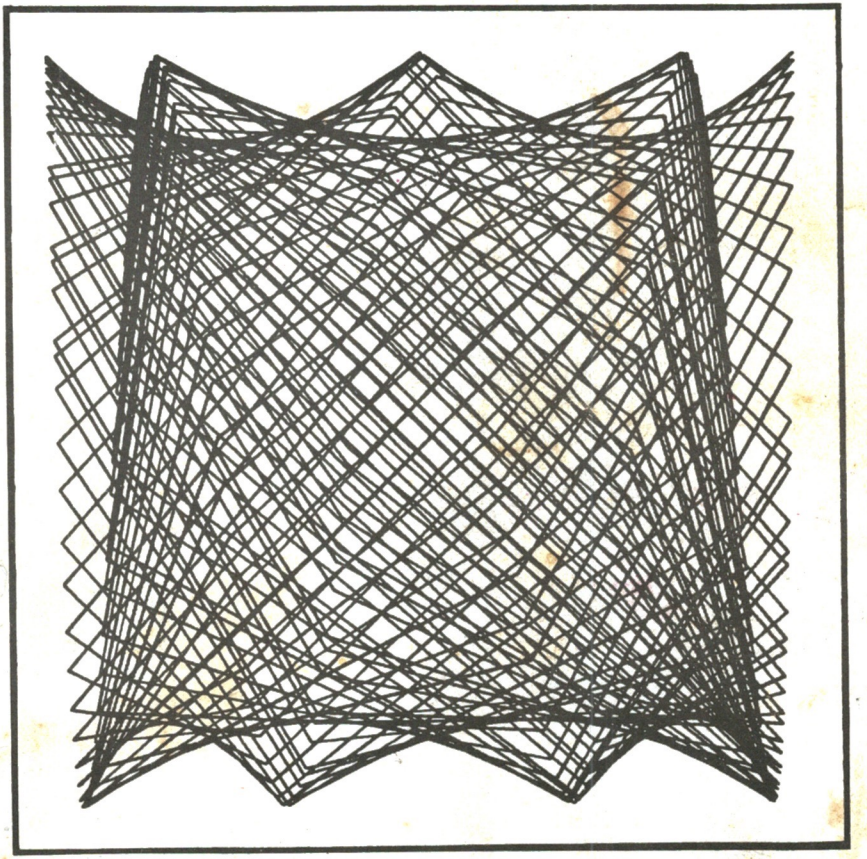
Der NDR-Klein-Computer

68008 Grundprogramm



Franzis'

SoftwareService

Das 68000/8 Grundprogramm

(C) 1985

Rolf–Dieter Klein
München

Rev 4.3

Inhaltsverzeichnis

Befehl	Index	Bedeutung
ADJ360	83	360 Grad Bereich
ASSEMBLE	65	Assembler starten
ASSERR	120	Fehlerzahl
AUFK	94	Schildkrötenrichtung setzen
AUFXY	92	Schildkröte setzen
AUTOFLIP	60	Bildseitenumschaltung
CHAR	63	Zeichen ausgeben
CI	12	Zeichen einlesen
CI2	32	Zeichen aus Speicher lesen
CIINIT2	31	Zeiger auf Textstart setzen
CLPG	17	Eine Bildschirmseite löschen
CLR	16	Alle Bildschirmseiten löschen
CLRSCREEN	20	Bildschirm löschen und Cursor setzen
CMD	26	Einen Befehl an GDP ausgeben
CMDPRINT	40	Befehlsausgabe mehrere Befehle an den GDP
CO	21	Zeichenausgabe auf den Bildschirm
CO2	33	Zeichen- und Steuerzeichenausgabe
COS	24	Cosinus*256 bilden
CRLF	99	Zeilenvorschub ausgeben
CRT	49	CO2 auf Bildschirm
CSTS	13	Zeichen da ?
CUROFF	82	Cursor ausblenden bei CO,CO2
CURON	81	Cursor erscheint bei CO,CO2
CURSAUS	62	Cursor ausblenden
CURSEIN	61	Cursor einblenden
DELAY	35	Warteschleife (Synchron zu Quarz)
DIVS32	73	32-Bit-Division
DRAWTO	9	Linie zur neuen X,Y-Position zeichnen
DREHE	2	Schildkröte dreht sich
EDIT	56	Editor aufrufen
ERAPEN	38	Schreiber in GDP auf Löschen
FIGUR	57	Figur-Befehl
FIRSTTIME	36	Schildkrötenneustart
FLINIT	74	Floppy-Variable definieren
FLOPPY	75	Floppy-Befehl ausführen
GETAD8	109	Wert vom 8 Bit AD-Umsetzer laden
GETAD10	110	Wert vom 10 Bit AD-Umsetzer laden
GETBASIS	89	Adresse des Grundprogramms laden

GETCOLOR	80	Farbcode laden
GETCURXY	101	Cursorposition laden
GETERR	54	Fehlercode lesen
GETFLOP	76	Floppy-Format feststellen
GETK	95	Schildkrötenrichtung laden
GETLINE	100	Zeilenadresse, Bildwidh. laden
GETNEXT	87	Offset für Symboltabelle
GETORG	68	Default-ORG-adresse laden
GETRAM	59	Speicherbereiche testen
GETSN	98	Seriennummer laden
GETSTX	66	Textstart-Adresse laden
GETSYM	86	Adresse, Symboltabelle laden
GETUHR	115	Uhrzeit und Datum einlesen
GETVAR	90	Adresse des Arbeitsspeichers laden
GETVERS	97	Version des Grundprogramms laden
GETXOR	78	Spezial-Mode laden
GETXY	103	Aktuelle GDP-Position laden
GRAPOFF	39	Schildkrötengraphik aus
HEBE	3	Schreibspur ausschalten
HIDE	47	Schildkröte ausblenden
KORXY	93	Schildkötenkoordinaten laden
LO	22	Zeichenausgabe auf den Drucker
LST	50	CO2 auf Drucker
LSTS	117	Drucker bereit ?
MOVETO	8	X,Y-Position im GDP setzen
MULS32	72	32-Bit-Multiplikation
NEWPAGE	27	Bildschirm Schreib- und Leseseite setzen
NIL	52	CO2 ignorieren
PO	15	Zeichen auf CAS-Baugruppe ausgeben
PUTNEXT	88	Offset für Symboltabelle setzen
PUTORG	69	Default-ORG-Adresse setzen
PUTSTX	67	Textstart neu festsetzen
PRINT8B	45	Binäre Ausgabe
PRINT4D	46	Dezimale Ausgabe
PRINT8D	70	Dezimale Ausgabe
PRINT2X	41	Sedezimale Ausgabe
PRINT4X	42	Sedezimale Ausgabe
PRINT6X	43	Sedezimale Ausgabe
PRINT8X	44	Sedezimale Ausgabe

PRINTV8D	71	Dezimale Ausgabe
PROGZGE	64	Zeichen frei wählbar, ausgeben
PRTSYM	84	Symboltabelle ausgeben
READ	11	Text einlesen
RELAN	118	Relais auf CAS2 einschalten
RELAUS	119	Relais auf CAS2 ausschalten
RI	14	Zeichen von CAS–Baugruppe einlesen
RND	96	Zufallszahl bestimmen
SCHR16TEL	19	Schrildkröte schreitet 1/16
SCHREITE	1	Schildkröte schreitet
SETFLIP	34	Umschaltrate für AUTOFLIP setzen
SIN	23	Sinus*256 bilden
SIZE	25	Textgröße setzen
SENKE	4	Schreibspur wird sichtbar
SET	7	Schildkröte setzen, absolut
SETA5	91	Ram–Zeiger A5 gültig belegen
SETCOLOR	79	Farbcode setzen
SETCURXY	102	Cursorposition setzen
SETDA	111	DA–Umsetzer setzen
SETERR	53	Fehlercode setzen
SETFIG	58	Figur einfrieren
SETPASS	55	Laufnummer setzen
SETPEN	37	Schreiber in GDP auf Setzen
SETUHR	116	Uhrzeit und Datum neu stellen
SETXOR	77	Spezial–Mode setzen
SHOW	48	Schildkröte einblenden
SI	104	Zeichen von serieller Schnittstelle
SIINIT	108	Baudrate etc. setzen
SISTS	106	Zeichen da?
SO	105	Zeichen an serielle Schnittstelle
SOSTS	107	Zeichen ausgegeben ?
SOUND	114	Sound–Generator setzen
SPEAK	112	Sprachausgabe mit 5 Parametern
SPEAK1	113	Sprachausgabe nur Phonemcode
SYMCLR	85	Symboltabelle löschen
SYNC	28	Vertikal–Synchronpuls testen
USR	51	CO2 auf Benutzerschnittstelle
WAIT	18	Warten bis GDP für Befehle bereit
WERT	29	Auswertung eines arithmetischen Ausdrucks
WRITE	10	Text auf dem Bildschirm ausgeben
ZUWEIS	30	Variable mit Wert eines Ausdrucks belegen

Befehlsübersicht

In der nachfolgenden Befehlsübersicht sind alle Befehle des Grundprogramms Version 4.2 aufgelistet. Dabei bedeutet (S): Schildkrötenbefehl.

Befehl	Index		Register
SCHREITE	1	(S)	D0
DREHE	2	(S)	D0
HEBE	3	(S)	
SENKE	4	(S)	
SET	7	(S)	D1 D2 D3
MOVETO	8		D1 D2
DRAWTO	9		D1 D2
WRITE	10		D0.B D1 D2 A0
READ	11		D0.B D1 D2 D3 A0
CI	12		->D0.L (-- .L ab Rev 4.2)
CSTS	13		->D0.L (-- sonst nur .B gueltig)
RI	14		->D0.L (--)
PO	15		D0.B
CLR	16		
CLPG	17		
WAIT	18		
SCHR16TEL	19	(S)	D0
CLRSCREEN	20		
CO	21		D0.B
LO	22		D0.B
SIN	23		D0 -> D0
COS	24		D0 -> D0
SIZE	25		D0.B
CMD	26		D0.B
NEWPAGE	27		D0 D1
SYNC	28		-> D0.B
WERT	29		A0 -> D0.L D1
ZUWEIS	30		A0 -> D0.L
CIINIT2	31		
CI2	32		-> D0.B
CO2	33		D0.B
SETFLIP	34		D0 D1
DELAY	35		D0.L
FIRSTTIME	36		
SETPEN	37		
ERAPEN	38		
GRAPOFF	39	(S)	

CMDPRINT	40	D0.B D1 D2 A0
PRINT2X	41	D0.B A0
PRINT4X	42	D0 A0
PRINT6X	43	D0.L A0
PRINT8X	44	D0.L A0
PRINT8B	45	D0.B A0
PRINT4D	46	D0 A0
HIDE	47	(S)
SHOW	48	(S)
CRT	49	
LST	50	
USR	51	
NIL	52	
SETERR	53	D0
GETERR	54	-> D0
SETPASS	55	D0
EDIT	56	
FIGUR	57	D0 A0
SETFIG	58	
GETRAM	59	-> A0 A1
AUTOFLIP	60	
CURSEIN	61	
CURSAUS	62	
CHAR	63	D0.B
PROGZGE	64	A0
ASSEMBLE	65	
GETSTX	66	-> D0.L
PUTSTX	67	D0.L
GETORG	68	-> D0.L
PUTORG	69	D0.L
PRINT8D	70	D0.L A0
PRINTV8D	71	D0.L A0
MULS32	72	D0.L D2.L -> D0.L D1.L
DIVS32	73	D0.L D1.L D2.L -> D0.L D1.L
FLINIT	74	
FLOPPY	75	D1 D2.B D3.B D4.B A0
GETFLOP	76	D4.B -> D4.B
SETXOR	77	D0
GETXOR	78	-> D0
SETCOLOR	79	D0

GETCOLOR	80		-> D0
CURON	81		
CUROFF	82		
ADJ360	83		D0 -> D0
PRTSYM	84		
SYMCLR	85		
GETSYM	86		-> D0.L A0
GETNEXT	87		-> D0.L
PUTNEXT	88		D0
GETBASIS	89		-> D0.L A0
GETVAR	90		-> D0.L A0
SETA5	91		-> A5
AUFXY	92	(S)	D1 D2
KORXY	93	(S)	-> D1 D2
AUFK	94	(S)	D0
GETK	95	(S)	-> D0
RND	96		D0 -> D0
GETVERS	97		-> D0.L
GETSN	98		-> D0.L
CRLF	99		
GETLINE	100		D0.B -> A0 A1 A2
GETCURXY	101		-> D1.L D2.L
SETCURXY	102		D1.B D2.B
GETXY	103		-> D1 D2
SI	104		-> D0.L
SO	105		D0.B
SISTS	106		-> D0.L
SOSTS	107		-> D0.L
SIINIT	108		D0.B D1.B
GETAD8	109		-> D0.L
GETAD10	110		-> D0.L
SETDA	111		D1.B D2.B
SPEAK	112		A0
SPEAK1	113		A0
SOUND	114		A0
GETUHR	115		A0
SETUHR	116		A0
LSTS	117		D0.L
RELAN	118		
RELAUS	119		
ASSERR	120		D0.L

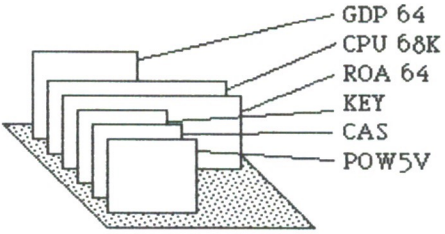
Einleitung

Das Grundprogramm der Version 4.3 bietet einige neue Befehle und Komfort, der in dem Grundprogramm der Version 3 noch nicht enthalten war. So gibt es jetzt 120 eingebaute Befehle. Ferner ist das Grundprogramm jetzt verschiebbar, da alle Befehle relokativ sind. Damit kann auch der Betrieb von CP/M 68K durchgeführt werden. Ebenfalls sind die Floppy-Unterprogramme eingebaut, die dafür nötig sind.

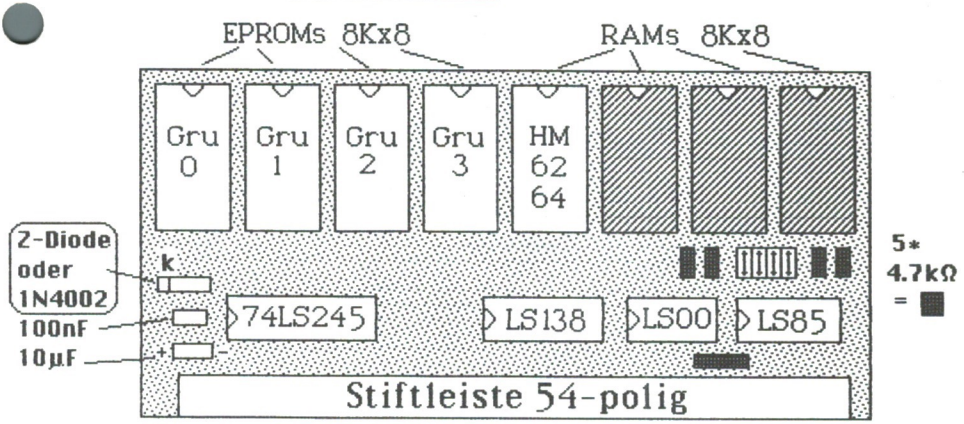
1. Inbetriebnahme

Das Grundprogramm gibt es in zwei Versionen. Zum einen für den 68008, und dann für den 68000.
Bild 1 zeigt die Anordnung für den 68008. Die EPROMs sind mit 0 bis 3 beschriftet und werden gemäß der Abbildung eingesetzt. Die Anordnung auf der Bus-Baugruppe spielt keine Rolle. Das Grundprogramm benötigt zum Betrieb mindestens ein 8Kx8 RAM. Wenn man aber größere Programme bearbeiten will, so sollte man möglichst zwei RAMs verwenden.
Bild 2 zeigt die Belegung der Adressjumper für die ROA-Karte, die auf Adresse 00000 angesprochen wird.
Wenn man die 68000 CPU verwendet, so sieht die Anordnung anders aus. Bild 3 zeigt das Schema. Es wird dann der große BUS verwendet. Beim 68000 unterscheidet man zwischen geraden und ungeraden Adressen (even und odd). Bild 4 zeigt die Anordnung auf dem BUS. Der große BUS besitzt in der Mitte eine doppelte Lochreihe. Die Leitungen D0 bis D7 und RD, WR, IORQ und MREQ werden in der Mitte davon aufgetrennt. Es wird eine doppelte Buchsenleiste bestückt. Die 68000 CPU-Karte besitzt eine doppelte Stiftleiste und dadurch wird der Datenbus und die Steuersignale in gerade und ungerade aufgeteilt. Alle Peripheriekarten werden auf der geraden (even) Seite untergebracht. Auf der ungeraden Seite kann man natürlich eigene Peripherie unterbringen. Alle Portadressen sind beim 68000 mit zwei multipliziert, daher aufpassen, wenn man mit dem 68000 arbeitet, denn unsere Beispiele beziehen sich immer auf die 68008 CPU.
Beispiel: Beim 68008 wird die GDP Karte auf der Adresse $FFFFFF70 angesprochen, beim 68000 ist es die Adresse $FFFFFF70*2 = $FFFFFFEE0.
Auf den Peripheriekarten werden aber dennoch die normalen Adressen eingestellt, die Verdopplung kommt nur daher, daß die Adresse A0 nicht mehr direkt auf dem Bus erscheint, sondern A1 die frühere Adresse A0 ist. Das interne A0 wird verwendet, um zwischen geraden (A0=0) und ungeraden (A0=1) Adressen zu unterscheiden.
Der 68000 hat den Vorteil, daß er schneller ist, da Befehle als ganzes Wort geholt werden, aber den Nachteil, daß Sie Speichererweiterungen immer doppelt brauchen. Der Geschwindigkeitsvorteil macht sich zudem nur bei speziellen Anwendungen bemerkbar. Daher soll im folgenden auch nur noch die 68008 CPU-Karte behandelt werden.

CPU68K - Busanordnung

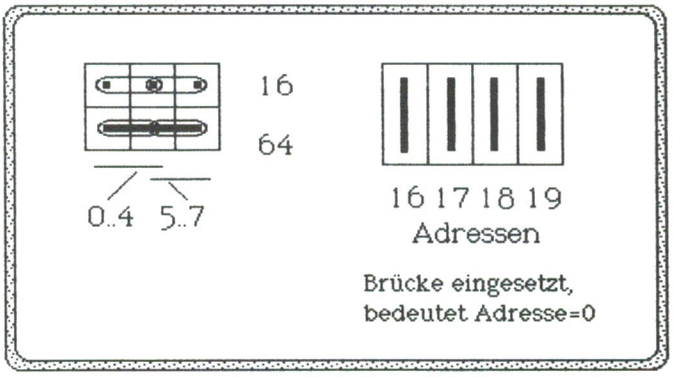


1



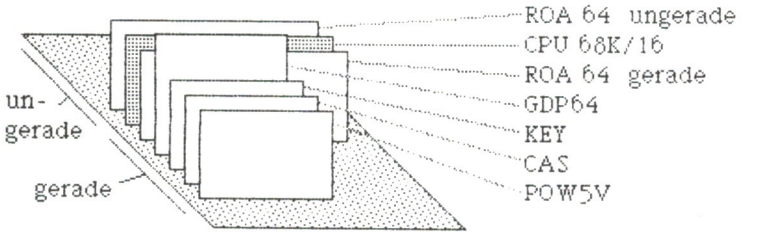
Einstellung der Brücken auf der ROA64

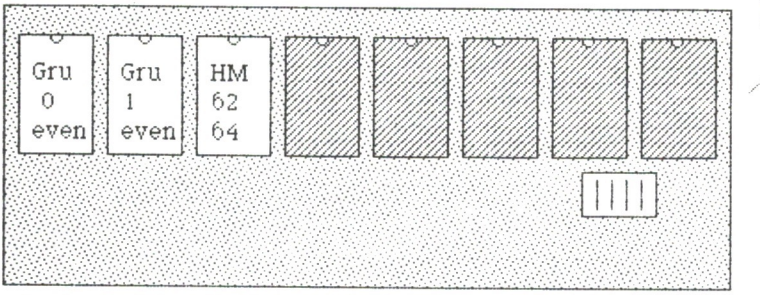
2



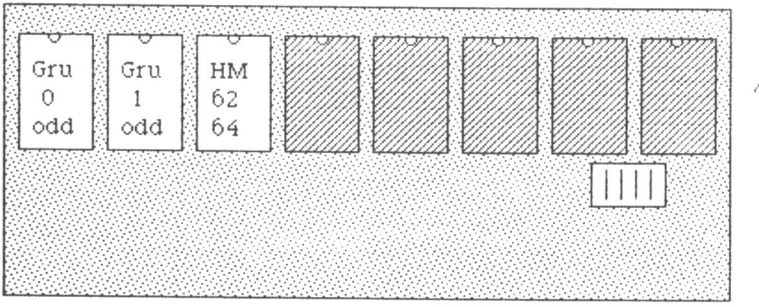
CPU68K/16 - Busanordnung

3





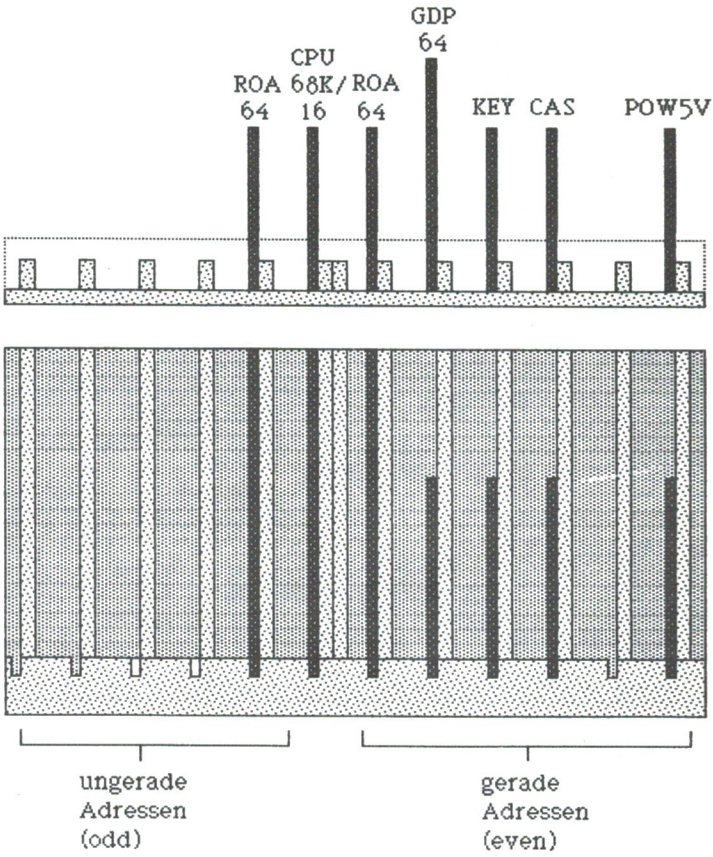
gerade Adresse



ungerade Adresse

CPU 68K/16 - Anschluß

4



IO–Adressbelegung

Bild 5 zeigt die Belegung der IO–Adressen, wie sie derzeit von unterschiedlichen Programmen verwendet wird. Sie müssen sich nicht unbedingt daran halten, jedoch ist es empfehlenswert um kompatibel zu bleiben. Ebenfalls unterstützt das Grundprogramm eine Vielzahl der Peripherie direkt, die dann natürlich auch auf den angegebenen Adressen zu finden sein muß.

Zu den Abkürzungen:

Hex 0,Hex 1	Hexio–Baugruppe
Sound	Karte mit einem Soundgenerator
Cent	Centronix–Port für Druckeranschluß
GDP/P	Seitenregister der GDP–Baugruppe
Key	Tastatur–Eingabe–Baugruppe
GDP	Graphik–Display Prozessor Port
Promer	EPROM–Programmier–Baugruppe
Hardc	Hardcopy–Baugruppe 1
Color	Farbauswahlport für Erweiterung
Farbt	Farbtabellen–RAM
Flo2	Floppy–Baugruppe,Floppycontroller
Flo2/S	Statusregister auf Flo2
Banksel	Bankselekt–Port auf BANK/BOOT
Relais	Port für Relais–Baugruppe
Cas	Port des 6850 auf der CAS–Baugruppe
Sprach	Port des SSI 263 der Karte SPRACH
Ad16x	A/D Umsetzer mit 16 Kanälen und 8 Bit
Ser	Serielle Schnittstelle, IC 6551
Da2x8	zwei D/A Umsetzer mit 8 Bit
Ad1x10	10 Bit A/D Umsetzer
Uhr	Port des Uhrenbausteins.

