bcs a56170              ; alle tasten abgefragt: ->
56162         dex                     ; bitzähler dekrementieren
56163         bne a56132              ; wenn nicht 0, nächstes bit ->
56165         sec
56166         pla                     ; dekoder-eingang
56167         rol a                   ; nach links rotieren
56168         bne a56113              ; immer ->
56170
56170 a56170  pla                     ; stack korrigieren
56171         lda *a198               ; offset in kodetabelle
56173         jmp (a1349)             ; ---> keylog (a56186)
56176
56176 ;-------------------------------  D E K O D E R - A B F R A G E
56176
56176 a56176  sta a64816              ; tastaturdekoder eingang
56179         sta a65288
56182         lda a65288              ; tastaturdekoder ausgang
56185         rts
56186
56186 ;-------------------------------  K E Y L O G
56186
56186 a56186  lda a1347               ; shflag
56189         cmp #3                  ; shift und C= gedrückt?
56191         bne a56218              ; nein: ->
56193         lda a1351               ; mode: shift-C=-tasten gesperrt?
56196         bmi a56250              ; ja: ->
56198         lda a1348               ; wiederholungsverzögerung abgelaufen?
56201         bne a56250              ; nein: ->
56203         lda a65299
56206         eor #4
56208         sta a65299
56211         lda #8                  ; wiederholungsverzögerung setzen
56213         sta a1348
56216         bne a56250              ; immer ->
```

```
56218 a56218  asl a
56219         cmp #8                ; ctrl-taste?
56221         bcc a56239            ; nein: ->
56223         lda #6
56225         ldx a2039
56228         bne a56239
56230         ldx *a198             ; offset in kode-tabelle
56232         cpx #13               ; ctrl-s?
56234         bne a56239            ; nein: ->
56236         stx *a240             ; print-warteflag setzen
56238         rts
56239
56239 a56239  tax
56240         lda a57374,x          ; adresse der kodetabelle für x
56243         sta *a236             ; in zeiger bringen
56245         lda a57374+1,x
56248         sta *a237
56250 a56250  ldy *a198             ; offset
56252         lda (a236),y          ; kode aus tabelle
56254         tax
56255         cpy a2038             ; lstx: selbe taste wie beim letzten mal?
56258         beq a56267            ; ja: ->
56260         ldy #16
56262         sty a1346             ; delay: anlaufverzögerung setzen
56265         bne a56321            ; immer ->
56267
56267 a56267  and #127
56269         bit a1344             ; rptflg: wiederholungsflag gesetzt?
56272         bmi a56296            ; gesetzt: ->
56274         bvs a56363            ; auto-wiederholung abgeschaltet: ->
56276         cmp #127              ; taste gedrückt?
56278         beq a56321            ; nein: ->
56280         cmp #20               ; 'delete'?
56282         beq a56296            ; ja: ->
56284         cmp #'                ; leertaste?
56286         beq a56296            ; ja: ->
56288         cmp #29               ; 'kursor nach rechts'?
56290         beq a56296            ; ja: ->
56292         cmp #17               ; 'kursor nach unten'?
56294         bne a56363            ; nein: ->
56296 a56296  ldy a1346             ; delay: anlaufverzögerung
56299         beq a56306            ; abgelaufen: ->
56301         dec a1346             ; delay dekrementieren
56304         bne a56363            ; noch nicht 0: ->
56306 a56306  dec a1345             ; kount: normalverzügerung
56309         bne a56363            ; nicht abgelaufen: ->
56311         ldy #4
56313         sty a1345             ; neu initialisieren
56316         ldy *a239             ; ndx: tastenpuffer-index
56318         dey
56319         bpl a56363
56321 a56321  nop
56322         nop
56323         lsr a1348             ; shift-C= verzögerung
56326         ldy *a198             ; aktueller offset
56328         sty a2038             ; als lstx = letzter offset speichern
56331         cpx #255              ; noch eine taste gedrückt?
56333         beq a56363            ; nein: fertig ->
56335         txa                   ; tastenkode
56336         ldx #0
56338         stx *a240             ; print-warteflag rücksetzen
```

```
56340          ldx #7
56342 a56342   cmp a56385,x            ; funktionstaste?
56345          beq a56364              ; ja: ->
56347          dex
56348          bpl a56342
56350          ldx *a239               ; ndx: tastenpuffer-index
56352          cpx a1343               ; maximalwert erreicht?
56355          bcs a56363              ; ja: ->
56357          sta a1319,x             ; tastenkode in puffer
56360          inx                     ; index erhöhen
56361          stx *a239               ; ndx: tastenpuffer-index
56363 a56363   rts
56364
56364 a56364   lda a1375,x             ; länge des textes
56367          sta a1373               ; in zeichenzähler speichern
56370          lda #0                  ; zeiger in puffer initialisieren
56372 a56372   dex                    ; nummer der taste dekrementieren
56373          bmi a56381              ; fertig: ->
56375          clc
56376          adc a1375,x             ; alle vorausgehenden längen addieren
56379          bcc a56372
56381
56381 a56381   sta a1374               ; zeiger auf anfang des textes (x)
56384          rts
56385
56385 a56385   .by 133 137 134 138 135 139 136 140     ; funktionstasten-kodes
56393
56393 ;------------------------------   P R I N T
56393
56393 a56393   sta *a206               ; kode merken
56395          pha
56396          txa
56397          pha
56398          tya
56399          pha
56400 a56400   lda *a240               ; print-warteflag gesetzt durch ctrl-s?
56402          bne a56400              ; ja: warten ->
56404          sta *a199               ; crsw: übernahmeflag rücksetzen
56406          lda #h,a55751-1         ; adresse des print-abschlu
56408          pha                     ; auf stack legen
56409          lda #l,a55751-1
56411          pha
56412          ldy *a202               ; kursorspalte
56414          lda *a206               ; datenbyte
56416          cmp #13                 ; 'return'?
56418          beq a56460              ; ja: ->
56420          cmp #141                ; 'shift-return'?
56422          beq a56460              ; ja: ->
56424          ldx a2027               ; letzter ausgegebener kode
56427          cpx #27                 ; 'esc'?
56429          bne a56434              ; nein: ->
56431          jmp a56838              ; ---> esc-sequenz bearbeiten
56434
56434 a56434   tax                    ; datenbyte
56435          bmi a56457              ; shift-zeichen: ->
56437          cmp #32                 ; steuerkode?
56439          bcc a56487              ; ja: ->
56441          cmp #96                 ; druckbarer kode?
56443          bcc a56449              ; ja: ->
56445          and #%11011111          ; (96,..,127) in (64,..,95) wandeln
56447          bne a56451              ; immer ->
```

```
56449 a56449  and #%00111111          ; (64,..,95) in (0,..,31) wandeln
56451 a56451  jsr a55738              ; ---> quote-modus umschalten bei '"'
56454         jmp a55771              ; ---> zeichen auf bildschirm, abschlu
56457
56457 a56457  jmp a56647              ; ---> shift-zeichen bearbeiten
56460
56460 ;-------------------------------  ( S H I F T - ) R E T U R N
56460
56460 a56460  jsr a57237              ; ---> kursor ans ende der zeile
56463         inx                     ; zeilennummer
56464         jsr a57162              ; ---> clrbit: zeilentabelle
56467         ldy a2023               ; linker rand
56470         sty *a202               ; =: kursorzeiger
56472         jsr a55841              ; ---> neue zeile
56475
56475 ;-------------------------------  E S C - O :  F L A G S  L ö S C H E N
56475
56475 a56475  lda #0
56477         sta *a207               ; insert-zähler
56479         sta *a194               ; rvs-flag
56481         sta *a203               ; quote-modus
56483         sta a1340               ; blink-flag
56486         rts
56487
56487 ;-------------------------------  S T E U E R K O D E S
56487
56487 a56487  cmp #27                 ; 'esc'?
56489         beq a56569              ; ja: ->
56491         ldx *a207               ; insert-zähler aktiv?
56493         beq a56498              ; nein: ->
56495 a56495  jmp a55775              ; ---> zeichen rvs ausgeben, abschlu
56498
56498 a56498  cmp #20                 ; 'delete'?
56500         bne a56505              ; nein: ->
56502         jmp a56729              ; ---> delete
56505
56505 a56505  ldx *a203               ; quote-modus
56507         bne a56495              ; gesetzt: ->
56509         cmp #18                 ; 'rvs on'?
56511         bne a56515              ; nein: ->
56513         sta *a194               ; rvs-flag setzen
56515 a56515  cmp #19                 ; 'home'?
56517         bne a56530              ; nein: ->
56519         cmp a2027               ; zum zweiten mal 'home'?
56522         bne a56527              ; nein: ->
56524         jsr a56944              ; ---> vollen bildschirm herstellen
56527 a56527  jmp a55450              ; ---> kursor in home-position
56530
56530 a56530  cmp #29                 ; 'kursor nach rechts'?
56532         beq a56570              ; ja: ->
56534         cmp #17                 ; 'kursor nach unten'?
56536         beq a56576              ; ja: ->
56538         cmp #14                 ; 'text-modus'?
56540         beq a56615              ; ja: ->
56542         cmp #8                  ; 'shift-C= aus'?
56544         beq a56622              ; ja: ->
56546         cmp #9                  ; 'shift-C= ein'?
56548         beq a56629              ; ja: ->
56550 a56550  ldx #15
56552 a56552  cmp a57651,x            ; farbkode?
56555         beq a56561              ; ja: ->
56557         dex
56558         bpl a56552
56560         rts
```

```
56561 a56561  pha
56562         jsr a53130              ; ---> farbkode aus tabelle holen
56565         sta a1339               ; und speichern
56568         pla
56569 a56569  rts
56570
56570 ;-------------------------------  K U R S O R  N A C H  R E C H T S
56570
56570 a56570  jsr a57279              ; ---> kursor rechts
56573         bcs a56579              ; neue zeile: ->
56575         rts
56576
56576 ;-------------------------------  K U R S O R  N A C H  U N T E N
56576
56576 a56576  jsr a55841              ; ---> zeilennr erhöhen, ggf. scroll
56579 a56579  jsr a57145              ; ---> getbit: zeilentabelle
56582         bcs a56587              ; bit gesetzt: ->
56584         sec
56585         ror *a196               ; flag für 'ganze zeile' setzen
56587 a56587  clc
56588 a56588  rts
56589
56589 ;-------------------------------  K U R S O R  N A C H  O B E N
56589
56589 a56589  ldx a2022               ; erste bildfensterzeile
56592         cpx *a205               ; = kursorzeile?
56594         bcs a56588              ; ja: ->
56596 a56596  jsr a56579              ; ---> getbit: zeilentabelle
56599         dec *a205               ; zeilennummer erniedrigen
56601         jmp a55464              ; ---> kursorzeiger setzen
56604
56604 ;-------------------------------  K U R S O R  N A C H  L I N K S
56604
56604 a56604  jsr a57300              ; ---> kursor links
56607         bcs a56588              ; kursor war in home-position: ->
56609         bne a56587              ; kursor nicht am rechten rand: ->
56611         inc *a205               ; zeilennummer erhöhen
56613         bne a56596              ; immer ->
56615
56615 ;-------------------------------  T E X T - M O D U S
56615
56615 a56615  lda a65299
56618         ora #4
56620         bne a56643
56622
56622 ;-------------------------------  S H I F T - C=  A U S
56622
56622 a56622  lda #128
56624         ora a1351
56627         bmi a56634
56629
56629 ;-------------------------------  S H I F T - C=  E I N
56629
56629 a56629  lda #127
56631         and a1351
56634 a56634  sta a1351
56637         rts
```

```
56638 ;------------------------------  G R A P H I K - M O D U S
56638
56638 a56638  lda a65299
56641         and #251
56643 a56643  sta a65299
56646         rts
56647
56647 ;------------------------------  S H I F T - Z E I C H E N
56647
56647 a56647  and #127                ; bit 7 löschen
56649         cmp #127                ; war es kode 255?
56651         bne a56655              ; nein: ->
56653         lda #94                 ; durch kode 94 (= 222-128) ersetzen
56655 a56655  cmp #32                 ; steuerkode?
56657         bcc a56662              ; ja: ->
56659         jmp a55769              ; ---> zeichen ausgeben, abschlu
56662
56662 a56662  ldx *a203               ; quote-modus gesetzt?
56664         beq a56671              ; nein: ->
56666 a56666  ora #64                 ; bit 6 setzen
56668         jmp a55775              ; ---> zeichen rvs ausgeben, abschlu
56671
56671 a56671  cmp #20                 ; 'insert'?
56673         beq a56782              ; ja: ->
56675         ldx *a207               ; insert-zähler aktiv?
56677         bne a56666              ; ja: ->
56679         cmp #17                 ; 'kursor nach oben'?
56681         beq a56589              ; ja: ->
56683         cmp #18                 ; 'rvs off'?
56685         bne a56691              ; nein: ->
56687         lda #0
56689         sta *a194               ; rvs-flag löschen
56691 a56691  cmp #29                 ; 'kursor nach links'?
56693         beq a56604              ; ja: ->
56695         cmp #19                 ; 'clear'?
56697         bne a56702              ; nein: ->
56699         jmp a55435              ; ---> bildschirm löschen
56702
56702 a56702  cmp #2                  ; 'blinken ein'?
56704         bne a56711              ; nein: ->
56706         lda #128
56708         sta a1340               ; blink-flag setzen
56711 a56711  cmp #4                  ; 'blinken aus'?
56713         bne a56720              ; nein: ->
56715         lda #0
56717         sta a1340               ; blink-flag rücksetzen
56720 a56720  cmp #14                 ; 'graphik-modus'?
56722         beq a56638              ; ja: ->
56724         ora #128                ; bit 7 wieder setzen
56726         jmp a56550              ; ---> wenn farbkode, ausgeben
56729
56729 ;------------------------------  D E L E T E
56729
56729 a56729  jsr a56604              ; ---> kursor nach links
56732         jsr a57334              ; ---> kursorposition retten
56735         bcs a56753              ; kursor in home-position: ->
56737 a56737  cpy a2024               ; kursor am rechten rand?
56740         bcc a56764              ; nein: ->
56742         ldx *a205               ; kursorzeile
56744         inx                     ; vermindern
```

```
56745          jsr a57147              ; ---> getbit: zeilentabelle
56748          bcs a56764              ; zeilenbit gesetzt: ->
56750          jsr a57343              ; ---> blank ausgeben
56753 a56753   lda *a204
56755          sta *a202               ; kursorspalte wiederherstellen
56757          lda *a254
56759          sta *a205               ; kursorzeile wiederherstellen
56761          jmp a55464              ; ---> kursorzeiger setzen
56764
56764 a56764   jsr a57279              ; ---> kursor nach rechts
56767          jsr a57135              ; ---> zeichen vom bildschirm lesen
56770          jsr a57300              ; ---> kursor nach links
56773          jsr a57361              ; ---> zeichen auf bildschirm schreiben
56776          jsr a57279              ; ---> kursor nach rechts
56779          jmp a56737              ; ---> zurück zum schleifenanfang
56782
56782 ;------------------------------  I N S E R T
56782
56782 a56782   jsr a57334              ; ---> kursorposition retten
56785          jsr a57237              ; ---> kursor an zeilenende
56788          cpx *a254               ; insert vor zeilenende?
56790          bne a56794
56792          cpy *a204
56794 a56794   bcc a56829              ; nein: ->
56796          jsr a55800              ; ---> scroll ab, wenn zeilenende
56799          bcs a56835              ; scroll gesperrt: ->
56801 a56801   jsr a57300              ; ---> kursor nach links
56804          jsr a57135              ; ---> zeichen vom bildschirm lesen
56807          jsr a57279              ; ---> kursor nach rechts
56810          jsr a57361              ; ---> zeichen auf bildschirm schreiben
56813          jsr a57300              ; ---> kursor nach links
56816          ldx *a205               ; ausgangsposition erreicht?
56818          cpx *a254
56820          bne a56801
56822          cpy *a204
56824          bne a56801              ; nein: ->
56826          jsr a57343              ; ---> blank schreiben
56829 a56829   inc *a207               ; insert-zähler inkrementieren
56831          bne a56835
56833          dec *a207               ; jedoch nicht über 255
56835 a56835   jmp a56753              ; ---> kursorposition wiederherstellen
56838
56838 ;------------------------------  E S C - S E Q U E N Z
56838
56838 a56838   and #127                ; bit 7 löschen
56840          sec                     ; kode
56841          sbc #65                 ; in bereich (0,...,22) verschieben
56843          cmp #23                 ; bereich überschritten?
56845          bcs a56857              ; ja: fertig ->
56847          asl a                   ; offset in tabelle erzeugen
56848          tax
56849          lda a56858+1,x          ; h,routinenadresse
56852          pha
56853          lda a56858,x            ; l,...
56856          pha
56857 a56857   rts                     ; routine aufrufen
```

```
56858 a56858  .si a57129-1          ;  esc-a   auto-insert ein
56860         .si a56928-1          ;       b  bildfenster ende
56862         .si a57126-1          ;       c  auto-insert aus
56864         .si a56992-1          ;       d  zeile löschen
56866         .si a56857-1          ;       e  rts
56868         .si a56857-1          ;       f  rts
56870         .si a56857-1          ;       g  rts
56872         .si a56857-1          ;       h  rts
56874         .si a56971-1          ;       i  zeile einfügen
56876         .si a57218-1          ;       j  kursor an zeilenanfang
56878         .si a57237-1          ;       k  kursor an zeilenende
56880         .si a57117-1          ;       l  scroll freigeben
56882         .si a57120-1          ;       m  scroll sperren
56884         .si a55432-1          ;       n  normaler bildschirm
56886         .si a56475-1          ;       o  flags löschen
56888         .si a57057-1          ;       p  zeile links löschen
56890         .si a57035-1          ;       q  zeile rechts löschen
56892         .si a56904-1          ;       r  rand um bildschirm
56894         .si a56857-1          ;       s  rts
56896         .si a56926-1          ;       t  bildfenster anfang
56898         .si a56857-1          ;       u  rts
56900         .si a57078-1          ;       v  scroll auf
56902         .si a57092-1          ;       w  scroll ab
56904
56904 ;----------------  E S C - R :  B I L D F E N S T E R  M I T  R A N D
56904
56904 a56904  jsr a56944            ; ---> vollen bildschirm herstellen
56907         jsr a55435            ; ---> bildschirm löschen
56910         lda #1
56912         tax
56913         jsr a56954            ; ---> bildfenster anfang
56916         lda #23
56918         ldx #38
56920         jsr a56935            ; ---> bildfenster ende
56923         jmp a55450            ; ---> home
56926
56926 ;--------------------  E S C - T :  B I L D F E N S T E R  A N F A N G
56926
56926 a56926  clc
56927         .by 36
56928
56928 ;-----------------------  E S C - B :  B I L D F E N S T E R  E N D E
56928
56928 a56928  sec
56929         ldx *a202             ; kursorspalte
56931         lda *a205             ; kursorzeile
56933         bcc a56954
56935 a56935  sta a2021             ; unterer rand
56938         stx a2024             ; rechter rand
56941         jmp a56960            ; ---> zeilentabelle löschen
56944
56944 ;-------------  V O L L E N  B I L D S C H I R M  H E R S T E L L E N
56944
56944 a56944  lda #24                ; gröte zeilennummer
56946         ldx #39                ; gröte spaltennummer
56948         jsr a56935             ; ---> =: unterer und rechter rand
56951         lda #0                 ; kleinste zeilennummer
56953         tax                    ; kleinste spaltennummer
56954 a56954  sta a2022              ; oberer rand
56957         stx a2023              ; linker rand
```

```
56960 a56960  lda #0                  ; zeilentabelle löschen
56962         ldx #4
56964 a56964  sta a2030-1,x
56967         dex
56968         bne a56964
56970         rts
56971
56971 ;-----------------------------  E S C - I :  Z E I L E  E I N F Ü G E N
56971
56971 a56971  jsr a55902              ; ---> scroll ab
56974         jsr a55457              ; ---> kursorzeiger setzen
56977         inx                     ; zeilenzeiger erhöhen
56978         jsr a57147              ; ---> getbit: zeilentabelle
56981         php
56982         jsr a57158              ; ---> putbit: zeilentabelle
56985         plp
56986         bcs a56991              ; zeilenbit war gesetzt: ->
56988         sec
56989         ror *a196               ; flag für 'ganze zeile' setzen
56991 a56991  rts
56992
56992 ;-----------------------------  E S C - D :  Z E I L E  L ö S C H E N
56992
56992 a56992  jsr a57223              ; ---> kursor an zeilenanfang
56995         lda a2022               ; erste bildfensterzeile
56998         pha                     ; merken
56999         lda *a205               ; kursorzeile
57001         sta a2022               ; =: erste bildfensterzeile
57004         lda a2028               ; scroll flag
57007         pha                     ; merken
57008         lda #128                ; log scroll
57010         sta a2028               ; einschalten
57013         jsr a55966              ; ---> scroll auf
57016         pla
57017         sta a2028               ; scroll flag wiederherstellen
57020         lda a2022               ; gelöschte zeile
57023         sta *a205               ; =: kursorzeile
57025         pla
57026         sta a2022               ; erste bildfensterzeile wiederherstellen
57029         sec
57030         ror *a196               ; flag für 'ganze zeile' setzen
57032         jmp a55457              ; ---> kursor zeiger setzen
57035
57035 ;----------------  E S C - Q :  Z E I L E  R E C H T S  L ö S C H E N
57035
57035 a57035  jsr a57334              ; ---> kursorposition retten
57038 a57038  jsr a56061              ; ---> zeile ab kursor löschen
57041         inc *a205               ; kursorzeile erhöhen
57043         jsr a55464              ; ---> kursorzeiger setzen
57046         ldy a2023               ; linker rand
57049         jsr a57145              ; ---> getbit: zeilentabelle
57052         bcs a57038              ; zeilenbit gesetzt: ->
57054 a57054  jmp a56753              ; ---> kursorposition wiederherstellen
57057
57057 ;------------------  E S C - P :  Z E I L E  L I N K S  L ö S C H E N
57057
57057 a57057  jsr a57334              ; ---> kursorposition retten
57060 a57060  jsr a57343              ; ---> blank ausgeben
57063         cpy a2023               ; linker rand erreicht?
57066         bne a57073              ; nein: ->
57068         jsr a57145              ; ---> getbit: zeilentabelle
57071         bcc a57054              ; zeilenbit nicht gesetzt: ->
57073 a57073  jsr a57300              ; ---> kursor links
57076         bcc a57060              ; immer ->
```