IO-Adressbelegung

	00	10	20	30	40	50	60	70
0	Hex 0	-	-	-	Sound	-	GDP/P	GDP
1	Hex 1	-	-	-	Sound	-	-	GDP
2	-	-	-	-	-	-	-	GDP
3	-	-	-	-	-	-	-	GDP
4	-	-	-	-	-	-	-	GDP
5	-	-	-	-	-	-	-	GDP
6	-	-	-	-	-	-	-	GDP
7	-	-	-	-	-	-	-	GDP
8	-	-	-	-	Cent	-	Key	GDP
9	-	-	-	-	Cent	-	Key	GDP
A	-	-	-	-	-	-	-	GDP
B	-	-	-	-	-	-	-	GDP
C	-	-	-	-	-	-	-	GDP
D	-	-	-	-	-	-	-	GDP
E	-	-	-	-	-	-	-	GDP
F	-	-	-	-	-	-	-	GDP

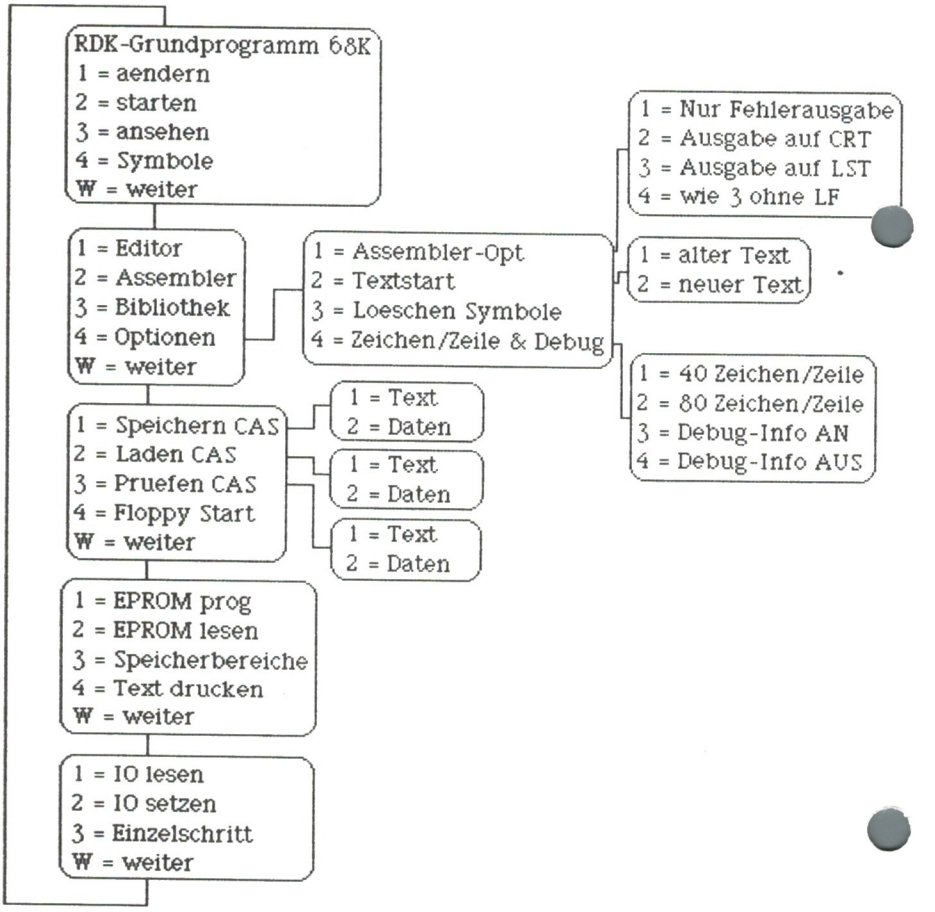
	80	90	A0	B0	C0	D0	E0	F0
0	Promer	-	Color	-	Flo2	-	Ad16x	Ser
1	Promer	-	-	-	Flo2	-	Ad16x	Ser
2	Promer	-	Farbt	-	Flo2	-	Ad16x	Ser
3	-	-	Farbt	-	Flo2	-	Ad16x	Ser
4	-	-	-	-	Flo2/S	-	Ad16x	-
5	-	-	-	-	Flo2/S	-	Ad16x	-
6	-	-	-	-	Flo2/S	-	Ad16x	-
7	-	-	-	-	Flo2/S	-	Ad16x	-
8	Hardc	-	-	-	Banksel	Sprach	Ad16x	Da 2x8
9	Hardc	-	-	-	Relais	Sprach	Ad16x	Da 2x8
A	Hardc	-	-	-	Cas	Sprach	Ad16x	-
B	Hardc	-	-	-	Cas	Sprach	Ad16x	-
C	-	-	-	-	-	Sprach	Ad16x	Ad 1x10
D	-	-	-	-	-	Sprach	Ad16x	Ad 1x10
E	-	-	-	-	-	Sprach	Ad16x	Uhr
F	-	-	-	-	-	Sprach	Ad16x	Uhr

Grundprogramm Menues

Bild 6 zeigt eine Übersicht aller Menues des Grundprogramms.
Hier kann aus Platzgründen nur eine kurze Anleitung gegeben werden, es daher auf das weitere Begleitmaterial, wie das Sonderheft, verwiesen, in dem mehr über Details steht.
Zuerst gelangt man in das Grundmenue. Mit "aendern" kann man Speicherinhalte ansehen und modifizieren. Dabei ist es möglich Bytes, Worte oder Langworte anzugeben, indem man .B, .W oder .L hinter den Datenwert stellt.
Mit "starten" kann man Programme starten. Dabei kann man symbolische Namen wie auch absolute Adressen angeben. Für die Syntax von Ausdrücken sei auf den Abschnitt WERT (29) und ZUWEIS (30) verwiesen.
Mit "ansehen" kann man Speicherbereiche ansehen und darin blättern.
Mit "Symbole" bekommt man eine Liste aller definierten Symbole. Diese kann man auch auf einem Drucker ausgeben, wenn man den Drucker im Optionen–Menue eingeschaltet hat.
Mit dem nächsten Menue werden der Editor, der Assembler und über die Bibliotheksfunktion (siehe auch Beispiel in Bild 7) auch andere Programme erreichbar. Mit "Optionen" gelangt man in ein getrenntes Menue, in dem man die Ausgabe von Assembler und Symboltabelle umsteuern kann. Entweder man schaltet auf "nur Fehlerausgabe", dann erhält man die Quelle bei der Übersetzung nicht mehr auf dem Bildschirm, oder auf "CRT", das ist dann so wie nach dem Einschalten, oder auf "LST", dann kommt die Quelle am Drucker an.
Mit "Wie 3 ohne LF" kann man Drucker bedienen, die einen automatischen Zeilenvorschub besitzen.
Mit "Textstart" kann man die Anfangsadresse der textbearbeitung umstellen. Normalerweise beginnen Texte bei der Adresse $9000, oberhalb des Grundprogramms. Man kann aber einen beliebigen Speicherbereich verwenden. Mit "neuer Text" wird dieser Speicherbereich dann festgelegt. Mit "alter Text" ist es möglich mehrere Textgebiete zu verwalten.
Mit "Loeschen Symbole" kann man die Symboltabelle löschen und damit Platz schaffen, denn die Symboltabelle wird mit hinzukommenden Symbolen immer größer und kann in das Textgebiet hineinlaufen.
Mit "Zeichen/Zeile & Debug" gelangt man wieder in ein Untermenue, man kann dort zwischen 40 Zeichen pro Zeile und 80 Zeichen pro Zeile wählen. Davon sind alle Ausgaben mit dem Unterprogram CO2 betroffen, so der Assembler, die Symboltabelle und der Editor. Andere Formate kann man auch selbst einstellen, dazu muß man ein Programm mit dem Unterprogramm SIZE schreiben. Mit "Debug–Info AN" wird eine zusätliche Information für die Einzelschritt–Funktion festgehalten und man erhält dann die jeweilige Quellzeile im Einzelschritt mit angegeben. Die Zusatzinformation wird direkt hinter dem Textgebiet angelegt und beinhaltet eine Zuordnung von Programmzählerstand zu Zeilenadresse. Achtung, die Debug–Tabelle benötigt u.U. viel Platz. Eine Kontroller bietet der Assembler, der immer die letzte Adresse der Debugtabelle mit angibt, wenn sie eingeschaltet ist.
Mit dem nächsten Menue wird die Baugruppe CAS und FLO2 bedient. Mit Speichern und Laden, kann man Daten und Programme auf Kassette speichern

und laden.
Mit "Floppy Start" wird der erste Sektor auf Spur 0 auf die Adresse $9800 geladen und gestartet, falls der erste Befehl ein NOP ist. Damit kann man z.B. das Betriebsystem CP/M 68K starten, oder das Mikrodos. Die Floppy-Art wird dabei automatisch festgestellt. Liegt ein Fehler vor, oder ist kein gültiges Programm vorhanden, so kehrt das Programm wieder ins Menue zurück.
Mit "EPROM prog" und "EPROM lesen" kann man die Baugruppe PROMER bedienen und unterschiedliche EPROM-Typen bedienen.
Mit "3=Speicherbereiche" erhält man die aktuelle RAM-Belegung auf dem Bildschirm, siehe Bild 8. Dort sind 4 8Kx8 Rams vorhanden, ein bereich geht von $8000 bis $BFFF und der andere von $18000 bis $1BFFF. Der Bereich $F0000 bis $FFFFF wird nicht geprüft, da er für Peripherie reserviert ist.
Mit "Text drucken" erhält man einen Ausdruck des Textspeichers über den Centronix Port.
Im nächsten Menue kann man IO-Port ansehen, aber auch Speicherzellen.
Mit "Einzelschritt" kann man Programme schrittweise durchlaufen.

Grundprogramm Menues



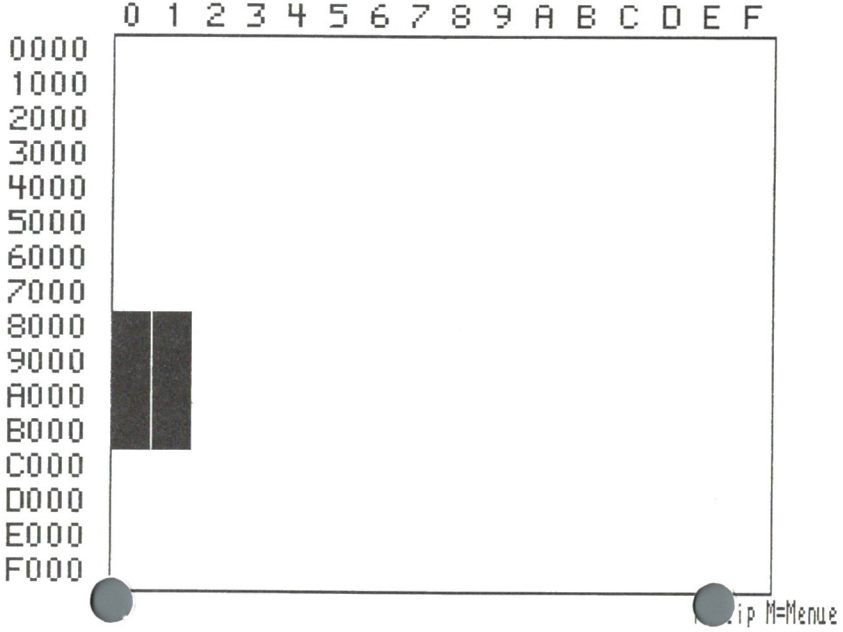
Bibliothek

```
Name   : UFORM68K
Start  : 01499A
Laenge : 0016F6
```

Starten J = JA
cr = weiter, M = Menue

RAM-Bereiche

Erster Bereich: 008000-00BFFF



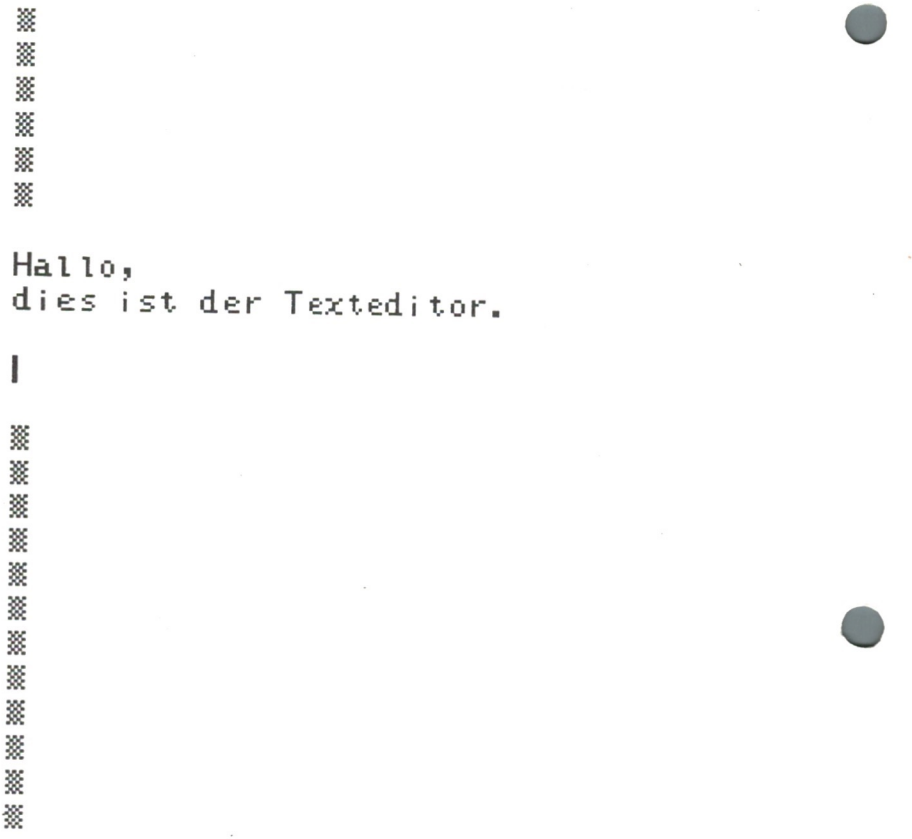
Der Texteditor.

Der Editor berarbeitet das aktuelle Textmenue. Nach dem Aufruf erscheint das Textfenster, daß nach dem Einschalten zunächst leer ist. Bild 9 zeigt ein Beispiel. Wenn man 80 Zeichen pro Zeile im Optionen-Menue eingestellt hat, so kann man, wie z.B. Bild 10 auch Textverarbeitung mit treiben. Die Befehle des Texteditors kann man sich jederzeit durch Drücken der Tasten CTRL und J (oder LF) auf den Bildschirm bringen (Bild 11 und 12). Durch Drücken weitere Tasten gelangt man wieder zurück.
Der Pfeil bedeutet, daß man die Taste CTRL mit dem nachfolgenden Buchstaben zusammen drücken muß. Dabei wird immer zuerst CTRL gedrückt und dann dazu die andere Taste.

Einige Besonderheiten:
Wenn man mit dem Befehl SUCHEN arbeitet, so kann man die Suche durch Drücken von CTRL-L fortsetzen.
Bei ERSETZEN, wird die Zeichenfolge erst dann durch eine neue ersetzt, wenn man die Taste "J" drückt. Wenn man "N" eingibt, so wird die Zeichenkette nicht ersetzt, stattdessen wird die nächste Zeichenkette gesucht.
Wenn man eine andere Taste drückt, wird die Suche abgebrochen.
Mit den Block-Befehlen, kann man Blöcke verschieben, löschen oder kopieren. Dabei erscheint ein Pfeil nach rechts als Marke für den Blockanfang und ein Pfeil nach links als Zeichen für das Blockende. Bei großen Blöcken kann kurzzeitig auch noch ein umgedrehtes T als Einfügemarke sichtbar sein.
Mit CTRL K und R kann man Blöcke aus einem anderen Speicherbereich lesen und einkopieren. Mit CTRL K und W kann man einen Block aus einem Text kopieren und zusätzlich beginnend bei einer anzugebenden Adresse ablegen.
Die Blockmarken werden beim Verlassen des Editors automatisch entfernt und stören nicht weiter.
Der Editor wird durch Drücken der Tasten CTRL und K und dann X verlassen. Wenn man versehentlich den RESET-Taster drückt ist der gerade sichtbare Textteil verloren.
Im unteren Bildschirmteil gibt der Editor einige Zusatzinformationen. Textstart ist die aktuelle Anfangsadresse des Textes. Fenster gibt die Adresse des Bildschirmfensters an. Und TOR ist das aktuelle Speicherende. Dabei wird der Bildschirminhalt nicht dazugerechnet. Wenn man die Gesamtgröße wissen will, so muß man mit CTRL-Q und C und dann CTRL-C den Text ganz aus dem Bildfenster herausschieben, so daß nur das Karo am linken Teil erscheint. Dann gibt TOR die effektive Endadresse an.
Achtung, das Karo muß man durch überschreiben weglöschen, sonst wird der Text nicht in den Speicher übernommen.

Bild 13 zeigt noch eine grobe Übersicht über die einzelnen Speicherbereiche, die wachsen können. Die Symboltabelle kommt vom System-RAM, und wächst in Richtung Text. Wenn sie zu groß wird, so wird

der Text überschrieben, daher immer zuerst die Quelle abspeichern, bevor man den Assembler aufruft, oder gar das Programm startet. Dann das Textgebiet, das ebenfalls wächst. Man kann es aber im gesamten Speicher irgendwohin legen. Am Ende des Textes wird vom Assembler noch die Debugtabelle gelegt, falls man das im Optionenmenue festgelegt hat. Dann das Programm, der Objektcode. Wenn man keine ORG-Anweisung gibt, so wird das Programm ab Adresse $9c00 abgelegt. Es wächst dann natürlich auch. Und schließlich der Stack, der immer automatisch an das Ende des phsyiklischen Speichers gelegt wird und zwar an das Ende des ersten zusammenhängenden Speicher-Bereichs.



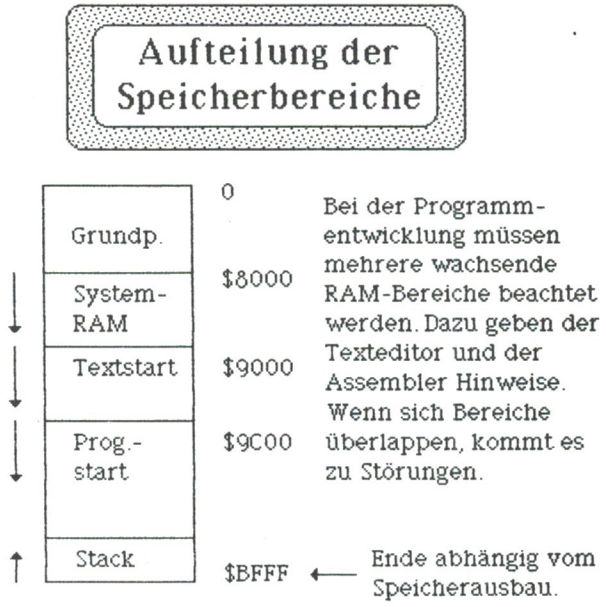
```
Hallo,
dies ist der Texteditor.
Man kann auch 80 Zeichen pro Zeile darstellen.
Gemischte Darstellung von deutschen Zeichen: äöüÄÖÜß
und amerikanischen Zeichen: {|} [\] ¬ ist jetzt möglich.
Textstart=00A000 Fenster=00A000 Tor=00A000      deut CTRL-J=Hilfe
```

REV4.3 -1-

^E=Zeile hoch ^X=Zeile runter
^S=Zeichen links ^D=Zeichen rechts
^A=Wort links ^F=Wort rechts
^Z=Zeile auf ^W=Zeile ab
^Y=Zeile loeschen ^N=Zeile einfuegen
^T=Loeschen rechts
^R=Seite zurueck ^C=Seite vor
^V=Einfuegemode ^P=Zeichensatz
^QC=an Textende ^QR=an Textanfang
^QF=suchen ^QA=ersetzen
^G,DEL=Zeichen loeschen
^L=wiederhole ^QF,^QA

^KX=ENDE

^KB=Block Anf. ^KK=Block Ende
^KY=Block loeschen
^KV=verschieben ^KC=kopieren
^KR=lesen ^KW=speichern
^KH=Marken loeschen



Der Assembler

Der Assembler versteht die Syntax, wie sie von Motorola für den 68000 und 68008 festgelegt wurde. Eine Befehlsliste und Hinweise sind in Bild 14 zusammengestellt. Es wird die Anschaffung eines Assembler-Handbuchs empfohlen.

Die Syntax von arithmetischen Ausdrücken, ist bei den Unterprogrammen WERT (29) und ZUWEIS (30) beschrieben.

Verwendete Abkürzungen:

cc	Bedingungscode (CC,CS,EQ,GE,GT,HI,LE,LS,LT,MI,NE,PL,VC,VS, T,F)
.x	Wortbreite (.B,.W,.L) bei Adressregistern ist nur .W und .L erlaubt)
reglist	Registerbereichsangabe, z.B. D0-D3/A1/A3-A6
konst	Konstante, dies kann auch ein komplexer Ausdruck mit Symbolen und arithmetischen Operationen sein
Dn,Dx,Dy	Angabe eines Datenregisters D0,D1....D7
An,Ax,Ay	Angabe eines Adressregisters A0,A1...A7
Rx,Ry	Daten oder Adressregister
marke	Zieladresse, oder Ausdruck
<ea>	Effektive Adresse:

Dn	Datenregister
An	Adressregister
(An)	indirekte Adresse
(An)+	mit Post-Increment
-(An)	mit Pre-Decrement
d(An)	mit Displacement
d(An,Rx)	mit Displacement und Index
wert.W	absolute Adresse 16 Bit
wert.L	absolute Adresse 32 Bit
d(PC)	PC-Relativ
d(PC,Rx)	mit Index
#konst	direkter Wert

Nicht alle Adressierarten sind bei allen Befehlen mit der <ea>-Angabe zulässig. Als Ziel ist z.B. der direkte Wert nicht erlaubt, als Ziel auch nicht die PC-Relativ-Adressierart.

d,marke,konst wert	Es sind immer komplexe Ausdrücke mit Symbolen und arithmetischen Operationen erlaubt.

Befehlssatz des 68008:

ABCD.B Dx,Dy	XNZVC	Addition von BCD-Zahlen
ABCD.B -(An),-(An)	XNZVC	Addition von BCD-Zahlen
ADD.x <ea>,Dn	XNZVC	Binäre Addition
ADD.x Dn,<ea>	XNZVC	Binäre Addition
ADDA.x <ea>,An	-	Binäre Addition mit Adressregister als Ziel
ADDI.x #konst,<ea>	XNZVC	Binäre Addition mit Konstante als Quellop.
ADDQ.x #konst,<ea>	XNZVC	Addiere schnell (Werte 1..8)
ADDX.x Dx,Dy	XNZVC	Addiere mit Extendbit
ADDX.x -(Ax),-(Ay)	XNZVC	Addiere mit Extendbit
AND.x <ea>,Dn	-NZVC	Logische Und-Verknüpfung
AND.x Dn,<ea>	-NZVC	Logische Und-Verknüpfung
ANDI.x #konst,<ea>	-NZVC	Logische Und-Verknüpfung mit einer Konst.
ANDI #konst,CCR	XNZVC	Und-Verknüpfung mit Condition-Code-Reg.
ANDI #konst,SR	XNZVC	Und-Verknüpfung mit Statusregister (priv)
ASL.x Dx,Dy	XNZVC	Links Schieben indirekt
ASL.x #konst,Dn	XNZVC	Links Schieben, um "konst" Bits
ASL <ea>	XNZVC	Links Schieben, um Eins
ASR.x Dx,Dy	XNZVC	Rechts Schieben indirekt
ASR.x #konst,Dn	XNZVC	Rechts Schieben, um "konst" Bits
ASR <ea>	XNZVC	Rechts Schieben, um Eins
Bcc marke	-	Bedingter Sprung, 16-Bit-Adressdistanz
Bcc.S marke	-	Bedingter Sprung, 8-Bit-Adressdistanz
BCHG Dn,<ea>	--Z--	Prüfe ein Bit und komplementiere es
BCHG #konst,<ea>	--Z--	Prüfe ein Bit und komplementiere es
BCLR Dn,<ea>	--Z--	Prüfe ein Bit und setze es auf Null
BCLR #konst,<ea>	--Z--	Prüfe ein Bit und setze es auf Null
BRA marke	-	Sprungbefehl, 16-Bit-Adressdistanz
BRA.S marke	-	Sprungbefehl, 8-Bit-Adressdistanz
BSET Dn,<ea>	--Z--	Prüfe ein Bit und setze es auf Eins
BSET #konst,<ea>	--Z--	Prüfe ein Bit und setze es auf Eins
BSR marke	-	Unterprogrammaufruf, 16-Bit-Adressdistanz
BSR.S marke	-	Unterprogrammaufruf, 8-Bit-Adressdistanz
BTST Dn,<ea>	--Z--	Prüfe ein Bit
BTST #konst,<ea>	--Z--	Prüfe ein Bit
CHK <ea>,Dn	-NZVC	Prüfe gegen Grenzbereich
CLR.x <ea>	-NZVC	Löschen den Inhalt eines Operanden
CMP.x <ea>,Dn	-NZVC	Vergleich zweier Operanden
CMPA.x <ea>,An	-NZVC	Vergleich mit einem Adressregister
CMPI.x #konst,<ea>	-NZVC	Vergleich einer Konstanten mit Operand
CMPM.x (Ax)+,(Ay)+	-NZVC	Vergleich zweier Speicherzellen
DBcc Dn,marke	-	Prüfe Bedingung, decrementiere und spinge
DBRA Dn,marke	-	Decrementiere und springe
DIVS <ea>,Dn	-NZVC	Division mit Vorzeichen
DIVU <ea>,Dn	-NZVC	Division ohne Vorzeichen
EOR.x Dn,<ea>	-NZVC	Logische Exklusiv-Oder-Verknüpfung

EORI.x #konst,<ea>	-NZVC	Logische Exklusiv-Oder-Verknupfung
EORI #konst,CCR	XNZVC	Exklusiv-Oder mit Condition-Code-Register
EORI #konst,SR	XNZVC	Exklusiv-Oder mit Status-Register (priv)
EXG Rx,Ry	-	Vertausche Registerinhalte
EXT.x Dn	-NZVC	Vorzeichenrichtige Erweiterung
ILLEGAL	-	Unzulässiger Befehl, fuhrt zur Exception
JMP <ea>	-	Springe absolut
JSR <ea>	-	Unterprogrammaufruf absolut
LEA <ea>,An	-	Lade effiktive Adresse in Adressregister
LINK An,#konst	-	Rette A7 und lege neuen Stackbereich an
LSL.x Dx,Dy	XNZVC	Logische Verschiebung nach Links, indirekt
LSL.x #konst	XNZVC	Logische Verschiebung nach Links
LSL.x <ea>	XNZVC	Logische Verschiebung nach Links
LSR.x Dx,Dy	XNZVC	Logische Verschiebung nach Rechts, indirekt
LSR.x #konst	XNZVC	Logische Verschiebung nach Rechts
LSR.x <ea>	XNZVC	Logische Verschiebung nach Rechts
MOVE.x <ea>,<ea>	-NZVC	Übertrage Daten von Quelle nach Ziel
MOVE <ea>,CCR	XNZVC	Übertrage ein Datum in Condition-Code-Reg.
MOVE <ea>,SR	XNZVC	Übertrage ein Datum in Status-Reg. (priv)
MOVE SR,<ea>	XNZVC	Übertrage Status-Reg. in Zieloperand
MOVE USP,An	-	Anwenderstackpointer nach An (priv)
MOVE An,USP	-	An nach Anwenderstackpointer (priv)
MOVEA.x <ea>,An	-	Übertrage Wert in Adressregister
MOVEM.x reglist,<ea>	-	Register nach Ziel speichern
MOVEM.x <ea>,reglist	-	Quelle nach Register
MOVEP.x Dx,d(Ay)	-	Inhalt von Dx nach Adresse d(Ay)
MOVEP.x d(Ay),Dx	-	Inhalt von Adresse d(Ay) nach Dx
MOVEQ #konst,Dn	-	Laden einer Konstanten, schnell
MULS <ea>,Dn	XNZVC	Multiplikation mit Vorzeichen
MULU <ea>,Dn	XNZVC	Multiplikation ohne Vorzeichen
NBCD <ea>	XNZVC	Negation von BCD-Zahlen
NEG.x <ea>	XNZVC	Negation
NEGX.x <ea>	XNZVC	Negation mit Berücksichtigung des X-Flags
NOP	-	Keine Operation (Nichtstubefehl)
NOT.x <ea>	XNZVC	Logisches Komplement
OR.x <ea>,Dn	-NZVC	Logische Oder-Verknüpfung
OR.x Dn,<ea>	-NZVC	Logische Oder-Verknüpfung
ORI.x #konst,<ea>	-NZVC	Logische Oder-Verknüpfung mit Konstanten
ORI #konst,CCR	XNZVC	Logisches Oder mit Condition-Code-Register
ORI #konst,SR	XNZVC	Logisches Oder mit Status-Register (priv)
PEA <ea>	-	Lege die effektive Adresse auf dem Stack ab
RESET	-	Rücksetzen externen Peripherie (priv)
ROL.x Dx,Dy	XNZVC	Rotiere links, indirekt
ROL.x #konst,Dy	XNZVC	Rotiere links, um Anzahl "konst"
ROL.x <ea>	XNZVC	Rotiere rechts, um Eins
ROR.x Dx,Dy	XNZVC	Rotiere rechts, indirekt
ROR.x #konst,Dy	XNZVC	Rotiere rechts, um Anzahl "konst"

ROR.x <ea>	XNZVC	Rotiere rechts, um Eins
ROXL.x Dx,Dy	XNZVC	Rotiere links, indirekt mit X-Flag
ROXL.x #konst,Dy	XNZVC	Rotiere links, um Anzahl "konst" mit X-Flag
ROXL.x <ea>	XNZVC	Rotiere rechts, um Eins mit X-Flag
ROXR.x Dx,Dy	XNZVC	Rotiere rechts, indirekt mit X-Flag
ROXR.x #konst,Dy	XNZVC	Rotiere rechts, um Anzahl "konst" mit X-Flag
ROXR.x <ea>	XNZVC	Rotiere rechts, um Eins mit X-Flag
RTE	XNZVC	Rückkehr aus Exception (priv)
RTR	XNZVC	Rückkehr mit Laden der Flags
RTS	-	Rückkehr aus dem Unterprogramm
SBCD Dx,Dy	XNZVC	Subtraktion von BCD-Zahlen
SBCD -(Ax),-(Ay)	XNZVC	Subtraktion von BCD-Zahlen
Scc	XNZVC	Setze ein Byte je nach Bedingung
STOP #konst	XNZVC	Lade Statusregister und HALT (priv)
SUB.x <ea>,Dn	XNZVC	Binäre Subtraktion
SUB.x Dn,<ea>	XNZVC	Binäre Subtraktion
SUBA.x <ea>,An	-	Binäre Subtraktion ins Adressregister
SUBI.x #konst,<ea>	XNZVC	Binäre Subrtraktion mit einer Konstanten
SUBQ.x #konst,<ea>	XNZVC	Subtrahiere schnelle (1..8)
SUBX.x Dx,Dy	XNZVC	Subtrahiere mit Extendbit
SUBX.x -(Ax),-(Ay)	XNZVC	Subtrahiere mit Extendbit
SWAP Dn	-NZVC	Tausche Inhalt von Registerhälften
TAS <ea>	-NZVC	Prüfe und setze ein Bit im Zieloperanden
TRAP #konst	-	Trap-Befehl
TRAPV	-	Trap bei Überlauf
TST.x <ea>	-NZVC	Prüfe den Inhalt eines Operanden
UNLK An	-	Hole den Stackpointer zurück

Pseudobefehle des Assemblers:

DC.x werteliste
Die durch Komma getrennten Werte der Werteliste werden im Speicher abgelegt.
Dabei sind bei DC.B auch Texte möglich, die zwischen zwei einfachen Anführungzeichen gesetzt werden.
Beispiel: DC.B 1,'Hallo',0
DC.W $AA55
DC.L alpha, 345, geht-4

DS.x wert
Ein Speicherbereich wird reserviert.
Dorthin werden keine Daten abgelegt.
Beispiel : DS.W 100

DS 0
Falls der Programmzahler auf einer ungeraden Adresse steht, wird ein Byte in das Programm eingefugt, sonst nicht. Die nächste freie Speicherzelle liegt damit immer auf einer geraden Adresse.

END
Ende des Assemblerprogramms. Aller folgender Text in der Quelle wird ignoriert.

symbol EQU wert
Der Wert des Ausdrucks "wert" wird dem Symbol zugewiesen. Jedes Symbol darf aber nur eine Zuweisung erhalten.
Beispiel: HUNDERT EQU 100

OFFSET wert
Wird zur Ablageadresse addiert, ohne jedoch bei Marken Einfluß zu haben.
Damit kann man Code z.B. für die Adresse 0 erzeugen und dennoch im RAM ablegen.
Beispiel:
OFFSET $9C00
ORG 0

ORG wert
Startadresse für die Ablage des Codes. Wenn kein OFFSET angeben wurde ist dies die Speicheradresse von wo ab das folgende Assemblerprogramm, oder Daten abgelegt wird.
Beispiel: ORG $A000

SCHREITE 1

(Schildkrötengraphik)
Der Befehle Schreite bewegt die Schildkröte auf dem Bildschirm in die Richtung, in die sie gerade blickt. Das Register D0.W gibt dabei die Anzahl der Schritte an. Ein Schritt entspricht in X-Richtung einem Bildpunkt und in Y-Richtung einem halben Bildpunkt. Damit wird erreicht, daß sich symmetrische Figuren auf dem Bildschirm ergeben.

Beispiel:

```
MOVE #10,D0     * Zehn Schritte in die
JSR @SCHREITE   * Blickrichtung gehen.
RTS             * Programmende
```

Wird ein negativer Wert im Register D0.W angegeben, so schreitet die Schildkröte rückwärts.

Beispiel:

```
MOVE #-10,D0    * Zehn Schritte zurueck gehen
JSR @SCHREITE
RTS
```

Wenn die angegebene Zahl der Schritte zu groß ist, verschwindet die Schildkröte aus dem sichtbaren Bereich. Der Wert sollte aber 4096 nicht überschreiten, sonst erscheint die Schildkröte wieder im Bildbereich.

Bei allen Befehlen der Schildkrötengraphik, werden automatisch zwei Bildseiten verwendet. Diese beiden Bildseiten werden alle 20ms umgeschaltet. Auf der Bildseite 0 wird die Graphik dargestellt und auf der Bildseite 1 die Schildkröte, die im folgenden auch manchmal Zeiger genannt wird. Der Zeiger besteht dabei aus einem Dreieck, dessen Spitze in die aktuelle Blickrichtung zeigt. Der Zeiger kann die Richtung in 45 Grad-Schritten anzeigen. Die Schildkröte kann aber intern auf Ein Grad genau ausgerichtet werden. Will man den Umschaltvorgang der Bildseiten nicht haben, so muß man mit speziellen Befehlen den Vorgang abschalten.

DREHE 2

(Schildkrötengraphik)
Die Schildkröte wird entgegen dem Uhrzeigersinn gedreht (mathematisch positiv). Der Winkel um den sie gedreht werden soll, wird dabei in Grad angegeben. Der Winkel steht im Register D0.W. Wenn eine negative Zahl vorliegt, so dreht sich die Schildkröte im Uhrzeigersinn.

Beispiel:

```
MOVE #45,D0     * Um 45 Grad entgegen dem
JSR @DREHE      * Uhrzeiger drehen.
RTS
```

Beispiel bei negativem Winkel:

```
MOVE #-30,D0    * Um 30 Grad im Uhrzeiger
JSR @DREHE      * drehen.
RTS
```

Wird ein Winkel angegeben, der größer als 359 Grad oder kleiner als 0 Grad ist, so er wird im Programm DREHE immer automatisch auf den Bereich 0 bis 359 Grad umgerechnet und dann der Zeiger danach ausgerichtet. Wichtig ist noch, daß sich der angegebene Winkel immer auf die aktuelle Stellung des Zeigers bezieht, nicht dagegen auf den Bildschirmrand. Dabei gehorcht der Zeiger der in LOGO üblichen Weise.

HEBE 3

(Schildkrötengraphik)
Die Schildkröte hinterläßt ab sofort keine Schreibspur mehr beim bewegen. Damit kann man isolierte Figuren zeichnen.
Zu dem Befehl gehört auch das Gegenstück "SENKE".

Beispiel:

```
JSR @HEBE
RTS
```

kombiniertes Beispiel:

```
MOVE #20,D0     * 20 Schritte schreiten
JSR @SCHREITE
JSR @HEBE       * dann Stift anheben
MOVE #20,D0     * wieder 20 Schritte gehen
JSR @SCHREITE   *
JSR @SENKE      * Stift absenken
MOVE #20,D0     * und 20 Schritte gehen
JSR @SCHREITE
RTS
```

Der Befehl wirkt nur auf Befehle der Schildkrötengrahik.

SENKE 4

(Schildkrötengraphik)
Der Schreibstift wird gesenkt. Von da an wird eine Schreibspur hinterlassen, wenn sich der Zeiger (sprich: Schildkröte) bewegt.
Zu diesem Befehl gehört auch das Kommando "HEBE".

Beispiel:

```
JSR @SENKE      * Schreibstift senken
RTS
```

kombiniertes Beispiel:

```
MOVE #20,D0     * 20 Schritte schreiten
JSR @SCHREITE
JSR @HEBE       * dann Stift anheben
MOVE #20,D0     * wieder 20 Schritte gehen
JSR @SCHREITE   *
JSR @SENKE      * Stift absenken
MOVE #20,D0     * und 20 Schritte gehen
JSR @SCHREITE
RTS
```

Der Befehl wirkt nur auf Befehle der Schildkrötengraphik.

SET 7

(Schildkrötengraphik)
Damit kann die Schildkröte direkt auf eine absolute Position gesetzt werden. Die Schildkröte hinterläßt dabei keine Schreibspur.

Register D1.W erhält die X-Koordinate. (Bereich 0 bis 511)
Register D2.W erhält die Y-Koordinate. (Bereich 0 bis 511)
Register D3.W erhält die Blickrichtung in Grad, dabei zeigt die Schildkröte bei 0 Grad nach Rechts und bei 90 Grad nach oben.

Beispiel:

```
MOVE #256,D1    * X=256
MOVE #256,D2    * Y=256
MOVE #90,D3     * Blick nach oben
JSR @SET        * und ausfuehren
RTS
```

kombiniertes Beispiel:

```
MOVE #0,D1      * X=0
MOVE #10,D2     * Y=10
MOVE #-45,D3    * Blick 45 Grad nach rechts unten
JSR @SET
MOVE #10,D0     * dann zehn Schritte vorwaerts
JSR @SCHREITE
RTS
```

Wichtig ist hier, daß sich der angegebene Winkel absolut auf den Bildschirmrand bezieht und nicht wie z.B. beim DREHE-Befehl auf die alte Position der Schildkröte.

MOVETO 8

Damit wird die nächste Schreib- oder Zeichenstelle, für Graphik oder Text vorbereitet. Dabei werden direkt die Register des Graphik-Prozessors eingestellt.
Als Parameter steht in Register D1.W die X-Koordinate. Dabei ist die Schreibstelle auf dem Bildschirm sichtbar, wenn die Koordinate im Bereich 0 bis 511 liegt, sonst zeichnet er im unsichtbaren. Dabei ist der gesamte Bildpunktraum 4096 groß. Also die Koordinate x=4096 entspricht der Koordinate 0 und 4097 dem Wert 1.
Im Register D2.W steht die y-Koordinate. Dabei ist hier der sichtbare Bereich von 0 bis 255 und es gibt ebenfalls 4096 verschiedene Koordinaten. Negative Werte sind auch zulässig, aber ebenfalls nicht im sichtbaren Bereich.

Beispiel:

```
                    * Einen Punkt setzen
MOVE #10,D1         * X=10
MOVE #30,D2         * Y=20
JSR @MOVETO         * Position setzen
MOVE #$80,D0        * PUNKT-Befehl
JSR @CMD            * Befehl an den GDP schicken
RTS
```

Der Punkt erscheint im linken unteren Bildbereich. Der Befehl CMD folgt noch später, MOVETO ist aber nur mit anderen Befehlen zusammen sinnvoll.

DRAWTO 9

Der Graphik-Prozessor zeichnet eine Linie, von der zuletzt eingestellten Position zu der neuen Position, die beim DRAWTO-Befehl angegeben wird. Register D1.W erhält dazu die X-Koordinate,
Register D2.W die Y-Koordinate. Für die Bereiche gilt das gleiche, wie beim MOVETO-Befehl (X von 0 bis 511 und Y von 0 bis 255).

Beispiel:

```
MOVE #0,D1      * Linie von x=0,y=0
MOVE #0,D2
JSR @MOVETO     * Startpunkt festlegen
MOVE #511,D1    * bis x=511
MOVE #255,D2    * und y=255
JSR @DRAWTO     * ergibt Diagonale
RTS
```

Beispiel mit unsichtbaren Start- und Endpunkten:

```
MOVE #-100,D1   * X=-100
MOVE #200,D2    * Y=200
JSR @MOVETO
MOVE #200,D1    * bis X=200
MOVE #-100,D2   * Y=-100
JSR @DRAWTO
RTS
```

Der Wiedereintritt in den Bildbereich wird vom Graphik-Prozessor korrekt ausgeführt, damit kann man Ausschnitte aus großen Zeichnungen durch verschieben der Anfangskoordinaten erzeugen.
Will man eine Linie löschen, so kann man mit dem Befehl ERAPEN die Schreibart "Löschen" vorwählen. Mit SETPEN wird sie wieder in den Normalzustand gesetzt.

WRITE 10

Ein Befehl, mit dem man Texte auf dem Bildschirm ausgeben kann.
Dazu wird im Register D0.B die Textgröße festgelegt,
in Register D1.W die X-Koordinate (0 bis 511).
und in Register D2.W die Y-Koordinate (0 bis 255).
In das Adreßregister A0 wird die Startadresse des auszugebenden Textes geladen. Der Text besteht aus ASCII-Zeichen, das Ende des Textes wird durch eine 0 (Code 00) angegeben.

1. Textgröße festlegen:
Man kann die Größe eines Schriftzeichens in X und Y-Richtung getrennt angeben. Dazu muß man beide Wert mit einer kleinen Formel zusammenbauen.

```
g :=  dx * 16 + dy
```

Wenn x oder y den Maximalwert von 16 annehmen soll, so muß man anstelle der Zahl 16 den Wert 0 einsetzen. Dies ist ein Sonderfall, und ist vom Hersteller des Graphikprozessors (EF 9366) so bestimmt.
Beispiel:

```
dx=1, dy=1

g := 1*16+1
```

also ist g=16. Man kann den Wert auch sedezimal darstellen, es ergibt sich g=$11. Dort kann man den Zusammenbau aus den beiden Werten noch besser erkennen.
Beispiel:

```
dx=16, dy=16
```

"g" nimmt dann den Wert 0 an (Sonderfall).

2. Textbuffer:
Der abzulegende Text kann im Assembler einfach mit

```
DC.B '....'
```

angegeben werden.
Die Ablage des Endes erreicht man mit

```
DC.B 0
```

Die Ausgabe von ganzen Textzeilen kann man dann ganz leicht durch mehrere WRITE-Befehle erreichen, deren Y-Koordinaten sich unterscheiden. Eigene Experimente verschaffen aber wohl schnell Aufschluß.

Beispiel:

```
                MOVE.B #$32,D0      * Breitschrift dx=3,dy=2
                MOVE #120,D1        * X= 120
                MOVE #128,D2        * Y= 128 (Mitte)
                LEA BUFFER,A0       * Adresse des Textes
                JSR @WRITE          * Text ausgeben
                RTS                 * Ende des Programms
BUFFER:         DC.B 'Hallo',0      * Text "Hallo" ausgeben
```

Achtung! Die 0 beim DC.B-Befehl, die hinter dem Komma steht darf nicht vergessen werden, sonst weis das Grundprogramm nicht, wann der Text zu Ende ist. Die merkwürdigsten Ausgaben sind dann die Folge.
Man kann den Bufferinhalt aber auch wie folgt definieren:

```
DC.B 'Hallo'
DC.B 0
```

Hier wird die 0 mit einem getrennen DC.B-Befehl abgelegt.

Achtung! Wenn hinter dem Text weitere Programmteile folgen, so kann die Fehlermeldung "Nicht auf Wortgrenze" erscheinen. Dann muß man den Befehl:

```
DS 0
```

vor den neuen Befehl schreiben. Damit wird automatisch ein leeres Byte eingefügt, wenn die nachfolgende Adresse ungerade sein sollte. 68000-Befehle dürfen nämlich nur auf geraden Adressen zu liegen kommen.

READ 11

Einlesen eines Textes von der Tastatur. Damit kann man Eingabefelder erzeugen, wie sie auch im Grundprogramm, z.B. zur Eingabe von "Adresse" oder "Name" verwendet werden.
Das Register D0.B erhält die Textgröße wie beim WRITE-Befehl.
Das Register D1.W erhält die X-Koordinate,
das Register D2.W die Y-Koordinate,
das Register D3.W die maximale Anzahl der einzulesenden Zeichen. Damit auch die Breite des Textfensters bestimmt.
Das Register A0 schließlich erhält die Adresse des Textbuffers, in dem nach dem Aufruf der eingegebene Text abgelegt sein wird.
Nach dem Aufruf steht in Register D4.W die wirkliche Anzahl der eingebenen Zeichen und in Register D5.B das Zeichen, das zuletzt eingetippt wurde. Damit ist es möglich zu unterscheiden, ob die Eingabe mit einem CR beendet wurde, oder durch ein anderes Zeichen, z.B. weil mehr Zeichen eingeben wurden, als in Register D3.W als Maximum zulässig waren. Im Buffer steht an letzter Stelle eine 0 (Code 00), um das Ende des eingebenen Textes einfach erkennen zu können. Daher muß für den Buffer immer ein Byte mehr reserviert werden, als es die Anzahl in D3 angibt.
Mit DEL oder "Pfeil links" kann man Zeichen löschen, solange man im Eingabefeld ist. Ein blinkender Cursor zeigt die aktuelle Textposition.
Alle anderern Steuertasten wie CR,LF führen zum Ende der Eingabe. Das Zeichen steht anschließend in Register D5.B wie schon gesagt.

Beispiel:

```
            MOVE.B #$22,D0      * dx=2, dy=2
            MOVE #10,D1         * X=10
            MOVE #100,D2        * Y=100
            MOVE #12,D3         * Maximal 12 Zeichen
            LEA BUFFER,A0       * Adresse des Buffers
            JSR @READ           * und einlesen
            RTS                 * Fertig
BUFFER:     DS.B 13             * Reservierter Speicher (+1 Reserve)
```

Achtung! Wenn X=0,1 oder Y=0,1 wird ein Teil der Umrahmung abgeschnitten, da der Textrahmen nicht bei der angebenen Koordinate anfängt, sondern zwei Punkte weiter links und weiter unten.
zeichnen.

CI 12

Ein Zeichen wird von der Tastatur gelesen. Dabei wird solange gewartet, bis das Zeichen auch wirklich eingegeben wird. Das Zeichen erscheint im ASCII-Code im Register D0.B.

Beispiel:

```
        JSR @CI         * Zeichen einlesen
        RTS             * Ergebnis ist D0.B
```

kombiniertes Beispiel:

```
START:  JSR @CI         * Zeichen einlesen
        CMP.B #$D,D0    * und warten bis CR
        BNE.S START     * gedrueckt wird.
        RTS             * Dann erst Rueckkehr
```

kombiniertes Beispiel:

```
START:  JSR @CI         * Zeichen einlesen
        CMP.B #'f',D0   * war es die Taste Klein-F
        BNE.S ST1       * nein, dann weiter
        MOVE #10,d0     * Ja, dann Schreiten
        JSR @SCHREITE   * in der Schildkroetensprache
        BRA.S START     * dann wieder zurueck
ST1:    CMP.B #'d',D0   * war es die Taste Klein-D
        BNE.S ST2       * nein, dann weiter
        MOVE #45,D0     * Ja, dann um 45 Grad drehen
        JSR @DREHE      *
        BRA.S START     * und wieder zurueck
ST2:    CMP.B #'h',d0   * war es die Taste Klein-H
        BNE.S ST3       * nein, dann weiter
        JSR @HEBE       * sonst HEBE-Befehl
        BRA.S START     * Dann zurueck
ST3:    CMP.B #'s',d0   * oder die Taste Klein-S
        BNE.S START     * Nein, dann zurueck
        JSR @SENKE      * Sonst SENKE-Befehl
        BRA.S START     * und dann erst zurueck
```

Dies ist eine Mini-Schildkrötensprache. Wenn man das Programm startet und die Taste "f" drückt, so werden 10 Schritte ausgeführt und eine kleine Linie erscheint. Mit der Taste "d" kann man die Schildkröte in 45 Grad-Schritten drehen. Mit der Taste "h" kann man verhindern, daß eine Schreibspur hinterlassen wird und mit der Taste "s" kann man sie wieder einschalten. Damit kann man kleine Zeichnungen erstellen.
Wer will kann das Programm zu einem richtigen kleinen CAD-System ausbauen.