```
57078 ;------------------------------  E S C - V :  S C R O L L  A U F
57078
57078 a57078  jsr a57334              ; ---> kursorposition retten
57081         txa
57082         pha
57083         jsr a55945              ; ---> scroll auf
57086         pla
57087         sta *a254               ; kursorzeile
57089         jmp a57054              ; ---> kursorposition wiederherstellen
57092
57092 ;------------------------------  E S C - W :  S C R O L L  A B
57092
57092 a57092  jsr a57334              ; ---> kursorposition retten
57095         jsr a57145              ; ---> getbit: zeilentabelle
57098         bcs a57103              ; zeilenbit gesetzt: ->
57100         sec
57101         ror *a196               ; flag für 'ganze zeile' setzen
57103 a57103  lda a2022               ; oberer rand
57106         sta *a205               ; =: kursorzeile
57108         jsr a55902              ; ---> scroll ab
57111         jsr a57162              ; ---> clrbit: zeilentabelle
57114         jmp a57054              ; ---> kursorposition wiederherstellen
57117
57117 ;-------------------------  E S C - L :  S C R O L L  F R E I G E B E N
57117
57117 a57117  lda #0
57119         .by 44
57120
57120 ;----------------------------  E S C - M :  S C R O L L  S P E R R E N
57120
57120 a57120  lda #128
57122         sta a2025
57125         rts
57126
57126 ;--------------------------  E S C - C :  A U T O - I N S E R T  A U S
57126
57126 a57126  lda #0
57128         .by 44
57129
57129 ;--------------------------  E S C - A :  A U T O - I N S E R T  E I N
57129
57129 a57129  lda #255
57131         sta a2026
57134         rts
57135
57135 ;-------------  Z E I C H E N  V O M  B I L D S C H I R M  L E S E N
57135
57135 a57135  ldy *a202              ; kursorzeiger
57137         lda (a234),y            ; farbkode
57139         sta a2029
57142         lda (a200),y            ; zeichen
57144         rts
57145
57145 ;------------------------------  Z E I L E N T A B E L L E
57145
57145 ;                               g e t b i t
57145
57145 a57145  ldx *a205               ; kursorzeile
57147 a57147  jsr a57190              ; ---> bitposition erzeugen
57150         and a2030,x             ; mit bit-tabelle maskieren
57153         cmp #1                  ; gesetztes bit setzt carry
57155         jmp a57173              ; ---> ausgang
```

```
57158 ;                                 p u t b i t
57158
57158 a57158  ldx *a205                 ; kursorzeile
57160 a57160  bcs a57177                ; cs: setbit, cc: clrbit
57162
57162 ;                                 c l r b i t
57162
57162 a57162  jsr a57190                ; ---> bitposition erzeugen
57165         eor #255                  ; alle bits flippen
57167         and a2030,x               ; mit bit-tabelle maskieren
57170 a57170  sta a2030,x               ; in tabelle zurückspeichern
57173 a57173  ldx a745                  ; byte-offset in tabelle
57176         rts
57177
57177 ;                                 s e t b i t
57177
57177 a57177  bit a2025                 ; scroll freigabe
57180         bvs a57147                ; bit 6 gesetzt: ->
57182         jsr a57190                ; ---> bitposition erzeugen
57185         ora a2030,x               ; mit bit-tabelle odern
57188         bne a57170                ; immer ->
57190
57190 ;                                 b i t p o s i t i o n   e r z e u g e n
57190
57190 a57190  stx a745                  ; kursorzeile
57193         txa
57194         and #7                    ; modulo 8
57196         tax
57197         lda a57210,x              ; bitwert holen
57200         pha
57201         lda a745                  ; kursorzeile
57204         lsr a                     ; dividiert durch 8
57205         lsr a
57206         lsr a
57207         tax                       ; byte offset in tabelle
57208         pla                       ; bitwert
57209         rts
57210
57210 a57210  .by 128 64 32 16 8 4 2 1         ; bitwerte
57218 ;
57218 ;------------  E S C - J :  K U R S O R  A N  Z E I L E N A N F A N G
57218
57218 a57218  ldy a2023                 ; linker rand
57221         sty *a202                 ; =: kursorspalte
57223 a57223  jsr a57145                ; ---> getbit: zeilentabelle
57226         bcc a57234                ; zeilenbit nicht gesetzt: ->
57228         dec *a205                 ; kursorzeile erniedrigen
57230         bpl a57223                ; war noch nicht die erste zeile: ->
57232         inc *a205                 ; auf erste zeile erhöhen
57234 a57234  jmp a55464                ; ---> kursorzeiger setzen
57237
57237 ;----------------  E S C - K :  K U R S O R  A N  Z E I L E N E N D E
57237
57237 a57237  inc *a205                 ; actuelle zeile erhöhen
57239         jsr a57145                ; ---> getbit: zeilentabelle
57242         bcs a57237                ; zeilenbit gesetzt: ->
57244         dec *a205                 ; kursorzeile erniedrigen
57246         jsr a55464                ; ---> kursorzeiger setzen
57249         ldy a2024                 ; rechter rand
57252         sty *a202                 ; =: kursorspalte
```

```
57254 a57254  jsr a57135            ; ---> zeichen vom bildschirm
57257         cmp #'                ; leerstelle?
57259         bne a57276            ; nein: ->
57261         cpy a2023             ; linker rand erreicht?
57264         bne a57271            ; nein: ->
57266         jsr a57145            ; ---> getbit: zeilentabelle
57269         bcc a57276            ; zeilenbit nicht gesetzt: ->
57271 a57271  jsr a57300            ; ---> kursor links
57274         bcc a57254            ; kursor nicht in home-position: ->
57276 a57276  sty *a195             ; position des letzten zeichens
57278         rts
57279
57279 ;------------------------------  K U R S O R  R E C H T S
57279
57279 a57279  pha
57280         ldy *a202             ; kursorspalte
57282         cpy a2024             ; am rechten rand?
57285         bcc a57295            ; nein: ->
57287         jsr a55841            ; ---> neue zeile
57290         ldy a2023             ; linker rand
57293         dey
57294         sec
57295 a57295  iny
57296         sty *a202             ; kursorspalte
57298         pla
57299         rts
57300
57300 ;------------------------------  K U R S O R  L I N K S
57300
57300 a57300  ldy *a202             ; kursorspalte
57302         dey
57303         bmi a57310            ; kursor war in spalte 0: ->
57305         cpy a2023             ; linker rand erreicht?
57308         bcs a57327            ; nein: ->
57310 a57310  ldy a2022             ; oberer rand
57313         cpy *a205             ; = kursorzeile?
57315         bcs a57333            ; ja: fertig ->
57317         dec *a205             ; kursorzeile erniedrigen
57319         pha
57320         jsr a55464            ; ---> kursorzeiger setzen
57323         pla
57324         ldy a2024             ; rechter rand
57327 a57327  sty *a202             ; =: kursorspalte
57329         cpy a2024             ; rechter rand
57332         clc
57333 a57333  rts
57334
57334 ;---------------------------  K U R S O R P O S I T I O N  R E T T E N
57334
57334 a57334  ldy *a202             ; kursorspalte
57336         sty *a204
57338         ldx *a205             ; kursorzeile
57340         stx *a254
57342         rts
```

```
57343 ;-----  Z E I C H E N  A U F  B I L D S C H I R M  S C H R E I B E N
57343
57343 a57343  lda #32
57345 a57345  ldy *a202
57347         sta (a200),y
57349         jsr a55476              ; ---> zeiger in color-ram setzen
57352         lda a1339               ; color
57355         ora a1340
57358         sta (a234),y
57360         rts
57361
57361 a57361  ldy *a202
57363         sta (a200),y
57365         jsr a55476              ; ---> zeiger in color-ram setzen
57368         lda a2029
57371         sta (a234),y
57373         rts
57374
57374 ;------------------------------  T A S T E N K O D E - T A B E L L E N
57374
57374 a57374  .si a57382              ; kode-tabelle 1: taste allein
57376         .si a57447              ;              2: shift + taste
57378         .si a57512              ;              3: C= + taste
57380         .si a57577              ;              4: ctrl + taste
57382
57382 a57382  .by 20 13 92 140 133 137 134 64 51 87
57392         .by 65 52 90 83 69 1 53 82 68 54
57402         .by 67 70 84 88 55 89 71 56 66 72
57412         .by 85 86 57 73 74 48 77 75 79 78
57422         .by 17 80 76 145 46 58 45 44 157 42
57432         .by 59 29 27 61 43 47 49 19 4 50
57442         .by 32 2 81 3 255
57447
57447 a57447  .by 148 141 169 136 138 135 139 186 35 215
57457         .by 193 36 218 211 197 1 37 210 196 38
57467         .by 195 198 212 216 39 217 199 40 194 200
57477         .by 213 214 41 201 202 94 205 203 207 206
57487         .by 17 208 204 145 62 91 221 60 157 192
57497         .by 93 29 27 95 219 63 33 147 4 34
57507         .by 160 2 209 131 255
57512
57512 a57512  .by 148 141 168 136 138 135 139 164 150 179
57522         .by 176 151 173 174 177 1 152 178 172 153
57532         .by 188 187 163 189 154 183 165 155 191 180
57542         .by 184 190 41 162 181 48 167 161 185 170
57552         .by 17 175 182 145 62 91 220 60 157 223
57562         .by 93 29 27 222 166 63 129 147 4 149
57572         .by 160 2 171 131 255
57577
57577 a57577  .by 255 255 28 255 255 255 255 255 28 23
57587         .by 1 159 26 19 5 255 156 18 4 30
57597         .by 3 6 20 24 31 25 7 158 2 8
57607         .by 21 22 18 9 10 146 13 11 15 14
57617         .by 255 16 12 255 132 27 255 130 255 255
57627         .by 29 255 27 6 255 255 144 255 255 5
57637         .by 255 255 17 255 255
57642
57642 ;------------------------------  R U N - T E X T
57642
57642 a57642  .by 'dL"*' 13 'run' 13
```

```
57651 ;------------------------------  F A R B - T A B E L L E N
57651
57651 a57651  .by 144 5 28 159 156 30 31 158 129 149  ; farbtasten-kodes
57661         .by 150 151 152 153 154 155
57667
57667 a57667  .by 0 113 50 99 68 53 70 119 72 41      ; farbkodes
57677         .by 90 107 92 109 46 95
57683
57683 ;------------------------------  T A L K
57683
57683 a57683  ora #%01000000          ; talk-bit
57685         .by 44
57686
57686 ;------------------------------  L I S T E N
57686
57686 a57686  ora #%00100000          ; listen-bit
57688 a57688  pha
57689         bit *a148               ; c3p0: byte im puffer?
57691         bpl a57703              ; nein: ->
57693         sec
57694         ror *a166
57696         jsr a57729              ; ---> byte ausgeben
57699         lsr *a148               ; pufferflag rücksetzen
57701         lsr *a166
57703 a57703  pla                     ; geräteadresse mit talk-/listen-bit
57704         sta *a149               ; bus-ausgabepuffer
57706         sei
57707         jsr a58054              ; ---> data 0 ausgeben
57710         jsr a58047              ; ---> clock 1 ausgeben
57713         lda *a1
57715         ora #%00000100
57717         sta *a1
57719 a57719  sei
57720         jsr a58047              ; ---> clock 1 ausgeben
57723         jsr a58054              ; ---> data 0 ausgeben
57726         jsr a58076              ; ---> 1 ms warten
57729
57729 ;------------------------------  B Y T E  A U S G E B E N
57729
57729 a57729  sei
57730         jsr a58054              ; ---> data 0 ausgeben
57733         jsr a58068              ; ---> data eingabe
57736         bcs a57833              ; data 1 empfangen: device not present ->
57738         jsr a58040              ; ---> clock 0 ausgeben
57741         bit *a166
57743         bpl a57759
57745 a57745  jsr a58068              ; ---> data eingabe
57748         bcc a57745              ; data 0: warten ->
57750 a57750  lda *a1                 ; eingabe
57752         cmp *a1                 ; verändert?
57754         bne a57750              ; ja: warten ->
57756         asl a                   ; data 1 empfangen?
57757         bcs a57750              ; ja: warten ->
57759 a57759  jsr a58068              ; ---> data eingabe
57762         bcc a57759              ; data 0: warten ->
57764         jsr a58047              ; ---> clock 1 ausgeben
57767         lda #8
57769         sta *a170               ; bitzähler setzen
```

```
57771 a57771  jsr a58068              ; ---> data eingabe
57774         bcc a57838              ; data 0 empfangen: timeout ->
57776         ror *a149               ; datenbyte ins carry rotieren
57778         bcs a57785              ; bit gesetzt: ->
57780         jsr a58061              ; ---> data 1 ausgeben
57783         bne a57788              ; immer ->
57785
57785 a57785  jsr a58054              ; ---> data 0 ausgeben
57788 a57788  jsr a58129              ; ---> 20 us warten
57791         jsr a58040              ; ---> clock 0 ausgeben
57794         jsr a58129              ; ---> 20 us warten
57797         lda *a1
57799         and #%11111110          ; data 0
57801         ora #%00000010          ; clock 1
57803         sta *a1                 ; ausgeben
57805         dec *a170               ; bitzähler dekrementieren
57807         bne a57771              ; noch bits auszugeben: ->
57809         txa
57810         pha
57811         ldx #120                ; zeitschleife ca. 1 ms
57813 a57813  lda *a1
57815         cmp *a1                 ; stabilität abwarten
57817         bne a57813
57819         asl a                   ; data 0 empfangen?
57820         bcc a57829              ; ja: ok ->
57822         dex                     ; wartezeit vermindern
57823         bne a57813              ; zeit noch nicht abgelaufen: warten ->
57825         pla
57826         tax
57827         bcs a57838              ; timout ->
57829
57829 a57829  pla
57830         tax
57831         cli
57832         rts
57833
57833 a57833  lda #128                ; 'device not present'
57835         jmp a57840              ; --->
57838
57838 a57838  lda #3                  ; 'timeout'
57840 a57840  jsr a62494              ; ---> in status bringen
57843         cli
57844         clc
57845         bcc a57922
57847
57847 ;-------------------------------  S E C O N D
57847
57847 a57847  sta *a149               ; sekundäradresse in bus-puffer
57849         jsr a57719              ; ---> byte ausgeben
57852 a57852  lda *a1
57854         and #%11111011          ; 'atn'
57856         sta *a1                 ; ausgeben
57858         rts
57859
57859 ;-------------------------------  T K S A
57859
57859 a57859  sta *a149               ; sekundäradresse in bus-puffer
57861         jsr a57719              ; ---> byte ausgeben
57864         bit *a144               ; status
57866         bmi a57922              ; device not present: ->
```

```
57868 a57868  sei
57869         jsr a58061              ; ---> data 1 ausgeben
57872         jsr a57852              ; ---> 'atn' ausgeben
57875         jsr a58040              ; ---> clock 0 ausgeben
57878 a57878  jsr a58068              ; ---> data eingabe
57881         bmi a57878              ; data 1 empfangen: warten ->
57883         cli
57884         rts
57885
57885 ;------------------------------  C I O U T
57885
57885 a57885  bit *a148               ; byte im puffer?
57887         bmi a57894              ; ja: ->
57889         sec
57890         ror *a148               ; pufferflag setzen
57892         bne a57899              ; immer ->
57894
57894 a57894  pha                     ; datenbyte merken
57895         jsr a57729              ; ---> byte aus puffer ausgeben
57898         pla                     ; datenbyte
57899 a57899  sta *a149               ; in puffer
57901         clc
57902         rts
57903
57903 ;------------------------------  U N T A L K
57903
57903 a57903  sei
57904         jsr a58047              ; ---> clock 1 ausgeben
57907         lda *a1
57909         ora #%00000100          ; 'atn'
57911         sta *a1                 ; ausgeben
57913         lda #95                 ; 'untalk'
57915         bne a57919
57917
57917 ;------------------------------  U N L I S T E N
57917
57917 a57917  lda #63                 ; 'unlisten'
57919 a57919  jsr a57688              ; ---> talk/listen
57922 a57922  jsr a57852              ; ---> 'atn' ausgeben
57925 a57925  txa
57926         ldx #20                 ; zeitschleife ca. 50 us
57928 a57928  dex
57929         bne a57928              ; warten ->
57931         tax
57932         jsr a58040              ; ---> clock 0 ausgeben
57935         jmp a58054              ; ---> data 0 ausgeben
57938
57938 ;--------------------------  A C P T R :  B Y T E  E M P F A N G E N
57938
57938 a57938  sei
57939         lda #0
57941         sta *a170               ; endeflag rücksetzen
57943         jsr a58040              ; ---> clock 0 ausgeben
57946         txa
57947         pha
57948 a57948  jsr a58068              ; ---> data eingabe
57951         bpl a57948              ; data 0 empfangen: warten ->
57953 a57953  ldx #32                 ; zeitschleife ca. 250 us
57955         jsr a58054              ; ---> data 0 ausgeben
```

```
57958 a57958  lda *a1
57960         cmp *a1
57962         bne a57958              ; stabilisierung abwarten
57964         asl a
57965         bpl a57998              ; data 0 empfangen: ->
57967         dex                     ; wartezeit vermindern
57968         bne a57958              ; zeit noch nicht abgelaufen: ->
57970         lda *a170               ; endeflag gesetzt?
57972         beq a57981              ; nein: ->
57974         pla
57975         tax
57976         lda #2                  ; 'read timeout'
57978         jmp a57840              ; ---> in status
57981
57981 a57981  jsr a58061              ; ---> data 1 ausgeben
57984         ldx #64                 ; zeitschleife ca. 160 us
57986 a57986  dex
57987         bne a57986              ; warten ->
57989         lda #%01000000          ; eoi-bit
57991         jsr a62494              ; ---> in status
57994         inc *a170               ; endeflag setzen
57996         bne a57953              ; immer ->
57998
57998 a57998  ldx #8                   ; bitzähler
58000 a58000  lda *a1
58002         asl a                   ; clock 1 empfangen?
58003         bpl a58000              ; nein: warten ->
58005         ror *a168               ; data in eingaberegister rotieren
58007 a58007  lda *a1                 ; dataport
58009         cmp *a1
58011         bne a58007              ; stabilisierung abwarten
58013         asl a
58014         bmi a58007              ; clock 1 empfangen: warten ->
58016         dex                     ; bitzähler erniedrigen
58017         bne a58000              ; noch nicht 0: ->
58019         stx *a17                ; eingabeflag
58021         pla
58022         tax
58023         jsr a58061              ; ---> data 1 ausgeben
58026         lda #64                 ; ??? sinnlos!
58028         bit *a144               ; status
58030         bvc a58035              ; eoi-bit nicht gesetzt: ->
58032         jsr a57925              ; ---> nach 50 us data 0, clock 0 ausg
58035 a58035  lda *a168               ; empfangenes datenbyte
58037         cli
58038         clc
58039         rts
58040
58040 ;------------------------------  B U S - A U S G A B E
58040
58040 a58040  lda *a1
58042         and #%11111101          ; clock 0
58044         sta *a1
58046         rts
58047
58047 a58047  lda *a1
58049         ora #%00000010          ; clock 1
58051         sta *a1
58053         rts
```

```
58054 a58054  lda *a1
58056         and #%11111110          ; data 0
58058         sta *a1
58060         rts
58061
58061 a58061  lda *a1
58063         ora #%00000001          ; data 1
58065         sta *a1
58067         rts
58068
58068 ;------------------------------  B U S - E M P F A N G
58068
58068 a58068  lda *a1                 ; data-port
58070         cmp *a1                 ; stabil?
58072         bne a58068              ; nein: warten ->
58074         asl a                   ; carry := data, bit 7 := clock
58075         rts
58076
58076 ;------------------------------  Z E I T S C H L E I F E N
58076
58076 a58076  jsr a58104              ; ---> timer setzen und starten (1 ms)
58079         lda #%00010000
58081 a58081  bit a65289              ; timer abgelaufen?
58084         beq a58081              ; nein: warten ->
58086         sta a65289
58089         rts
58090
58090 a58090  jsr a58108              ; ---> timer setzen und starten (16 ms)
58093         lda #%00010000
58095 a58095  bit a65289              ; timer abgelaufen?
58098         beq a58095              ; nein: warten ->
58100         sta a65289
58103         rts
58104
58104 a58104  lda #4                  ; 4 * 256 zyklen
58106         bne a58110
58108
58108 a58108  lda #64                 ; 64 * 256 zyklen
58110 a58110  php
58111         pha
58112         sei
58113         lda #0
58115         sta a65282              ; l,timer
58118         pla
58119         sta a65283              ; h,...
58122         lda #%00010000
58124         sta a65289              ; timer starten
58127         plp
58128         rts
58129
58129 a58129  txa
58130         ldx #5                  ; wartezeit 5*5+12 zyklen = ca. 18 us
58132 a58132  dex
58133         bne a58132
58135         tax
58136         rts
```

```
58137 ;------------------------------  P L A Y - T A S T E  P R Ü F E N
58137
58137 a58137  sec                     ; einsprung bei save
58138         .by 36
58139 a58139  clc                     ; einsprung bei load
58140         lda a64784              ; play-taste am recorder
58143         and #4
58145         beq a58210              ; gedrückt: ->
58147         php
58148         jsr a64472              ; ---> meldung ausgeben
58151         .by 13 'press play ' 0
58164         plp
58165         bcc a58180              ; load: ->
58167         jsr a64472              ; ---> meldung ausgeben
58170         .by '& record ' 0
58180 a58180  jsr a64472              ; ---> meldung ausgeben
58183         .by 'on tape' 0
58191 a58191  jsr a64459              ; ---> stoptaste gedrückt?
58194         bcs a58211              ; ja: ->
58196         lda a64784              ; play-taste am recorder
58199         and #%00000100          ; gedrückt?
58201         bne a58191              ; nein: warten ->
58203         jsr a64472              ; ---> meldung ausgeben
58206         .by 13 'ok' 0
58210 a58210  clc
58211 a58211  rts
58212
58212 ;-------------------  K A S S E T T E N P O R T  E I N S C H A L T E N
58212
58212 a58212  sei
58213         lda a65286
58216         and #%11101111
58218         sta a65286
58221         lda a65290
58224         and #%11111101
58226         ora #%00001000
58228         sta a65290
58231         rts
58232
58232 ;-------------------  K A S S E T T E N P O R T  A U S S C H A L T E N
58232
58232 a58232  sei
58233         lda a65286
58236         ora #%00010000
58238         sta a65286
58241         lda a65290
58244         and #%11110111
58246         ora #%00000010
58248         sta a65290
58251         cli
58252         rts
58253
58253 ;------------------------------  M O T O R  E I N S C H A L T E N
58253
58253 a58253  php
58254         sec
58255         ror a2044               ; motor-flag setzen
58258         lda *a1
58260         and #%11110101
58262         sta *a1                 ; einschalten
58264         ldx #30
```

```
58266 a58266  jsr a58090              ; ---> 16 ms warten
58269         dex
58270         bne a58266              ; insgesamt ca. 0.5 s warten ->
58272         plp
58273         rts
58274
58274         .by 'c1984commodore'
58288 ;
58288 ;------------------------------  M O T O R  A U S S C H A L T E N
58288
58288 a58288  lda *a1
58290         ora #%00001000
58292         sta *a1
58294         rts
58295
58295 ;-----------------------  K A S S E T T E N P U F F E R  L ö S C H E N
58295
58295 a58295  ldy #0
58297         lda #'
58299 a58299  sta (a182),y
58301         iny
58302         cpy #192
58304         bne a58299
58306         rts
58307
58307 ;---------------  P U F F E R Z E I G E R  I N I T I A L I S I E R E N
58307
58307 a58307  pha
58308         lda #l,a819
58310         sta *a182
58312         lda #h,a819
58314         sta *a183
58316         pla
58317         rts
58318
58318 ;-----------------------  A B B R U C H  D U R C H  S T O P T A S T E
58318
58318 a58318  jsr a64459              ; ---> stoptaste gedrückt?
58321         bcc a58339              ; nein: ->
58323         jsr a58288              ; ---> motor ausschalten
58326         jsr a58232              ; ---> kassettenport ausschalten
58329         ldx a1982               ; stackzeiger
58332         txs                     ; wiederherstellen
58333         lda #0
58335         sta a1982
58338         sec
58339 a58339  rts
58340
58340 ;------------------------------  T I M E R  S E T Z E N
58340
58340 a58340  lda a65289              ; bit 3 = 1 wenn timer a abgelaufen
58343         and a65290              ; bit 3 = 1 wenn kassettenport eingesch
58346         and #%00001000
58348         bne a58351
58350         rts
```

```
58351 a58351  sta a65289              ; timer starten
58354         sei
58355         lda #l,13200            ; 13.2 ms
58357         sta a65280              ; l,timer a
58360         lda #h,13200
58362         sta a65281              ; h,...
58365         ldx a1983
58368         txs
58369         sec
58370         rts
58371
58371 a58371  lda #l,18600            ; 18.6 ms
58373         sta a65280              ; l,timer a
58376         lda #h,18600
58378         sta a65281              ; h,...
58381         lda #%00001000
58383         sta a65289              ; timer starten
58386         rts
58387
58387 ;-------------------------------  P U L S   S C H R E I B E N
58387
58387 a58387  sec
58388         bcs a58391
58390
58390 a58390  clc
58391 a58391  sty a1994
58394         stx a1995
58397         ldy a1992
58400         ldx a1993
58403         lda #%00010000
58405 a58405  bit a65289              ; timer b abgelaufen?
58408         beq a58405              ; nein: warten ->
58410         sty a65282              ; l,timer b
58413         stx a65283              ; h,...
58416         sta a65289              ; timer b starten
58419         lda *a1
58421         eor #%00000010          ; datenpegel flippen
58423         sta *a1
58425         php
58426         jsr a58318              ; ---> abbruch, falls stoptaste gedrückt
58429         plp
58430         ldy a1994
58433         ldx a1995
58436         bcs a58390              ; cs: einmal wiederholen ->
58438         rts
58439
58439 ;------------------------------  Z E I T E N   S E T Z E N
58439
58439 a58439  lda #l,846              ; 846 us
58441         sta a1992
58444         lda #h,846
58446         sta a1993
58449         rts
58450
58450 a58450  lda #l,208              ; 208 us
58452         sta a1992
58455         lda #h,208
58457         sta a1993
58460         rts
```

```
58461 a58461  lda #l,420              ; 420 us
58463         sta a1992
58466         lda #h,420
58468         sta a1993
58471         rts
58472
58472 ;-------------------------------  B I T  0  S C H R E I B E N
58472
58472 a58472  jsr a58450              ; ---> zeit laden (208 us)
58475         jsr a58387              ; ---> puls schreiben
58478         jsr a58461              ; ---> zeit laden (420 us)
58481         jmp a58387              ; ---> puls schreiben
58484
58484 ;-------------------------------  B I T  1  S C H R E I B E N
58484
58484 a58484  jsr a58461              ; ---> zeit laden (420 us)
58487         jsr a58387              ; ---> puls schreiben
58490         jsr a58450              ; ---> zeit laden (208 us)
58493         jmp a58387              ; ---> puls schreiben
58496
58496 ;-------------------------------  S T A R T B I T  S C H R E I B E N
58496
58496 a58496  jsr a58439              ; ---> zeit laden (846 us)
58499         jsr a58387              ; ---> puls schreiben
58502         jsr a58461              ; ---> zeit laden (420 us)
58505         jmp a58387              ; ---> puls schreiben
58508
58508 ;-------------------------------  B Y T E  S C H R E I B E N
58508
58508 a58508  sta *a167               ; datenbyte
58510         lda #1
58512         sta a1969               ; byte-parity
58515         jsr a58496              ; ---> startbit schreiben
58518         ldx #8
58520 a58520  ror *a167               ; datenbyte ins carry rotieren
58522         bcs a58533              ; bit = 1: ->
58524         inc a1969               ; parity erhöhen
58527         jsr a58472              ; ---> bit 0 schreiben
58530         jmp a58536              ; --->
58533
58533 a58533  jsr a58484              ; ---> bit 1 schreiben
58536 a58536  dex                     ; schon 8 bits geschrieben?
58537         bne a58520              ; nein: ->
58539         ror a1969               ; parity
58542         bcs a58550              ; = 1: ->
58544         jsr a58472              ; ---> bit 0 schreiben
58547         jmp a58553              ; --->
58550
58550 a58550  jsr a58484              ; ---> bit 1 schreiben
58553 a58553  rts
58554
58554 ;---------------------------  B Y T E - S T R I N G  S C H R E I B E N
58554
58554 a58554  tsx
58555         stx a1982               ; stack-zeiger merken
58558         lda *a1                 ; datenausgang
58560         ora #%00000010          ; hoch legen
58562         sta *a1
58564         jsr a58450              ; ---> zeit laden (208 us)
```

```
58567          ldy #1
58569          sty a65283              ; h,timer b
58572          lda #%00010000          ; timer b starten
58574          sta a65289
58577          bit *a247               ; 2. pass?
58579          bpl a58585              ; ja: ->
58581          ldy #64
58583          ldx #254
58585 a58585   jsr a58387              ; ---> puls schreiben
58588          dex                     ; 64 * 254 pulse von je 208 us
58589          bne a58585              ; ergeben mit ebensolangen pausen
58591          dey                     ; einen vorspann von ca. 6.8 s
58592          bne a58585
58594          ldy #9                  ; nach dem vorspann folgt ein
58596 a58596   tya                     ; 'countdown' 9, 8, ..., 1
58597          ora *a247               ; im 1. pass ist bit 7 gesetzt
58599          jsr a58508              ; ---> byte schreiben
58602          dey
58603          bne a58596
58605          ldy #0
58607          sty *a245               ; block-parity initialisieren
58609          lda *a248               ; filetyp
58611          beq a58622              ; = 0: nicht schreiben ->
58613          eor *a245
58615          sta *a245               ; block-parity berechnen
58617          lda *a248               ; filetyp
58619          jsr a58508              ; ---> byte schreiben
58622 a58622   lda (a186),y            ; datenbyte
58624          pha
58625          eor *a245               ; block-parity berechnen
58627          sta *a245
58629          pla
58630          jsr a58508              ; ---> byte schreiben
58633          inc *a186               ; quellzeiger erhöhen
58635          bne a58639
58637          inc *a187
58639 a58639   inc a1011               ; byte-zähler erhöhen
58642          bne a58622
58644          inc a1012
58647          bne a58622
58649          lda *a245               ; block-parity
58651          jsr a58508              ; ---> byte schreiben
58654          jsr a58461              ; ---> zeit laden (420 us)
58657          jsr a58387              ; ---> puls schreiben
58660          jsr a58450              ; ---> zeit laden (208 us)
58663          ldy #1
58665          ldx #194
58667 a58667   jsr a58387              ; ---> puls schreiben
58670          dex                     ; 194 * 2 * 208 us = 80 ms
58671          bne a58667              ; nachspann
58673          dey
58674          bne a58667
58676          rts
```

```
58677 ;-------------------  P U F F E R  A U F  B A N D  S C H R E I B E N
58677
58677 a58677  jsr a58137            ; ---> play-taste prüfen ggf. anfordern
58680         jsr a58212            ; ---> kassettenport einschalten
58683         jsr a58253            ; ---> motor einschalten
58686         bcs a58726            ; stoptaste gedrückt: ->
58688         lda #128
58690         sta *a247             ; 1. pass
58692 a58692  lda *a182             ; pufferanfang
58694         sta *a186             ; in quellzeiger
58696         lda *a183
58698         sta *a187
58700         lda #l,65345          ; 65345 = 65536 - pufferlänge 191
58702         sta a1011             ; in byte-zähler
58705         lda #h,65345
58707         sta a1012
58710         jsr a58554            ; ---> block schreiben
58713         bcs a58726            ; fehler: ->
58715         lda *a247             ; 1. pass?
58717         bpl a58725            ; nein: ->
58719         lda #0
58721         sta *a247             ; 2. pass
58723         bpl a58692
58725
58725 a58725  clc
58726 a58726  jsr a58288            ; ---> motor ausschalten
58729         jmp a58232            ; ---> kassettenport ausschalten
58732
58732 ;---------  H E A D E R B L O C K  A U F  B A N D  S C H R E I B E N
58732
58732 a58732  jsr a58307            ; ---> pufferzeiger initialisieren
58735         jsr a58295            ; ---> kassettenpuffer löschen
58738         ldy #0
58740         lda *a178             ; l,anfangsadresse
58742         sta (a182),y          ; in puffer
58744         iny
58745         lda *a179             ; h,anfangsadresse
58747         sta (a182),y          ; in puffer
58749         iny
58750         lda *a157             ; l,endadresse
58752         sta (a182),y          ; in puffer
58754         iny
58755         lda *a158             ; h,endadresse
58757         sta (a182),y          ; in puffer
58759         iny
58760         sty a1971             ; offset in puffer
58763         ldy #0
58765         sty a1970             ; offset in filenamen
58768 a58768  ldy a1970
58771         cpy *a171             ; fnlen
58773         beq a58797            ; filename zu ende: ->
58775         lda #a175             ; l,fnadr
58777         sta a2015
58780         jsr a2009             ; ---> lda (a),y: zeichen aus filename
58783         ldy a1971
58786         sta (a182),y          ; in puffer
58788         inc a1970             ; offsets inkrementieren
58791         inc a1971
58794         jmp a58768            ; ---> weitermachen
58797
58797 a58797  jmp a58677            ; ---> block auf band schreiben
```

```
58800 ;-- S P E I C H E R B E R E I C H  A U F  B A N D  S C H R E I B E N
58800
58800 a58800  jsr a58137              ; ---> play-taste prüfen ggf. anfordern
58803         jsr a58212              ; ---> kassettenport einschalten
58806         jsr a58253              ; ---> motor einschalten
58809         bcs a58858              ; stoptaste gedrückt: ->
58811         lda #128
58813         sta *a247               ; 1. pass
58815 a58815  lda *a178               ; anfangsadresse
58817         sta *a186               ; in quellzeiger
58819         lda *a179
58821         sta *a187
58823         clc
58824         lda *a157               ; ende des speicherbereichs
58826         sbc *a178               ; - anfangsadresse
58828         eor #255                ; invertieren
58830         sta a1011               ; und in bytezähler bringen
58833         lda *a158
58835         sbc *a179
58837         eor #255
58839         sta a1012
58842         jsr a58554              ; ---> bereich schreiben
58845         bcs a58858              ; fehler: ->
58847         lda *a247               ; 1. pass?
58849         bpl a58857              ; nein: ->
58851         lda #0
58853         sta *a247               ; 2. pass
58855         bpl a58815
58857
58857 a58857  clc
58858 a58858  jsr a58288              ; ---> motor ausschalten
58861         jmp a58232              ; ---> kassettenport ausschalten
58864
58864 ;------------------------------  E O T - B L O C K  S C H R E I B E N
58864
58864 a58864  jsr a58295              ; ---> kassettenpuffer löschen
58867         lda #5
58869         sta *a248               ; filetyp 'end-of-tape'
58871         jmp a58677              ; ---> block auf band schreiben
58874
58874 ;------------------------------  P U L S  L E S E N
58874
58874 a58874  .by 64 0 128
58877
58877 a58877  ldx a1976               ; zeit für kurzen puls
58880         ldy a1977
58883 a58883  lda a1979               ; zeit für langen puls
58886         pha
58887         lda a1978
58890         pha
58891         lda #%00010000
58893 a58893  bit *a1                 ; data
58895         beq a58893              ; = 0: warten ->
58897 a58897  bit *a1                 ; data
58899         bne a58897              ; = 1: warten ->
58901         stx a65282              ; timer b auf kurzen puls setzen
58904         sty a65283
58907         pla
58908         sta a65284              ; timer c auf langen puls setzen
58911         pla
58912         sta a65285
```

```
58915          lda #%01010000
58917          sta a65289              ; timer b und c starten
58920 a58920   lda *a1                 ; data
58922          cmp *a1                 ; stabil?
58924          bne a58920              ; nein: warten ->
58926          and #%00010000          ; data
58928          bne a58883              ; = 1: nochmal von vorn ->
58930          jsr a58318              ; ---> abbruch, falls stoptaste gedrückt
58933          lda #%00010000
58935 a58935   bit *a1                 ; data
58937          bne a59010              ; = 1: kurzen puls gelesen ->
58939          bit a65289              ; timer b abgelaufen?
58942          beq a58935              ; nein: warten ->
58944 a58944   lda *a1                 ; data
58946          cmp *a1                 ; stabil?
58948          bne a58944              ; nein: warten ->
58950          and #%00010000          ; data
58952          bne a59010              ; = 1: kurzen puls gelesen ->
58954          lda #%01000000
58956 a58956   bit a65289              ; timer c abgelaufen?
58959          beq a58956              ; nein: warten ->
58961 a58961   lda *a1                 ; data
58963          cmp *a1                 ; stabil?
58965          bne a58961              ; nein: warten ->
58967          and #%00010000          ; data
58969          bne a59015              ; = 1: langen puls gelesen ->
58971          lda a1980               ; wartezeit verdoppeln
58974          sta a65282              ; l,timer b
58977          lda a1981
58980          sta a65283              ; h,...
58983          lda #%00010000
58985          sta a65289              ; timer b starten
58988          lda #%00010000
58990 a58990   bit a65289              ; timer b abgelaufen?
58993          beq a58990              ; nein: warten ->
58995 a58995   lda *a1                 ; data
58997          cmp *a1                 ; stabil?
58999          bne a58995              ; nein: warten ->
59001          and #%00010000          ; data
59003          beq a59020              ; = 0: kein puls, fehler ->
59005          bit a58874+2            ; doppelt langer puls setzt n-flag
59008          bmi a59018
59010
59010 a59010   bit a58874              ; kurzer puls setzt v-flag
59013          bvs a59018
59015
59015 a59015   bit a58874+1            ; langer puls setzt z-flag
59018 a59018   clc                     ; ok-flag
59019          rts
59020
59020 a59020   sec                     ; fehler-flag
59021          rts
59022
59022 ;-------------------------------  B I T   L E S E N
59022
59022 a59022   .by 64 0 128
59025 ;
59025 a59025   jsr a58877              ; ---> puls lesen
59028          bcs a59091              ; kein puls: fehler ->
59030          bvs a59050              ; kurzer puls: ->
59032          bpl a59036              ; langer puls: ->
59034          bmi a59075              ; doppeltlanger puls: ->
```

```
59036 a59036  jsr a58877              ; ---> puls lesen
59039         bcs a59091              ; kein puls: fehler ->
59041         bvs a59045              ; kurzer puls: bit 1 ->
59043         bvc a59091              ; sonst fehler ->
59045
59045 a59045  bit a59022+1            ; bit 1 setzt z-flag
59048         clc                     ; ok-flag
59049         rts
59050
59050 a59050  jsr a58877              ; ---> puls lesen
59053         bvs a59059              ; kurzer puls: vorspann ->
59055         bpl a59070              ; langer puls: bit 0 ->
59057         bmi a59091              ; sonst: fehler ->
59059
59059 a59059  jsr a58877              ; ---> puls lesen
59062         bcs a59091              ; kein puls: fehler ->
59064         bvs a59059              ; kurzer puls: vorspann, weiterlesen ->
59066         bpl a59091              ; langer puls: fehler ->
59068         bmi a59075              ; doppeltlanger puls: startbit ->
59070
59070 a59070  bit a59022              ; bit 0 setzt v-flag
59073         clc                     ; ok-flag
59074         rts
59075
59075 a59075  jsr a58877              ; ---> puls lesen
59078         bcs a59091              ; kein puls: fehler ->
59080         bvs a59091              ; kurzer puls: fehler ->
59082         bpl a59086              ; langer puls: startbit ->
59084         bmi a59091              ; sonst fehler ->
59086
59086 a59086  bit a59022+2            ; startbit setzt n-flag
59089         clc                     ; ok-flag
59090         rts
59091
59091 a59091  sec                     ; fehler-flag
59092         rts
59093
59093 ;-------------------------------  S T A R T B I T  S U C H E N
59093
59093 a59093  tsx
59094         stx a1983               ; stack-zeiger merken
59097         clc
59098         ror a1996               ; startbit-flag
59101         cli
59102 a59102  jsr a59025              ; ---> bit lesen
59105         bcs a59102              ; kein bit: weiterlesen ->
59107         bvs a59102              ; bit 0: weiterlesen ->
59109         bpl a59102              ; bit 1: weiterlesen ->
59111         jsr a58371              ; ---> timer a setzen und starten (18.6 ms)
59114         clc
59115         rts
59116
59116 ;-------------------------------  B Y T E  L E S E N
59116
59116 a59116  bit a1996               ; startbit-flag gesetzt?
59119         bmi a59202              ; ja: fehler ->
59121         jsr a59093              ; ---> startbit suchen
59124         bcs a59202              ; stoptaste gedrückt: ->
```

```
59126 a59126  lda #1
59128         sta a1969              ; byte-parity
59131         ldx #8
59133         stx a1973              ; bit-zähler
59136         sec
59137         ror a1996              ; startbit-flag setzen
59140 a59140  jsr a59025             ; ---> bit lesen
59143         bcs a59202             ; kein bit: fehler ->
59145         bvs a59151             ; bit 0 gelesen: ->
59147         bpl a59164             ; bit 1 gelesen: ->
59149         bmi a59202             ; sonst fehler ->
59151
59151 a59151  clc
59152         ror *a167              ; bit 0 einrotieren
59154         inc a1969              ; byte-parity erhöhen
59157         dec a1973              ; schon 8 bits gelesen?
59160         bne a59140             ; nein: ->
59162         beq a59172
59164
59164 a59164  sec
59165         ror *a167              ; bit 1 einrotieren
59167         dec a1973              ; schon 8 bits gelesen?
59170         bne a59140             ; nein: ->
59172 a59172  jsr a59025             ; ---> parity-bit lesen
59175         bcs a59202             ; kein bit: fehler ->
59177         bvs a59183             ; bit 0 gelesen: ->
59179         bpl a59192             ; bit 1 gelesen: ->
59181         bmi a59202
59183
59183 a59183  lda a1969              ; byte-parity
59186         and #1                 ; = 0?
59188         bne a59202             ; nein: fehler ->
59190         beq a59199
59192
59192 a59192  lda a1969              ; byte-parity
59195         and #1                 ; = 1?
59197         beq a59202             ; nein: fehler ->
59199 a59199  clc                    ; ok-flag
59200         bcc a59203
59202
59202 a59202  sec                    ; fehler-flag
59203 a59203  sei                    ; warum??
59204         php
59205         clc
59206         ror a1996              ; startbit-flag rücksetzen
59209         plp
59210         rts
59211
59211 ;------------------------------  B Y T E - S T R I N G  L E S E N
59211
59211 a59211  tsx
59212         stx a1982              ; stack-zeiger merken
59215         lda *a147              ; verify-flag
59217         beq a59222             ; 'load': ->
59219         sec
59220         ror *a147              ; verify-flag setzen
59222 a59222  jsr a58253             ; ---> motor einschalten
59225         jsr a58212             ; ---> kassettenport einschalten
59228         lda a1984              ; anfangsadresse
59231         sta *a182              ; in zielzeiger
59233         lda a1985
59236         sta *a183
```

```
59238          lda a1986              ; länge des strings, invertiert,
59241          sta a1013              ; in bytezähler
59244          lda a1987
59247          sta a1014
59250          jsr a59677             ; ---> lesen vorbereiten
59253          ldy #0
59255          sty a1974              ; fehlerzähler 1
59258          sty a1975              ; fehlerzähler 2
59261          sty *a245              ; block-parity
59263          sty *a177              ; fehlerbilanz
59265          sty *a248              ; filetyp
59267          dey
59268          sty *a169              ; fehlerflag 1. pass
59270          sty *a146              ; fehlerflag 2. pass
59272          bit a1968              ; wird in puffer gelesen?
59275          bpl a59296             ; nein: ->
59277          jsr a59116             ; ---> byte lesen
59280          bcs a59293             ; fehler: ->
59282          lda *a167              ; gelesenes byte
59284          sta *a248              ; =: filetyp
59286          eor *a245              ; block-parity
59288          sta *a245
59290          jmp a59296             ; --->
59293
59293 a59293   sec
59294          ror *a248              ; bit 7 im filetyp setzen
59296 a59296   jsr a59116             ; ---> byte lesen
59299          bcs a59326             ; fehler: ->
59301          lda *a167              ; gelesenes byte
59303          ldy #0
59305          bit *a147              ; verify-flag
59307          bpl a59317             ; 'load': ->
59309          cmp (a182),y           ; mit byte im speicher vergleichen
59311          beq a59317             ; gleich: ok ->
59313          sty *a169              ; fehlerflag 1. pass setzen
59315          bne a59319             ; immer ->
59317
59317 a59317   sta (a182),y           ; gelesenes byte speichern
59319 a59319   eor *a245              ; block-parity
59321          sta *a245
59323          jmp a59356             ; --->
59326
59326 a59326   ldy a1974              ; fehlerzähler 1
59329          cpy #30                ; schon = 30?
59331          bcs a59351             ; ja: zu viele fehler ->
59333          lda *a182              ; fehleradresse für vergleich
59335          sta a1079,y            ; im 2. pass merken
59338          lda *a183
59340          sta a1109,y
59343          inc a1974              ; fehlerzähler 1 erhöhen
59346          inc *a177              ; fehlerbilanz erhöhen
59348          jmp a59356             ; --->
59351
59351 a59351   lda #255
59353          sta a1974              ; fehlerzähler 1 := 255
59356 a59356   inc *a182              ; zielzeiger erhöhen
59358          bne a59362
59360          inc *a183
59362 a59362   inc a1013              ; bytezähler erhöhen
59365          bne a59296
59367          inc a1014
59370          bne a59296             ; noch nicht alle bytes gelesen: ->
```