Die Eingabe läßt sich aber auch umleiten, wenn man z.B. eine eigene Eingaberoutine geschrieben hat. Dazu muß man in die Speicherzelle 802B (IOSTATB) den Wert 6 schreiben. Dann erfolgt beim Aufruf von CI ein Sprung auf die Adresse $8018 (USERCI). Dort kann man einen Sprung auf die eigene Routine hinsetzen. Im Register D0.W wird der Wert 2 an das Benutzerprogramm übergeben, um die CI-Routine von der CI2-Routine unterscheiden zu können, den diese verwendet bei einer Umschaltung die gleiche Sprungadresse (USERCI), jedoch mit dem Wert 0 im Register D0.W.
Wenn man CI wieder normalisieren will, so muß man den Wert 0 auf die Adresse $802B schreiben.
Gleichzeitig mit CI wird auch CSTS umgeschaltet und zwar auf die Adresse $801E (USERCSTS), denn CI und CSTS gehören immer zusammen.

CSTS 13

Eng verwandt mit dem CI-Befehl ist der Befehl CSTS. Damit ist es möglich abzufragen, ob eine Taste gedrückt wurde oder nicht. Im Register D0.B erscheint eine 0, wenn keine Taste gedrückt wurde, sonst ein Wert ungleich 0. Das Zeichen wird aber nicht eingelesen, sondern muß dann mit dem Befehl CI geholt werden. Auch bleibt der Status solange erhalten, bis das Zeichen mit CI geholt wurde.

Beispiel:

```
        JSR @CSTS       * Status holen
        RTS             * D0.B=0, wenn kein Zeichen da.
```

kombiniertes Beispiel:

```
START:  MOVE #1,D0      * Zeichnen eines Kreises
        JSR @SCHREITE   * (360 - Eck)
        MOVE #1,D0      * Abbruch, bei Tastendruck
        JSR @DREHE
        JSR @CSTS       * Fragen, ob Taste gedrueckt
        BEQ.S START     * nein, dann zurueck
        RTS             * Ja, dann aufhoeren
```

kombiniertes Beispiel:

```
START:     CLR D4          * Geschwindigkeit 0
SCHLEIFE:  MOVE D4,D0      * Schildkroete bewegen
           JSR @SCHREITE   *
           JSR @CSTS       * Wenn nicht Taste gedrueckt
           BEQ.S SCHLEIFE  * Dann zurueck
           JSR @CI         * Sonst abfragen des Zeichens
           CMP.B #'e',d0   * Aufwaerts = Klein-E
           BNE.S ST1       * nein, dann weiter
           ADDQ #1,D4      * Geschwindigkeit erhoehen
           BRA.S SCHLEIFE  * und zurueck
ST1:       CMP.B #'x',d0   * Abwaerts = Klein-X
           BNE.S SCHLEIFE  * nein, dann zurueck
           SUBQ #1,D4      * sonst Geschw. vermindern
           BRA.S SCHLEIFE  * und zurueck
```

Der Ablauf ist sehr schnell, und aus dem obigen Programm läßt sich wohl leicht ein Geschicklichkeitsspiel machen.
Wenn man auf die Adresse $802B (IOSTATB) den Wert 6 schreibt, so erfolgt ein Sprung auf die Adresse $801E (USERCSTS), wenn man die Routine CSTS aufruft. Gleichzeitig wird auch die CI-Routine umgeleitet. Man erhält als Kennung den Wert 2 im Register D0.W.
Siehe auch CI.

RI 14

Ein Zeichen wird von der CAS-Schnittstelle gelesen. Dabei wird solange gewartet, bis ein Zeichen ankommt. Das Zeichen erscheint im ASCII-Code im Register D0.B

Beispiel:

```
        JSR @RI         * Zeichen einlesen
        RTS             * Zeichen in D0.B
```

kombiniertes Beispiel:

```
START:    LEA BUFFER,A0    * Zieladresse in A0
SCHLEIFE: JSR @RI          * Zeichen lesen
          MOVE.B D0,(A0)+  * Zeichen in Speicher ablegen
          BRA.S SCHLEIFE   * wiederholen
BUFFER:   DS.B 1000        * z.B. 1000 Zeichen
```

Das Programm kann zum Beispiel dazu genutzt werden um ein Band mit Daten in den Speicher einzulesen. Dabei ist aufgrund des verwendeten PE-Verfahrens, das erste eingelesene Zeichen ungueltig. Bei Daten, die z.B. vom Grundprogramm stammen, wird daher am Anfang immer ein unbedeutender Datenstrom den eigentlichen Daten vorweggeschickt.
Das obige Programm kann nur durch einen RESET erlöst werden. Denn es ist hier ja nicht bekannt, wieviele Zeichen kommen.
Mit "ANSEHEN" kann man sich den Inhalt des Buffers dann anschauen.

PO 15

Ein Zeichen wird über die CAS-Baugruppe ausgeben. Dabei steht das Zeichen zuvor im Register D0.B.

Beispiel:

```
START:   MOVE.B #'A',D0      * Ausgabe des Zeichens "A"
         JSR @PO
         RTS
```

komplettes Beispiel für Schreiben und Lesen:

```
SCHREIBE: MOVE #10-1,D3      * 10 Synchronisationsbytes
SCH1:     MOVE.B #$FF,D0     * z.B. FF verwenden
          JSR @PO            * und ausgeben
          DBRA D3,SCH1       *
          MOVE.B #0,D0       * Kennung f}r den Start
          JSR @PO            * ausgeben.
          LEA AUSBUF,A0      * Adresse des Textes laden
SCH2:     MOVE.B (A0)+,D0    * Datenbyte laden
          JSR @PO            * ausgeben
          BNE SCH2           * bis das Zeichen "NUL" erscheint
          RTS                * Dann Ausgabe beendet
AUSBUF:   DC.B 'HALLO',0     * Text
          DS 0               * angleich auf Even-Adresse
                             *
LIES:     JSR @RI            * Zeichen holen
          CMP.B #$FF,D0      * Suchen nach den Sync-Bytes
          BNE.S LIES         * immer weiter
LIES1:    JSR @RI            * dann muss 0 oder FF folgen
          CMP.B #$FF,D0      * Bei FF ok, zurueck
          BEQ.S LIES1
          CMP.B #0,D0        * Sonst muss es 0 sein
          BNE.S LIES         * oder bei Stoerung zurueck
          LEA EINBUF,A0      * Ziel fuer Eingabe
LIES2:    JSR @RI            * Zeichen holen
          MOVE.B D0,(A0)+    * und ablegen
          BNE.S LIES2        * bis 0 aufgetreten
          LEA EINBUF,A0      * Dann des Text auf
          MOVE #$22,D0       * dem Bildschirm ausgeben
          MOVE #2,D1         * X=2
          MOVE #128,D2       * Y=128
          JSR @WRITE         * und dann
          RTS                * Ende
EINBUF:   DS.B 100           * Hier Platz zum Einlesen
```

Der Aufzeichungsvorgang mit "SCHREIBE" geht bei diesem Beispiel sehr schnell, daher meldet sich das Menü auch sofort wieder.
Bei der Wiedergabe muß dann der Text auf dem Bildschirm erscheinen, dazu wird das Programm "LIES" gestartet.

In der Praxis muß man dieses einfache Programm noch um eine Prüfmöglichkeit erweitern. Denn treten beim Lesen Fehler auf, so muß eine Kontrollmöglichkeit vorhanden sein.
Meißt wird dazu eine Prüfsumme verwendet. Das heißt man summierte alle Datenbytes auf, verwendet aber nur z.B. 16 Bit des Eregbnis und gibt dieses Ergebnis als zwei Bytes am Schluß mit aus. Beim Einlesen bildet man erneut die Prüfsumme und vergleicht diese mit dem aufgezeichneten Wert. So ist es auch im Grundprogramm selbst realisiert, wenn man Texte oder Daten aufzeichnet. Ferner ist es nützlich die Anzahl der Datenbytes mit am Anfang aufzuzeichnen um auch beliebige Daten abspeicher zu können, die auch z.B. eine 0 enthalten.

CLR 16

Alle vier Bildseiten werden gelöscht. Der Vorgang dauert ca. 20ms*4 also ca. 80ms.
Die aktuelle Lese- und Schreibseite wird anschließend auf 0 eingestellt, so daß Bildschirmausgaben auf der Seite 0 erfolgen und auch dort sichtbar sind.

Beipsiel:

```
        JSR @CLR        * Loeschen des Bildschirms
        RTS
```

Will man nur eine Bildseite loeschen, so kann man auch einen direkten Befehl für den Graphikprozessor verwenden.
Dabei muß man aber darauf achten, daß Schreib- und Lese-Seite identisch sind, denn der Löschbefehl des GDPs arbeitet nur dann einwandfrei.

Beispiel löschen der Seite 0:

```
START:  CLR D0     * Seite 0 als Schreibseite anwaehlen
        CLR D1     * Seite 0 auch Leseseite
        JSR @NEWPAGE
        MOVE.B #4 * GDP - Loeschbefehl
        JSR @CMD   * Befehl direkt ausgeben
        RTS
```

Das Löschen dauert hierbei nur etwa 20ms und ist damit die schnellste Methode den Bildschirm zu löschen.

Damit kann eine unsichtbare Seite gelöscht werden. Der Vorgang dauert hierbei aber wesentlich länger als der schnelle Löschbefehl des GDPs, denn man kann bei unterschiedlicher Schreib– und Leseseite nicht den normalen Löschbefehl (Code 4 oder 7) verwenden.
Im Grundprogramm wird das Löschen der unsichtbaren Seite durch überschreiben mit dem Befehl 0A (5 x 8 Block) in der Schriftgröße 16 erreicht.
Den CLPG–Befehl benötigt man, wenn man auf einer unsichtbaren Bildseite ein neues Bild aufbauen will und dabei eine andere Bildseite sieht.

Beispiel:

```
        JSR @CLPG
        RTS
```

Beispiel:

```
START:     MOVE #1,SCHREIBSEITE * Start bei Schreib=1
           MOVE #0,LESESEITE    * Lese=0
           MOVE #0,D4           * Zaehler fuer SIN,COS
SCHLEIFE:  MOVE SCHREIBSEITE,D0
           MOVE LESESEITE,D1
           JSR @NEWPAGE         * Seite einstellen
           JSR @CLPG            * erst mal loeschen
           MOVE #256,D1         * X=256
           MOVE #128,D2         * Y=128
           JSR @MOVETO          * Startpunkt  Mitte
           MOVE D4,D0
           JSR @COS             * X=COS256(phi)+256
           ADD #256,D0
           MOVE D0,D1           * X-Koordinate
           MOVE D4,D0
           JSR @SIN             * Y=SIN256(phi)/2+128
           EXT.L D0             * Fuer DIV
           DIVS #2,D0
           ADD #128,D0          * Damit Zentrisch
           MOVE D0,D2           * Y-Koordinate
           JSR @DRAWTO          * Ziellinie
           EOR #1,SCHREIBSEITE  * Seiten wechseln
           EOR #1,LESESEITE
           ADD #10,D4           * 10 Grad dazu
           BRA SCHLEIFE         * und wiederholen
SCHREIBSEITE: DC.W 0
LESESEITE:    DC.W 0
```

Wenn man die Graphik schneller haben will, so verwendet man nicht den

CLPG-Befehl, sondern löscht das alte Bild, dadurch daß man den GDP auf Löschen umstellen (ERAPEN) und dann das alte Bild erneut schreibt. Dann darf man nicht vergessen den GDP wieder auf Schreiben (SETPEN) umzustellen. CLPG benötigt man eigentlich nur dann, wenn man das alte Bild nicht mehr rekonstruieren kann, oder wenn das erneute Einschreiben länger braucht als der CLPG-Befehl Zeit beansprucht.

WAIT 18

Immer wenn man direkt Daten auf die GDP-Ports ausgeben will, muß man solange warten, bis der GDP den letzten Befehl ausgeführt hat. Dies kann man direkt mit dem WAIT-Befehl tun.

Beispiel:

```
        JSR @WAIT
        RTS
```

ausführliches Beispiel:

```
            CPU EQU 1             * (68008=1 68000=2 68020=4)
            GDP EQU $FFFFFF70     * GDP Adresse
START:      CLR D3                * Size merken
            CLR D1                * X=0,Y=0
            CLR D2
            JSR @MOVETO           * Position einstellen
SCHLEIFE:   JSR @WAIT             * Warten bis fertig
            ADD.B #1,D3           * Groesse veraendern
            MOVE.B D3,GDP*CPU+3*CPU  * CSIZE-Register
            MOVE.B #'A',GDP*CPU   * Buchstabe "A" ausgeben
SYN:        JSR @SYNC             * 20 ms Warten
            BEQ SYN               * dann erst loeschen
            JSR @CLR              *
            BRA SCHLEIFE          * immer weiter so
```

Will man die Ausgabe Flimmerfrei haben, so muß man mit zwei Bildebenen arbeiten.
Das Programm gibt den Buchstaben "A" in allen möglichen Größen aus.

SCHR16TEL 19

(Schildkrötengraphik)
Damit ist es möglich um 1/16 Bildpunkt zu schreiten. Dieser Befehl wird z.B. für Kreise mit unterschiedlichen Radien benötigt.
Die Anzahl der Schritte wird in Register D0.W übergeben.

Beispiel:

```
MOVE #16,D0          * Schreitet um einen Bildpunkt
JSR @SCHR16TEL       * weiter
RTS
```

Aufgabe: Kreis mit gegebenem Radius zeichnen lassen:
Für einen Kreis mit gegebenem Radius gibt es eine bekannte Formel um den Umfang zu bestimmen:

$$U = 2 * PI * R$$

Wobei PI = 3.141592 ... , R = Radius und U = Umfang.

Das Kreisprogramm habe die allgemeine Form:

```
KREIS:
     MOVE #36-1,D3       * 36 Mal
SCHLEIFE:
     MOVE #ANZAHL,D0     * zu berechnen
     JSR @SCHR16TEL
     MOVE #10,D0         * 10 Grad drehen
     JSR @DREHE          * fuer Schnelligkeit
     DBRA D3,SCHLEIFE    * und min. Fehler beim
     RTS                 * Berechnen
```

Es gilt nun ANZAHL zu berechnen.
Der Umfang ergibt sich zu:

$$U = 36 * ANZAHL / 16$$

Wobei als Einheit 1 Bildpunkt gewählt wurde.
Damit läßt sich die Unbekannte ANZAHL ermitteln:

ANZAHL = (2 * PI * R) / 36 * 16

Man kann die Formel mit einer EQU–Anweisung annähern und es ergibt sich:

ANZAHL EQU (R*1000)/358

Die Multiplikation mit 1000 ist nötig um eine ausreichende Genauigkeit zu erhalten. Jetzt kann man einmal mit unterschiedlichen Werten für R experimentieren.
Man kann einen Kreis auch mit der Schleife für 360 Schritte konstruieren, jedoch ergibt sich dann ein Fehler durch die interne Rechengenauigkeit und die Umfangsformel kann nicht mehr angewandt werden.
Das es sich eigentlich nur um ein 36–Eck handelt ist kaum zu sehen und hängt natürlich von dem Durchmesser des Kreises ab.

CLRSCREEN 20

Der Bildschirm wird gelöscht. Dabei werden alle vier Bildseiten gelöscht. Ferner erscheint wird die Textausgabe vorbereitet und es erscheint ein blinkendes Cursorfeld links oben auf dem Bildschirm. Immer wenn man die Ausgaberoutinen CO und CO2 verwenden will, muß man mit CLRSCREEN die Ausgabe vorbereiten und den Cursor einschalten.

Beispiel:

```
JSR @CLRSCREEN
RTS
```

ein Beispiel mit einer Bildschirmausgabe:

```
START:     JSR @CLRSCREEN       * Bildschirm loeschen
           LEA TEXT,A0          * Adresse des Textes
SCHLEIFE:  MOVE.B (A0)+,D0      * Zeichen holen
           BEQ.S ENDE           * =0 dann Ende erreicht
           MOVEM.L A0,-(A7)     * Retten des Pointers
           JSR @CO              * Sonst Zeichen ausgeben
           MOVEM.L (A7)+,A0
           BRA.S SCHLEIFE
ENDE:      RTS                  * Rueckkehr
TEXT:      DC.B 'Ausgabe auf dem',$d,$a
           DC.B 'Bildschirm mit CO',$d,$a
           DC.B 'und CLRSCREEN',0
```

Der Cursor blinkt danach direkt hinter der letzten Textstelle.

Damit wird ein Zeichen im Textmode ausgegeben. Das Zeichen steht zuvor in Register D0.B. Dabei können auch Steuerzeichen ausgegeben werden. CO verändert anschließend die Registerinhalte, weshalb Vorsicht geboten ist. Eine zweite Ausgaberoutine (CO2) tut das nicht und sollte daher bevorzugt verwendet werden. CO ist aber nur reine Bildschirmausgabe, während CO2 auch noch andere Aufgaben besitzt. Daher kann es nötig werden auch einmal CO zu verwenden.
Bevor man CO verwenden kann muß CLRSCREEN aufgerufen worden sein, sonst erscheint kein Cursor auf dem Bildschirm.
Die Zeichenbreite wird bei der Ausgabe so gewählt, wie man sie zuvor im OPTIONEN-Menue eingestellt hat (40 oder 80 Zeichen pro Zeile).

Beispiel:

```
START:   JSR @CLRSCREEN      * Bildschirm loeschen
         MOVE.B #'A',D0      * Zeichen "A" auf dem
         JSR @CO             * Bildschirm ausgeben
         RTS
```

Beispiel eines Terminals:

```
START:   JSR @CLRSCREEN      * Bildschirm loeschen
SCHLEIFE: JSR @CI            * Zeichen holen
         CMP.B #1,D0         * CTRL-A bricht ab
         BEQ.S ENDE          * wenn gedrueckt
         JSR @CO             * sonst ausgeben
         BRA.S SCHLEIFE      * und wiederholen
ENDE:    RTS                 * Abbruch
```

Mit dem Programm kann man auf dem Bildschirm schreiben, ähnlich wie beim Editor. Aber es werden andere Steuertasten verwendet. Bei der CHERRY-Tastatur funktionieren jetzt die Cursortasten.

Nun zu den Befehlen:

Im Textbereich von $20 bis $7F werden die normalen ASCII-Zeichen ausgegeben.
Zeichen die einen Code größer als $80 besitzten sind Sonderzeichen, wie "Ä" etc. Ab Version 4.0 des Grundprogramms werden auch diese direkt ausgegeben um einen gemischen Betrieb mit den Klammern, wie "[" etc. zu ermöglichen.
Wenn ein Zeichen ganz rechts eingegeben wird, so erfolgt ein Sprung zur nächsten Zeile und das Zeichen erscheint links auf der nächsten Zeile.
Wenn ein Zeichen auf der letzten Zeile ganz rechts ausgegeben wird, so erfolgt ein Scrollvorgang des Bildschirms. Der Bildschirm wird dabei um eine Zeile nach oben verschoben.

Programmbeispiel zur Ausgabe aller Zeichen $20 bis $7F:

```
START:     JSR @CLRSCREEN      * erst Bildschirm loeschen
           MOVE.B #$20,D0      * erstes Zeichen
SCHLEIFE:  MOVEM.W D0,-(A7)    * Zeichen merken
           JSR @CO             * dann ausgeben
           MOVEM.W (A7)+,D0    * wieder holen
           ADD.B #1,D0         * naechstes Zeichen
           CMP.B #$80,D0       * Ende erreicht ?
           BNE.S SCHLEIFE      * Nein, dann weiter
           RTS                 * OK.
```

Achtung! Man sieht nur dann alle Zeichen, wenn man mit dem OPTIONEN-Menue auf 80 Zeilen pro Zeile umgeschaltet hat.

CTRL-Befehle

Hier werden Codes im Bereich 0 bis $1F als Befehl interpretiert. Dabei gilt der Code $1B als besonderer Code, da dort weitere Befehle, die ESC-Befehle angesprochen werden.
Neben dem Code steht der Control-Code. Wenn man die Taste CTRL und gleichzeitig die angegebene Taste nach dem Bindestrich drückt, erhält man den gewünschten Code.

Code 8: CTRL-H

Der Cursor wandert ein Zeichen zurück. Man nennt dieses Zeichen auch BACKSPACE (BS), was soviel wie Rückwärtsschritt bedeutet.
Befindet sich der Cursor schon auf der linken Seite des Bildschirmes, so wandert er eine Zeile nach oben und kommt ganz rechts zum Vorschein. Wenn der Cursor in der obersten Zeile ganz links war, so kann man ihn nicht mehr weiter bewegen.

Code 9: CTRL-I

Der Cursor wird ein Zeichen nach rechts verschoben. War der Cursor ganz rechts auf der Zeile, so erfolgt ein Sprung in die nächste Zeile. War der Cursor dabei auch auf der letzten Zeile, so wird der Bildschirminhalt um eine Zeile nach oben verschoben.

Code $A: CTRL-J

Der Cursor wandert eine Zeile nach unten. Wenn der Cursor auf der Zeile 23 (unterste Zeile) war, so wandert er nicht weiter nach unten, sondern der gesamte Bildschirminhalt wird um eine Zeile nach oben verschoben. Diese Funktion nennt man auch Scroll. Der Befehl wird auch LINEFEED genannt.

Code $B: CTRL-K

Der Cursor wird eine Zeile nach oben transportiert. Er verändert dabei seine X-Koordinate nicht. Wenn der Cursor schon in der obersten Zeile war, so verändert er seine Position nicht weiter.

Code $C: CTRL-L

Der Cursor wird ein Zeichen nach rechts verschoben. War der Cursor ganz rechts auf der Zeile, so erfolgt ein Sprung in die nächste Zeile. War der Cursor dabei auch auf der letzten Zeile, so wird der Bildschirminhalt um eine Zeile nach oben verschoben.
Der Code $C hat die gleiche Wirkung wie der Code 9 und wurde aus Kompatibilitätsgründen mit eingeführt.

Code $D: CTRL-M

Der Cursor wird auf die linke Seite der Zeile gestellt, in der er sich vor dem Aufruf schon befunden hat. Der Befehl wird auch CARRIAGE RETURN oder kurz CR genannt.

Code $16: CTRL-V

Der Cursor wird eine Zeile nach unten transportiert. Wenn er auf der untersten Zeile stand, so wird er aber nicht weiter nach unten gesetzt. Es erfolgt auch kein Scroll-Vorgang.
(Erst ab Version 4.0)

Code $1A: CTRL-Z

Die Wirkung ist die Gleiche, wie der Aufruf des CLRSCREEN-Kommandos. Der Bildschirm wird gelöscht und der Cursor wandert in die linke, obere Ecke.

Code $1E: CTRL-↑

Der Cursor wandert in die linke obere Ecke, der Bildschirm bleibt aber erhalten. Der Befehl wiurd auch HOME genannt.

ESC-Befehle

Bei den ESC-Befehlen werden mehrere Codes übertragen. Auch sie dienen der Cursor- und Bildschirmsteuerung.
Zum Test wird zunächst, wie angegeben die ESC-Taste gedrückt und dann die Zeichensequenz.

Code $1B $23 $yy $xx: ESC =..

Damit kann der Cursor absolut positoniert werden. Die neue X,Y-Position wird in Zeicheneinheiten übergeben. In der X-Richtung gibt es die Werte 0 bis 79 und in der Y-Richtung die Werte 0 bis 23. Für die Ausgabe wird jetzt aber noch zuvor jeweils der Wert 32 addiert. Damit ergeben sich lesbare ASCII-Zeichen für die Koordinaten.