```
59372          lda a1974              ; fehlerzähler 1
59375          sta a1975              ; in fehlerzähler 2 kopieren
59378          jsr a59116             ; ---> byte lesen
59381          lda a1975              ; waren fehler im 1. pass?
59384          bne a59392             ; ja: kein parity-vergleich ->
59386          lda *a167              ; gelesenes byte
59388          cmp *a245              ; = berechnete block-parity?
59390          bne a59395             ; nein: ->
59392 a59392   jmp a59402             ; --->
59395
59395 a59395   lda *a247              ; 2. pass beendet?
59397          bmi a59402             ; nein: ->
59399          jmp a59575             ; ---> fehlerabschlu
59402
59402 a59402   lda *a247              ; 1. pass beendet?
59404          bmi a59417             ; ja: 2. pass ->
59406          lda a1975              ; fehlerzahl im 1. pass
59409          beq a59414             ; = 0: fertig ->
59411          jmp a59575             ; ---> fehlerabschlu
59414
59414 a59414   jmp a59591             ; ---> ok, motor und datenport abschalten
59417
59417 a59417   lda #0
59419          sta a1974              ; fehlerzähler 1
59422          sta *a245              ; block-parity
59424          lda a1984              ; anfangsadresse
59427          sta *a182              ; in zielzeiger
59429          lda a1985
59432          sta *a183
59434          lda a1986              ; bereichslänge
59437          sta a1013              ; in bytezähler
59440          lda a1987
59443          sta a1014
59446          jsr a59677             ; ---> lesen vorbereiten
59449          bit a1968              ; wird in puffer gelesen?
59452          bpl a59475             ; nein: ->
59454          jsr a59116             ; ---> byte lesen
59457          bit *a248              ; filetyp im 1. pass fehlerfrei?
59459          bpl a59469             ; ja: ->
59461          lda *a167              ; gelesenes byte
59463          sta *a248              ; =: filetyp
59465          bcc a59469             ; ok: ->
59467          ror *a248              ; sonst bit 7 setzen
59469 a59469   lda *a248
59471          eor *a245              ; block-parity
59473          sta *a245
59475 a59475   jsr a59116             ; ---> byte lesen
59478          ror a1988              ; bei fehler bit 7 setzen
59481          lda *a167              ; gelesenes byte
59483          eor *a245              ; block-parity
59485          sta *a245
59487          bit a1975              ; fehlerzahl im 1. pass
59490          bmi a59542             ; grüer als 30: ->
59492          ldy a1974              ; fehlerzahl im 2. pass
59495          cpy a1975              ; = fehlerzahl im 1. pass?
59498          beq a59542             ; ja: ->
59500          lda a1079,y            ; fehleradresse im 1. pass
59503          cmp *a182              ; = aktuelle zieladresse?
59505          bne a59542
59507          lda a1109,y
59510          cmp *a183
59512          bne a59542             ; nein: ->
```

```
59514          inc a1974              ; fehlerzähler 1 erhöhen
59517          lda a1988              ; letztes gelesenes byte fehlerhaft?
59520          bmi a59542             ; ja: nicht abspeichern ->
59522          ldy #0
59524          lda *a167              ; gelesenes byte
59526          bit *a147              ; verify-flag
59528          bpl a59538             ; 'load': ->
59530          cmp (a182),y           ; mit byte im speicher vergleichen
59532          beq a59538             ; gleich: ->
59534          sty *a146              ; fehlerflag 2. pass setzen
59536          bne a59542             ; immer ->
59538
59538 a59538   dec *a177              ; fehlerbilanz erniedrigen
59540          sta (a182),y           ; byte abspeichern
59542 a59542   inc *a182              ; zielzeiger erhöhen
59544          bne a59548
59546          inc *a183
59548 a59548   inc a1013              ; bytezähler erhöhen
59551          bne a59475
59553          inc a1014
59556          bne a59475             ; noch nicht alle bytes gelesen: ->
59558          jsr a59116             ; ---> byte lesen
59561          lda *a177              ; alle fehler korrigiert?
59563          bne a59575             ; nein: ->
59565          lda *a146              ; fehlerflag 1. pass
59567          and *a169              ; fehlerflag 2. pass
59569          beq a59583             ; mindestens 1 fehler in 1 pass: ->
59571          lda *a248              ; filetyp
59573          bpl a59591             ; fehlerfrei: ->
59575 a59575   lda #%01100000
59577          jsr a62494             ; ---> in status
59580          sec                    ; fehler-flag
59581          bcs a59592
59583
59583 a59583   lda #%00010000
59585          jsr a62494             ; ---> in status
59588          sec                    ; fehler-flag
59589          bcs a59592
59591
59591 a59591   clc                    ; ok-flag
59592 a59592   jsr a58288             ; ---> motor ausschalten
59595          jsr a58232             ; ---> kassettenport ausschalten
59598          rts
59599
59599 ;--------------------------  B L O C K   I N   P U F F E R   L E S E N
59599
59599 a59599   .se a819               ; kassettenpuffer-anfang
59601          .se 65345              ; = 65536 - pufferlänge 191
59603
59603 a59603   ldy #3
59605 a59605   lda a59599,y           ; anfang und länge übernehmen
59608          sta a1984,y
59611          dey
59612          bpl a59605
59614          sty a1968              ; flag für 'lesen in puffer' setzen
59617          lda *a147              ; verify-flag
59619          pha                    ; merken
59620          iny
59621          sty *a147              ; verify-flag := 0
59623          sty a1337              ; offset in kassettenpuffer
59626          jsr a59211             ; ---> block lesen
59629          pla
59630          sta *a147              ; verify-flag wiederherstellen
59632          jmp a58307             ; ---> pufferzeiger initialisieren
```

```
59635 ;------------------------------  P R O G R A M M F I L E  L E S E N
59635
59635 a59635  lda *a178               ; ladeanfang
59637         sta a1984               ; für zielzeiger
59640         lda *a179
59642         sta a1985
59645         clc
59646         lda *a157               ; länge berechnen
59648         sbc *a178
59650         eor #255                ; invertieren
59652         sta a1986               ; für bytezähler übernehmen
59655         lda *a158
59657         sbc *a179
59659         eor #255
59661         sta a1987
59664         clc
59665         ror a1968               ; flag für 'lesen in puffer' rücksetzen
59668         jmp a59211              ; ---> file einlesen
59671
59671 ;------------------------------  L E S E N  V O R B E R E I T E N
59671
59671 a59671  .se 258                 ; zeitvorgabe kurzer puls
59673         .se 514                 ; zeitvorgabe langer puls
59675         .se 525                 ; zusätzlich für doppeltlangen puls
59677
59677 a59677  ldx #5
59679 a59679  lda a59671,x            ; zeiten aus tabelle übernehmen
59682         sta a1976,x
59685         dex
59686         bpl a59679
59688 a59688  lda #10                 ; zähler setzen
59690         sta a1989
59693 a59693  jsr a58877              ; ---> puls lesen
59696         bcs a59688              ; kein puls: nochmal ->
59698         bvc a59688              ; kein kurzer puls: nochmal ->
59700         dec a1989               ; zähler erniedrigen
59703         bne a59693              ; bis 10 kurze pulse gelesen wurden
59705 a59705  lda #0
59707         sta *a186               ; zellen für zeitberechnung löschen
59709         sta *a187
59711         ldy #16                 ; zähler für pulse
59713 a59713  ldx #0                  ; zähler für zeitschleifen
59715         lda #%00010000
59717 a59717  bit *a1                 ; data
59719         beq a59717              ; = 0: warten ->
59721 a59721  bit *a1                 ; data
59723         bne a59721              ; = 1: warten ->
59725 a59725  inx                     ; zähler erhöhen
59726         beq a59705              ; überlauf: nochmal von vorn ->
59728         bit *a1                 ; data
59730         beq a59725              ; = 0: zähler x erhöhen und warten ->
59732 a59732  inx                     ; zähler erhöhen
59733         beq a59705              ; überlauf: nochmal von vorn ->
59735         bit *a1                 ; data
59737         bne a59732              ; = 1: zähler x erhöhen und warten ->
59739         txa                     ; zählerstand
59740         clc
59741         adc *a186               ; zum gesamtzeit-zähler addieren
59743         sta *a186
59745         lda #0
59747         adc *a187
59749         sta *a187
```

```
59751          dey                    ; schon 16 takte gemessen?
59752          bne a59713             ; nein: ->
59754          lsr *a187              ; gesamte gemessene zeit
59756          ror *a186              ; durch 4 dividieren
59758          lsr *a187
59760          ror *a186
59762          lda *a186
59764          sta a1976              ; ergebnis als kurze pulszeit,
59767          asl a                  ; verdoppeltes ergebnis
59768          sta a1978              ; als lange pulszeit und
59771          sta a1980              ; als verlängerungszeit übernehmen
59774          lda *a187
59776          sta a1977
59779          rol a
59780          sta a1979
59783          sta a1981
59786 a59786   jsr a58877             ; ---> puls lesen
59789          bcs a59786             ; kein puls: weiterlesen ->
59791          bvs a59786             ; kurzer puls: weiterlesen ->
59793          bpl a59786             ; langer puls: weiterlesen ->
59795          jsr a58877             ; ---> puls lesen
59798          bcs a59786             ; kein puls: weiterlesen ->
59800          bvs a59786             ; kurzer puls: weiterlesen ->
59802          bmi a59786             ; doppeltlanger puls: weiterlesen ->
59804          clc                    ; startbit wurde gelesen
59805          ror a1996              ; startbit-flag rücksetzen
59808          jsr a58371             ; ---> timer a setzen und starten (18.6 ms)
59811          lda #3
59813          sta a1990              ; zähler
59816          jsr a59126             ; ---> byte lesen
59819          bcc a59824             ; ok: ->
59821          dec a1990              ; zähler dekrementieren
59824 a59824   jsr a59116             ; ---> byte lesen
59827          bcc a59837             ; ok: ->
59829          dec a1990              ; zähler dekrementieren
59832          bne a59837             ; noch nicht 3 fehler: ->
59834          jmp a59677             ; ---> lesevorbereitung wiederholen
59837
59837 a59837   lda *a167              ; gelesenes byte
59839          and #15                ; unteres halbbyte isolieren
59841          cmp #1                 ; = 1 (ende des countdowns)?
59843          bne a59824             ; nein: weiterlesen ->
59845          lda *a167              ; gelesenes byte
59847          and #128               ; bit 7 isolieren
59849          sta *a247              ; pass 1: 128, pass 2: 0
59851          rts
59852
59852 ;------------------------------  F I L E H E A D E R   L E S E N
59852
59852 a59852   jsr a59603             ; ---> block von band in puffer lesen
59855          bcs a59934
59857          lda *a248              ; filetyp
59859          cmp #5                 ; 'end of tape'?
59861          beq a59930             ; ja: ->
59863          cmp #1                 ; 'programm'?
59865          beq a59875             ; ja: ->
59867          cmp #3
59869          beq a59875
59871          cmp #4                 ; 'daten'?
59873          bne a59852             ; nein: ->
```

```
59875 a59875  tax
59876         bit *a154              ; meldungen zugelassen?
59878         bpl a59927             ; nein: ->
59880         jsr a64472             ; ---> meldung ausgeben
59883         .by 13 'found ' 0
59891         ldy #4
59893 a59893  lda (a182),y           ; dateinamen
59895         jsr a65490             ; ---> bsout
59898         iny
59899         cpy #21                ; insgesamt 16 zeichen
59901         bne a59893
59903         ldx #255
59905 a59905  jsr a58090             ; ---> 16 ms warten
59908         jsr a58090             ; ---> 16 ms warten
59911         dex
59912         beq a59927             ; nach ca. 8 s: ->
59914         lda #127
59916         jsr a56176             ; ---> tastaturdekoder abfragen
59919         cmp #127               ; stoptaste gedrückt?
59921         beq a59934             ; ja: ->
59923         cmp #223               ; C=-taste gedrückt?
59925         bne a59905             ; nein: warten ->
59927 a59927  clc
59928         lda *a248              ; filetyp
59930 a59930  rts
59931
59931         nop
59932         nop
59933         nop
59934
59934 a59934  lda #0
59936         rts
59937
59937 ;---------------------------  H E A D E R  I D E N T I F I Z I E R E N
59937
59937 a59937  jsr a59852             ; ---> block vom band lesen, meldung
59940         bcs a59987             ; abbruch: ->
59942         cmp #5                 ; 'end of tape'-block?
59944         beq a59989             ; ja: ->
59946         ldy #4
59948         sty a1971              ; offset in kassettenpuffer
59951         ldy #0
59953         sty a1970              ; offset in filenamen
59956 a59956  cpy *a171              ; fnlen: ende des namens erreicht?
59958         beq a59991             ; ja: ->
59960         lda #a175              ; l,fnadr
59962         sta a2015
59965         jsr a2009              ; ---> lda (a),y: zeichen aus filenamen
59968         ldy a1971
59971         cmp (a182),y           ; = zeichen des namens im puffer?
59973         bne a59937             ; nein: nächsten block lesen ->
59975         inc a1970              ; offsets erhöhen
59978         inc a1971
59981         ldy a1970
59984         jmp a59956             ; ---> weiter vergleichen
59987
59987 a59987  lda #0
59989 a59989  sec
59990         rts
59991
59991 a59991  clc
59992         lda *a248              ; filetyp
59994         rts
```

```
59995 ;------------------------------  R S - 2 3 2  R O U T I N E N
59995
59995 a59995  lda a2004              ; rs-status
59998         and #%00010000         ; bit 4 gesetzt?
60000         beq a60052             ; nein: ->
60002         lda a64784
60005         and #2
60007         beq a60052
60009         ldx #0
60011         bit a2000
60014         bpl a60025
60016         lda a1999
60019         stx a2000
60022         jmp a60041             ; --->
60025
60025 a60025  bit a1998              ; datenbyte im ausgaberegister?
60028         bpl a60052             ; nein: fertig ->
60030         bit a2006
60033         bmi a60052
60035         lda a1997              ; datenbyte
60038         stx a1998              ; ausgaberegister freigeben
60041 a60041  sta a64768             ; datenbyte an rs-232
60044         lda a2004              ; rs-status
60047         and #%11101111         ; bit 4 löschen
60049         sta a2004
60052 a60052  rts
60053
60053 a60053  lda a2004              ; rs-status
60056         and #%00001000
60058         beq a60144
60060         lda a2004              ; rs-status
60063         and #%11110111         ; bit 3 löschen
60065         sta a2004
60068         lda a64768             ; datenbyte von rs-232
60071         beq a60098             ; = 0: ->
60073         sta a2005              ; zwischenspeichern
60076         cmp *a252              ; = 255 (von open)
60078         bne a60087
60080         lda #0
60082         sta a2006
60085         beq a60144             ; fertig ->
60087
60087 a60087  cmp *a253              ; = 255 (von open)
60089         bne a60098
60091         lda #255
60093         sta a2006
60096         bne a60144             ; fertig ->
60098
60098 a60098  lda a2003              ; bytezähler
60101         cmp #63
60103         beq a60144             ; fertig ->
60105         cmp #56
60107         bne a60124
60109         lda *a253
60111         beq a60124
60113         sta a1999
60116         lda #255
60118         sta a2000
60121         sta a2007
```

```
60124 a60124  ldx a2001               ; pufferzeiger
60127         inx                     ; erhöhen
60128         txa
60129         and #63                 ; modulo 64
60131         sta a2001               ; wieder abspeichern
60134         tax
60135         lda a2005               ; datenbyte
60138         sta a1015,x             ; in puffer
60141         inc a2003               ; bytezähler erhöhen
60144 a60144  rts
60145
60145 ;------------------------------  E I N G A B E  V O N  R S - 2 3 2
60145
60145 a60145  lda a2003               ; puffer leer?
60148         beq a60202              ; ja: ->
60150         php
60151         sei
60152         ldx a2002               ; pufferzeiger
60155         inx                     ; erhöhen
60156         txa
60157         and #63                 ; modulo 64
60159         sta a2002               ; und wieder abspeichern
60162         plp
60163         tax
60164         lda a1015,x             ; byte aus puffer
60167         pha                     ; merken
60168         dec a2003               ; bytezähler dekrementieren
60171         lda a2003
60174         cmp #8                  ; noch 8 bytes im puffer?
60176         bne a60203              ; nein: ->
60178         bit a2007
60181         bpl a60203
60183         lda *a252
60185         beq a60203
60187         sta a1999
60190         sec
60191         ror a2000
60194         lsr a2007
60197 a60197  bit a2008               ; rs-kanal offen?
60200         bpl a60213              ; nein: ->
60202 a60202  pha
60203 a60203  lda a2004               ; rs-status
60206         and #%01001111
60208         eor #%01000000
60210         sta *a144               ; status
60212         pla
60213 a60213  clc
60214         rts
60215
60215 ;------------------------------  A U S G A B E  A U F  R S - 2 3 2
60215
60215 a60215  bit a1998               ; ausgaberegister frei?
60218         bmi a60215              ; nein: warten ->
60220         sta a1997               ; datenbyte in ausgaberegister
60223         sec
60224         ror a1998               ; ausgabe-sperre setzen
60227         jmp a60202              ; ---> abschlu
```

```
60230 ;-----------  R S - 2 3 2 - A R B E I T S B E R E I C H  L ö S C H E N
60230
60230 a60230  lda #0
60232         ldx #11
60234 a60234  sta a1997,x             ; bereich 1997,...,2008 löschen
60237         dex
60238         bpl a60234
60240         sta a64769              ; rs-232-status
60243         sta *a252
60245         sta *a253
60247         rts
60248
60248 ;------------------------------  I / O  - M E L D U N G E N
60248
60248 a60248  .by 13 'i/o error ' 163
60260         .by 13 'searching for' 160
60275         .by 13 'press play on tapE'
60294         .by 'press record & play on tapE'
60321         .by 13 'loadinG'
60329         .by 13 'saving' 160
60337         .by 13 'verifyinG'
60347         .by 13 'found' 160
60354         .by 13 'ok' 141
60358
60358 a60358  bit *a154               ; meldungen
60360         bpl a60375              ; gesperrt: ->
60362 a60362  lda a60248,y            ; zeichen aus text
60365         php
60366         and #127
60368         jsr a65490              ; ---> bsout
60371         iny
60372         plp
60373         bpl a60362              ; nicht letztes zeichen: ->
60375 a60375  clc
60376         rts
60377
60377 ;------------------------------  G E T I N
60377
60377 a60377  lda *a152               ; eingabekanal
60379         bne a60407              ; nicht tastatur: ->
60381         lda *a239               ; tastenpuffer-index
60383         ora a1373               ; funktionstasten-index
60386         beq a60450              ; beide = 0: ->
60388         sei
60389         jmp a55489              ; ---> kode aus tastenpuffer holen
60392
60392 ;------------------------------  B A S I N
60392
60392 a60392  lda *a152               ; eingabekanal
60394         bne a60407              ; nicht tastatur: ->
60396         lda *a202               ; spaltenzeiger
60398         sta *a197               ; letzter spaltenwert
60400         lda *a205               ; zeilenzeiger
60402         sta *a196               ; letzter zeilenwert
60404         jmp a55653              ; ---> zeichen vom bildschirm holen
60407
60407 a60407  cmp #3                  ; bildschirm?
60409         bne a60442              ; nein: ->
60411         ora *a199               ; übernahmeflag
60413         sta *a199
60415         lda a2024               ; rechter bildfensterrand
60418         sta *a195               ; spalte des letzten zeichens
60420         jmp a55653              ; ---> zeichen vom bildschirm holen
```

```
60423 a60423  jsr a64442              ; ---> x und y retten
60426         cmp #1                  ; kassette?
60428         bne a60436              ; nein: ->
60430         jsr a60452              ; ---> byte von kassette holen
60433         jmp a64452              ; ---> x und y wiederherstellen
60436
60436 a60436  jsr a60145              ; ---> byte von rs-232 holen
60439         jmp a64452              ; ---> x und y wiederherstellen
60442
60442 a60442  bcc a60423              ; nicht bus: ->
60444         lda *a144               ; status
60446         beq a60555              ; ok: ->
60448         lda #13                 ; cr-kode
60450 a60450  clc
60451         rts
60452
60452 ;-----------------------  B Y T E  V O N  K A S S E T T E  H O L E N
60452
60452 a60452  ldy a1337               ; pufferzeiger
60455         cpy #191                ; schon am pufferende?
60457         bcc a60465              ; nein: ->
60459         jsr a59603              ; ---> block in puffer lesen
60462         bcc a60452              ; ok: ->
60464         rts
60465
60465 a60465  ldy a1337               ; pufferzeiger
60468         lda (a182),y            ; byte aus puffer
60470         pha                     ; merken
60471         iny                     ; offset in puffer erhöhen
60472         cpy #191                ; pufferende erreicht?
60474         bcs a60485              ; ja: ->
60476         lda (a182),y            ; nächstes byte aus puffer
60478         bne a60485              ; nicht kode 0: ->
60480         lda #64                 ; eoi-bit
60482         jsr a62494              ; ---> in status odern
60485 a60485  inc a1337               ; pufferzeiger erhöhen
60488         pla                     ; datenbyte aus puffer
60489         clc
60490         rts
60491
60491 ;------------------------------  B S O U T
60491
60491 a60491  pha                     ; datenbyte
60492         lda *a153               ; ausgabekanal
60494         cmp #3                  ; bildschirm?
60496         bne a60502              ; nein: ->
60498         pla                     ; datenbyte
60499         jmp a56393              ; ---> auf bildschirm ausgeben
60502
60502 a60502  bcc a60508              ; tastatur, kasette, rs-232?
60504         pla                     ; datenbyte
60505         jmp a60639              ; ---> ausgabe auf bus
60508
60508 a60508  jsr a64439              ; ---> a, x, y retten
60511         cmp #1                  ; kassette?
60513         bne a60548              ; nein: ->
60515         ldy a1337               ; pufferzeiger
60518         cpy #191                ; puffer voll?
60520         bcc a60533              ; nein: ->
```

```
60522          jsr a58677              ; ---> puffer auf band schreiben
60525          bcs a60542              ; fehler: ->
60527          lda #2
60529          sta *a248               ; filetyp = 'daten'
60531          ldy #0                  ; offset in puffer
60533 a60533   pla                     ; datenbyte
60534          sta (a182),y            ; in puffer schreiben
60536          iny                     ; offset erhöhen
60537          sty a1337               ; pufferzeiger
60540          bcc a60552              ; immer ->
60542
60542 a60542   pla                     ; stack korrigieren
60543          lda #0
60545          jmp a64452              ; ---> x und y wiederherstellen
60548
60548 a60548   pla                     ; datenbyte
60549          jsr a60215              ; ---> auf rs-232 ausgeben
60552 a60552   jmp a64449              ; ---> a, x, y wiederherstellen
60555
60555 ;-------------------------------   A C P T R
60555
60555 a60555   stx *a186
60557          bit *a249               ; iec-bus?
60559          bvs a60566              ; ja: ->
60561          ldx *a186
60563          jmp a57938              ; ---> byte von serial bus holen
60566
60566 a60566   lda *a249               ; iec-bus-flag
60568          and #48                 ; offset
60570          tax                     ; = 48/0 f gerät #8/9
60571          lda #%10000100
60573          sta a65216,x
60576 a60576   lda a65218,x
60579          bmi a60576
60581          lda #0
60583          sta a65219,x
60586          sta a65218,x
60589 a60589   lda a65218,x
60592          bpl a60589
60594          lda a65217,x
60597          and #3
60599          cmp #3
60601          bne a60605
60603          lda #64                 ; eoi-bit
60605 a60605   jsr a62494              ; ---> in status odern
60608          lda a65216,x            ; datenbyte
60611          pha
60612          lda #%01000000
60614          sta a65218,x
60617 a60617   lda a65218,x
60620          bmi a60617
60622          lda #255
60624          sta a65219,x
60627          lda #0
60629          sta a65216,x
60632          sta a65218,x
60635          jmp a60884              ; --->
60638
60638          nop
```

```
60639 ;------------------------------  C I O U T
60639
60639 a60639  bit *a249               ; iec-bus?
60641         bmi a60646              ; ja: ->
60643         jmp a57885              ; ---> byte auf serial bus senden
60646
60646 a60646  pha
60647         sta a1512
60650         lda #%10000011
60652 a60652  stx *a186
60654         pha
60655         lda *a249               ; iec-bus-flag
60657         and #48                 ; offset
60659         tax                     ; = 48/0 für gerät #8/9
60660         pla
60661         sta a65216,x
60664 a60664  lda a65218,x
60667         bmi a60664
60669         lda a1512               ; datenbyte
60672         sta a65216,x
60675         lda #0
60677         sta a65218,x
60680 a60680  lda a65218,x
60683         bpl a60680
60685         lda a65217,x
60688         and #3
60690         jsr a62494              ; ---> in status odern
60693         jmp a60891              ; ---> abschlu
60696
60696 ;------------------------------  C H K I N
60696
60696 a60696  jsr a61160              ; ---> file in tabelle suchen
60699         beq a60704              ; gefunden: ->
60701         jmp a62073              ; ---> i/o-error #3: 'file not open'
60704
60704 a60704  jsr a61176              ; ---> file-daten aus tabelle übernehmen
60707         beq a60726              ; tastatur: ->
60709         cmp #3
60711         beq a60726              ; bildschirm: ->
60713         bcs a60730              ; bus: ->
60715         cmp #2
60717         bne a60759              ; kassette: ->
60719         jsr a60197              ; ---> chkin von rs-232
60722         bcs a60729              ; fehler: ->
60724         lda *a174               ; geräteadresse
60726 a60726  sta *a152               ; =: eingabekanal
60728         clc
60729 a60729  rts
60730
60730 a60730  tax
60731         jsr a60922              ; ---> talk
60734         bit *a144               ; status
60736         bmi a60756              ; bit 7 gesetzt: fehler ->
60738         lda *a173               ; sekundäradresse
60740         bpl a60748
60742         jsr a60947              ; ---> 'atn' ausgeben
60745         jmp a60751              ; --->
60748
60748 a60748  jsr a60954              ; ---> tksa
60751 a60751  txa
60752         bit *a144               ; status
60754         bpl a60726              ; ok: ->
60756 a60756  jmp a62079              ; ---> i/o-error #5: 'device not present'
```

```
60759 a60759  ldx *a173              ; sekundäradresse
60761         cpx #96                ; = 96?
60763         beq a60726             ; ja: ->
60765         jmp a62082             ; ---> i/o-error #6: not input file
60768
60768 ;-------------------------------  C K O U T
60768
60768 a60768  jsr a61160             ; ---> file in tabelle suchen
60771         beq a60776             ; gefunden: ->
60773         jmp a62073             ; ---> i/o-error #3: file not found
60776
60776 a60776  jsr a61176             ; ---> file-daten aus tabelle übernehmen
60779         bne a60784             ; nicht tastatur: ->
60781 a60781  jmp a62085             ; ---> i/o-error #7: not output file
60784
60784 a60784  cmp #3
60786         beq a60801             ; bildschirm ->
60788         bcs a60805             ; bus ->
60790         cmp #2
60792         bne a60833             ; kassette ->
60794         jsr a60197             ; ---> rs-232
60797         bcs a60804             ; fehler: ->
60799         lda *a174              ; geräteadresse
60801 a60801  sta *a153              ; ausgabekanal
60803         clc
60804 a60804  rts
60805
60805 a60805  tax
60806         jsr a60972             ; ---> listen
60809         bit *a144              ; status
60811         bmi a60830             ; bit 7 gesetzt: fehler ->
60813         lda *a173              ; sekundäradresse
60815         bpl a60822
60817         jsr a60997             ; ---> 'atn' serial bus
60820         bne a60825             ; immer ->
60822
60822 a60822  jsr a61005             ; ---> second
60825 a60825  txa
60826         bit *a144              ; status
60828         bpl a60801             ; ok: ->
60830 a60830  jmp a62079             ; ---> i/o-error #5: 'device not present'
60833
60833 a60833  ldx *a173              ; sekundäradresse
60835         cpx #96                ; = 96?
60837         beq a60781             ; ja: fehler ->
60839         bne a60801             ; ok ->
60841
60841 ;-------------------------------  I E C - B U S  P R Ü F E N
60841
60841 a60841  pha
60842         stx *a186
60844         ldx #48                ; offset für geräteadresse 8
60846         lda *a174              ; geräteadresse
60848         cmp #8                 ; = 8 oder 9?
60850         beq a60858
60852         cmp #9
60854         bne a60879             ; nein: ->
60856         ldx #0                 ; offset für geräteadresse 9
60858 a60858  lda #85                ; ist adresse
60860         sta a65216,x           ; a65216+offset
60863         eor a65216,x           ; eine ram-adresse?
60866         bne a60879             ; nein: kein iec-bus ->
```

```
60868          lda a65217,x            ; ist an der nächsten adresse
60871          and #2                  ; bit 1 gesetzt?
60873          bne a60879              ; ja: kein iec-bus ->
60875          stx *a249               ; offset abspeichern
60877          clc                     ; flag für 'iec-bus vorhanden'
60878          .by 36
60879 a60879   sec                     ; flag für 'kein iec-bus'
60880          ldx *a186               ; x wiederherstellen
60882          pla
60883          rts
60884
60884 ;------------------------------   I E C - A B S C H L U S S
60884
60884 ;                                 e i n g a b e
60884
60884 a60884   lda a65218,x
60887          bpl a60884
60889          bmi a60896
60891
60891 ;                                 a u s g a b e
60891
60891 a60891   lda #0
60893          sta a65216,x
60896 a60896   lda #64
60898          sta a65218,x
60901          ldx *a186
60903          pla
60904          clc
60905          rts
60906
60906 ;------------------------------   R U C K S A C K   V O N   I O I N I T
60906
60906 a60906   sta a65266
60909          sta a65221
60912          sta a65218
60915          dex
60916          stx a65219
60919          jmp a53226              ; --->
60922
60922 ;------------------------------   T A L K
60922
60922 a60922   jsr a60841              ; ---> iec-bus?
60925          bcc a60930              ; ja: ->
60927          jmp a57683              ; ---> talk serial bus
60930
60930 a60930   pha
60931          lda #64
60933          sta a1512               ; ausgaberegister
60936          lda *a249               ; iec-bus-flag
60938          ora #64                 ; talk-bit setzen
60940          sta *a249
60942          lda #%10000001
60944          jmp a60652              ; ---> 'talk' ausgeben
60947
60947 ;------------------------------   T K A T N
60947
60947 a60947   bit *a249               ; iec-bus?
60949          bvs a61004              ; ja: ->
60951          jmp a57868              ; ---> 'atn' serial bus
```

```
60954 ;------------------------------  T K S A
60954
60954 a60954  bit *a249               ; iec-bus?
60956         bvs a60961              ; ja: ->
60958         jmp a57859              ; ---> talk serial bus
60961
60961 a60961  pha
60962         lda *a173               ; sekundäradresse
60964         sta a1512               ; in ausgaberegister
60967         lda #%10000010
60969         jmp a60652              ; ---> sekundäradresse ausgeben
60972
60972 ;------------------------------  L I S T E N
60972
60972 a60972  jsr a60841              ; ---> iec-bus?
60975         bcc a60980              ; ja: ->
60977         jmp a57686              ; ---> listen serial bus
60980
60980 a60980  pha
60981         lda #32                 ; listen-bit
60983         sta a1512               ; ausgaberegister
60986         lda *a249               ; iec-bus-flag
60988         ora #128                ; listen-bit setzen
60990         sta *a249
60992         lda #%10000001
60994         jmp a60652              ; ---> 'listen' ausgeben
60997
60997 ;------------------------------  L S N A T N
60997
60997 a60997  bit *a249               ; iec-bus?
60999         bmi a61004              ; ja: fertig ->
61001         jmp a57852              ; ---> 'atn' serial bus
61004
61004 a61004  rts
61005
61005 ;------------------------------  S E C O N D
61005
61005 a61005  bit *a249               ; iec-bus?
61007         bmi a61012              ; ja: ->
61009         jmp a57847              ; ---> second serial bus
61012
61012 a61012  pha
61013         sta a1512               ; sekundäradresse in ausgaberegister
61016         lda #%10000010
61018         jmp a60652              ; ---> sekundäradresse ausgeben
61021
61021 ;------------------------------  C L O S E
61021
61021 a61021  ror *a186               ; carry-bit merken
61023         jsr a61165              ; ---> file in tabelle suchen
61026         beq a61030              ; gefunden: ->
61028         clc
61029         rts
61030
61030 a61030  jsr a61176              ; ---> file-daten aus tabelle übernehmen
61033         txa                     ; geräteadresse
61034         pha
61035         lda *a174               ; log adresse
61037         beq a61130              ; tastatur: ->
61039         cmp #3
61041         beq a61130              ; bildschirm: ->
```

```
61043          bcs a61109             ; bus: ->
61045          cmp #2
61047          bne a61057             ; kassette: ->
61049          php
61050          sei
61051          jsr a60230             ; ---> arbeitsbereich löschen
61054          plp
61055          beq a61130             ; immer ->
61057
61057 a61057   lda *a173              ; sekundäradresse
61059          and #15
61061          beq a61130             ; = 0: 'read' ->
61063          ldy a1337              ; kassettenpuffer-zeiger
61066          cpy #191
61068          bcc a61084             ; puffer noch nicht voll: ->
61070          jsr a58677             ; ---> puffer auf band schreiben
61073          bcs a61093             ; fehler: ->
61075          lda #2                 ; filetyp: 'daten'
61077          sta *a248
61079          ldy #0                 ; pufferzeiger rücksetzen
61081          sty a1337
61084 a61084   lda #0
61086          sta (a182),y           ; puffer mit kode '0' abschlie^en
61088          jsr a58677             ; ---> puffer auf band schreiben
61091          bcc a61097             ; ok: ->
61093 a61093   pla
61094          lda #0
61096          rts
61097
61097 a61097   lda *a173              ; sekundäradresse
61099          cmp #98                ; 'end-of-tape'-block gefordert?
61101          bne a61130             ; nein: ->
61103          jsr a58864             ; ---> 'eot'-block schreiben
61106          jmp a61130             ; ---> file-tabelle fortschreiben
61109
61109 a61109   bit *a186              ; aufruf mit 'sec'?
61111          bpl a61127             ; nein: ->
61113          lda *a174              ; geräteadresse
61115          cmp #8                 ; kleiner als 8?
61117          bcc a61127             ; ja: ->
61119          lda *a173              ; sekundäradresse
61121          and #15
61123          cmp #15                ; kanal 15?
61125          beq a61130             ; ja: ->
61127 a61127   jsr a61969             ; ---> floppy-kanal schlieen
61130 a61130   pla
61131          tax                    ; file-daten aus tabelle entfernen
61132          dec *a151
61134          cpx *a151
61136          beq a61158
61138          ldy *a151
61140          lda a1289,y
61143          sta a1289,x
61146          lda a1299,y
61149          sta a1299,x
61152          lda a1309,y
61155          sta a1309,x
61158 a61158   clc
61159          rts
```

```
61160 ;-------------------------  F I L E   I N   T A B E L L E   S U C H E N
61160
61160 a61160  lda #0
61162         sta *a144              ; status löschen
61164         txa                    ; log adresse
61165 a61165  ldx *a151              ; file-tabellenindex
61167 a61167  dex
61168         bmi a61191             ; nicht gefunden: ->
61170         cmp a1289,x
61173         bne a61167
61175         rts
61176
61176 a61176  lda a1289,x            ; log adresse
61179         sta *a172              ; la
61181         lda a1309,x            ; sekundäradresse
61184         sta *a173              ; sa
61186         lda a1299,x            ; geräteadresse
61189         sta *a174              ; fa
61191 a61191  rts
61192
61192 ;------------------------------  C L A L L
61192
61192 a61192  lda #0
61194         sta *a151              ; ldtnd: filetabelle löschen
61196
61196 ;------------------------------  C L R C H
61196
61196 a61196  ldx #3
61198         cpx *a153              ; ausgabekanal
61200         bcs a61205             ; nicht bus: ->
61202         jsr a61219             ; ---> unlisten
61205 a61205  cpx *a152              ; eingabekanal
61207         bcs a61212             ; nicht bus: ->
61209         jsr a61243             ; ---> untalk
61212 a61212  stx *a153              ; ausgabekanal = bildschirm
61214         lda #0
61216         sta *a152              ; eingabekanal = tastatur
61218         rts
61219
61219 ;------------------------------  U N L I S T E N
61219
61219 a61219  bit *a249              ; iec-bus?
61221         bmi a61226             ; ja: ->
61223         jmp a57917             ; ---> unlisten serial bus
61226
61226 a61226  pha
61227         lda #63
61229         sta a1512              ; ausgaberegister
61232         lda *a249              ; iec-bus-flag
61234         and #127               ; listen-bit löschen
61236         sta *a249
61238         lda #%10000001
61240         jmp a60652             ; ---> 'unlisten' ausgeben
61243
61243 ;------------------------------  U N T A L K
61243
61243 a61243  bit *a249              ; iec-bus?
61245         bvs a61250             ; ja: ->
61247         jmp a57903             ; ---> untalk serial bus
```

```
61250 a61250  pha
61251         lda #95
61253         sta a1512              ; ausgaberegister
61256         lda *a249              ; iec-bus-flag
61258         and #191               ; talk-bit löschen
61260         sta *a249
61262         lda #%10000001
61264         jmp a60652             ; ---> 'untalk' ausgeben
61267
61267 ;------------------------------  O P E N
61267
61267 a61267  ldx *a172              ; log adresse
61269         jsr a61160             ; ---> in tabelle suchen
61272         bne a61277             ; nicht gefunden: ->
61274         jmp a62070             ; ---> i/o-error #2: 'file open'
61277
61277 a61277  ldx *a151              ; file-tabellenindex
61279         cpx #10                ; kleiner als 10?
61281         bcc a61286             ; ja: ->
61283         jmp a62067             ; ---> i/o-error #1: 'too many files'
61286
61286 a61286  inc *a151              ; file-tabellenindex
61288         lda *a172              ; log adresse
61290         sta a1289,x            ; in la-tabelle
61293         lda *a173              ; sekundäradresse
61295         ora #%01100000         ; bits 5 und 6 setzen
61297         sta *a173              ; wieder abspeichern
61299         sta a1309,x            ; und in sa-tabelle
61302         lda *a174              ; geräteadresse
61304         sta a1299,x            ; in fa-tabelle
61307         beq a61318             ; tastatur: ->
61309         cmp #3
61311         beq a61318             ; bildschirm: ->
61313         bcc a61320             ; kassette oder rs-232: ->
61315         jsr a61445             ; ---> openi: bus eröffnen
61318 a61318  clc                    ; ok-flag
61319         rts
61320
61320 a61320  cmp #2
61322         bne a61368             ; kassette ->
61324
61324 ;------------------------------  O P E N   R S - 2 3 2
61324
61324         jsr a60230             ; ---> arbeitsbereich löschen
61327         tax                    ; = 255
61328 a61328  inx
61329         beq a61342             ; ??? immer! rest wird nie ausgeführt ->
61331         stx a64771             ; 6551 control register
61334         cpx a64771
61337         beq a61328
61339         jmp a62079             ; ---> i/o-error #5: 'device not present'
61342
61342 a61342  sec
61343         ror a2008              ; rs-open-flag setzen
61346         lda #a175              ; 1,adresse des file-namens
61348         sta a2015
61351         ldy #0
61353         jsr a2009              ; ---> lda (a),y: 1. zeichen aus filenamen
61356         sta a64771             ; 6551 control register
61359         iny
61360         jsr a2009              ; ---> lda (a),y: 2. zeichen aus filenamen
61363         sta a64770             ; 6551 command register
61366         clc
61367         rts
```

```
61368 ;------------------------------  O P E N  K A S S E T T E
61368
61368 a61368  lda *a173               ; sekundäradresse
61370         and #15
61372         bne a61418              ; ungleich 0: 'write' ->
61374         jsr a58139              ; ---> play-taste prüfen ggf anfordern
61377         bcs a61417              ; stoptaste gedrückt: ->
61379         jsr a61792              ; ---> 'searching for ...' ausgeben
61382         lda *a171               ; fnlen: länge des filenamens
61384         beq a61396              ; kein filename: ->
61386         jsr a59937              ; ---> file header auf band suchen
61389         bcc a61407              ; gefunden: ->
61391         beq a61417
61393 a61393  jmp a62076              ; ---> i/o-error #4: 'file not found'
61396
61396 a61396  jsr a59852              ; ---> nächsten block in puffer lesen
61399         beq a61417
61401         bcs a61393              ; keinen block gefunden: ->
61403         cmp #5                  ; 'end of tape'-block?
61405         beq a61393              ; ja: ->
61407 a61407  ldy #191                ; pufferzeiger initialisieren
61409         sty a1337
61412         lda #2                  ; kennzeichen f 'datenblock'
61414         sta *a248
61416 a61416  clc
61417 a61417  rts
61418
61418 a61418  jsr a58137              ; ---> play-taste prüfen ggf. anfordern
61421         bcs a61417              ; stoptaste gedrückt: ->
61423         lda #4                  ; kennzeichen f 'headerblock'
61425         sta *a248
61427         jsr a58732              ; ---> file header auf band schreiben
61430         bcs a61444
61432         lda #2                  ; kennzeichen f 'datenblock'
61434         sta *a248
61436         ldy #0
61438         sty a1337               ; pufferzeiger
61441         sty a1335
61444 a61444  rts
61445
61445 ;------------------------------  O P E N  B U S
61445
61445 a61445  lda *a173               ; sa: sekundäradresse
61447         bmi a61416              ; > 127: kanal bereits offen ->
61449         ldy *a171               ; fnlen: filename länge
61451         beq a61416              ; kein filename: fertig ->
61453         lda #0
61455         sta *a144               ; status
61457         lda *a174               ; geräteadresse
61459         jsr a60972              ; ---> listen
61462         bit *a144               ; status
61464         bmi a61477              ; gesetzt: ->
61466         lda *a173               ; sa
61468         ora #%11110000
61470         jsr a61005              ; ---> second
61473         lda *a144               ; status
61475         bpl a61482              ; ok: ->
61477 a61477  pla
61478         pla
61479         jmp a62079              ; ---> i/o-error #5: 'device not present'
```

```
61482 a61482  lda *a171             ; fnlen: filename länge
61484         beq a61504            ; kein filename: ->
61486         ldy #0
61488 a61488  lda #a175             ; l,fnadr: filename adresse
61490         sta a2015
61493         jsr a2009             ; ---> lda (a),y
61496         jsr a60639            ; ---> ciout: zeichen auf bus ausgeben
61499         iny
61500         cpy *a171             ; filename zu ende?
61502         bne a61488            ; nein: ->
61504 a61504  jmp a61987            ; ---> unlisten
61507
61507 ;------------------------------  L O A D  /  V E R I F Y
61507
61507 a61507  stx *a180             ; l,ladeadresse
61509         sty *a181             ; h,...
61511         jmp (a814)            ; ---> load-routine (a61514)
61514
61514 a61514  sta *a147             ; verck: verify flag
61516         lda #0
61518         sta *a144             ; status
61520         lda *a174             ; geräteadresse
61522         bne a61527            ; nicht tastatur: ->
61524 a61524  jmp a62091            ; ---> illegal device number
61527
61527 a61527  cmp #3
61529         beq a61524            ; bildschirm: ->
61531         bcs a61540            ; bus: ->
61533         cmp #2
61535         beq a61524
61537         jmp a61680            ; ---> laden von kassette
61540
61540 a61540  ldy *a171             ; fnlen: länge des file namens
61542         bne a61547            ; filename vorhanden: ->
61544         jmp a62088            ; ---> file name missing
61547
61547 a61547  ldx *a173             ; sekundäradresse
61549         jsr a61792            ; ---> 'searching for ...' ausgeben
61552         lda #96               ; sekundäradresse f 'load'
61554         sta *a173
61556         jsr a61445            ; ---> open serial bus
61559         lda *a174             ; geräteadresse
61561         jsr a60922            ; ---> talk
61564         lda *a173             ; sekundäradresse
61566         jsr a60954            ; ---> tksa: sekundäradresse ausgeben
61569         jsr a60555            ; ---> acptr: byte vom bus holen
61572         sta *a157             ; eal: l,ladeadresse
61574         lda *a144             ; status
61576         lsr a
61577         lsr a
61578         bcs a61672            ; timout: ->
61580         jsr a60555            ; ---> acptr: byte vom bus holen
61583         sta *a158             ; eah: h,ladeadresse
61585         txa                   ; sekundäradresse nicht null:
61586         bne a61596            ; linkbytes als ladeadresse verwenden ->
61588         lda *a180             ; sonst: ladeadresse aus (a180,a181)
61590         sta *a157
61592         lda *a181
61594         sta *a158
61596 a61596  jsr a61833            ; ---> 'loading' bzw. 'verifying' ausgeben
```