Beispiel: ESC =!A

positioniert den Cursor in Spalte 33 (X-Position) und in Reihe 1 (Y-Position). X=0, Y=0 ist dabei die linke obere Ecke des Bildschirms.

Code $1B $24: ESC $

Der Zeichensatz wird auf Deutsch umgestellt. Wird nun das Zeichen "[" ausgegeben, so erscheint auf dem Bildschirm das Zeichen "Ä".

Code $1B $25: ESC %

Der Zeichensatz wird wieder auf amerikanisch umgestellt. Dabei sind die eckigen und geschweiften Klammern verfügbar.
Deutsche Zeichen kann man ab Version 4.0 der Grundprogramms aber dennoch ausgeben, wenn man einfach das Bit 7 des betreffenden ASCII-Codes setzt.

Code $1B $45: ESC E

Eine leere Zeile wird eingefügt. Dabei wandert alles von der aktuellen Cursorstelle um eine Zeile nach unten. Die letzte Zeile geht verloren.

Code $1B $51: ESC Q

Ein Leerzeichen wird an der aktuellen Cursorstelle eingefügt. Dabei wandern alle auf der Zeile befindlichen Zeichen um eins nach rechts. Wenn ganz rechts ein Zeichen stand, so geht es verloren.
Achtung beim Ausprobieren. Nach der ESC–Taste das große Zeichen Q eingeben (SHIFT und Q).

Code $1B $52: ESC R

Die Zeile in der sich der Cursor befindet wird gelöscht und alle Zeilen darunter werden um eine Zeile nach oben geschoben.

Code $1B $54: ESC T

Alle Zeichen von der Cursorstelle bis zum Ende der Zeile werden gelöscht.

Code $1B $57: ESC W

Das Zeichen an der Cursorstelle wird gelöscht und die restliche Zeile wandert um eine Zeichen–Stelle nach links.

Code $1B $59: ESC Y

Alle Zeichen, beginnend bei der Cursorstelle, werden bis zum Ende der Bildseite gelöscht.

LO 22

Das Zeichen, das in Register D0.B steht, wird über die Druckerschnittstelle ausgegeben. Dabei erfolgt die Ausgabe über die CENT-Baugruppe.

Beispiel:

```
MOVE.B #'A',D0      * Zeichen A
JSR @LO             * ausgeben
RTS
```

Beispiel für Textausgabe auf den Drucker:

```
START:     JSR @CIINIT2        * Aus RAM lesen
SCHLEIFE:  JSR @CI2            * Zeichen lesen
           BEQ.S ENDE          * =0, dann Ende
           JSR @LO             * auf den Drucker ausgeben
           BRA.S SCHLEIFE      *
ENDE:      RTS
```

Mit dem obigen Programm ist es möglich einen Text, der z.B. mit dem Editor eingegeben wurde, auf dem Drucker auszugeben. wenn man das Programm startet, so druckt es sich zunächst selbst aus.
Will man einen anderen Text ausgeben, so muß man entweder einen neuen Text mit dem Editor eingeben, oder mit der Funktion TEXTSTART (OPTIONEN-Menue) einen neuen Textanfang auswählen.

SIN 23

Damit steht die SIN-Funktion zur Verfügung. Der Sinus wird dabei mit Hilfe einer Tabelle ermittelt, was sehr schnelll geht. Damit man mit einem Integer-Bereich auskommt, ist der Sinuswert mit 256 multipliziert und die Nachkommastellen sind gerundet. Der Bereich ist somit:

+/- 256

Der Parameter für die Berechnung wird im Register D0.W als vorzeichenbehaftete Größe in Grad eingegeben.
Anschließend erhält man das Ergebnis im Register D0.W, ebenfalls als Vorzeichenbehaftete Größe im Bereich von -256 bis +256.
Intern wird die Routine SIN dazu verwendet um die Berechnungen für die Schildkrötengraphik durchzuführen.

Beispiel

```
MOVE #10,D0     * 10 Grad
JSR @SIN        * Sinus berechnen
RTS             * Ergebnis ist 256*sin(10)
```

Da der Sinus sehr schnell berechnet wird, ist die Anwendung vorwiegend für Echtzeitaufgaben, wie Rotation von Körpern oder Fouriersynthese zu sehen.

Beispiel:

```
START:     CLR D1          * X-Koordinate und Winkel
SCHLEIFE:  MOVE #128,D2    * Y=128
           JSR @MOVETO     * Startposition
           MOVE D1,D0      * Dann Y-Wert ermitteln
           JSR @SIN        * dazu Sinus berechnen
           EXT.L D0        * Fuer Div vorbereiten
           DIVS #4,D0      * Nun Bereich -64 bis +64
           ADD #128,D0     * Basis verschieben
           MOVE D0,D2      * Neuer Wert
           JSR @DRAWTO     * Linie ausgeben
           ADD #1,D1       * Von 0 bis 511 Grad (und x)
           CMP #512,D1     * Wenn 512 erreicht, dann
           BNE SCHLEIFE    * beenden, sonst zurueck
           RTS
```

Die Sinuskurve wird dabei als Flächengraphik ausgegeben.

Beispiel einer Fouriersynthese:

```
* y=k*(sin(x)+sin(3*x)/3+sin(5*x)/5+.....)
* Synthese Rechteckschwingung.
START:    CLR D1          * X-Koordinate
          CLR D3          * Startwinkel
SCHLEIFE: MOVE #128,D2    * Y=128
          JSR @MOVETO     * Startposition
          CLR D2          * Vorbereiten fuer Summe
          MOVE #1,D4      * Term 1
          BSR SINTEIL     * berechnen
          MOVE #3,D4      * Term 2
          BSR SINTEIL     * berechnen
          MOVE #5,D4      * Term 3
          BSR SINTEIL     * berechnen
          MOVE #7,D4      * Term 4
          BSR SINTEIL     * berechnen, dann Stop der Reihe
          EXT.L D2        * Normieren
          DIVS #4,D2      * Bereich anpassen
          ADD #128,D2     * Basislinie
          JSR @DRAWTO     * Linie ausgeben
          ADD #2,D3       * In 2 Grad-Schritten
          ADD #1,D1       * Von 0 bis 511 Grad (und x)
          CMP #512,D1     * Wenn 512 erreicht, dann
          BNE SCHLEIFE    * beenden, sonst zurueck
          RTS
SINTEIL:  MOVE D3,D0      * z.B. bei 3
          MULU D4,D0      * sin(3*x)
          JSR @SIN
          EXT.L D0        * Wegen DIV
          DIVS D4,D0      * sin(3*x)/3 etc.
          ADD D0,D2
          RTS
```

Die Fouriersynthese erfüllt wichtige Aufgaben in der Technik. Hier soll aber nicht weiter darauf eingegangen werden, sondern dies sei nur ein Beispiel für die Anwendung der schnellen SIN-Funktion.

COS 24

Für die COS-Routine gilt das gleiche wie schon bei SIN. Im Register D0.W wird der Wert des Parameters in Grad übergeben und das Ergebnis steht dann nach dem Aufruf wieder im Register D0.W. Der Wert ist dabei cos(x)*256 und die Nachkommastellen entfallen. Der Wert ist Vorzeichenbehaftet und im Zweierkomplement dargestellt.

Beispiel:

```
        MOVE #-10,D0    * Winkel= -10 Grad
        JSR @COS        * Berechnen von cos(-10)*256
        RTS
```

kombiniertes Beispiel:

```
START:      CLR D3          * phi=Winkelstart=0
SCHLEIFE:   MOVE #256,D1    * Bildmitte einstellen
            MOVE #128,D2    * man beachte die
            JSR @MOVETO     * unterschiedliche Aufloesung
            MOVE D3,D0      * X=COS(phi)+256
            JSR @COS
            ADD #256,D0     *
            MOVE D0,D1      * X-Koordinate
            MOVE D3,D0      * Y=SIN(phi)/2+128
            JSR @SIN
            EXT.L D0        * wegen DIV
            DIVS #2,D0      * auf +/- 128 bringen
            ADD #128,D0
            MOVE D0,D2      * ergibt Y-Koordinate
            JSR @DRAWTO     * Verbindung zeichnen
            ADD #1,D3       * naechster Winkel
            CMP #360,D3     * von 0 bis 359 Grad
            BNE SCHLEIFE    *
            RTS
```

Wenn man das obige Programm startet, so ergibt sich eine Kreisfläche. Wenn man den Sinus genauer als 1 Grad benötigt, so muß man die Zwischenpunkte aproximieren.

SIZE 25

Zeichengröße setzen. Im Register D0.B wird die Zeichengröße für die nachfolgenden CO- oder CO2-Ausgaben bestimmt. Dabei kann man unterschiedliche Größen aber nicht so ohne weiteres mischen.
Nachdem man eine neue Größe gesetzt hat, sollte man den Bildschirm löschen. Die Codierung der Größe ist dabei genauso, wie beim WRITE-Befehl. Zwei Codes sind dabei besonders interessant:

Mit $11 wird auf 80 Zeichen pro Zeile und 24 Zeilen umgeschaltet. Mit $21 auf 40 Zeichen pro Zeilen ebenfalls mit 24 Zeilen. Alle Zeichen, die nach der 40 Spalte ausgegeben werden, bleiben unsichtbar.
Diese beiden Codes werden auch im OPTIONEN-Menue eingestellt, wenn man die Zeichengröße festsetzt.
Andere Größen sind prinzipiell möglich, jedoch ergeben sich dabei Bildausschnitte, ggf. auch in der Zeilenrichtung. Eigene Versuch schaffen hier aber schnell Aufschluß.

Beispiel:

```
START:   MOVE #$11,D0    * Umschalten auf 80 Zeichen/Zeile.
         JSR @SIZE       * Erst neue Groesse einstellen.
         JSR @CLRSCREEN * Dann Bildschirm loeschen.
         RTS
```

Beispiel2:

```
START:   MOVE #$21,D0    * Umschalten auf 40 Zeichen/Zeile.
         JSR @SIZE       * Erst neue Groesse einstellen.
         JSR @CLRSCREEN * Dann Bildschirm loeschen.
         RTS
```

Beispiel mit Übergröße:

```
                                * Doppelter Zeilenabstand und
START:    MOVE #$32,D0          * umschalten auf 30 Zeichen/Zeile.
          JSR @SIZE             * Erst neue Groesse einstellen.
          JSR @CLRSCREEN        * Dann Bildschirm loeschen.
          RTS
```

Wenn man nach Aufruf dieses Programms wieder in den Editor zurückkommt, so erscheint ein ganz großer Cursor auf dem Bildschirm und auch der Text ist gedehnt. Man sieht aber nur einen Bildausschnitt. Wenn der Cursor ganz oben steht, so wird er abgeschnitten, denn der Start liegt ein halbes Zeichen oberhalb (nur bei einem Grundprogramm der Version kleiner 4.2). Auch erscheinen auf dem Bildschirm nur ca. 12 Zeilen. Die restlichen Zeilen sind weiter unten unsichbar. Will man sie sehen, so kann man mit CTRL–Z den Bildschirm nach oben schieben. Es ist jetzt möglich den Cursor in ein unsichtbares Gebiet zu schieben. Will man ihn wieder sehen, so gibt man z.B. CTRL–Q, SHIFT–R ein und der Cursor ist links oben.
Diese Darstellungsart ist z.B. gut, wenn man eine Gruppe von Personen Einzelheiten bei der Eingabe zeigen will.
Man kann die Textgröße einfach zurückstellen, in dem man sie im OPTIONEN–Menue wieder auf 80 oder 40 Zeichen pro Zeile zurückstellt.

CMD 26

Damit kann man einen Befehl direkt an den Graphik-Prozessor ausgeben. Der befehl steht dazu zuvor im Register D0.B. Mit der Ausführung wird dann solange gewartet, bis der GDP bereit ist und dann wird der Befehl an den GDP übergeben.
Eine genaue Liste alle Befehle findet man bei der GDP-Beschreibung des EF9366.
Hier aber ein paar besonders wichtige Codes:

0	Schreiben.
1	Loeschen.
2	Stift senken.
3	Stift heben.
4	Bildschirm loeschen.
5	X- und Y-Register auf 0 setzen.
6	wie 4 und 5 zusammen.
7	Alle Register auf 0 (CSIZE auf $11) und wie 6.
8	Lichtblitz fuer Lichtgriffel und abtasten.
9	Lichtgriffel abtasten ohne Lichtblitz.
$A	5 x 8 Block zeichnen.
$B	4 x 4 Block zeichnen.
$C	Bildschirm setzen (hell oder dunkel).
$D	X-Register auf 0 setzen.
$E	Y-Register auf 0 setzen.
$F	DMA-Zugriff, n i c h t v e r w e n d e n.

Die Codes $10 bis $1F sind für Vektoren gedacht, und die Codes $80 bis $FF für Kurzvektoren. Die Vektoren sind aber über DRAWTO einfacher zu bedienen und die Kurzvektoren über den FIGUR-Befehl.

Die Code $20 bis $7F sind die Darstellung der lesbaren ASCII-Zeichen. Damit kann man Texte sehr schnell auf dem Bildschirm ausgeben.

Beispiel:

```
MOVE.B #$C,D0        * Bildschirm Weiss stellen
JSR @CMD             * ausfuehren
RTS
```

Der Bildschirm leuchtet nach Ausführung des Programms hell auf. Damit kann man inverse Darstellung vorbereiten, wenn man danach Zeichen oder Graphik im Löschmode (CMD mit D0=1 oder ERAPEN) ausgibt.

komplexeres Beispiel:

```
START:    JSR @WAIT              * Init Textgroesse
          MOVE.B #$22,$FFFFFF73 * In GDP-Register 3
          CLR D1                 * X=0
          CLR D2                 * Y=0
SCHLEIFE: JSR @MOVETO            * Position einstellen
          MOVE #0,D0             * Schreiben
          BSR SCHREIBE           * Text ausgeben
SY1:      JSR @SYNC              * 20 ms warten
          BEQ.S SY1              * sonst zu schnell
          JSR @MOVETO            * wieder zurueck
          MOVE #1,D0             * Loeschen dann
          BSR SCHREIBE
          ADD #1,D1              * Diagonal
          ADD #1,D2              * bewegen
          CMP #256,D2            * Wenn groesser 256
          BCS SCHLEIFE           * dann stop.
          RTS
SCHREIBE: JSR @CMD               * Befehl ausfuehren
          LEA TEXT,A0            * Text ausgeben
SCHL2:    MOVE.B (A0)+,D0        * Zeichen holen
          BEQ.S ENDE             * =0, dann Ende
          JSR @CMD               * und ausgeben
          BRA.S SCHL2            * bis fertig
ENDE:     RTS
TEXT:     DC.B 'Bewegter Text',0
```

Durch den Befehl SYNC wird erreicht, das der Text ohne Flimmern über den Bildschirm wandert.
Man kann zur Probe auch mal die beiden Zeilen "JSR @SYNC" und "BEQ.S SY1" weglassen.
Will man die Schrift schnell bewegen, so addiert man einfach einen größeren Wert auf D1 und D2.

NEWPAGE 27

Die GDP-Baugruppe besitzt vier Bildseiten mit 512 x 256 Punkten. Wenn man die Bildseite auswählen will, so kann man den Befehl NEWPAGE verwenden. Im Register D0.B steht die Seitennummer der Schreibseite und im Register D1.B die Seitennummer der Leseseite. Alle Graphik-Befehle werden auf der Schreibseite ausgeführt und das Bild auf dem Bildschirm stammt von der Leseseite. Damit ist es also möglich auf einer unsichtbaren Seite Bilder zu erstellen und dabei eine andere Seite zu betrachten. Dies wird benötigt um flimmerfreie Graphiken zu erzeugen.

Als Wert für die Schreib- und Leseseite kann man eine Zahl von 0 bis 3 wählen.
Wenn man die Schildkrötengraphik verwendet, so wird die Schildkröte selbst immer auf der Seite 1 gezeichnet und die Graphik normalerweise auf der Seite 0.

Beispiel:

```
        CLR D0          * Schreibseite=0
        CLR D1          * Leseseite=0
        JSR @NEWPAGE    * Seite anwaehlen.
        RTS
```

Beispiel:

```
START:      MOVE #1,D0      * Dummy SChreite
            JSR @SCHR16TEL  * Turtle an
            JSR @HIDE       * dann Flip ausschalten
            MOVE #1,D4      * Schreibseite = 1
            MOVE #0,D5      * Leseseite = 0
            MOVE #1,D3      * Info fuer Bild
SCHLEIFE:   MOVE D4,D0      * Schreibseite
            MOVE D5,D1      * Leseseite
            JSR @NEWPAGE    * Seite anwaehlen
            JSR @CLPG       * Seite unsichtbar loeschen
            BSR BILD        * Bild aufbauen
            EOR #1,D4       * Schreibseite austauschen
            EOR #1,D5       * Leseseite austauschen
            CMP #400,D3     * Max Wert
            BCS SCHLEIFE    * wieder zurueck
            RTS             * Ende
BILD:       MOVE #36-1,D6   * Kreis zeichnen,
BISCHL:     MOVE D3,D0      * der groesser wird
            JSR @SCHR16TEL  *
            MOVE #10,D0     * 10 Grad
            JSR @DREHE
            DBRA D6,BISCHL  * Bis beendet
            ADDQ #5,D3      * Neuer Wert
            RTS
```

Mit den ersten beiden Befehlen "SCHR16TEL" und "HIDE" wird erreicht, daß die Schildkröte sich zuerst ein Stück bewegt und dann wird sie mit "HIDE" abgestellt.Dann erst kann man die zweite Bildseite auch mitverwenden.
Eine andere Möglichkeit ist es auch, das alte Bild zu löschen indem die Graphik mit ERAPEN neu gezeichnet wird.
In dem nachfolgenden Beispiel wird ausserdem noch eine Drehung des Kreises durchgeführt.
weiteres Beispiel:

```
START:    MOVE #1,D0      * Dummy SChreite
          JSR @SCHR16TEL  * Turtle an
          JSR @HIDE       * dann Flip ausschalten
          MOVE #1,D4      * Schreibseite = 1
          MOVE #0,D5      * Leseseite = 0
          MOVE #1,D3      * Info fuer Bild
          MOVE D4,D0      * um Schildkroete zu loeschen
          MOVE D5,D1      * wird die Seite 1 geloescht
          JSR @NEWPAGE    * erst einstellen
          JSR @CLPG       * dann loeschen
SCHLEIFE: MOVE D4,D0      * Schreibseite
          MOVE D5,D1      * Leseseite
          JSR @NEWPAGE    * Seite anwaehlen
          MOVE #1,D0      * altes Bild loeschen
          JSR @CMD
          BSR BILD        * D3 = alter Stand
          MOVE #12,D0     * dann drehen
          JSR @DREHE      * fuer naechstes Bild
          ADDQ #5,D3      * neues Bild setzen (5+5)
          CLR D0          * Schreib-Mode einstellen
          JSR @CMD
          BSR BILD        * Bild aufbauen
          MOVE #-6,D0     * und Korrektur fuer
          JSR @DREHE      * naechstes Bild
          SUB #10,D3      * Korrektur
          EOR #1,D4       * Schreibseite austauschen
          EOR #1,D5       * Leseseite austauschen
          CMP #400,D3     * Max Wert
          BCS SCHLEIFE    * wieder zurueck
          JSR @GRAPOFF    * verhindert auftauchen der
          RTS             * Schildkroete, dann Ende
```

```
BILD:     MOVE #36-1,D6   * Kreis zeichnen,
BISCHL:   MOVE D3,D0      * der groesser wird
          JSR @SCHR16TEL *
          MOVE #10,D0     * 10 Grad
          JSR @DREHE
          DBRA D6,BISCHL * Bis beendet
          ADDQ #5,D3      * Neuer Wert
          RTS
```

Mit dem NEWPAGE–Befehl kann man auf diese Weise einfach bewegte Graphik erstellen. Ein weiterer Befehl, der SYNC–Befehl kann dann dazu verwendet werden den Bildwechsel mit der Graphikerstellung zu synchronisieren und damit keine störenden Bildschnitte mehr zu erhalten.

SYNC 28

Damit ist es möglich den Vertikal-Synchronisationsimpuls des GDP abzufragen. Immer wenn ein Durchgang erfolgte, wird $FFFF im Register D0 übergeben, sonst der Wert 0. Damit kann man Abläufe mit dem Bildwechsel synchronisieren. Der SYNC-Befehl liefert aber nur einmal den Wert $FFFF, auch wenn man sehr schnell hintereinander abfragt und sich immer noch in der vertikalen Austastlücke befindet. Dies ist nötig um auch Zeitsynchronisationen zu ermöglichen. Wenn man SYNC dauernd abfragt, so liefert es alle 20ms den Wert $FFFF. Damit kann man quarzgenaue Messungen durchführen, denn auf der GDP-Baugruppe befindet sich ein Quarz, der auch für diesen Takt veranwortlich ist.

Beispiel:

```
START:    JSR @SYNC       * V-Sync pruefen
          BEQ.S START     * =0 dann nicht da
          RTS             * V-Sync ist da.
```

Beispiel einer Zeitmessung:

```
UHR:      CLR D4          * Zeitzaehler
SCHLEIFE: MOVE #50-1,D3   * 50*20ms
WARTE:    JSR @SYNC       * Messen
          BEQ WARTE       * bis 20 ms
          DBRA D3,WARTE   * bis 1 sec
          ADD #1,D4
          LEA BUFFER,A0   * Ausgabe durchfuehren
          MOVE D4,D0      * muss klein 20ms sein
          EXT.L D0        * erst Minuten
          DIVS #60,D0     * ausgeben
          MOVE.L D0,-(A7)   * merken
          JSR @PRINT4D      * Dezimal ausgeben
          MOVE.B #':',(A0)+ * trennen
          MOVE.L (A7)+,D0   * dann Sekunden
          SWAP D0           * ist Modulo-Rest
          JSR @PRINT4D      * bei DIVS
          MOVE.B #' ',(A0)+ * zum Loeschen der
          CLR.B (A0)        * 1er Stelle
          LEA BUFFER,A0   * und auf Bildschirm
          MOVE #$22,D0    * Schriftgroesse
          MOVE #2,D1      * X=2
          MOVE #128,D2    * Y=128
          JSR @WRITE      * und ausgeben
          BRA SCHLEIFE
BUFFER:   DS.B 100        * Textbuffer
```

Neben der Anwendung für die Zeitmessung ist der SYNC-Befehl wichtig, um flickerfreie, bewegte Graphiken zu erhalten.

weiteres Beispiel:

```
START:    CLR D1          * X=0
SCHLEIFE: JSR @SETPEN     * erst mal schreiben
          CLR D2          * X=n,Y=0
          JSR @MOVETO     * von da an
          MOVE #255,D2    * bis Y=255
          JSR @DRAWTO     * Linie
WARTE:    JSR @SYNC       * dann auf Sync warten
          BEQ.S WARTE
          JSR @ERAPEN     * und Linie wieder loeschen
          CLR D2          * Endpunkt X=n, Y=0
          JSR @DRAWTO     *
          ADD #1,D1       * Neuer Wert fuer X
          CMP #512,D1     * bis aus dem Bildfeld
          BNE.S SCHLEIFE  * verschwunden
          RTS             * Dann fertig.
```

Wenn man das Programm startet, so läuft eine Linie vom linken Bildteil in den rechten Bildteil.
Wenn man die Befehle "JSR @SYNC" und "BEQ.S WARTE" einmal wegläßt, so sieht man den gleichen Vorgang, jedoch erscheint die Linie unterbrochen und die wandert auch viel über den Bildschirm. Die Unterbrechungen kommen daher, weil man nun Lösch- und Schreibvorgang überlagert sieht.

WERT 29

Jetzt kommt ein sehr komplexer Befehl. Damit ist es möglich Ausdrücke zu berechnen. Dabei können Symbole, wie auch mathematische Zeichen verwendet werden. Dieser Befehl wird im Grundprogramm dazu verwendet, um z.B. im Assembler arithmetische Ausdrücke zu berechnen. Ebenfalls dient der Befehl der Eingabe von Startadressen etc., wo immer es gilt eine Zahl au einer Formel oder einer einfachen Eingabe zu ermitteln. Dabei können auch Symbole verwendet werden, wie z.B. im Assembler definiert wurden.

Für die Parameter gilt folgendes:

```
A0        Adresse des Buffers in dem der Text steht.
```

nach dem Aufruf:

```
D0.L      Ergebnis der Berechnung.
D1.W      Fehlercode.
```

für den Fehlercode in D0.W gilt:

```
0         Syntax-Fehler aufgetreten
1         Bytewert geladen
2         Wortwert geladen
3         Langwortwert geladen
4         -reserviert-
5         Symbolaufruf ohne Definition des Symbols
$FFFF     Interner Fehler
```

Als Opertionen sind zulässig:

```
+         Addition 32 Bit
-         Subtraktion 32 Bit
*         Multiplikation (ab Version 4.0 mit 32 Bit)
/         Division (ab Version 4.0 mit 32 Bit)
\         Modulo-Division ab Version 4.0
( )       Klammerrechnung
!         Oder-Verknuepfung 32 Bit (ab Version 4.0)
&         Und-Verknuepfung 32 Bit  (ab Version 4.0)
```

*	Als Operand ist der Programmstand beim Assembler
?	Ist die aktuelle Textanfangsadresse
'...'	Text, wird als ASCII-Wert(e) geladen
~	Nicht-Operation 32 Bit
-	Vorzeichen einer Zahl
$nnnnnnnn	sedezimale Zahl
dddddddd	dezimale Zahl
%bbbbbbbb	binaere Zahl
@symbol	Adresse eines festen Symbols
!symbol	Index eines festen Symbols, fuer TRAP-Befehl

Die nachfolgenden Zusätze stehen hinter einem Ausdruck:

.B	Bytegroesse markieren
.W	Wortgroesse markieren
.L	Langwortgroesse markieren

Beispiel:

```
START:  LEA TEXT,A0     * Adresse des Textes laden
        JSR @WERT
        RTS             * Ergebnis in D0.L
TEXT:   DC.B '23*3/4',0
```

Das Beispiel liefert den Wert sedezimal $11 oder dezimal 17.
Man kann das überprüfen, wenn man das Programm im Einzelschritt durchläuft.

weiteres Beispiel:

```
START:  LEA BUFFER,A0   * Text eingeben
        MOVE #$22,D0    * Groesse
        MOVE #2,D1      * X=2
        MOVE #128,D2    * Y=128
        MOVE #30,D3     * Anzahl der Zeichen
        JSR @READ       * und lesen.
        LEA BUFFER,A0   * Dann auswerten
        JSR @WERT       * und danach den Wert ausgeben
        LEA BUFFER,A0   * wieder Adresse laden
        JSR @PRINT8X    * z.B. sedezimal
        LEA BUFFER,A0   * fuer Ausgabe
        MOVE #$22,D0    * Textgroesse
        MOVE #2,D1      * X=2
        MOVE #200,D2    * Y=200
        JSR @WRITE      * auf den Bildschirm
        BRA START       * immer weiter
BUFFER: DS.B 100        * Textbuffer
```

Man versuche dann einmal folgende Eingabe:

```
?
*
3*4
10/2
10\3
!SCHREITE
@SCHREITE
?+1
3*(4+5)
5!12
'A'
'AB'
%1010
```

Man beachte das die Ausgabe hier sedezimal ist. Wenn man eine dezimale Ausgabe will, so kann man PRINT4D oder PRINT8D (Version 4.0) verwenden. Dabei muß man aber den Buffer mit Leerzeichen auffüllen, sonst bleibt ggf. der Rest der vorherigen Zeile stehen, falls die neue Ausgabe weniger Stellen besitzt als die alte Ausgabe.

Achtung! Zwischen den Operationen dürfen keine Leerzeichen stehen. Dies war nötig, da man z.B. im Assembler sonst nicht unterscheiden könnte ob das Zeichen "*" als Kommentar oder als Operation gemeint ist.
Beispiel:

```
MOVE #3,$54  * 3 ist der Ladewert
```

Wenn Leerzeichen erlaubt wären ergäbe sich

```
$54*3
```

als Adresse für den Transport.

ZUWEIS 30

Damit ist es möglich einem Symbol einen Wert zuzuordnen.
Dies geschieht in Form einer Zuweisung:

SYMBOL := wert

Auf der rechten Seite steht ein beliebiger Ausdruck, wie er bei dem Befehl WERT möglich war. Auf der linken Seite ein Name, den man selbst gewählt hat. Diesem Namen wird dann der Wert zugeordnet.
Man muß dabei immer eine Zuweisung ausführen. Wenn keine Zuweisung im Text steht, so wird ein CARRY zurückgemeldet.
Bei einer Fehleingabe wird die Zuweisung damit nicht ausgeführt.

Beispiel:

```
START:  LEA TEXT,A0     * Textadresse laden
        JSR @ZUWEIS     * ausfuehren
        RTS
TEXT:   DC.B 'neu:=75',0
```

Nach Ausführung der Anweisung findet man in der Symboltabelle den Namen "NEU". Er besitzt den Wert 75 oder sedezimal $4B.

ACHTUNG! bei älteren Versionen des Grundprogramms (kleiner 4.0) steht der Name ZUWES in der Suchtabelle, man mußte daher ZUWES anstelle von ZUWEIS eingeben, um mit dem Befehl arbeiten zu können.

kombiniertes Beispiel:

```
START:  LEA BUFFER,A0   * Text eingeben
        MOVE #$22,D0    * Groesse
        MOVE #2,D1      * X=2
        MOVE #128,D2    * Y=128
        MOVE #30,D3     * Anzahl der Zeichen
        JSR @READ       * und lesen.
        LEA BUFFER,A0   * Dann auswerten
        JSR @ZUWEIS     * und danach den Wert ausgeben
        BCC OK          * Ok, war Zuweisung
        JSR @WERT       * sonst mal mit WERT probieren
OK:     LEA BUFFER,A0   * wieder Adresse laden
        JSR @PRINT8X    * z.B. sedezimal
        LEA BUFFER,A0   * fuer Ausgabe
        MOVE #$22,D0    * Textgroesse
        MOVE #2,D1      * X=2
        MOVE #200,D2    * Y=200
        JSR @WRITE      * auf den Bildschirm
        BRA START       * immer weiter
BUFFER: DS.B 100        * Textbuffer
```

Man probiere einmal folgende Eingaben:

```
NEU:=5
5+NEU
TEXT:=?
TEXT+1
NEU:=NEU+1
NEU
```

Achtung! Zwischen den Operationen dürfen keine Leerzeichen stehen.
Auf der Anzeige erscheint erscheint jeweils der Wert des berechneten Ausdrucks.

CIINIT2 31

Der aktuelle Textzeiger wird auf den Textanfang gesetzt. Dieser Befehl wird zusammen mit dem Befehl CI2 verwendet, um Texte aus dem Speicher zu erreichen. Es gibt dabei vier wichtige Textzeiger, die auch im Editor verwendet werden.

```
STXTXT      * Ist die Adresse des Textanfangs
AKTTXT      * Ist die Adresse der aktuellen Textstelle
ETXTXT      * Ist die Adresse auf das Textende (Zeichen 0)
EOTTXT      * Ist die Adresse auf das Ende eines Textgebietes
```

EOTTXT wird dabei zur Zeit nicht weiter ausgewertet.

Die Speicheraufteilung eines einfachen Textes ist dabei so:

```
            ---------------------
STXTXT---->!HALLO,DIES IST EIN !
           !TEXT IM SPEICHER ER!
AKTTXT---->!WURDE MIT DEM      !
           !EDITOR GESPEICHERT.!
ETXTXT---->!0                  !
           ---------------------
```

Die 0 am Textende besitzt dabei die Codierung 00 und ist nicht mit dem ASCII-Zeichen "0" mit der Codierung $30 zu verwechseln.
AKTTXT zeigt hier auf das W von "WURDE".
STXTXT zeigt auf den ersten Buchstaben im Text.

Mit CIINIT2 wird AKTTXT auf den Wert von STXTXT gesetzt. Beide zeigen dann auf den Textanfang.
Der Textanfang ist durch das OPTIONEN-Menue umdefinierbar (2 = TEXTSTART).

Beispiel:

```
START:    JSR @CIINIT2          * Textstart setzen
          RTS
```

kombiniertes Beispiel:

```
START:    JSR @CIINIT2          * Textstart setzen
          JSR @CLRSCREEN        * Bildschirm loeschen
SCHLEIFE: JSR @CI2              * Text loeschen
          BEQ ENDE              * 0=Ende erreicht
          JSR @CO2              * und ausgeben
          BRA.S SCHLEIFE        * wieder zurueck
ENDE:     RTS
```

Mit diesem Beispiel wird der Textinhalt auf dem Bildschirm ausgegeben. Wenn man dabei nur das Programm eingegeben hat, und den TEXTSTART nicht verändert hat, so gibt es sich selbst auf dem Bildschirm aus.

Wenn man im OPTIONEN–Menue einen neuen Textstart wählt, so wird das Textende (ETXTXT) auf den Textanfang gesetzt, und der Code 00 an diese Stelle eingetragen.
Wenn man einen "alten Text" anwählt, so wird das Ende des Textes gesucht und dorthin ETXTXT gelegt, so daß ETXTXT auf das Byte mit dem Code 00 zeigt.

Damit ist es möglich Texte aus dem Speicher oder mit Hilfe einer eigenen Routine zu lesen.
Normalerweise wird vom Speicher gelesen. Dabei wird der Zeiger AKTTXT verwendet. Mit dem CIINIT2 Befehl kann dieser auf STXTXT, dem Textstart zurückgesetzt werden. Das Ergebnis des Aufrufs ist im Register D0.B abgelegt. Wenn das Textende erreicht ist, also das Zeichen 0 (Codierung 0) gefunden wurde, so wird zusätzlich das CARRY-Flag gesetzt. Es genügt aber auch einfach auf 0 abzufragen. Ferner wird beim Textende auch eine Speicherzelle mit dem Namen ERRFLAG auf den Wert 2 gesetzt. Diese Speicherzelle wird dazu verwendet, eine evtl. unterdrückte Ausgabe mit CO2 wieder einzuschalten. Im Assembler wird diese Funktion dazu verwendet ein ausgeschaltetes Listing (1 = Nur Fehlerausgabe) im Fehlerfall einzuschalten, um den Fehler auf den Bildschirm zu bringen.
Jetzt gibt es noch eine zweite Möglichkeit Texte einzulesen. Wenn man auf die Speicherzelle $802B den Wert 5 schreibt, so werden die Eingaben über einen Sprungvektor gelesen. Wenn man den Sprungvektor, der auf Adresse $8018 steht durch einen Sprung auf eine eigene Routine umsteuert, so kann man Eingaben auch durch eine eigene Routine erzeugen. Der Sprungvektor USERCI ($802B) ist nach dem RESET auf die Routine CI geschaltet.

Beispiel:

```
START:     JSR @CIINIT2    * Textstart auf Anfang
SCHLEIFE:  JSR @CI2        * Zeichen einlesen
           BEQ ENDE        * =0, dann Ende erreicht
           JSR @LO         * auf dem Drucker ausgeben
           BRA SCHLEIFE    * wieder zurueck
ENDE:      RTS             * Textende erreicht
```

Wird das Beispiel gestartet, so wird der aktuelle Text auf dem Drucker ausgegeben.

weiteres Beispiel:

```
START:   MOVE.B #5,$802B      * CI2 umdirigieren
         RTS
```

Wenn man nach dem Start des Programms den Assembler erneut startet, so erscheint nur ein blinkender Cursor auf dem Bildschirm.
Jetzt kann man dem Assembler direkt Befehle eingeben, z.B. um die Befehlssyntax kennenzulernen.
Dabei gibt man zuerst den Text "END" und dann die Taste CR (Carriage Return) ein. Jetzt erst erscheint die Überschrift des Assemblers auf dem Bildschirm. Die Eingabe selbst wird nicht auf dem Bildschirm ausgegeben. Der Assembler führt zwei Durchläufe aus. Beim ersten Durchlauf werden nur die Symbole definiert und dieser wurde durch die Eingabe von END übersprungen. Nun folgt der zweite Durchlauf, bei dem auch der erzeugte Code auf dem Bildschirm ausgegeben wird. Wenn man nun z.B. "NOP" und CR eingibt, so wird die Zeile mit dem Code ausgegeben. Wenn man einen Fehler macht, so wird die Fehlermeldung auch auf dem Bildschirm erscheinen. Man kann jedoch weitere Befehle eingeben.
Wenn man die Eingaben beendet hat, so gibt man einfach "END" und CR ein. Will man den Assembler wieder normal verwenden, so muß man die Speicherzelle $802B wieder auf 0 setzen, am einfachsten in dem man RESET-Taste drückt.

Wenn CIINIT2 aufgerufen wurde, so wird beim nächsten CI2-Aufruf im Register D0.W der Wert 1 an die USERCI-Routine übergeben, sonst der Wert 0. Dies ist nötig um Mehrpassvorgänge auch mit USERCI zu steuern, z.B. bei CP/M68K um die Quelle von der Diskette mehrfach zu lesen. Diese Funktion ist aber erst ab der Version 4.0 des Grundprogramms enthalten.

Zunächst besitzt CO2 die gleiche Bedeutung, wie der CO-Befehl, nämlich die Ausgabe eines Zeichens, das im Register D0.B steht auf die Console. Aber es gibt noch mehr Möglichkeiten. Zunächst kann die Ausgabe auf einen Drucker (LST) oder auf eine eigene Routine (USR) oder ins Leere (NIL) umgelenkt werden. Von dieser Funktion wird auch im Assembler Gebrauch gemacht.
Ferner erfolgt die Ausgabe nur, wenn die Speicherzelle PASSFLAG den Wert=2 enthält. Dieser Wert ist normalerweise aber voreingestellt. Die Abfrage ist speziell für den Assembler gedacht, wird aber auch von anderen Übersetzern genutzt, die mit dem Grundprogramm zusammenarbeiten. Dann gibt es noch die Möglichkeit die Ausgabe anzuhalten. Dazu wird die Taste CTRL-S gedrückt, dann erfolgt keine Ausgabe mehr und man hat somit die Möglichkeit ein Listing anzuhalten. Will man die Ausgabe fortsetzen, so muß man die Taste CTRL-Q drücken. Diese Anhaltefunktion gilt auch, wenn man die Ausgabe auf den Drucker oder eine eigene Routine umgelenkt hat.
Die Umsteuerung geschieht über die Speicherzelle $802A (IOSTAT). Folgende Codes sind für die Umsteuerung möglich:

1	Unterdrueckte Ausgabe, es wird ERRFLAG geprueft und nur wenn es gleich 1 oder 2 ist, wird ausgegeben.
2	Ausgabe auf den Bildschirm. Danach wird die Schreibseite auf 0 gestellt.
3	Ausgabe auf den Drucker.
4	Ausgabe auf den Drucker, aber das Zeichen LF (Code $A) wird unterdrueckt. Bei manchen Druckern ist dies noetig.
5	Ausgabe auf USRCO. Dabei steht ein Sprungvektor auf der Adresse $8024. Der Sprung wird nach dem RESET auf die Routine CO geschaltet.

Es gibt aber noch eine bedeutende Funktion. Wenn man den Code 01 (CTRL-A) im Register D0.B beim Aufruf der CO2-Routine angibt, so werden alle nachfolgenden Zeichen bis zum Line-Feed (Code $A) in einem Buffer (AUSBUF) zwischengespeichert. Danach erfolgt die Ausführung.

Dazu gibt es folgende Befehle (adr und wert können beliebige Ausdrücke sein, siehe WERT, so z.B. auch selbst definierte Symbole):

E adr wert0 wert1 wert2 wert3 wert4 wert5 wert6 wert7

Ausführen eines Programms "adr" mit den Parametern wert0 bis wert7. Die Parameter können auch weggelassen werden. Dabei werden die Parameter von 0 bis 7 und die Register D0.L bis D7.L gespeichert, bevor das Programm "adr" aufgerufen wird. Damit kann man in Hochsprachen Maschinen-unterprogramme aufrufen, auch wenn dies ursprünglich im Sprachkonzept nicht enthalten war.
PASCAL-Beispiel:

```
WRITELN(chr(1),'@SCHREITE 50');
```

G adr

Ein Byte wird von der Adresse "adr" geholt und bei der nächsten CI-Eingabe wird der Wert dort eingelesen. Als Trennung wird bei der CI-Eingabe als letztes Zeichen ein Leerzeichen mit ausgegeben. Mit diesem Befehl ist es möglich von beliebigen Programmiersprachen aus die Werte von Speicherzellen abzufragen. Eine CO2-Ausgabe, die während der Eingabe erfolgt wird solange unterdrückt, bis die Eingabe beendet ist, um unerwünsche ECHO-Effekte auf dem Bildschirm zu vermeiden.
PASCAL-Bespiel:

```
WRITELN(CHR(1),'G $8000');
READ(I);
```

In die Variable I (als Integer deklariert) wird der Inhalt der Speicherzelle $8000 gelesen.

P adr wert

Auf die Adresse "adr" wird der Wert "wert" geschrieben. Dabei wird nur ein Byte adressiert.
Damit kann man von Hochsprachen aus Werte im Speicher modifizieren.

PASCAL-Beispiel:

```
WRITELN(CHR(1),'P $802B 5');
```

Im übrigen gelten alle Befehle, die schon beim CO-Befehl gezeigt wurden. CO2 ist also ein Unterprogramm, das es ganz schön in sich hat. Hier nur ein paar einfache Beispiele:

```
START:   JSR @CLRSCREEN       * Init des Cursor
         MOVE.B #'A',D0       * Buchstabe "A" ausgeben
         JSR @CO2
         RTS
```

Die Ausgabe des Buchstabens "A" kann man ganz leicht auf den Drucker umlenken, wenn man im OPTIONEN-Menue die Ausgabe auf den Drucker umstellt.
weiteres Beispiel:

```
START:   JSR @CLRSCREEN       * Erst mal Init
         MOVE #3-1,D3         * Auessere Schleife
SCHL0:   LEA TEXT,A0          * Ausgabetextadresse
SCHLEIFE: MOVE.B (A0)+,D0     * Zeichen laden
         BEQ ENDE             * 0 sein Ende
         MOVEM.L D3/A0,-(A7) * A0 wird ggf, zerstoert.  -*
         JSR @CO2             * und ausgeben             *
         MOVEM.L (A7)+,D3/A0 * bei Rev 4.0 nicht noetig -*
         BRA SCHLEIFE         * wiederholen
ENDE:    DBRA D3,SCHL0        * Text dreimal ausgeben
         RTS                  * Ende
TEXT:    DC.B 1,'E @SCHREITE 100',$A
         DC.B 1,'E @DREHE 120',$A,0
```

Nach Start des Programms erscheint ein Dreieck auf dem Bildschirm.

SETFLIP 34

Die GDP-Baugruppe besitzt vier Bildseiten. Mit SETFLIP kann man eine automatische Umschaltrate beinflussen, die z.B. mit AUTOFLIP ausgeführt wird. Die Umschaltung erfolgt auch dann, wenn man z.B. nach Ausführung eines Programms die Anweisung "M=Menue" erhält oder wenn man Befehle der Schildkrötensprache ausführt.
Es gibt zwei Arten der automatischen Umschaltung. Den Wechsel zwischen zwei Bildseiten, oder den Wechsel zwischen vier Bildseiten.
Dazu übergibt man beim Aufruf zwei Parameter. Im Register D0.B steht die Wechselrate für die Zweiseitenumschaltung und im Register D1.B die Wechselrate für die Vierseitenumschaltung. Der Wert 0 bedeutet, keine Umschaltung und jeder Wert <> 0 ist die Umschaltrate in 20 Milisekunden-Schritten.
Beispiel:

```
START:   JSR @CLRSCREEN      * Cursor auf Bildschirm bringen
         MOVE #50,D0         * Blinkrate auf 1 sec
         CLR D1              * einstellen
         JSR @SETFLIP
         RTS
```

Der Cursor im linken oberen Feld blinkt anschließend im Sekundentakt.

weiteres Beispiel:

```
START:   JSR @WAIT           * Warten, bis GDP fertig
         MOVE.B #$33,$FFFFFF73   * Groesse setzen
         CLR D1
         MOVE #128,D2
         JSR @MOVETO         * Start X=0,Y=128
         CLR D3
         CLR D4              * Start bei Seite 0
         MOVE #40-1,D5       * Viele Buchstaben
SCHLEIFE: MOVE D3,D0         * Schreibseite
         MOVE D4,D1          * Leseseite
         AND #3,D0           * nur 0..3 sinnvoll
         JSR @NEWPAGE        * Seite setzen
         MOVE.B #'A',D0      * Buchstabe A ausgeben
         JSR @CHD            * auf verschiedene Seiten
         ADD #1,D3           * der Schreibseite
         DBRA D5,SCHLEIFE    * an versch. Positionen
         CLR D0              * Zweiseitenumschaltung=0
         MOVE #5,D1          * Viers. = 5*20 ms
         JSR @SETFLIP        * und setzen
         JSR @CI             * Warten bis Taste gedr}ckt
         RTS
```

Eine Buchstabenreihe wandert über die Bildschirmmitte. Wenn man eine Taste drückt, so erscheint blinkend die "M=Menue"-Anweisung auf dem Bildschirm. In der Routine CI wird genauso wie beim "M=Menue" die Routine AUTOFLIP automatisch aufgerufen, die den Umschaltvorgang steuert. Man kann AUTOFLIP auch als Unterprogramm verwenden, wenn man die Umschaltung nutzen möchte, ohne das CI verwendet wird.

Die automatische Bildschirmumschaltung mit vier Seiten ermöglich bewegte Graphiken, die sehr komplex sein können, wenn man die Bewegung in vier Teilphasen zerlegen kann. Beispiel: Drehen von Zahnrädern (auch 3D), Abläufe mit bewegenden Pfeilen etc.

DELAY 35

Die Routine DELAY gestattet quarzgenaue Verzögerungen im 100 Millisekundenraster. Dabei wird während der Verzögerungszeit das automatische Seitenumschalten aufrechterhalten, so daß sich beim Ablauf keine Störungen bei der Bildschirmumschaltung durch SETFLIP ergeben. Dies ist insbesondere bei Bildern mit der Schildkrötengraphik wichtig. Die Verzögerung erfolgt im 100 ms - Raster. Im Register D0.L wird dann die Anzahl der 1/100 sec angeben. Wenn die Zahl = 0 ist, so erfolgt keine Ausführung des Befehls.

Beispiel:

```
START:   MOVE.L #10,D0  * 1 Sek verzoegern
         JSR @DELAY
         RTS
```

weiteres Beispiel:

```
START:     MOVE #36-1,D3   * Kreis zeichnen
SCHLEIFE:  MOVE #10,D0     * Schreite 10
           JSR @SCHREITE
           MOVE #10,D0     * Drehe 10
           JSR @DREHE
           DBRA D3,SCHLEIFE
           MOVE.L #20,D0   * 2 Sek verzoegern
           JSR @DELAY
           JSR @CLR        * Dann Bildschirm loeschen
           MOVE #4-1,D3    * Dann Quadrat
SCHL2:     MOVE #100,D0    * Schreite 100
           JSR @SCHREITE
           MOVE #90,d0     * Drehe 90
           JSR @DREHE
           DBRA D3,SCHL2
           MOVE.L #20,D0   * und wieder warten
           JSR @DELAY
           JSR @CLR        * Dann Bildschirm loeschen
           JSR @CSTS       * bis eine Taste gedrueckt
           BEQ START       * fortfahren,
```

Eine Buchstabenreihe wandert über die Bildschirmmitte. Wenn man eine Taste drückt, so erscheint blinkend die "M=Menue"-Anweisung auf dem Bildschirm. In der Routine CI wird genauso wie beim "M=Menue" die Routine AUTOFLIP automatisch aufgerufen, die den Umschaltvorgang steuert. Man kann AUTOFLIP auch als Unterprogramm verwenden, wenn man die Umschaltung nutzen möchte, ohne das CI verwendet wird.

Die automatische Bildschirmumschaltung mit vier Seiten ermöglich bewegte Graphiken, die sehr komplex sein können, wenn man die Bewegung in vier Teilphasen zerlegen kann. Beispiel: Drehen von Zahnrädern (auch 3D), Abläufe mit bewegenden Pfeilen etc.

DELAY 35

Die Routine DELAY gestattet quarzgenaue Verzögerungen im 100 Millisekundenraster. Dabei wird während der Verzögerungszeit das automatische Seitenumschalten aufrechterhalten, so daß sich beim Ablauf keine Störungen bei der Bildschirmumschaltung durch SETFLIP ergeben. Dies ist insbesondere bei Bildern mit der Schildkrötengraphik wichtig. Die Verzögerung erfolgt im 100 ms – Raster. Im Register D0.L wird dann die Anzahl der 1/100 sec angeben. Wenn die Zahl = 0 ist, so erfolgt keine Ausführung des Befehls.