```
61599 a61599  lda #253              ; timeout bit
61601         and *a144
61603         sta *a144             ; rücksetzen
61605         jsr a65505            ; ---> stoptaste gedrückt?
61608         bne a61613            ; nein: ->
61610         jmp a61951            ; ---> close bus
61613
61613 a61613  jsr a60555            ; ---> acptr: byte vom bus holen
61616         tax
61617         lda *a144             ; status
61619         lsr a
61620         lsr a
61621         bcs a61599            ; timeout: ladeversuch wiederholen ->
61623         txa
61624         ldy *a147             ; verck: verify flag
61626         beq a61652            ; 'load': ->
61628         ldy #0
61630         sta a1991             ; geladenes byte zwischenspeichern
61633         lda #a157
61635         sta a2015
61638         jsr a2009             ; ---> lda (a),y: byte im speicher
61641         cmp a1991             ; mit geladenem byte identisch?
61644         beq a61654            ; ja: ->
61646         lda #16               ; 'verify error'-bit
61648         jsr a62494            ; ---> in status odern
61651         .by 44
61652 a61652  sta (a157),y          ; geladenes byte speichern
61654 a61654  inc *a157             ; ladezeiger erhöhen
61656         bne a61660
61658         inc *a158
61660 a61660  bit *a144             ; eoi-bit gesetzt?
61662         bvc a61599            ; nein: ->
61664         jsr a61243            ; ---> untalk
61667         jsr a61969            ; ---> close bus
61670         bcc a61675            ; ok: ->
61672 a61672  jmp a62076            ; ---> file not found
61675
61675 a61675  ldx *a157             ; l,lade-endadresse
61677         ldy *a158             ; h,...
61679 a61679  rts
61680
61680 a61680  jsr a58139            ; ---> play-taste prüfen ggf. anfordern
61683         bcs a61679            ; stoptaste gedrückt: ->
61685         jsr a61792            ; ---> 'searching for ...' ausgeben
61688 a61688  lda *a171             ; fnlen: filename-länge
61690         beq a61701            ; kein filename: ->
61692         jsr a59937            ; ---> file header auf band suchen
61695         bcc a61708            ; gefunden: ->
61697         beq a61679            ; 'end-of-tape'-block gefunden: ->
61699         bcs a61672            ; header nicht gefunden: ->
61701
61701 a61701  jsr a59852            ; ---> block von band lesen
61704         beq a61679            ; 'end-of-tape'-block gefunden: ->
61706         bcs a61672            ; keinen block gefunden: ->
61708 a61708  lda *a248             ; filetyp
61710         cmp #1                ; 'programm'?
61712         beq a61732            ; ja: ->
61714         cmp #3                ; mit sekundäradresse 1 geschrieben?
61716         bne a61688            ; nein: weitersuchen ->
```

```
61718 a61718  ldy #0
61720         lda (a182),y          ; l,ladeadresse aus file header
61722         sta *a180
61724         iny
61725         lda (a182),y          ; h,...
61727         sta *a181
61729         jmp a61736            ; --->
61732
61732 a61732  lda *a173             ; sekundäradresse
61734         bne a61718            ; nicht null: ladeadresse übernehmen ->
61736 a61736  sec
61737         ldy #2
61739         lda (a182),y          ; l,endadresse aus header
61741         ldy #0
61743         sbc (a182),y          ; - l,anfangsadresse aus header
61745         tax                   ; = l,file-länge
61746         ldy #3
61748         lda (a182),y          ; h,endadresse aus header
61750         ldy #1
61752         sbc (a182),y          ; - h,anfangsadresse aus header
61754         tay                   ; = h,file-länge
61755         clc
61756         txa                   ; l,file-länge
61757         adc *a180             ; + l,ladeanfangsadresse
61759         sta *a157             ; = l,endadresse im speicher
61761         tya                   ; h,file-länge
61762         adc *a181             ; + h,ladeanfangsadresse
61764         sta *a158             ; = h,endadresse im speicher
61766         lda *a180             ; l,ladeanfangsadresse
61768         sta *a178             ; l,ladezeiger
61770         lda *a181             ; h,ladeanfangsadresse
61772         sta *a179             ; h,ladezeiger
61774         jsr a61833            ; ---> 'loading ...' ausgeben
61777         jsr a59635            ; ---> file vom band laden
61780         bcc a61675            ; ok: ->
61782         lda #29               ; 'load'
61784         bit *a147             ; verck: verify-flag
61786         bpl a61679            ; load: ->
61788         lda #28               ; 'verify'
61790         bne a61679            ; immer ->
61792
61792 a61792  lda *a154             ; msgflg: flag f meldungen
61794         bpl a61832            ; nicht gesetzt: ->
61796         ldy #12               ; 'searching'
61798         jsr a60362            ; ---> ausgeben
61801         lda *a171             ; fnlen: filename-länge
61803         beq a61832            ; kein filename: ->
61805         ldy #23               ; 'for'
61807         jsr a60362            ; ---> ausgeben
61810 a61810  ldy *a171             ; fnlen: filename-länge
61812         beq a61832            ; kein filename: ->
61814         ldy #0
61816 a61816  lda #a175             ; l,fnadr: adresse des filenamens
61818         sta a2015
61821         jsr a2009             ; ---> lda (a),y
61824         jsr a65490            ; ---> bsout
61827         iny
61828         cpy *a171             ; filename schon ganz ausgegeben?
61830         bne a61816            ; nein: ->
61832 a61832  rts
```

```
61833 a61833  ldy #73                 ; 'loading'
61835         lda *a147               ; verck: verify-flag
61837         beq a61841              ; load: ->
61839         ldy #89                 ; 'verifying'
61841 a61841  jmp a60358              ; ---> meldung ausgeben
61844
61844 ;-------------------------------  S A V E
61844
61844 a61844  stx *a157               ; l,endadresse
61846         sty *a158               ; h,...
61848         tax                     ; zeiger auf anfangsadresse
61849         lda *a0,x               ; l,anfangsadresse
61851         sta *a178
61853         lda *a1,x               ; h,...
61855         sta *a179
61857         jmp (a816)              ; ---> (a61860)
61860
61860 a61860  lda *a174               ; log adresse
61862         bne a61867              ; nicht tastatur: ->
61864 a61864  jmp a62091              ; ---> illegal device number
61867
61867 a61867  cmp #3                  ; bildschirm?
61869         beq a61864              ; ja: fehler ->
61871         cmp #2                  ; rs-232?
61873         beq a61864              ; ja: fehler ->
61875         bcc a62004              ; kassette: ->
61877         lda #97                 ; sekundäradresse f save
61879         sta *a173               ; sa
61881         ldy *a171               ; fnlen: filename-länge
61883         bne a61888              ; filename vorhanden: ->
61885         jmp a62088              ; ---> 'file name missing'
61888
61888 a61888  jsr a61445              ; ---> open bus
61891         jsr a61992              ; ---> 'saving ...' ausgeben
61894         lda *a174               ; fa: geräteadresse
61896         jsr a60972              ; ---> listen
61899         lda *a173               ; sa: sekundäradresse
61901         jsr a61005              ; ---> second
61904         ldy #0
61906         lda *a179               ; h,save-anfangsadresse
61908         sta *a156               ; h,savezeiger
61910         lda *a178               ; l,save-anfangsadresse
61912         sta *a155               ; l,savezeiger
61914         lda *a155
61916         jsr a60639              ; ---> ciout: byte auf bus ausgeben
61919         lda *a156               ; h,savezeiger
61921         jsr a60639              ; ---> ciout: byte auf bus ausgeben
61924 a61924  sec
61925         lda *a155               ; save-zeiger
61927         sbc *a157               ; - save-endadresse
61929         lda *a156
61931         sbc *a158
61933         bcs a61966              ; nicht negativ: fertig ->
61935         lda #a155               ; save-zeiger
61937         sta a2015
61940         jsr a2009               ; ---> lda (a),y
61943         jsr a60639              ; ---> ciout: byte auf bus ausgeben
61946         jsr a65505              ; ---> stoptaste gedrückt?
61949         bne a61958              ; nein: ->
61951 a61951  jsr a61969              ; ---> close bus
61954         lda #0
61956         sec
61957         rts
```

```
61958 a61958  inc *a155              ; save-zeiger inkrementieren
61960         bne a61924
61962         inc *a156
61964         bne a61924             ; immer ->
61966
61966 a61966  jsr a61219             ; ---> unlisten
61969 a61969  bit *a173              ; sekundäradresse
61971         bmi a61990             ; bit 7 gesetzt: ->
61973         lda *a174              ; geräteadresse
61975         jsr a60972             ; ---> listen
61978         lda *a173              ; skundäradresse
61980         and #%11101111
61982         ora #%11100000
61984         jsr a61005             ; ---> second
61987 a61987  jsr a61219             ; ---> unlisten
61990 a61990  clc
61991         rts
61992
61992 a61992  lda *a154              ; msgflg: flag f meldungen
61994         bpl a62052             ; nicht gesetzt: ->
61996         ldy #81                ; 'saving'
61998         jsr a60362             ; ---> meldung ausgeben
62001         jmp a61810             ; ---> filenamen ausgeben
62004
62004 a62004  jsr a58137             ; ---> record-taste prüfen ggf anfordern
62007         bcs a62050             ; stoptaste gedrückt: ->
62009         jsr a61992             ; ---> 'saving ...' ausgeben
62012         ldx #3
62014         lda *a173              ; sekundäradresse
62016         and #1
62018         bne a62022             ; enthält 1: ->
62020         ldx #1
62022 a62022  stx *a248              ; filetyp
62024         jsr a58732             ; ---> file header auf band schreiben
62027         bcs a62050             ; fehler: ->
62029         lda #0
62031         sta *a248              ; filetyp
62033         jsr a58800             ; ---> file auf band schreiben
62036         bcs a62050             ; fehler: ->
62038         lda *a173              ; sekundäradresse
62040         and #2                 ; enthält 2?
62042         beq a62049             ; nein: ->
62044         jsr a58864             ; ---> 'end-of-tape'-block schreiben
62047         bcs a62050             ; fehler: ->
62049 a62049  clc                    ; 'ok'-flag
62050 a62050  lda #0
62052 a62052  rts
62053
62053 ;------------------------------  S T O P
62053
62053 a62053  lda *a145              ; stkey
62055         cmp #127               ; stoptaste gedrückt?
62057         bne a62066             ; nein: ->
62059         php
62060         jsr a65484             ; ---> clrch
62063         sta *a239              ; ndx: tastaturpuffer löschen
62065         plp
62066 a62066  rts
```

```
62067 ;------------------------------  I / O - F E H L E R
62067
62067 a62067  lda #1                  ; 'too many files"
62069         .by 44
62070 a62070  lda #2                  ; 'file open'
62072         .by 44
62073 a62073  lda #3                  ; 'file not open'
62075         .by 44
62076 a62076  lda #4                  ; 'file not found'
62078         .by 44
62079 a62079  lda #5                  ; 'device not present'
62081         .by 44
62082 a62082  lda #6                  ; 'not input file'
62084         .by 44
62085 a62085  lda #7                  ; 'not output file'
62087         .by 44
62088 a62088  lda #8                  ; 'missing file name'
62090         .by 44
62091 a62091  lda #9                  ;  'illegal divice number'
62093         pha
62094         jsr a65484              ; ---> clrch
62097         ldy #0
62099         bit *a154               ; msgflg: flag f meldungen
62101         bvc a62113              ; nicht gesetzt: ->
62103         jsr a60362              ; ---> meldung ausgeben
62106         pla
62107         pha
62108         ora #48                 ; nummer der meldung
62110         jsr a65490              ; ---> bsout
62113 a62113  pla
62114         sec
62115         rts
62116
62116 ;------------------------------  N M I ,  S T A R T
62116
62116 a62116  ldx #255
62118         sei
62119         txs                     ; prozessor-stack initialisieren
62120         cld
62121         jsr a53158              ; ---> modultab anlegen, vorh moduln init
62124         jsr a62219              ; ---> ioinit
62127         jsr a53009              ; ---> stoptaste dekodieren
62130         php                     ; ergebnis merken
62131         bmi a62140              ; taste nicht gedrückt: ->
62133         lda #165
62135         cmp a1288               ; ramtas bereits ausgeführt?
62138         beq a62143              ; ja: ->
62140 a62140  jsr a62290              ; ---> ramtas: speicherpruefung
62143 a62143  jsr a62158              ; ---> restor: vektoren initialisieren
62146         jsr a55374              ; ---> editor reset
62149         plp                     ; war stoptaste gedrückt?
62150         bmi a62155              ; nein: ->
62152         jmp a62533              ; ---> monitor call
62155
62155 a62155  jmp a32768              ; ---> basic kaltstart
62158
62158 ;------------------------------  R E S T O R
62158
62158 a62158  ldx #l,a62187           ; l,vektorentabelle
62160         ldy #h,a62187           ; h,...
62162         clc
```

```
62163 ;------------------------------  V E C T O R
62163
62163 a62163  stx *a184               ; bei aufruf mit 'sec' werden die
62165         sty *a185               ; vektoren aus dem bereich a786,...,a817
62167         ldy #31                 ; an die durch (x,y) vorgegebene
62169 a62169  lda a786,y              ; adresse kopiert.
62172         bcs a62176
62174         lda (a184),y            ; nach 'clc' erfolgt der kopiervorgang
62176 a62176  sta a786,y              ; in umgekehrter richtung
62179         bcc a62183
62181         sta (a184),y
62183 a62183  dey
62184         bpl a62169
62186         rts
62187
62187 a62187  .si a52802             ; (a786) interrupt intern (ohne i/o)
62189         .si a52750              ; (a788) interrupt
62191         .si a62540              ; (a790) monitor break entry
62193         .si a61267              ; (a792) open
62195         .si a61021              ; (a794) close
62197         .si a60696              ; (a796) chkin
62199         .si a60768              ; (a798) ckout
62201         .si a61196              ; (a800) clrch
62203         .si a60392              ; (a802) basin
62205         .si a60491              ; (a804) bsout
62207         .si a62053              ; (a806) stop
62209         .si a60377              ; (a808) getin
62211         .si a61192              ; (a810) clall
62213         .si a62540              ; (a812) monitor break entry
62215         .si a61514              ; (a814) load (a=0) / verify (a>0)
62217         .si a61860              ; (a816) save
62219
62219 ;------------------------------  I O I N I T
62219
62219 a62219  lda #%00001111          ; bits 0,...,3: output
62221         sta *a0                 ; bits 4,...,7: input
62223         lda #%00001000
62225         sta *a1                 ; kassettenmotor abschalten
62227         ldx #255
62229         stx a64784              ;
62232         stx a65267              ; iec-port #8: output
62235         inx
62236         stx a65268
62239         stx a65264              ; iec-port #8: daten
62242         lda #64
62244         sta a65269
62247         jsr a60906              ; --->
62250 a62250  lda a62264,x            ; bildschirm-controller
62253         sta a65280,x            ; programmieren
62256         inx
62257         cpx #26
62259         bne a62250
62261         jmp a60230              ; --->
62264
62264 a62264  .by 241 57 0 0 0 0 27 8 0 0     ; parameter für bildschirm-
62274         .by 2 204 0 0 0 0 0 0 4 208     ; controller
62284         .by 8 113 91 117 119 110
```

```
62290 ;------------------------------  R A M T A S
62290
62290 a62290  lda #0
62292         tay
62293 a62293  sta a2,y
62296         sta a512,y
62299         sta a768,y
62302         sta a1024,y
62305         sta a1792,y
62308         iny
62309         bne a62293
62311         ldx #8
62313         stx *a159
62315 a62315  lda 65525,x            ; speicherobergrenze ermitteln
62318         sta 65525,x
62321         cmp 16373,x
62324         bne a62327
62326         iny
62327 a62327  cmp 32757,x
62330         bne a62334
62332         dec *a159
62334 a62334  dex
62335         bne a62315
62337         cpy #8
62339         beq a62348
62341         lda *a159
62343         bne a62353
62345         ldy #127               ; memtop = 32758
62347         .by 44
62348 a62348  ldy #63                ; memtop = 16374
62350         ldx #246
62352         .by 44
62353 a62353  ldy #253               ; memtop = 64768
62355         clc
62356         jsr a62511             ; ---> memtop
62359         lda #16
62361         sta a1330              ; membot = 4096
62364         ldx #58
62366 a62366  lda a62418-1,x         ; funktionstasten programmieren
62369         sta a1375-1,x
62372         dex
62373         bne a62366
62375         stx a1373
62378         ldx #11
62380 a62380  lda a53171,x           ; ram-laderoutine kopieren
62383         sta a2009,x
62386         dex
62387         bpl a62380
62389         ldx #15
62391 a62391  lda a57667,x           ; farbkodes ins ram kopieren
62394         sta a275,x
62397         dex
62398         bpl a62391
62400         lda #165
62402         sta a1288              ; flag f 'ramtas ausgeführt'
62405         lda #4
62407         sta a2042
62410         lda #24
62412         sta a2043
62415         rts
62416
62416         .by 234 234
```

```
62418 ;------------------------------  F U N K T I O N S T A S T E N
62418
62418 a62418  .by 7 6 10 7 6 4 5 5    ; längen der texte
62426
62426         .by 'graphic'
62433         .by 'dload"'
62439         .by 'directory' 13
62449         .by 'scnclr' 13
62456         .by 'dsave"'
62462         .by 'run' 13
62466         .by 'list' 13
62471         .by 'help' 13
62476 ;
62476 ;------------------------------  S E T N A M
62476
62476 a62476  sta *a171               ; fnlen: filename-länge
62478         stx *a175               ; l,fnadr: adresse des filenamens
62480         sty *a176               ; h,...
62482         rts
62483
62483 ;------------------------------  S E T L F S
62483
62483 a62483  sta *a172               ; la: logische adresse
62485         stx *a174               ; fa: geräteadresse
62487         sty *a173               ; sa: sekundäradresse
62489         rts
62490
62490 ;------------------------------  S E T M S G
62490
62490 a62490  sta *a154               ; msgflg: flag f meldungen
62492
62492 ;------------------------------  R E A D S T
62492
62492 a62492  lda *a144               ; status
62494 a62494  ora *a144
62496         sta *a144
62498         rts
62499
62499 ;------------------------------  S E T T M O
62499
62499 a62499  sta a1333               ; timeout-flag
62502         rts
62503
62503 ;------------------------------  M E M T O P
62503
62503 a62503  bcc a62511             ; cc: obergrenze setzen ->
62505         ldx a1331               ; l,speicher-obergrenze f basic
62508         ldy a1332               ; h,...
62511 a62511  stx a1331
62514         sty a1332
62517         rts
62518
62518 ;------------------------------  M E M B O T
62518
62518 a62518  bcc a62526             ; cc: untergrenze setzen ->
62520         ldx a1329               ; l,speicher-untergrenze f basic
62523         ldy a1330               ; h,...
62526 a62526  stx a1329
62529         sty a1330
62532         rts
```

```
62533 ;------------------------------  M O N I T O R  C A L L
62533
62533 a62533  ldx #0
62535         stx a1364               ; prozessor-status
62538         beq a62552
62540
62540 ;------------------------------  M O N I T O R  B R E A K
62540
62540 a62540  cld
62541         ldx #5
62543 a62543  pla                     ; brk-adr, p, a, x, y vom stack holen
62544         sta a1362,x             ; und ab a1362 speichern
62547         dex
62548         bpl a62543
62550         ldx #9                  ; f ausgabe 'break'
62552 a62552  stx a2036
62555         lda #%11000000          ; ermöglicht alle meldungen
62557         sta *a154               ; msgflg
62559         tsx
62560         stx a1368
62563         ldx a2036
62566         jsr a53094              ; ---> 'monitor' bzw. 'break' ausgeben
62569         lda a65286
62572         ora #16
62574         sta a65286
62577         lda #0
62579         sta *a161
62581         sta *a162
62583         cli
62584
62584 ;------------------------------  R
62584
62584 a62584  ldx #15
62586         jsr a53094              ; ---> register-kopfleiste ausgeben
62589         lda a1362               ; h,break-adresse
62592         jsr a64272              ; ---> ausgeben
62595         ldy #0
62597 a62597  lda a1363,y             ; l,break-adresse, p, a, x, y, stp
62600         jsr a64261              ; ---> ausgeben, ' ' ausgeben
62603         iny
62604         cpy #6
62606         bcc a62597
62608         bcs a62613              ; cr, eingabeschleife ->
62610
62610 a62610  jsr a64267              ; ---> '?' ausgeben
62613 a62613  jsr a64314              ; ---> cr ausgeben
62616         ldx #0
62618         stx *a243               ; zeiger in eingabe-puffer
62620 a62620  jsr a65487              ; ---> basin
62623         sta a512,x              ; eingabe in puffer speichern
62626         inx
62627         cmp #13
62629         bne a62620
62631         dex
62632         stx *a244               ; puffer-endezeiger
62634 a62634  jsr a64319              ; ---> zeichen aus puffer
62637         beq a62613              ; ende erreicht oder ':' gefunden: ->
62639         cmp #'
62641         beq a62634              ; leerstellen überlesen ->
```