Beispiel:

```
START:   MOVE.L #10,D0  * 1 Sek verzoegern
         JSR @DELAY
         RTS
```

weiteres Beispiel:

```
START:   MOVE #36-1,D3  * Kreis zeichnen
SCHLEIFE: MOVE #10,D0   * Schreite 10
         JSR @SCHREITE
         MOVE #10,D0    * Drehe 10
         JSR @DREHE
         DBRA D3,SCHLEIFE
         MOVE.L #20,D0  * 2 Sek verzoegern
         JSR @DELAY
         JSR @CLR       * Dann Bildschirm loeschen
         MOVE #4-1,D3   * Dann Quadrat
SCHL2:   MOVE #100,D0   * Schreite 100
         JSR @SCHREITE
         MOVE #90,d0    * Drehe 90
         JSR @DREHE
         DBRA D3,SCHL2
         MOVE.L #20,D0  * und wieder warten
         JSR @DELAY
         JSR @CLR       * Dann Bildschirm loeschen
         JSR @CSTS      * bis eine Taste gedrueckt
         BEQ START      * fortfahren,
         RTS            * sonst Ende.
```

Die Darstellung des Bildes wechselt alle 2 Sekunden zwischen einem Kreis und einem Quadrat.

FIRSTTIME 36

Dadurch wird die Schildkrötengraphik initialisiert. Die Schildkröte wird in die Bildmitte gesetzt und die Bildseitenautomatik für die Umschaltung wird aktiviert. Ferner wird die Schreib– und Leseseite auf 0 gestellt. FIRSTTIME wird auch automatisch aufgerufen, wenn man einen der Schildkrötenbefehle verwendet. Es ist also nur in Sonderfällen nötig, diesen Befehl zu verwenden. Ein Fall ist z.B., wenn man mit mehreren Ebenen arbeitet und die Schildkröte auf dem Bildschirm nicht haben möchte, dann ist folgende Sequenz interessant:

Beispiel:

```
START:   JSR @FIRSTTIME      * Schildkroete an
         JSR @HIDE           * aber nicht darstellen
         RTS
```

Die Schildkröte wird dann auch nicht am Anfang gesetzt, da ein Setzen nur erfolgt wenn AUTOFLIP aufgerufen wird und das geschieht nur bei den anderen Schildkrötenbefehlen. HIDE jedoch setzt den Wert von FLIP auf 0, somit findet keine automatische Bildseitenumschaltung statt und dann wird auch bei späterem Aufruf von AUTOFLIP die Schildkröte nicht eingeblendet.
Will man AUTOFLIP später verwenden um alle Bildseiten nacheinander darzustellen, so muß man mit GRAPOFF, die Schildkrötengraphik ausschalten.

Beispiel:

```
START:    JSR @FIRSTTIME      * Schildkroete an
          JSR @HIDE           * Seitenautomatik aus
          MOVE #4-1,D4        * Schreibseite 3
          CLR D5              * Leseseite 0
SCHLEIFE: MOVE D4,D0          * Schreibseite
          MOVE D5,D1          * Leseseite
          JSR @NEWPAGE        * Seite einstellen
          BSR RAD             * Rad dort zeichnen
          MOVE #45,D0         * Dann drehen
          JSR @DREHE          * bis alle vier Seiten
          DBRA D4,SCHLEIFE    * beschrieben
          JSR @GRAPOFF        * bevor AUTOFLIP dran ist,
          CLR D0              * Zweiseiteflip aus
          MOVE #3,D1          * aber Vierseitenflip an
          JSR @SETFLIP        *
          JSR @CI             * z.B. jetzt
          RTS                 * Wenn M=Menue kommt
RAD:      MOVEM D3/D4,-(A7)   * retten
          MOVE #6-1,D4        * Drehteil
RAD1:     MOVE #3-1,D3        * Dreieck
RAD2:     MOVE #100,D0        * Schreite 100
          JSR @SCHREITE
          MOVE #120,D0        * Drehe 120
          JSR @DREHE
          DBRA D3,RAD2        * Beide Schleifen
          MOVE #60,D0         * Dann noch drehen
          JSR @DREHE
          DBRA D4,RAD1
          MOVEM (A7)+,D3/D4
          RTS
```

Nach dem Start des Programms wird eine Figur ausgegeben, die sich dreht.

SETPEN 37

Damit wird der Graphikprozessor in den Schreibmode mit gesenktem Schreibstift versetzt.
Alle nachfolgenden Graphikbefehle erfolgen also mit weißer Linie, auf ggf. dunklem Grund.

Beispiel:

```
          JSR @SETPEN
          RTS
```

Mit dem Befehl kann man also vorhergehende Befehl rücksetzen, die einen Löschvorgang bewirken. Nach dem Befehl CLR und CLPG, wie auch CLRSCREEN wird jedoch der GDP automatisch auf Schreiben umgeschaltet.

ERAPEN 38

Schreibstift auf Löschen umschalten. Dabei wird der Stift auch gesenkt. Nun kann man dunkle Linie auf hellem Grund schreiben oder bestehende Graphikteile löschen.

Beispiel:

```
JSR @ERAPEN
RTS
```

weiteres Beispiel:

```
START:     CLR D1           * X=0
           MOVE #128,D2     * Y=128
           JSR @WAIT        * Groesse einstellen
           MOVE.B #$33,$FFFFFF73
SCHLEIFE:  JSR @MOVETO      * Start festlegen
           MOVE.B #'A',D0   * Buchstabe setzen
           JSR @CMD         * mit CMD
SY1:       JSR @SYNC        * 20ms
           BEQ SY1
           JSR @MOVETO      * wieder zurueck
           JSR @ERAPEN      * Dann Buchstaben
           MOVE.B #'A',D0   * loeschen
           JSR @CMD
           JSR @SETPEN      * und dann wieder schreiben
           ADD #1,D1        * neues X
           CMP #512,D1      * Wenn > 511 dann Stop
           BCS SCHLEIFE     * sonst wiederholen
           RTS
```

Der Buchstabe "A" wandert von links nach rechts über den Bildschirm. Wenn der Buchstabe das Bild verlassen hat, meldet sich "M=Menue".

GRAPOFF 39

(Schildkrötengraphik)
Mait dem Befehl wird die Schildkrötengraphik ausgeschalten, daß heißt die Schildkröte wird von da an nicht mehr auf die Bildebene 1 gezeichnet. Sonst ändern sich keine Einstellungen. Wenn man danach einen Schildkrötenbefehl, wie SCHREITE gibt, so beginnt die Schildkröte wieder in der Mitte des Bildschirms zu zeichnen.
Intern wird die Variable FIRST mit dem Wert 1 belegt.

Beispiel:

```
        JSR @GRAPOFF
        RTS
```

weiteres Beispiel:

```
START:  MOVE #100,D0    * Linie zeichnen
        JSR @SCHREITE
        JSR @GRAPOFF    * dann ausschalten
        MOVE #90,D0     * dann drehen
        JSR @DREHE      * und
        MOVE #100,D0    * wieder schreiten
        JSR @SCHREITE   *
        RTS             * Ende
```

Auf dem Bildschirm sind zwei von der Bildmitte ausgehende Linien zu sehen.

CMDPRINT 40

Damit kann man eine Befehlssequenz direkt an den GDP ausgeben. Ferner stellt man zuvor mit Register D0.B die Schriftgröße ein (wie Register 3 des EF9366), in D1.W steht die X-Koordinate und in D2.W die Y-Koordinate. Das Register A0 enthält die Adresse bei der sich die Daten befinden. Eine binäre 0 (Code 00) beendet den Text. Man kann also damit nicht den Befehl "Auf Schreiben schalten" mit dem Code 00 an den GDP ausgeben, sondern muß dies ggf. mit SETPEN oder mit CMD direkt tun. Der Befehl eignet sich sowohl zur Textausgabe, als auch zur Ausgabe von Graphikbefehlen, den sogenannten Kurzvektoren, mit denen man sehr schnell kleine Figuren zeichnen kann.

Beispiel:

```
START:   LEA DATEN,A0        * Adresse der Daten laden
         MOVE #$22,D0        * Textgroesse dx=2,dy=2
         MOVE #50,D1         * X=50
         MOVE #120,D2        * Y=120
         JSR @CMDPRINT       * und ausfuehren
         RTS                 * Ende
DATEN:   DC.B 'Hallo',0      * Quelle
```

Da die Anwendung der Kurzvektoren so interessant ist, hier den prinzipielle Aufbau eines Kurzvektors:

```
B7   B6   B5   B4   B3   B2   B1   B0
1    --dx---   --dy---   --Richtung--
```

Am ersten Bit erkennt der GDP, das es sich um einen Kurzvektor handelt. Mit dx wird die Anzahl der Punkte in X-Richtung beschrieben, mit dy die Anzahl der Punkte in Y-Richtung. "Richtung" legt die Richtung des Vektors fest. Da der GDP hierbei nur in 45 Grad-Schritten arbeitet, kann man dx und dy immer mit dem gleichen Wert belegen. 00 bedeutet dabei kein Schritt ausführen, 01 bedeutet einen Schritt, 10 sind zwei Schritte und 11 drei Schritte. Mit dem Code $80 kann man z.B. einen einzelnen Punkt setzen.

Richtungscode:

```
      010
    011 001
110         000
  111   101
      100
```

Beispiel:

```
START:  LEA DATEN,A0        * Adresse der Daten laden
        MOVE #$22,D0        * Textgroesse dx=2,dy=2
        MOVE #50,D1         * X=50
        MOVE #120,D2        * Y=120
        JSR @CMDPRINT       * und ausfuehren
        RTS                 * Ende
DATEN:  DC.B %11111000      * nach Rechts mit 3 Schritten
        DC.B %11111001      * schraeg Rechts nach Oben
        DC.B %11111010      * nach Oben
        DC.B %11111011      * schraeg Links nach Oben
        DC.B %11111110      * nach Links
        DC.B %11111111      * schraeg Links nach Unten
        DC.B %11111100      * nach Unten
        DC.B %11111101      * schraeg Rechts nach Unten
        DC.B 0
```

Auf dem Bildschirm erscheint ein kleines Achteck.

PRINT2X 41

Sedezimale Ausgabe eines Wertes. Dabei werden zwei sedezimale Ziffern ausgegeben. Der Wert steht zuvor im Register D0.B. Im Register A0 ist die Zieladresse für die sedezimalen Ziffern angegeben. Dabei wird das Ergebnis im ASCII-Code abgelegt. A0 zeigt anschließend auf den nächsten freien Platz und dort ist eine 0 (Code 00) abgelegt, so daß man den Text mit den Befehlen wie WRITE direkt ausgeben kann.

Beispiel:

```
        LEA ZIEL,A0          * Zielbufferadresse
        MOVE.B #$1A,D0       * Wert ins Register D0.B
        JSR @PRINT2X         * und ausgeben
        RTS                  * Ende
ZIEL:   DS.B 10              * Zielbuffer.
```

Im Zielbuffer wird der Text '1A',0 abgelegt.

Beispiel mit Bildschirmausgabe:

```
START:    CLR D4             * Zaehler
SCHLEIFE: LEA ZIEL,A0        * Adresse laden
          MOVE.B D4,D0       * Wert laden
          JSR @PRINT2X       * und ausgeben
          LEA ZIEL,A0        * dann auf den Bildschirm
          MOVE.B #$22,D0     * Schriftgroesse
          MOVE #2,D1         * X-Koordinate
          MOVE #128,D2       * Y-Koordinate
          JSR @WRITE         * und ausgeben
          MOVE #5,D0         * 1/2 Sek warten
          JSR @DELAY
          ADDQ #1,D4         * Zaehler erhoehen
          CMP #256,D4        * Bis Ende erreicht
          BNE SCHLEIFE       * alle Codes ausgeben
          RTS
ZIEL:     DS.B 10            * eigentlich nur 3 Bytes noetig.
```

Das Programm zählt sedezimal von 0 bis $FF und gibt das Ergebnis alle 1/2 Sekunde auf dem Bildschirm aus. Wenn man den Befehl DELAY wegläßt erfolgt die Ausgabe so schnell, daß man nichts mehr erkennen kann.

PRINT4X 42

Sedezimale Ausgabe eines Wertes. Dabei werden vier sedezimale Ziffern ausgegeben. Der Wert steht zuvor im Register D0.W. Im Register A0 ist die Zieladresse für die sedezimalen Ziffern angegeben. Dabei wird das Ergebnis im ASCII-Code abgelegt. A0 zeigt anschließend auf den nächsten freien Platz und dort ist eine 0 (Code 00) abgelegt, so daß man den Text mit den Befehlen wie WRITE direkt ausgeben kann.

Beispiel:

```
        LEA ZIEL,A0          * Zielbufferadresse
        MOVE.W #$12AB,D0     * Wert ins Register D0.W
        JSR @PRINT4X         * und ausgeben
        RTS                  * Ende
ZIEL:   DS.B 10              * Zielbuffer.
```

Im Zielbuffer wird der Text '12AB',0 abgelegt.

PRINT6X 43

Sedezimale Ausgabe eines Wertes. Dabei werden sechs sedezimale Ziffern ausgegeben. Der Wert steht zuvor im Register D0.L. Im Register A0 ist die Zieladresse für die sedezimalen Ziffern angegeben. Dabei wird das Ergebnis im ASCII-Code abgelegt. A0 zeigt anschließend auf den nächsten freien Platz und dort ist eine 0 (Code 00) abgelegt, so daß man den Text mit den Befehlen wie WRITE direkt ausgeben kann.

Beispiel:

```
        LEA ZIEL,A0          * Zielbufferadresse
        MOVE.L #$12ABCD,D0   * Wert ins Register D0.L
        JSR @PRINT6X         * und ausgeben
        RTS                  * Ende
ZIEL:   DS.B 10              * Zielbuffer.
```

Im Zielbuffer wird der Text '12ABCD',0 abgelegt.

PRINT8X 44

Sedezimale Ausgabe eines Wertes. Dabei werden acht sedezimale Ziffern ausgegeben. Der Wert steht zuvor im Register D0.L. Im Register A0 ist die Zieladresse für die sedezimalen Ziffern angegeben. Dabei wird das Ergebnis im ASCII-Code abgelegt. A0 zeigt anschließend auf den nächsten freien Platz und dort ist eine 0 (Code 00) abgelegt, so daß man den Text mit den Befehlen wie WRITE direkt ausgeben kann.

Beispiel:

```
        LEA ZIEL,A0           * Zielbufferadresse
        MOVE.L #$1234ABCD,D0  * Wert ins Register D0.B
        JSR @PRINT8X          * und ausgeben
        RTS                   * Ende
ZIEL:   DS.B 10               * Zielbuffer.
```

Im Zielbuffer wird der Text '1234ABCD',0 abgelegt.

PRINT8B 45

Binaere Ausgabe eines Wertes. Dabei werden acht Dualstellen ausgegeben. Der Wert steht zuvor im Register D0.B. Im Register A0 ist die Zieladresse für die dualen Ziffern angegeben. Dabei wird das Ergebnis im ASCII-Code abgelegt. A0 zeigt anschließend auf den nächsten freien Platz und dort ist eine 0 (Code 00) abgelegt, so daß man den Text mit den Befehlen wie WRITE direkt ausgeben kann.

Beispiel:

```
        LEA ZIEL,A0          * Zielbufferadresse
        MOVE.B #$1A,D0       * Wert ins Register D0.B
        JSR @PRINT8B         * und ausgeben
        RTS                  * Ende
ZIEL:   DS.B 10              * Zielbuffer.
```

Im Zielbuffer wird der Text '00011010',0 abgelegt.

Beispiel mit Bildschirmausgabe:

```
START:    CLR D4             * Zaehler
SCHLEIFE: LEA ZIEL,A0        * Adresse laden
          MOVE.B D4,D0       * Wert laden
          JSR @PRINT8B       * und ausgeben
          LEA ZIEL,A0        * dann auf den Bildschirm
          MOVE.B #$22,D0     * Schriftgroesse
          MOVE #2,D1         * X-Koordinate
          MOVE #128,D2       * Y-Koordinate
          JSR @WRITE         * und ausgeben
          MOVE #5,D0         * 1/2 Sek warten
          JSR @DELAY
          ADDQ #1,D4         * Zaehler erhoehen
          CMP #256,D4        * Bis Ende erreicht
          BNE SCHLEIFE       * alle Codes ausgeben
          RTS
ZIEL:     DS.B 10            * eigentlich nur 9 Bytes noetig.
```

Das Programm zählt von 0 bis 255 und gibt das Ergebnis alle 1/2 Sekunde in dualer Schreibweise auf dem Bildschirm aus. Wenn man den Befehl DELAY wegläßt erfolgt die Ausgabe so schnell, daß man nichts mehr erkennen kann.

PRINT4D 46

Dezimale Ausgabe eines Wertes. Der Wert steht zuvor im Register D0.W. Im Register A0 ist die Zieladresse für die dezimalen Ziffern angegeben.
Dabei wird das Ergebnis im ASCII-Code abgelegt. A0 zeigt anschließend auf den nächsten freien Platz und dort ist eine 0 (Code 00) abgelegt, so daß man den Text mit den Befehlen wie WRITE direkt ausgeben kann.
Achtung die Ausgabe erfolgt ohne Vorzeichen. Wenn man Zweierkomplemet-Zahlen verwendet ,so muß man bei einer negativen Zahl das Vorzeichen selbst mit ausgeben.
Bei der dezimalen Ausgabe werden immer nur soviele Stellen ausgegeben, wie sie durch die Zahl belegt sind. Daher muß man, wenn man Zahlen übereinanderschreibt den Zielbuffer mit Leerzeichen auffüllen.

Beispiel:

```
        LEA ZIEL,A0          * Zielbufferadresse
        MOVE.W #$123A,D0     * Wert ins Register D0.B
        JSR @PRINT4D         * und ausgeben
        RTS                  * Ende
ZIEL:   DS.B 10              * Zielbuffer.
```

Im Zielbuffer wird der Text '4666',0 abgelegt.

Beispiel mit Bildschirmausgabe:

```
START:    MOVE #$FFFF,D4      * Zaehler
SCHLEIFE: LEA ZIEL,A0         * Adresse laden
          MOVE D4,D0          * Wert laden
          JSR @PRINT4D        * und ausgeben
          MOVE #6-1,D3        * max Stellen
LOESCHE:  MOVE.B #' ',(A0)+   * auffuellen mit
          DBRA D3,LOESCHE     * Leerzeichen
          CLR.B (A0)          * und Ende kennzeichnen
          LEA ZIEL,A0         * dann auf den Bildschirm
          MOVE.B #$22,D0      * Schriftgroesse
          MOVE #2,D1          * X-Koordinate
          MOVE #128,D2        * Y-Koordinate
          JSR @WRITE          * und ausgeben
          MOVE #5,D0          * 1/2 Sek warten
          JSR @DELAY
          DBRA D4,SCHLEIFE    * alle Codes ausgeben
          RTS
ZIEL:     DS.B 20
```

Das Programm zählt von 65535 bis 0 und gibt das Ergebnis alle 1/2 Sekunde in dualer Schreibweise auf dem Bildschirm aus. Wenn man den Befehl DELAY wegläßt erfolgt die Ausgabe schneller.

HIDE 47

(Schildkrötengraphik)
Wenn man den Befehl eingibt, so wird die automatische Seitenumschaltung abgestellt und die Schildkröte wird nicht mehr ins Bild eingeblendet.

Beispiel:

```
START:   MOVE #10,D0     * erst mal schreiten
         JSR @SCHREITE
         JSR @HIDE       * Dann Flip ausschalten
         RTS             * Ende
```

Will man vermeiden, das die Schildkröte noch auf der Bildseite 1 gezeichnet wird, weil man diese Seite auch verwenden will, so empfiehlt sich folgende Methode:

```
START:   JSR @FIRSTTIME * Schildkr|tenmode an
         JSR @HIDE      * aber gar nicht erst zeichnen lassen
         MOVE #10,D0    * Dann zeichnen
         JSR @SCHREITE
         JSR @GRAPOFF   * und wichtig, dann ausschalten
         RTS
```

Wenn man das Program startet erscheint eine stehende Linie auf dem Bildschirm. Wenn man nun die Taste "F=Flip" betätigt, so werden beide Bildseiten umgeschaltet, aber keine Schildkröte erscheint.

SHOW 48

(Schildkrötengraphik)
Eine durch HIDE ausgeschaltete Bildschirmseitenumschaltung wird wieder rückgängig gemacht. Die Schildkröte wird dann wieder eingeblendet.

Beispiel:

```
START:    MOVE #10,D0     * erst mal Schreiten
          JSR @SCHREITE
          JSR @HIDE       * Dann abschalten
          JSR @SHOW       * Dann wieder anschalten
          RTS
```

Beispiel:

```
START:    MOVE #100,D0    * erst mal Schreiten
          JSR @SCHREITE
SCHLEIFE: JSR @HIDE       * dann abschalten
          MOVE #10,D0     * 1 Sek warten
          JSR @DELAY
          JSR @SHOW       * dann einschalten
          MOVE #10,D0     * wieder 1 Sek warten
          JSR @DELAY
          JSR @CSTS       * solange bis Taste gedrueckt
          BEQ SCHLEIFE    * dann erst anhalten.
          RTS
```

Alle Sekunde wechselt die Darstellung zwischen "Schildkröte" sichtbar und "Schildkröte" unsichtbar.

CRT 49

Die CO2-Ausgabe ist auf verschiedene Ausgabegeräte umstellbar. Mit dem Aufruf von CRT wird die Ausgabe wieder auf den Bildschirm gestellt. Dieser Befehl ist insbesondere für höhere Programmiersprachen wichtig, um die Ausgaben auf unterschiedliche Geräte zu steuern. Mit CRT wird aber meißt die Ausgabe zurückgesetzt.

Beispiel:

```
JSR @CRT
RTS
```

Pascal-Beispiel:

```
PROGRAM TEST(OUTPUT);
BEGIN
WRITELN(CHR(1),'E @LST'); { Ausgabe auf Drucker }
WRITELN('Drucker gibt aus');
WRITELN(CHR(1),'E @CRT'); { Ausgabe auf CRT }
WRITELN('Bildschirm gibt aus');
END.
```

LST 50

Die Ausgabe über den CO2-Befehl wird auf den Drucker umgesteuert. Damit kann man z.B. von PASCAL aus den Drucker über die normale WRITELN-Anweisung bedienen.

Beispiel:

```
JSR @LST        * umschalten auf den Drucker
RTS
```

Die Funktion ist genauso wie im Optionenmenue, wenn man die Ausgabe des Assemblers auf den Drucker umstellt.

Pascal-Beispiel:

```
PROGRAM TEST(OUTPUT);
BEGIN
WRITELN(CHR(1),'E @LST');
WRITELN('Kommt auf dem Drucker')
END.
```

USR 51

Die Ausgabe des CO2-Befehls wird auf die Benutzerschnittstelle umgelenkt. Siehe auch Beschreibung des CO2-Befehls.

Beispiel:

```
JSR @USR
RTS
```

Pascal-Beispiel:

```
PROGRAM TEST(OUTPUT);
BEGIN
WRITELN(CHR(1),'E @USR');
WRITELN('Kommt ueber die Benutzerroutine')
END.
```

Wenn man das Programm ausführt erscheint die Ausgabe normalerweise auf dem Bildschirm. Um sie umzuleneken muß man zusätzlich den Benutzervektor im RAM durch ein kleines Assemblerprogramm umsteuern (Siehe CO2-Beschreibung).

NIL 52

Alle Ausgabe mit dem Befehl CO2 werden unterdrückt, es sein denn, die Variable ERRFLAG wurde umbesetzt (siehe CO2-Beschreibung).
Diese Funktion ist vorwiegend für Übersetzer gedacht, bei denen man das Listing abschalten möchte um die Übersetzung zu beschleunigen.

Beispiel:

```
JSR @NIL
RTS
```

Die Ausfährung des Befehls entspricht der Eingabe im Optionenmenue "1 = Nur Fehlerausgabe".

SETERR 53

Damit wird die Variable ERRFLAG gesetzt. Im Register D0.W steht der Wert. Wenn ERRFLAG einen Wert ungleich 0 annimmt, so wird eine evtl. mit NIL unterdrückte CO2-Ausgabe wieder freigegeben. Diese Funktion ist speziell für Übersetzer gedacht, die im Fehlerfall Meldungen an die Console geben müssen.