```
62643          ldx #15
62645 a62645   cmp a62832,x          ; zeichen in kommando-tabelle suchen
62648          beq a62655            ; gefunden: ->
62650          dex
62651          bpl a62645
62653          bmi a62610            ; zeichen nicht in tabelle: ->
62655
62655 a62655   cpx #13               ; kommando 'l', 's' oder 'v'?
62657          bcs a62673            ; ja: ->
62659          txa
62660          asl a                 ; offset in adressentabelle
62661          tax
62662          lda a62848+1,x        ; h,routinenadresse
62665          pha
62666          lda a62848,x          ; l,...
62669          pha
62670          jmp a64173            ; ---> adr übernehmen, rts ruft routine
62673
62673 a62673   sta a1371             ; kommando (l, s, v)
62676          jmp a63086            ; ---> kommando ausführen
62679
62679 ;------------------------------  M
62679
62679 a62679   bcs a62689            ; keine adresse: ->
62681          jsr a64347            ; ---> adr in zgr (a161/a162) kopieren
62684          jsr a64173            ; ---> 2. adresse übernehmen
62687          bcc a62695            ; adresse angegeben: ->
62689 a62689   lda #11               ; zeilenzähler setzen
62691          sta *a241
62693          bne a62709            ; immer ->
62695
62695 a62695   jsr a64356            ; ---> länge des bereichs berechnen
62698          lsr a                 ; durch 8 dividieren
62699          ror *a241             ; und in zeilenzähler bringen
62701          lsr a
62702          ror *a241
62704          lsr a
62705          ror *a241
62707          sta *a242
62709 a62709   jsr a65505            ; ---> stoptaste gedrückt?
62712          beq a62727            ; ja: ->
62714          jsr a62874            ; ---> zeile bytes & zeichen ausgeben
62717          lda #8
62719          jsr a64406            ; ---> (a) zum zeiger addieren
62722          jsr a64370            ; ---> zeilenzähler dekrementieren
62725          bcs a62709            ; kein unterlauf: ->
62727 a62727   jmp a62613            ; ---> cr, eingabeschleife
62730
62730 ;------------------------------  ;
62730
62730 a62730   bcs a62727            ; keine adresse: fertig ->
62732          lda *a241
62734          ldy *a242
62736          sta a1363             ; l,programmadresse
62739          sty a1362             ; h,...
62742          ldy #0
62744 a62744   jsr a64173            ; ---> byte übernehmen
62747          bcs a62727            ; zeilenende: fertig ->
62749          lda *a241
62751          sta a1364,y           ; p, a, x, y, stp speichern
62754          iny
62755          cpy #5                ; schon 5 bytes übernommen?
62757          bcc a62744            ; nein: ->
62759          bcs a62727            ; immer ->
```

```
62761 ;------------------------------   >
62761
62761 a62761  bcs a62782              ; keine adresse: ->
62763         jsr a64347              ; ---> adr in zgr (a161/a162) kopieren
62766         ldy #0
62768 a62768  jsr a64173              ; ---> byte übernehmen
62771         bcs a62782              ; zeilenende: ->
62773         lda *a241
62775         sta (a161),y            ; byte unter adresse speichern
62777         iny
62778         cpy #8                  ; schon 8 bytes übernommen?
62780         bcc a62768              ; nein: ->
62782 a62782  jsr a64472              ; ---> 'esc o kursor nach oben' ausgeben
62785         .by 27 'o' 145 0
62789         jsr a62874              ; ---> zeile bytes & zeichen ausgeben
62792         jmp a62613              ; ---> cr, eingabeschleife
62795
62795 ;------------------------------   G
62795
62795 a62795  bcs a62807              ; keine adresse: ->
62797         lda *a241               ; aktuelle adresse
62799         sta a1363               ; als programmadresse übernehmen
62802         lda *a242
62804         sta a1362
62807 a62807  ldx a1368               ; stack zeiger
62810         txs                     ; setzen
62811         ldx #0
62813 a62813  lda a1362,x             ; programmadr u status auf stack legen
62816         pha
62817         inx
62818         cpx #3
62820         bne a62813
62822         ldx a1366               ; register setzen
62825         ldy a1367
62828         lda a1365
62831         rti                     ; go
62832
62832 ;------------------------------   K O M M A N D O - T A B E L L E
62832
62832 a62832  .by 'xmrgtcda.hf>;lsv'
62848
62848 a62848  .si a32771-1            ; x
62850         .si a62679-1            ; m
62852         .si a62584-1            ; r
62854         .si a62795-1            ; g
62856         .si a62929-1            ; t
62858         .si a62926-1            ; c
62860         .si a63268-1            ; d
62862         .si a63775-1            ; a
62864         .si a63775-1            ; .
62866         .si a62990-1            ; h
62868         .si a63242-1            ; f
62870         .si a62761-1            ; >
62872         .si a62730-1            ; ;
```

```
62874 ;--------  1 ZEILE BYTES & ZEICHEN AUSGEBEN
62874
62874 a62874  jsr a64314            ; ---> cr ausgeben
62877         lda #62               ; '>'
62879         jsr a65490            ; ---> bsout
62882         jsr a64251            ; ---> adresse ausgeben
62885         ldy #0
62887 a62887  jsr a53142            ; ---> lda (a161),y: byte holen
62890         jsr a64261            ; ---> ausgeben, ' ' ausgeben
62893         iny
62894         cpy #8                ; schon 8 bytes ausgegeben?
62896         bcc a62887            ; nein: ->
62898         jsr a64472            ; ---> ':' ausgeben, 'rvs' einschalten
62901         .by ':' 18 0
62904         ldy #0
62906 a62906  jsr a53142            ; ---> lda (a161),y: byte holen
62909         and #127              ; bit 7 löschen
62911         cmp #32               ; steuerkode?
62913         bcs a62917            ; nein: ->
62915         lda #'.               ; durch '.' ersetzen
62917 a62917  jsr a65490            ; ---> bsout
62920         iny
62921         cpy #8                ; schon 8 zeichen ausgegeben?
62923         bcc a62906            ; nein: ->
62925         rts
62926
62926 ;------------------------------  C
62926
62926 a62926  lda #0
62928         .by 44
62929
62929 ;------------------------------  T
62929
62929 a62929  lda #128
62931         sta *a187
62933         jsr a64416            ; ---> quelladresse u blocklänge holen
62936         bcs a62986            ; keine adressen: fehler ->
62938         jsr a64173            ; ---> zieladresse übernehmen
62941         bcs a62986            ; keine angabe: fehler ->
62943         jsr a64314            ; ---> cr ausgeben
62946         ldy #0
62948 a62948  jsr a53142            ; ---> lda (a161),y: byte holen
62951         bit *a187
62953         bpl a62957            ; 'compare': ->
62955         sta (a241),y
62957 a62957  cmp (a241),y
62959         beq a62969
62961         jsr a65505            ; ---> stoptaste gedrückt?
62964         beq a62983            ; ja: ->
62966         jsr a64251            ; ---> quelladresse ausgeben
62969 a62969  inc *a241             ; zieladresse erhöhen
62971         bne a62975
62973         inc *a242
62975 a62975  jsr a64404            ; ---> quelladresse erhöhen
62978         jsr a64390            ; ---> zeichenzähler dekrementieren
62981         bcs a62948            ; kein unterlauf: ->
62983 a62983  jmp a62613            ; ---> cr, eingabeschleife
62986
62986 a62986  jmp a62610            ; ---> fehler
62989
62989         nop
```

```
62990 ;------------------------------  H
62990
62990 a62990  jsr a64416              ; ---> adresse und blocklänge holen
62993         bcs a62986              ; keine angabe: fehler ->
62995         ldy #0
62997         jsr a64319              ; ---> zeichen aus puffer
63000         cmp #''
63002         bne a63022
63004         jsr a64319              ; ---> zeichen aus puffer
63007 a63007  sta a605,y
63010         iny
63011         jsr a64319              ; ---> zeichen aus puffer
63014         beq a63043              ; trennzeichen: ->
63016         cpy #32                 ; schon 32 kodes übernommen?
63018         bne a63007              ; nein: ->
63020         beq a63043
63022
63022 a63022  sty a1372
63025         jsr a64171              ; ---> byte übernehmen
63028 a63028  lda *a241
63030         sta a605,y
63033         iny
63034         jsr a64173              ; ---> byte übernehmen
63037         bcs a63043              ; zeilenende: ->
63039         cpy #32                 ; schon 32 bytes übernommen?
63041         bne a63028              ; nein: ->
63043 a63043  sty a1371               ; zeichenzähler
63046         jsr a64314              ; ---> cr ausgeben
63049 a63049  ldx #0
63051         ldy #0
63053 a63053  jsr a53142              ; ---> lda (a161),y: byte holen
63056         cmp a605,x              ; und mit vorgabe vergleichen
63059         bne a63076              ; nicht gleich: ->
63061         iny
63062         inx
63063         cpx a1371               ; alle bytes verglichen?
63066         bne a63053              ; nein: ->
63068         jsr a65505              ; ---> stoptaste gedrückt?
63071         beq a62983              ; ja: ->
63073         jsr a64251              ; ---> adresse ausgeben
63076 a63076  jsr a64404              ; ---> zeiger in block inkrementieren
63079         jsr a64390              ; ---> blocklänge dekrementieren
63082         bcs a63049              ; kein unterlauf: ->
63084         bcc a62983              ; immer ->
63086
63086 ;------------------------------  L , S , V
63086
63086 a63086  ldy #1                  ; vorbesetzung 1 für
63088         sty *a174               ; fa: geräteadresse
63090         sty *a173               ; sa: sekundäradresse
63092         dey                     ; vorbesetzung 0 für
63093         sty *a171               ; fnlen: filename-länge
63095         sty *a144               ; status
63097         sty *a147               ;
63099         lda #h,a605
63101         sta *a176               ; h,fnadr: adresse des filenamens
63103         lda #l,a605
63105         sta *a175               ; l,...
63107 a63107  jsr a64319              ; ---> zeichen aus puffer
63110         beq a63206              ; trennzeichen: ->
```

```
63112          cmp #'                  ; leerstellen überlesen
63114          beq a63107
63116          cmp #'"
63118          bne a63143              ; fehler ->
63120          ldx *a243               ; zeiger in eingabe-puffer
63122 a63122   cpx *a244               ; pufferende erreicht?
63124          bcs a63206              ; ja: ->
63126          lda a512,x              ; zeichen aus puffer
63129          inx
63130          cmp #'"                 ; ende des filenamens?
63132          beq a63147              ; ja: ->
63134          sta (a175),y            ; filenamen in monitor-puffer kopieren
63136          inc *a171               ; fnlen inkrementieren
63138          iny
63139          cpy #17                 ; schon 17 zeichen gelesen?
63141          bcc a63122              ; nein: ->
63143 a63143   jmp a62610              ; ---> fehler
63146
63146          nop
63147
63147 a63147   stx *a243               ; zeiger in eingabe-puffer
63149          jsr a64319              ; ---> zeichen aus puffer
63152          jsr a64173              ; ---> geräteadresse übernehmen
63155          bcs a63206              ; keine adresse: ->
63157          lda *a241               ; geräteadresse
63159          beq a63143              ; = 0: fehler ->
63161          cmp #3
63163          beq a63143              ; = 3: fehler ->
63165          sta *a174               ; fa: geräteadresse
63167          jsr a64173              ; ---> anfangsadresse übernehmen
63170          bcs a63206              ; keine angabe: ->
63172          jsr a64347              ; ---> adr in zgr (a161/a162) kopiere
63175          jsr a64173              ; ---> endadresse übernehmen
63178          bcs a63143              ; keine angabe: fehler ->
63180          jsr a64314              ; ---> cr ausgeben
63183          ldx *a241               ; l,endadresse
63185          ldy *a242               ; h,...
63187          lda a1371               ; kommando
63190          cmp #'s
63192          bne a63143              ; fehler ->
63194          lda #0
63196          sta *a173               ; sa: sekundäradresse
63198          lda #a161               ; zeiger auf anfangsadresse
63200          jsr a65496              ; ---> save
63203 a63203   jmp a62613              ; ---> cr, eingabeschleife
63206
63206 a63206   lda a1371               ; kommando
63209          cmp #'v
63211          beq a63219
63213          cmp #'l
63215          bne a63143              ; fehler ->
63217          lda #0
63219 a63219   jsr a65493              ; ---> load
63222          lda *a144               ; status
63224          and #16                 ; lese- oder verifizierfehler?
63226          beq a63203              ; nein: zur eingabeschleife ->
63228          lda a1371               ; kommando
63231          cmp #'l
63233          beq a63143              ; fehler ->
63235          ldx #42                 ; 'error'
63237          jsr a53094              ; ---> meldung ausgeben
63240          bmi a63203              ; zur eingabeschleife ->
```

```
63242 ;------------------------------  F
63242
63242 a63242  jsr a64416              ; ---> adresse und blocklänge holen
63245         bcs a63143              ; nicht angegeben: fehler ->
63247         jsr a64173              ; ---> füllkode holen
63250         bcs a63143              ; keine angabe: fehler ->
63252         ldy #0
63254 a63254  lda *a241               ; füllkode
63256         sta (a161),y            ; abspeichern
63258         jsr a64404              ; ---> bereichszeiger inkrementieren
63261         jsr a64390              ; ---> blocklänge dekrementieren
63264         bcs a63254              ; kein unterlauf: ->
63266         bcc a63203              ; zur eingabeschleife ->
63268
63268 ;------------------------------  D
63268
63268 a63268  bcs a63278              ; keine adresse: ->
63270         jsr a64347              ; ---> adresse in zeiger kopieren
63273         jsr a64173              ; ---> endadresse übernehmen
63276         bcc a63284              ; endadresse vorhanden: ->
63278 a63278  lda #20
63280         sta *a241               ; zeilenzähler auf 20 setzen
63282         bne a63287              ; immer ->
63284
63284 a63284  jsr a64356              ; ---> bereichslänge ermitteln
63287 a63287  jsr a64314              ; ---> cr ausgeben
63290         jsr a65505              ; ---> stoptaste gedrückt?
63293         beq a63203              ; ja: zur eingabeschleife
63295         jsr a63314              ; ---> zeile disassemblieren
63298         inc *a246               ; operandenlänge + 1
63300         lda *a246               ; = befehlslänge
63302         jsr a64406              ; ---> adresse um befehlslänge erhöhen
63305         lda *a246               ; befehlslänge
63307         jsr a64372              ; ---> von endadresse subtrahieren
63310         bcs a63287              ; ergebnis positiv: weitermachen ->
63312         bcc a63203              ; zur eingabeschleife ->
63314
63314 a63314  lda #'.
63316         jsr a65490              ; ---> bsout
63319         jsr a64264              ; ---> ' ' ausgeben
63322 a63322  jsr a64251              ; ---> adresse ausgeben
63325         jsr a64264              ; ---> ' ' ausgeben
63328         ldy #0
63330         jsr a53142              ; ---> lda (a161),y: opkode holen
63333         jsr a63444              ; ---> opkode analysieren
63336         pha                     ; tabellen-offset merken
63337         ldx *a246               ; zahl der operandenbytes
63339         inx                     ; befehlslänge
63340 a63340  dex                     ; alle bytes ausgegeben?
63341         bpl a63354              ; nein: ->
63343         jsr a64472              ; ---> '   ' ausgeben
63346         .by '   ' 0
63350         jmp a63360              ; --->
63353
63353         nop
```

```
63354 a63354  jsr a53142              ; ---> lda (a161),y: objekt-byte holen
63357         jsr a64261              ; ---> und als hex-ziffer ausgeben
63360 a63360  iny
63361         cpy #3                  ; maximalzahl 3
63363         bcc a63340              ; noch nicht erreicht: ->
63365         pla                     ; tabellenoffset für opkode
63366         ldx #3
63368         jsr a63515              ; ---> mnemo zum opkode ausgeben
63371         ldx #6                  ; operandenwort erzeugen
63373 a63373  cpx #3
63375         bne a63397
63377         ldy *a246               ; operandenlänge
63379         beq a63397              ; = 0: ->
63381 a63381  lda a1355               ; adressierungsflag
63384         cmp #232                ; 1,8*157: branch-befehl
63386         jsr a53142              ; ---> lda (a161),y: operandenbyte
63389         bcs a63420              ; branch-befehl: ->
63391         jsr a64272              ; ---> ausgeben
63394         dey
63395         bne a63381
63397 a63397  asl a1355               ; adressierungsflag
63400         bcc a63416              ; bit nicht gesetzt: ->
63402         lda a63631-1,x          ; symbol aus tabelle
63405         jsr a65490              ; ---> bsout
63408         lda a63637-1,x          ; zweites symbol aus tabelle
63411         beq a63416              ; = 0: nicht ausgeben ->
63413         jsr a65490              ; ---> bsout
63416 a63416  dex
63417         bne a63373
63419         rts
63420
63420 ; z i e l a d r e s s e   f ü r   b r a n c h   b e r e c h n e n
63420
63420 a63420  jsr a63432              ; ---> operand zur adresse addieren
63423         clc
63424         adc #1                  ; ergebnis inkrementieren
63426         bne a63429
63428         inx
63429 a63429  jmp a64255              ; ---> zieladresse ausgeben
63432
63432 a63432  ldx *a162               ; h,adresse
63434         tay                     ; operand
63435         bpl a63438              ; kleiner als 128: ->
63437         dex                     ; sonst rückwärtssprung
63438 a63438  adc *a161               ; operand + 1,adresse
63440         bcc a63443              ; kein übertrag: ->
63442         inx
63443 a63443  rts
63444
63444 ; o p k o d e   a n a l y s i e r e n
63444
63444 a63444  tay                     ; befehlskode
63445         lsr a                   ; bit 0 gesetzt?
63446         bcc a63459              ; nein: ->
63448         lsr a                   ; bit 1 gesetzt?
63449         bcs a63474              ; ja: fehler ->
63451         cmp #34                 ; opkode 137?
63453         beq a63474              ; ja: fehler ->
63455         and #7
63457         ora #128
```

```
63459 a63459  lsr a
63460         tax
63461         lda a63549,x            ; offset für adressierungs-flag
63464         bcs a63470              ; bit 0 gesetzt: unteres halbbyte ->
63466         lsr a                   ; sonst oberes halbbyte
63467         lsr a                   ; nach unten schieben
63468         lsr a
63469         lsr a
63470 a63470  and #15                 ; unteres halbbyte isolieren
63472         bne a63478              ; ergebnis nicht null: ok ->
63474 a63474  ldy #128                ; fehler-einsprung
63476         lda #0
63478 a63478  tax
63479         lda a63617,x            ; adressierungs-flag
63482         sta a1355               ; merken
63485         and #3
63487         sta *a246               ; operandenlänge
63489         tya                     ; opkode
63490         and #%10001111
63492         tax
63493         tya
63494         ldy #3                  ; ermittlung des offsets
63496         cpx #%10001010
63498         beq a63511
63500 a63500  lsr a
63501         bcc a63511
63503         lsr a
63504 a63504  lsr a
63505         ora #%00100000
63507         dey
63508         bne a63504
63510         iny
63511 a63511  dey
63512         bne a63500
63514         rts
63515
63515 ; m n e m o   z u m   o p k o d e   a u s g e b e n
63515
63515 a63515  tay                     ; opkode-offset
63516         lda a63643,y            ; bytes aus tabellen holen
63519         sta *a159
63521         lda a63707,y
63524         sta *a160
63526 a63526  lda #0
63528         ldy #5                  ; jeweils 5 bits ergeben 1 buchstaben
63530 a63530  asl *a160
63532         rol *a159
63534         rol a
63535         dey
63536         bne a63530
63538         adc #'?                 ; ergibt buchstabenkode, bei 0: '?'
63540         jsr a65490              ; ---> bsout
63543         dex
63544         bne a63526
63546         jmp a64264              ; ---> ' ' ausgeben
```

```
63549 ;------------------------------  T A B E L L E N
63549
63549 a63549  .by 64 2 69 3 208 8 64 9 48 34
63559         .by 69 51 208 8 64 9 64 2 69 51
63569         .by 208 8 64 9 64 2 69 179 208 8
63579         .by 64 9 0 34 68 51 208 140 68 0
63589         .by 17 34 68 51 208 140 68 154 16 34
63599         .by 68 51 208 8 64 9 16 34 68 51
63609         .by 208 8 64 9 98 19 120 169
63617
63617 a63617  .by 0 33 129 130 0 0 89 77 145 146 134 74 133 157
63631
63631 a63631  .by ',),#($'
63637
63637 a63637  .by 'y' 0 'x$$' 0
63643
63643 a63643  .by 28 138 28 35 93 139 27 161
63651         .by 157 138 29 35 157 139 29 161
63659         .by 0 41 25 174 105 168 25 35
63667         .by 36 83 27 35 36 83 25 161
63675         .by 0 26 91 91 165 105 36 36
63683         .by 174 174 168 173 41 0 124 0
63691         .by 21 156 109 156 165 105 41 83
63699         .by 132 19 52 17 165 105 35 160
63707
63707 a63707  .by 216 98 90 72 38 98 148 136
63715         .by 84 68 200 84 104 68 232 148
63723         .by 0 180 8 132 116 180 40 110
63731         .by 116 244 204 74 114 242 164 138
63739         .by 0 170 162 162 116 116 116 114
63747         .by 68 104 178 50 178 0 34 0
63755         .by 26 26 38 38 114 114 136 200
63763         .by 196 202 38 72 68 68 162 200
63771
63771         .by 13 32 32 32
63775
63775 ;------------------------------  A
63775
63775 a63775  bcc a63780              ; adresse angegeben: ->
63777         jmp a62610             ; ---> eingabeschleife
63780
63780 a63780  jsr a64347             ; ---> adresse in zeiger kopieren
63783 a63783  ldx #0
63785         stx *a120
63787 a63787  jsr a64319             ; ---> zeichen aus puffer
63790         bne a63799             ; kein trennzeichen: ->
63792         cpx #0                 ; trennzeichen am pufferanfang?
63794         bne a63799             ; nein: ->
63796         jmp a62613             ; ---> eingabeschleife
63799
63799 a63799  cmp #'                 ; leerstellen überlesen
63801         beq a63783
63803         sta a1356,x            ; befehlswort nach
63806         inx                    ; a1356,...,a1358 kopieren
63807         cpx #3
63809         bne a63787
63811 a63811  dex
63812         bmi a63832
```

```
63814          lda a1356,x
63817          sec                    ; jedes byte des befehlswortes
63818          sbc #63                ; wird in den bereich 1,...,27
63820          ldy #5                 ; transformiert, das sind 5 bits;
63822 a63822   lsr a                  ; aus den 3 * 5 bits werden
63823          ror *a120              ; 2 bytes in a119,a120 erzeugt
63825          ror *a119
63827          dey
63828          bne a63822
63830          beq a63811
63832
63832 a63832   ldx #2
63834 a63834   jsr a64319             ; ---> zeichen aus puffer
63837          beq a63869             ; trennzeichen:
63839          cmp #'                 ; leerstellen überlesen
63841          beq a63834
63843          jsr a64125             ; ---> hex-ziffer?
63846          bcs a63862             ; nein: ->
63848          jsr a64139             ; ---> zwei hex-ziffern in byte wandeln
63851          ldy *a241              ; voriges byte
63853          sty *a242              ; hochschieben
63855          sta *a241              ; neues byte speichern
63857          lda #'0
63859          sta *a119,x
63861          inx
63862 a63862   sta *a119,x
63864          inx
63865          cpx #30
63867          bcc a63834
63869 a63869   stx *a159               ; ende-offset merken
63871          ldx #0
63873          stx a1359              ; testkode
63876 a63876   ldx #0
63878          stx a1360              ; vergleichs-offset
63881          lda a1359              ; testkode
63884          jsr a63444             ; ---> disassemblieren
63887          ldx a1355              ; adressierungsflag
63890          stx *a160              ; merken
63892          tax                    ; opkode-offset
63893          lda a63707,x           ; byte aus zweiter tabelle
63896          jsr a64094             ; ---> vergleichen
63899          lda a63643,x           ; byte aus erster tabelle
63902          jsr a64094             ; ---> vergleichen
63905          ldx #6                 ; beide bytes waren gleich
63907 a63907   cpx #3
63909          bne a63930
63911          ldy *a246              ; operandenlänge
63913          beq a63930             ; = 0: ->
63915 a63915   lda a1355              ; adressierungsflag
63918          cmp #232               ; branchbefehl?
63920          lda #'0
63922          bcs a63954             ; ja: ->
63924          jsr a64091             ; ---> vergleichen
63927          dey
63928          bne a63915
63930 a63930   asl a1355              ; adressierungsflag bitweise ins carry
63933          bcc a63949             ; flag nicht gesetzt: ->
63935          lda a63631-1,x         ; symbol aus tabelle
63938          jsr a64094             ; ---> vergleichen
63941          lda a63637-1,x         ; zweites symbol aus tabelle
63944          beq a63949             ; = 0: ->
63946          jsr a64094             ; ---> vergleichen
```

```
63949 a63949  dex
63950         bne a63907
63952         beq a63960
63954
63954 a63954  jsr a64091              ; ---> vergleichen
63957         jsr a64091              ; ---> vergleichen
63960 a63960  lda *a159               ; ende-offset
63962         cmp a1360               ; = vergleichs-offset?
63965         beq a63970              ; ja: ->
63967         jmp a64106              ; ---> testkode erhöhen
63970
63970 a63970  ldy *a246               ; operandenlänge
63972         beq a64026              ; = 0: ->
63974         lda *a160               ; adressierungsflag
63976         cmp #157                ; branch?
63978         bne a64018              ; nein: ->
63980         lda *a241               ; zieladresse
63982         sbc *a161               ; - befehlsadresse
63984         sta a1361               ; = operand
63987         lda *a242
63989         sbc *a162
63991         bcc a64002              ; negativ: rückwärtssprung ->
63993         bne a64114              ; nicht null: fertig ->
63995         ldx a1361               ; operand
63998         bmi a64114              ; negativ: fertig ->
64000         bpl a64011              ; sonst ->
64002
64002 a64002  tay                     ; h-byte
64003         iny                     ; = 255?
64004         bne a64114              ; nein: fertig ->
64006         ldx a1361               ; l-byte
64009         bpl a64114              ; kleiner als 128: fertig ->
64011 a64011  dex
64012         dex
64013         txa
64014         ldy *a246               ; operandenlänge
64016         bne a64021              ; nicht null: ->
64018 a64018  lda a240,y              ; operanden
64021 a64021  sta (a161),y            ; abspeichern
64023         dey
64024         bne a64018
64026 a64026  lda a1359               ; opkode
64029         sta (a161),y            ; abspeichern
64031         jsr a64309              ; ---> auf zeilenanfang zurücksetzen
64034         ldx #40
64036         jsr a53094              ; ---> 'a  error' ausgeben
64039         jsr a63322              ; ---> adresse ausgeben
64042         inc *a246               ; operandenlänge + 1
64044         lda *a246               ; = befehlslänge
64046         jsr a64406              ; ---> adresse um befehlslänge erhöhen
64049         lda #'a                 ; 'a'
64051         sta a1319               ; in tastaturpuffer
64054         lda #'
64056         sta a1319+1
64059         sta a1319+6
64062         lda *a162               ; h,adresse
64064         jsr a64288              ; ---> in 2 hex-ziffern wandeln
64067         sta a1319+2             ; und in tastaturpuffer
64070         stx a1319+3
```

```
64073          lda *a161              ; l,adresse
64075          jsr a64288             ; ---> in 2 hex-ziffern wandeln
64078          sta a1319+4            ; und in tastaturpuffer
64081          stx a1319+5
64084          lda #7                 ; 7 kodes sind im puffer
64086          sta *a239              ; = tastaturpuffer-index
64088          jmp a62613             ; ---> eingabeschleife
64091
64091 ; t e s t k o d e   m i t   e i n g a b e   v e r g l e i c h e n
64091
64091 a64091   jsr a64094             ; ---> byte vergleichen
64094 a64094   stx a2035
64097          ldx a1360              ; vergleichs-offset
64100          cmp *a119,x            ; (a) mit eingabe vergleichen
64102          beq a64117             ; gleich: ->
64104          pla                    ; rücksprungadresse entfernen
64105          pla
64106 a64106   inc a1359              ; testkode erhöhen
64109          beq a64114             ; alle kodes getestet: fertig ->
64111          jmp a63876             ; ---> nächsten kode testen
64114
64114 a64114   jmp a62610             ; ---> eingabeschleife
64117
64117 a64117   inx
64118          stx a1360              ; vergleichs-offset erhöhen
64121          ldx a2035
64124          rts
64125
64125 ; p r ü f e n   o b   h e x - z i f f e r
64125
64125 a64125   cmp #'a
64127          bcc a64132
64129          cmp #'g
64131          rts
64132
64132 a64132   cmp #'0
64134          bcc a64158
64136          cmp #':
64138          rts
64139
64139 ; z w e i   h e x - z i f f e r n   i n   b y t e   w a n d e l n
64139
64139 a64139   jsr a64160             ; ---> erste ziffer in halbbyte
64142          asl a                  ; halbbyte nach oben schieben
64143          asl a
64144          asl a
64145          asl a
64146          sta a1372              ; und merken
64149          jsr a64319             ; ---> zeichen aus puffer
64152          jsr a64160             ; ---> zweite ziffer in halbbyte
64155          ora a1372              ; mit erstem halbbyte odern
64158 a64158   sec
64159          rts
64160
64160 a64160   cmp #58                ; dezimal-ziffer?
64162          php
64163          and #15
64165          plp
64166          bcc a64170             ; ja: ->
64168          adc #8                 ; + carry macht 10 aus 'a' usw.
64170 a64170   rts
```

```
64171 ;------------------------------
64171
64171 a64171  dec *a243
64173
64173 ;------------------------------  A D R E S S E  Ü B E R N E H M E N
64173
64173 a64173  lda #0
64175         sta *a241               ; adressenspeicher löschen
64177         sta *a242
64179         sta a2036               ; ziffernzähler rücksetzen
64182 a64182  jsr a64319              ; ---> zeichen aus puffer
64185         beq a64245              ; trennzeichen: ->
64187         cmp #'                  ; leerstellen überlesen
64189         beq a64182
64191 a64191  cmp #'
64193         beq a64241
64195         cmp #',
64197         beq a64241
64199         cmp #'0                 ; hex-ziffer?
64201         bcc a64246              ; nein: fehler ->
64203         cmp #'g
64205         bcs a64246              ; nein: fehler ->
64207         cmp #':
64209         bcc a64217
64211         cmp #'a
64213         bcc a64246              ; nein: fehler ->
64215         sbc #8
64217 a64217  sbc #47
64219         asl a
64220         asl a
64221         asl a
64222         asl a
64223         ldx #4
64225 a64225  asl a
64226         rol *a241               ; bits in adressenspeicher rotieren
64228         rol *a242
64230         dex                          /
64231         bne a64225
64233         inc a2036
64236         jsr a64319              ; ---> zeichen aus puffer
64239         bne a64191              ; kein endezeichen: nächste ziffer ->
64241 a64241  lda a2036               ; anzahl der ziffern
64244         clc
64245 a64245  rts
64246
64246 a64246  pla
64247         pla
64248         jmp a62610              ; ---> fehler
64251
64251 a64251  lda *a161               ; l,adresse
64253         ldx *a162               ; h,...
64255 a64255  pha
64256         txa                     ; h-byte
64257         jsr a64272              ; ---> ausgeben
64260         pla                     ; l-byte
64261 a64261  jsr a64272              ; ---> ausgeben
64264 a64264  lda #'
64266         .by 44
64267 a64267  lda #'?
64269         jmp a65490              ; ---> bsout
```

```
64272 a64272  stx a2035
64275         jsr a64288              ; ---> byte in 2 hex-ziffern wandeln
64278         jsr a65490              ; ---> 1. ziffer ausgeben
64281         txa                     ; 2. ziffer
64282         ldx a2035
64285         jmp a65490              ; ---> ausgeben
64288
64288 a64288  pha                     ; byte auf stack legen
64289         jsr a64298              ; ---> unteres halbbyte in hex-ziffer
64292         tax                     ; 2. hex-ziffer
64293         pla                     ; byte erinnern
64294         lsr a                   ; oberes halbbyte nach unten schieben
64295         lsr a
64296         lsr a
64297         lsr a
64298 a64298  and #15                 ; unteres halbbyte isolieren
64300         cmp #10                 ; kleiner als 10?
64302         bcc a64306              ; ja: ->
64304         adc #6                  ; sonst 7 addieren (carry!)
64306 a64306  adc #48                 ; ascii-kode erzeugen
64308         rts
64309
64309 a64309  lda #145                 ; 'kursor nach oben'
64311         jsr a65490              ; ---> bsout
64314 a64314  lda #13                  ; cr
64316         jmp a65490              ; ---> bsout
64319
64319 ; z e i c h e n   a u s   p u f f e r   l e s e n
64319
64319 a64319  stx a2035               ; x merken
64322         ldx *a243               ; pufferzeiger
64324         cpx *a244               ; = pufferende?
64326         bcs a64343              ; ja: ->
64328         lda a512,x              ; zeichen aus puffer
64331         cmp #':                 ; trennzeichen?
64333         beq a64343              ; ja: ->
64335         inc *a243               ; pufferzeiger erhöhen
64337 a64337  php
64338         ldx a2035               ; x erinnern
64341         plp
64342         rts
64343
64343 a64343  lda #0                   ; simuliert trennzeichen
64345         beq a64337
64347
64347 ; a d r e s s e   i n   z e i g e r   k o p i e r e n
64347
64347 a64347  lda *a241
64349         sta *a161
64351         lda *a242
64353         sta *a162
64355         rts
64356
64356 ; z e i g e r   v o n   a d r e s s e   s u b t r a h i e r e n
64356
64356 a64356  sec
64357         lda *a241
64359         sbc *a161
64361         sta *a241
64363         lda *a242
64365         sbc *a162
64367         sta *a242
64369         rts
```

```
64370 ; a d r e s s e n  d e k r e m e n t i e r e n
64370
64370 a64370  lda #1
64372 a64372  sta a2035
64375         sec
64376         lda *a241
64378         sbc a2035
64381         sta *a241
64383         lda *a242
64385         sbc #0
64387         sta *a242
64389         rts
64390
64390 a64390  sec
64391         lda *a159
64393         sbc #1
64395         sta *a159
64397         lda *a160
64399         sbc #0
64401         sta *a160
64403         rts
64404
64404 ; z e i g e r  i n k r e m e n t i e r e n
64404
64404 a64404  lda #1
64406 a64406  clc
64407         adc *a161
64409         sta *a161
64411         bcc a64415
64413         inc *a162
64415 a64415  rts
64416
64416 ; a d r e s s e  u n d  b l o c k l ä n g e  h o l e n
64416
64416 a64416  bcs a64438
64418         jsr a64347            ; ---> adresse in zeiger kopieren
64421         jsr a64173            ; ---> adresse übernehmen
64424         bcs a64438            ; nicht vorhanden: ->
64426         jsr a64356            ; ---> 1. adr von 2. adr subtrahieren
64429         lda *a241             ; ergebnis
64431         sta *a159             ; nach a159,a160
64433         lda *a242
64435         sta *a160
64437         clc
64438 a64438  rts
64439
64439 ; r e g i s t e r  r e t t e n
64439
64439 a64439  sta a272
64442 a64442  stx a274
64445         sty a273
64448         rts
64449
64449 a64449  lda a272
64452 a64452  ldx a274
64455         ldy a273
64458         rts
```

```
64459 ; s t o p t a s t e   p r ü f e n
64459
64459 a64459  stx *a250
64461         jsr a53009              ; ---> stoptaste dekodieren
64464         ldx *a250
64466         eor #128                ; wenn stoptaste gedrückt,
64468         asl a                   ; wird carry gesetzt
64469         lda #0
64471         rts
64472
64472 ;------------------------------  M E L D U N G   A U S G E B E N
64472
64472 a64472  pha                     ; register retten
64473         tya
64474         pha
64475         txa
64476         pha
64477         tsx
64478         inx
64479         inx
64480         inx
64481         inx
64482         lda a256,x              ; rücksprungadresse vom stack holen
64485         sta *a188
64487         inx
64488         lda a256,x
64491         sta *a189
64493         inc *a188               ; und um 1 erhöhen
64495         bne a64499              ; ergibt anfang der meldung
64497         inc *a189
64499 a64499  ldy #0
64501 a64501  lda (a188),y            ; zeichen aus meldung
64503         beq a64511              ; endemarke: ->
64505         jsr a65490              ; ---> bsout
64508         iny
64509         bne a64501
64511
64511 a64511  tya                     ; länge der meldung
64512         tsx
64513         inx
64514         inx
64515         inx
64516         inx
64517         clc
64518         adc *a188               ; zum anfang der meldung addieren
64520         sta a256,x              ; ergebnis als neue rücksprungadresse
64523         lda #0                  ; auf stack legen
64525         adc *a189
64527         inx
64528         sta a256,x
64531         pla                     ; register wiederherstellen
64532         tax
64533         pla
64534         tay
64535         pla
64536         rts
64537
64537 ;------------------------------  I O B A S E
64537
64537 a64537  ldx #l,a64768
64539         ldy #h,a64768
64541         rts
```

```
64542 ;------------------------------  M O D U L - R E S E T
64542
64542 a64542  ldx #3                  ; modul-zeiger initialisieren
64544         stx *a150
64546         lda #0
64548 a64548  sta a1516,x             ; modul-tabelle löschen
64551         dex
64552         bpl a64548
64554 a64554  ldx *a150
64556         lda a64635,x            ; 0, 5, 10, 15
64559         tax
64560         sta a64976,x            ; modul einschalten
64563         ldy #2
64565 a64565  lda a32775,y            ; kennzeichen 'cbm' suchen
64568         cmp a64598,y
64571         bne a64593              ; nicht vorhanden: ->
64573         dey
64574         bpl a64565
64576         lda a32774              ; modul-nummer (basic: 0)
64579         ldx *a150
64581         sta a1516,x             ; in tabelle schreiben
64584         cmp #1
64586         bne a64593
64588         stx *a251
64590         jsr a32768              ; ---> initialisierung
64593 a64593  dec *a150
64595         bpl a64554
64597         rts
64598
64598 a64598  .by 'cbm'
64601
64601 a64601  sei
64602         ldx #3
64604 a64604  lda a1516,x             ; modul (x) vorhanden?
64607         beq a64625              ; nein: ->
64609         txa
64610         pha
64611         lda a64635,x            ; 0, 5, 10, 15
64614         tax
64615         sta a64976,x            ; modul einschalten
64618         stx *a251               ; schalterkopie
64620         jsr a32768              ; ---> initialisierung
64623         pla
64624         tax
64625 a64625  dex
64626         bne a64604
64628         sta a64976              ; basic einschalten
64631         stx *a251               ; schalterkopie
64633         cli
64634         rts
64635
64635 a64635  .by 0 5 10 15
64639
64639 ;------------------------------  M O D U L - Z U G R I F F
64639
64639 a64639  sta a64976,x            ; modul (x) einschalten
64642         tax                     ; vorheriger modul
64643         lda (a190),y
64645         sta a64976,x            ; auf vorherigen modul zurückschalten
64648         rts
```

```
64649 ;------------------------------  M O D U L - A U F R U F
64649
64649 a64649  pha                     ; aktueller modul
64650         stx *a251               ; modul (x) einschalten
64652         sta a64976,x
64655         ldx a1523               ; register setzen
64658         lda a1524
64661         pha
64662         lda a1522
64665         plp
64666         jsr a64688              ; ---> routine aufrufen
64669         sta a1522               ; register abspeichern
64672         php
64673         pla
64674         sta a1524
64677         stx a1523
64680         pla                     ; alten modul wieder einschalten
64681         sta *a251
64683         tax
64684         sta a64976,x
64687         rts
64688
64688 a64688  jmp (a1520)             ; ---> routine aufrufen
64691
64691 ;------------------------------  P U L S
64691
64691 a64691  pha
64692         txa
64693         pha
64694         tya
64695         pha
64696 a64696  sta a64976              ; basic einschalten
64699         jmp a52736              ; ---> interrupt-routine
64702
64702 a64702  ldx *a251               ; schalterkopie
64704         sta a64976,x            ; modul wieder einschalten
64707         pla
64708         tay
64709         pla
64710         tax
64711         pla
64712         rti
64713
64713 ;------------------------------
64713
64713 a64713  ldx *a251               ; schalterkopie
64715         sta a64976,x            ; modul einschalten
64718         jmp (a766)              ; ---> modul aufrufen
64721
64721 ;------------------------------
64721
64721 ; bereich 64721...64752 enthält kode 255
64721
64721 ;------------------------------  S P R U N G L I S T E  1
64721
64753 a64753  jmp a64713              ; ---> modul einschalten und aufrufen
64756
64756 a64756  jmp a64601              ; ---> vorhandene moduln initialisieren
64759
64759 a64759  jmp a64639              ; ---> lda (a190),y aus dem modul (x)
```

```
64762 a64762  jmp a64649              ; ---> jsr (a1520) in modul (x)
64765
64765 a64765  jmp a64696              ; ---> interrupt-routine
64768
64768 ;-------------------------------
64768
64768 ; bereich 64768...65352 enthält die i/o-adressen
64768
64768 ;-------------------------------  S P R U N G L I S T E  2
64768
65353 a65353  jmp a47042              ; ---> key definieren
65356
65356 a65356  jmp a56393              ; ---> print
65359
65359 a65359  jmp a64472              ; ---> meldung ausgeben
65362
65362 a65362  jmp a62533              ; ---> monitor call
65365
65365 ;-------------------------------
65365
65365 ; bereich 65365...65406 enthält kode 255
65365
65365         .by 138 131
65367
65367 ;-------------------------------  K E R N A L - S P R U N G L I S T E
65367
65409 a65409  jmp a55374              ; ---> cint: editor initialisieren
65412
65412 a65412  jmp a62219              ; ---> ioinit: i/o initialisieren
65415
65415 a65415  jmp a62290              ; ---> ramtas: ram etc. initialisieren
65418
65418 a65418  jmp a62158              ; ---> restor: vektoren wiederherstellen
65421
65421 a65421  jmp a62163              ; ---> vector: vektoren ändern
65424
65424 a65424  jmp a62490              ; ---> setmsg: ausgabe ein-/ausschalten
65427
65427 a65427  jmp a61005              ; ---> second
65430
65430 a65430  jmp a60954              ; ---> tksa
65433
65433 a65433  jmp a62503              ; ---> memtop
65436
65436 a65436  jmp a62518              ; ---> membot
65439
65439 a65439  jmp a56081              ; ---> scnkey
65442
65442 a65442  jmp a62499              ; ---> settmo
65445
65445 a65445  jmp a60555              ; ---> acptr
65448
65448 a65448  jmp a60639              ; ---> ciout
65451
65451 a65451  jmp a61243              ; ---> untlk
65454
65454 a65454  jmp a61219              ; ---> unlsn
65457
65457 a65457  jmp a60972              ; ---> listen
65460
65460 a65460  jmp a60922              ; ---> talk
```

```
65463 a65463  jmp a62492            ; ---> readst
65466
65466 a65466  jmp a62483            ; ---> setlfs
65469
65469 a65469  jmp a62476            ; ---> setnam
65472
65472 a65472  jmp (a792)            ; ---> open  (a61267)
65475
65475 a65475  jmp (a794)            ; ---> close (a61021)
65478
65478 a65478  jmp (a796)            ; ---> chkin (a60696)
65481
65481 a65481  jmp (a798)            ; ---> ckout (a60768)
65484
65484 a65484  jmp (a800)            ; ---> clrch (a61196)
65487
65487 a65487  jmp (a802)            ; ---> basin (a60392)
65490
65490 a65490  jmp (a804)            ; ---> bsout (a60491)
65493
65493 a65493  jmp a61507            ; ---> load
65496
65496 a65496  jmp a61844            ; ---> save
65499
65499 a65499  jmp a53037            ; ---> settim
65502
65502 a65502  jmp a53030            ; ---> rdtim
65505
65505 a65505  jmp (a806)            ; ---> stop  (a62053)
65508
65508 a65508  jmp (a808)            ; ---> getin (a60377)
65511
65511 a65511  jmp (a810)            ; ---> clall (a61192)
65514
65514 a65514  jmp a52976            ; ---> udtim
65517
65517 a65517  jmp a55348            ; ---> screen
65520
65520 a65520  jmp a55353            ; ---> plot
65523
65523 a65523  jmp a64537            ; ---> iobase
65526
65526 a65526  sta a65342            ; rom einschalten
65529         jmp a62116            ; ---> nmi
65532
65532         .si a65526            ; start
65534         .si a64691            ; puls
```

COMMODORE-plus/4 · ROM-LISTING

Das Listing des Commodore plus/4 ROM-Inhaltes stellt für alle Assemblerprogrammierer eine unerläßliche Informationsquelle dar. Auf über 260 Seiten findet sich das lückenlose Assembler-Quell-Code-Listing, daneben ist jede Codesequenz in deutscher Sprache kommentiert.

Mit dieser Information wird es möglich, die umfangreichen und ausgeklügelten Möglichkeiten des Commodore plus/4 in eigenen Assemblerprogrammen zu verwenden, ohne das Rad ständig neu erfinden zu müssen. Programmierer, die mit dem C-64 oder der 3xxx/4xxx/8xxx-Serie vertraut sind, können sich schnell einen Überblick über Unterschiede und Gemeinsamkeiten der jeweiligen ROM-Inhalte verschaffen und so mit minimalem Aufwand ihre vorhandenen Programme umstellen.

Commodore

Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore GmbH
Kinskygasse 40–44
A-1232 Wien

Artikel-Nr. 584000/10.84 Änderungen vorbehalten ISBN-Nr. 3-89133-006-5