Beispiel:

```
START:  JSR @CLRSCREEN      * CO2 vorbereiten
        JSR @NIL            * dann ausschalten
        MOVE #1,D0          * Fehler setzen
        JSR @SETERR         *
        MOVE.B #'A',D0      * Dann Ausgabe
        JSR @CO2            * durchfuehren
        RTS
```

Der Buchstabe "A" erscheint auf dem Bildschirm, obwohl durch den Aufruf von NIL die Ausgabe abgeschaltet wurde. Wenn man den Aufruf von SETERR wegläßt, so erscheint keine Meldung.
Wenn man nach einem Aufruf von SETERR mit D0.W <>0, anschließend SETERR mit einem Wert von 0 in D0.W aufruft, so wird die Ausgabe wieder unterdrückt.

GETERR 54

Der Inhalt der Variablen ERRFLAG wird ins Register D0.W gelesen. Damit kann man sich über den Fehlerstatus informieren. ERRFLAG wird zum Beispiel automatisch auf den Wert 2 gesetzt, wenn man CI2 solange aufruft, bis über das Ende des Textes gelesen wird. Dann wird auch ERRFLAG auf 2 gesetzt. Wichtig ist dabei, daß eine dann nachfolgende Ausgabe mit CO2 wieder erfolgt, falls sie mit NIL unterdrückt wurde, und aller Text nach der Fehlerbedingung auf den Bildschirm ausgegeben wird.

Beispiel:

```
        JSR @GETERR         * Fehlercode in D0.W
        RTS
```

SETPASS 55

Auch ein Befehl, der für Übersetzer interessant ist. Damit wird die Variable PASSFLAG gesetzt. Der Wert steht zuvor in Register D0.W. Dabei ist wichtig, daß eine Ausgabe über CO2 nur dann erfolgt, wenn der Wert in PASSFLAG = 2 ist. Sonst wird die Ausgabe unterdrückt. Der Wert 2 wird sonst automatisch vom Grundprogramm vorbelegt, um die Ausgabe freizuschalten. Wichtig ist dieser Befehl für Übersetzer, die beim ersten Durchlauf noch keine Ausgabe auf dem Bildschirm erzeugen sollen. Dann wird z.B. PASSFLAG = 1 gesetzt. Erst beim zweiten Durchlauf wird z.B. beim Assembler eine Ausgabe erzeugt. Dann wird PASSFLAG durch den Befehl SETPASS auf 2 gesetzt und die Ausgabe durchgeführt.

Beispiel:

```
START:  JSR @CLRSCREEN      * Bildschirm loeschen
        MOVE #1,D0          * Pass auf 1 setzen
        JSR @SETPASS
        MOVE.B #'A',D0      * Dann Buchstaben ausgeben
        JSR @CO2            * erscheint aber nicht
        MOVE #2,D0          * Pass auf 2 setzen
        JSR @SETPASS
        MOVE.B #'B',D0      * Nun muss Buchstabe "B"
        JSR @CO2            * auf dem Bildschirm
        RTS                 * erscheinen.
```

Nur der Buchstabe "B" wird auf dem Bildschirm ausgegeben.

EDIT 56

Damit kann man den TEXT-EDITOR als Unterprogramm aufrufen und so Eingaben vom Benutzer verlangen. Der Text wird beginnen bei der Adresse die durch STXTXT bestimmt ist abgelegt. Diese Adresse kann man ebenfalls verändern, um so ein freies Textgebiet verwenden zu können.

Beispiel:

```
START:    JSR @EDIT       * Editor aufrufen
          RTS             * Ende des Editierens
```

weiteres Beispiel:

```
START:    JSR @GETSTX          * Alten Wert merken
          MOVE.L D0,-(A7)      * auf den Stack
          MOVE.L #TEXT,D0      * Adresse Textstart
          JSR @PUTSTX          * Textstart definieren
          JSR @EDIT            * Dann Editieren
          JSR @CLRSCREEN       * und auf dem Bildschirm
          JSR @CIINIT2         * ausgeben
SCHLEIFE: JSR @CI2             * erst Lesen
          BEQ.S ENDE           * =0 dann Ende
          JSR @CO2             * auf den Bildschirm
          BRA.S SCHLEIFE
ENDE:     MOVE.L (A7)+,D0      * alter Wert zurueck
          JSR @PUTSTX          *
          RTS
TEXT:     DC.B 'Hallo',0       * Defaulttext
                               * Ende ist offen
```

Nach dem Start des Programms erscheint der Text "Hallo" auf dem Bilschirm und man befindet sich im Editor. Jetzt kann man beliebige Texte eingeben und wenn man den Editor verläßt, so wird der Text auf dem Bildschirm ausgegeben. Wichtig ist, daß man die 0 bei TEXT: nicht als Abschluß vergisst, sonst ist das Textende nicht definiert.
Anstelle von CO2 kann man auch LO aufrufen und damit den eingegebenen Text auf dem Drucker ausgeben.

FIGUR 57

Damit kann man bewegliche Figuren definieren, die man über den Bildschirm bewegen kann. Im Register D0.W wird dazu die Größe der Figur angeben. Ist der Wert = 1, so erhält man die kleinste Figur. Mit anderen Werten kann man die Größe jeweils mit dem Wert multiplizieren, der angegeben wird. Im Register D1.W steht die X-Koordinate und im Register D2.W die Y-Koordinate.
Das Register A0 enthält die Adresse der Figurendefinition.
Um eine Figur zeichnen zu können muß sie zunächst definiert werden. Dazu wird die Figur in einzelne Vektoren zerlegt und aus ihnen zusammengesetzt. Die Codierung ist dabei wie folgt:

Ferner gibt es:

```
 8   =   Schreibstift heben
 9   =   Schreibstift senken
10   =   Ende der Tabelle
```

Wenn man bei einem erneuten Aufruf die Größe oder die X-Koordinate oder die Y-Koordinate oder die Adresse der Figur ändert, so wird zunächst die alte Figur gelöscht und dann die neue gezeichnet. Dadurch ergibt sich der Bewegungseffekt. Wenn man als neue Größe den Wert 0 im Register D0.W angibt, so wird nur die alte Figur gelöscht, aber keine neue gezeichnet.

Beispiel:

```
START:     CLR D1          * X-Koordinate
           MOVE #100,D2    * Y-Koordinate
SCHLEIFE:  LEA TABELLE,A0  * Figurendefinition
           MOVE #3,D0      * Vergroesserung
           JSR @FIGUR      * und setzen
SY1:       JSR @SYNC       * alle 20 ms bewegen
           BEQ.S SY1
           ADD #4,D1       * X:=X+4
           CMP #512,D1     * Wenn groesser als 512
           BCS.S SCHLEIFE  * dann fertig
           RTS
TABELLE:   DC.B 0,1,2,3,4,5,6,7,10
```

Ein Kringel bewegt sich von links nach rechts über den Bildschirm.

Eine besondere Möglichkeit ist es aber auch die Adresse zu wechseln, um so sich selbst bewegende Figuren über den Bildschirm bewegen zu können.

Beispiel:

```
START:     CLR D1          * X-Koordinate
           MOVE #128,D2    * Y-Koordinate
SCHLEIFE:  LEA FIG1,A0     * Figur 1, Adresse
           BTST #3,D1      * =0, alle 8 Punkte
           BEQ.S SCHL1     * dann FIG1 ausgeben
           LEA FIG2,A0     * sonst FIG2
SCHL1:     MOVE #4,D0      * Groesse
           JSR @FIGUR      * dann ausgeben
           MOVE #2-1,D4    * besser 40ms warten
SY1:       JSR @SYNC       *
           BEQ SY1         *
           DBRA D4,SY1     * und wiederholen
           ADD #2,D1       * um zwei Schritte wandern
           CMP #512,D1     * bis Bildende
           BCS SCHLEIFE    * wiederholen
           RTS
FIG1:      DC.B 0,0,0,0,1,2,2,4,4,0,0,2,2
           DC.B 3,4,4,8,6,9,6,4,2,0,8,2,9
           DC.B 4,4,5,6,6,6,6,7,10
FIG2:      DC.B 0,0,0,0,1,2,4,4,2,2,0,0,2
           DC.B 3,4,4,8,6,9,6,4,2,0,8,2,9
           DC.B 4,4,5,6,6,6,6,7,10
```

Achtung! Man sollte die Figuren durch den Wert in D0.W nicht zu groß machen. Wenn man mit einem Wert größer als 3 arbeitet wird die Ausgabe wieder langsam, da das Programm FIGUR dies durch aneinandersetzen einzelner Kurzvektoren macht. Werte zwischen 1 und 3 werden aber durch den GDP vergrößert, was sehr schnell geht. Daher lieber die Figur aus mehreren Vektoren zusammensetzen und möglichst eine Größe zwischen 1 und 3 wählen. Eigene Experimente schaffen aber schnell Aufschluß.
Muß man sehr große Gebilde bewegen ist es besser die Routinen MOVETO und DRAWTO zu verwenden und das Löschen selbst zu verwalten.

SETFIG 58

Damit kann man eine zuletzt mit FIGUR gezeichnete Figur auf dem Bildschirm fixieren. Wenn man anschließend FIGUR mit neuen Parametern aufruft, verschwindet die Figur nicht von dem Bildschirm.

Beispiel:

```
START:  LEA FIG1,A0     * Adresse Figur laden
        MOVE #4,D0      * Groesse
        MOVE #10,D1     * X=10
        MOVE #100,D2    * Y=100
        JSR @FIGUR      * Figur zeichnen
        JSR @SETFIG     * Figur fixieren
        LEA FIG1,A0     * nochmals
        MOVE #10,D0     * neue Groesse
        MOVE #100,D1    * X=100
        MOVE #180,D2    * Y=180
        JSR @FIGUR      * neue Figur
        RTS
FIG1:   DC.B 0,1,2,3,4,5,6,7,10
```

Nach Ausführung des Programms sind zwei Figuren auf dem Bildschirm sichtbar.
Wenn man eine Figur löschen möchte ohne eine neue zu setzen, so gibt man einfach 0 im Register D0.W an, bevor man FIGUR aufruft.

GETRAM 59

Dies ist ein Befehl, der einen Speicherbereich ermitteln kann in dem sich RAM befindet. Die Suche nach RAM beginnt bei der Adresse, die im Register A0 steht. Dabei steht nach dem Aufruf im Register A0 die Adresse der nach dem Bereich folgenden Nicht-Ram-Zelle und im Register A1 die Adresse der ersten gefunden RAM-Zelle. Ein Carry wird geliefert, falls kein RAM-Bereich gefunden wurde.
Bevor die Suche beginnt, wird die angegebene Adresse auf eine 1K-Seitengrenze gebracht. Die Suche erfolgt bis zum Bereich 16MByte-64K.
Der Inhalt des geprüften Bereichs wird nicht zerstört.
Wenn man alle RAM-Bereiche erfahren will, so muß man GETRAM solange aufrufen bis ein Carry erscheint. man mekr sich dazwischen jeweils die Speicherbereich. A0 steht nach jedem Aufruf so, daß es vom nächstn GETRAM wieder direkt verwendet werden kann.

Beispiel:

```
START:  LEA $8000,A0    * Suchen vom RAM-Anfang aus
        JSR @GETRAM     * A0,A1 sind Ergebnis
        RTS
```

Mit der Einzelschrittfunktion kann man das vorherige Ergebnis überprüfen.
Dazu startet man das Programm im Einzelschritt (vorher im OPTIONEN-MENUE Debug anschalten). Nach dem Aufruf des Befehls GETRAM erscheinen in den Registern A0 und A1 die Werte.

Beispiel:

```
START:    JSR @CLRSCREEN * Bildschirm loeschen zuvor
          LEA $8000,A0   * komfortable Version
SCHLEIFE: JSR @GETRAM    * Alle Bereiche ausgeben
          BCS ENDE       * Carry, dann Ende
          MOVEM.L A0/A1,-(A7)
          MOVE.L A0,D4   * retten
          LEA BUFFER,A0  * Bereich ausgeben
          MOVE.L A1,D0   * Start
          JSR @PRINT8X   * sedezimal ausgeben
          MOVE.B #'-',(A0)+ * bis
          MOVE.L D4,D0   * Endadr+1
          SUBQ.L #1,D0   * letzte RAM-Zelle
          JSR @PRINT8X   * auch sedezimal
          MOVE.B #$D,(A0)+  * CR
          MOVE.B #$A,(A0)+  * LF
          CLR.B (A0)     * Ende
          LEA BUFFER,A0  * Fuer CO2-Ausgabe
SCHL2:    MOVE.B (A0)+,D0
          BEQ.S SKIP     * =0 dann Ende
          JSR @CO2       * sonst ausgeben
          BRA.S SCHL2    * usw.
SKIP:     MOVEM.L (A7)+,A0/A1
          BRA SCHLEIFE
ENDE:     RTS
BUFFER:   DS.B 100
```

Auch mit nur einem RAM–Baustein werden mehrere Adressen ausgeben. Dies kommt daher, daß die Dekodierung beim 68008 nicht eindeutig ist und der RAM–Speicher auf mehreren Adressen anspricht.
Bei der CPU–Karte 68K mit dem 68000 ist es ähnlich, jedoch ist hier die Dekodierung bis auf 2Mbyte ausgeführt und daher erscheinen nur halb soviele doppelte Bereiche, wie beim 68008.

AUTOFLIP 60

Das Unterprogram steuert die automatische Bildseitenumschaltung. Dabei ist die Umschaltrate von den Variablen FLIP und FLIP1 abhängig. Die Variablen kann man mit dem Befehl SETFLIP belegen. Sie werden aber auch automatisch durch die Schildkrötenbefehle besetzt. Alle 20 ms werden die Variablen abgefragt. FLIP steuert dabei die Umschaltung zwischen zwei Bildseiten. ist FLIP=1, so wird alle 20ms gewechselt. Dabei wird entweder zwischen Seite 0 und 1 oder zwischen Seite 2 und 3 umgeschaltet, je nach dem welche Leseseite zuvor aktiv war. Die Schreibseite wird nich beeinflußt. Wenn FLIP1 = 1 ist und FLIP = 0, so wird alle 20ms zwischen den Vier Bildseiten umgeschalten. Dabei wird von 0 auf 1, von 2 und 3 und von 3 auf 0 geschaltet. Ein Wert größer 1 bewirkt eine Multiplizierung mit jeweils 20ms. Also im Sekundentakt umschalten bedeutet FLIP = 50 oder FLIP1 = 50.
AUTOFLIP wird sonst z.B. in CI und bei Anwendung der Schildkrötenbefehle automatisch aufgerufen. Man muß AUTOFLIP nur dann verwenden, wenn man z.B. in einer großen Schleife Berechnungen durchführt oder lange Zeit nicht auf CI oder die Schildkrötenbefehle zurückgreift, aber dennoch eine Seitenumschaltung erreichen möchte.

Beispiel:

```
START:     MOVE #100,D0    * Schreite 100
           JSR @SCHREITE   * Damit FLIP=1
SCHLEIFE:  JSR @AUTOFLIP   * Bildwechsel aufrecht erhalten
           BRA SCHLEIFE
```

Die Schildkröte bleibt weiterhin eingeblendet. Wenn man AUTOFLIP nicht verwendet, so ist nur eine Seite sichtbar, jedoch ist nicht einmal definiert, welche Seite. Mit NEWPAGE könnte man aber zumindest das noch definieren.
Das Programm ist übrigends nur durch RESET zu stoppen.

CURSEIN 61

Einschalten des Cursors. Normalerweise wird dies automatisch durchgeführt, wenn man die Routine CI aufruft. Wenn man aber eine große Schleife durchläuft, und will den Cursor auf dem Bildschirm sehen, so muß man diese Routine verwenden. Bevor ein Aufruf von CO,CO2 oder CI erfolgt muß man aber den Cursor mit CURSAUS wieder ausschalten. Ferner muß man mit AUTOFLIP die Seiteumschaltung aufrechterhalten (in DELAY z.B. wird automatisch AUTOFLIP aufgerufen).

Beispiel:

```
START:     JSR @CLRSCREEN      * Cursor aktivieren
           JSR @CURSEIN        * Cursor einblenden
           MOVE #100,D0        * 10 Sek anlassen
           JSR DELAY           * dort AUTOFLIP
           JSR @CURSAUS        * dann ausschalten
SCHLEIFE: BRA SCHLEIFE         * und HALT
```

Wenn man die Schleife wegläßt wird der Cursor wieder sichtbar, denn bei der Rückkehr ins Grundprogramm wird CI aufgerufen und dort der Cursor wieder eingeschaltet. Das obige Programm läßt 10 Sek. einen Cursor auf dem Bildschirm sichtbar werden. Dann wird der Bildschirm dunkel.

CURSAUS 62

Der Cursor, der zuvor mit CURSEIN eingeschaltet wurd, wird wieder abgestellt. Anwendung:

Normalerweise wird bei CSTS kein Cursor auf den Bildschirm gebracht. Es gibt aber Fälle bei denen man in einer Schleife durch CSTS abfragt, ob ein Zeichen angekommen ist und es dann erst mit CI einliest (z.B. das 68000 FORTH von Laboratory Microsystems für CP/M68K tut es so).

Beispiel:

```
START:     JSR @CLRSCREEN      * Bildschirm loeschen
SCHLEIFE: JSR @CURSEIN         * Cursor ein
WARTE:     JSR @AUTOFLIP       * und Seitenumschaltung
           JSR @CSTS           * Taste da?
           BEQ WARTE           * nein, dann warten
           JSR @CURSAUS        *
           JSR @CI             * jetzt Zeichen einlesen
           JSR @CO2            * und ausgeben
           BRA SCHLEIFE        * usw.
```

Wichtig ist, daß man den Cursor ausschaltet, bevor man die CO2-Ausgabe akiviert, da sonst z.B. bei Scroll das alte Zeichen falsch auf dem Bildschirm zurückgeschrieben wird.

CHAR 63

Ausgabe eines Zeichens, das im Register D0.B steht auf den Bildschirm. Dabei wird wie bei CO verfahren, jedoch sind Steuerfunktionen nicht verwirklicht. Damit können alle Zeichen von 0 bis $FF verwendet werden. Das es sich um eine interne Funktion handelt ist aber bei der Anwendung Vorsicht geboten.
Man sollte wenn nicht anders möglich CO oder CO2 verwenden.

Beispiel:

```
START:   MOVE.B #'A',D0
         JSR @CHAR       * Buchstaben ausgeben
         RTS
```

Der Buchstabe "A" erscheint auf dem Bildschirm.

PROGZGE 64

Damit ist es möglich einen Programmierbaren Zeichengenrator zu verwenden. man definiert sich dazu in einer 8x6 Matrix die Zeichen und kann sie mit diesem Befehl in der aktuelle Schriftgröße auf dem Bildschirm ausgeben.
Dazu zeigt das Register A0 auf den Datenblock für das Zeichen, die Ausgabe erfolgt an der aktuellen Koordinatenstelle, die man mit MOVETO zuvor noch einstellen kann.
Im Datenblock wird das Zeichen von links nach rechts abgelegt. Dabei entspricht Bit 7 der untersten Zeile und Bit 0 der obersten Zeile eines Zeichens. Wenn ein Bit auf 1 gesetzt ist, so erscheint es als Punkt hell auf dem Bildschirm.

Beispiel:

```
START:  MOVE #100,D1    * X=100
        MOVE #128,D2    * X=128
        JSR @MOVETO     * Dort anfangen
        JSR @WAIT       * im GDP Groesse
        MOVE.B #$55,$FFFFFF73 * ablegen
        LEA DATA,A0     * Zeichendaten
        JSR @PROGZGE    * Zeichen ausgeben
        RTS
DATA:   DC.B %10011110 * Ohm-Zeichen
        DC.B %11100001
        DC.B %00000001
        DC.B %11100001
        DC.B %10011110
        DC.B %00000000
```

Es soll nicht verschwiegen werden, daß der Bildaufbau dabei relativ langsam erfolgt. Daher sollte man einer vektoriellen Darstellung den Vorzug geben, wann immer sie möglich ist.

ASSEMBLE 65

Damit kann der Assembler als Unterprogramm aufgerufen werden. Dieser Befehl ist insbesondere für Übersetzer (Compiler) interessant, man kann jedoch auch andere Effekte damit erreichen. So ist es möglich Assemblercode als Ergebnis eines Programms zu erzeugen und automatisch zu übersetzen.
Die Übersetzung beginnt beim aktuellen Textstart (STXTXT), der z.B. mit dem Befehl PUTSTX verändert werden kann.

Beispiel:

```
START:  JSR @GETSTX          * D0.L alter Start
        MOVE.L D0,-(A7)      * retten
        MOVE.L #TEXT,D0      * dort beginnen
        JSR @PUTSTX          * zu uebersetzen
        JSR @ASSEMBLE        * durchfuehren
        BSR NEU              * wird erst noch uebersetzt
        MOVE.L (A7)+,D0      * zurueck
        JSR @PUTSTX          * sonst Textanfang falsch
        RTS
TEXT:   DC.B 'ORG NEU',$D,$A
        DC.B ' MOVE #100,D0',$D,$A
        DC.B ' JSR @SCHREITE',$D,$A
        DC.B ' RTS',$D,$A,0
        DS 0
NEU:    DS.W 100             * Hier kommt Code hin
```

Bei Start des Programms wird zunächst der Assembler aufgerufen und dann das als Text angegebene Programm übersetzt und anschließend gestartet.

GETSTX 66

Damit kann man die aktuelle Textadresse ermitteln. Diese Adresse ist in der Variablen STXTXT gespeichert und wird nach dem Aufruf im Register D0.L übergeben.

Beispiel:

```
START:  JSR @GETSTX          * Adresse nach D0.L holen
        MOVE.L D0,-(A7)      * retten
        MOVE.L #NEU,D0       * Neue setzen
        JSR @PUTSTX          * z.B.
        JSR @EDIT            * dann editieren
        MOVE.L (A7)+,D0      * alle Adresse
        JSR @PUTSTX          * wieder zuruecklegen
        RTS
NEU:    DC.B 0               * leerer Text dort
```

Mit GETSTX kann man die ursprüngliche Adresse holen und z.B. retten, wenn man AKTTXT mit PUTSTX verändert.

PUTSTX 67

Damit kann man den Textanfang umdefinieren, genauso wie im OPTIONEN-MENUE. Im Register D0.L steht die neue Adresse. Das Textende wird automatisch gesetzt, dabei gibt eine 0 (Code 00) im Speicher an, wo sich das Textende befindet. Das Programm sucht nach der 0, daher Achtung, wenn man sie vergisst kann die Suche u.U. lange dauern.

Beispiel:

```
START:  MOVE.L #$9800,D0     * neuer Textstart
        CLR.B $9800          * Ende festlegen
        JSR @PUTSTX          * und Start definieren
        RTS
```

Wenn man das Programm startet, so passsiert zunüchst noch nichts aufregendes. erst wenn man anschließend in den Editor zurückgeht, so fehlt der Programmtext. In der unteren Zeile sieht man, das der Textanfang nun auf der Adresse $9800 liegt. Wenn man den alten Programmtext wieder sehen will, so kann man dies z.B. mit dem OPTIONEN-MENUE durch Eingabe eines neuen Textstartes tun, oder man schreibt nochmals das obere Programm, jedoch diesmal mit $9000.

GETORG 68

Damit kann man den Defaultwert des Programmzählers für die Codeablage durch den Assembler abfragen. Im Register D0.L steht dann das Ergebnis.

Beispiel:

```
START:   JSR @GETORG     * Ergebnis im Register D0.L
         LEA BUFFER,A0   * Fuer Ausgabe vorbereiten
         JSR @PRINT8X    * ausgeben in den Buffer
         LEA BUFFER,A0   * dann auf den Bildschirm
         MOVE #$22,D0    * Schriftgroesse
         MOVE #2,D1      * X=2
         MOVE #128,D2    * Y=128
         JSR @WRITE      * und ausgeben
         RTS
BUFFER:  DS.B 100
```

Normalerweise erscheint der Wert $9C00 auf dem Bildschirm. Dorthin wird nämlich der Assemblercode gelegt, wenn man kein ORG-Anweisung gibt.

PUTORG 69

Damit kann man den Defaultwert für Assembler-Codeablage ändern. Im Register D0.L steht dann die neue Adressen. Dies ist z.B. für Übersetzungsprogramme interessant, die automatisch die Adressen für den Code bestimmen sollen.

Beispiel:

```
START:   MOVE.L #ENDE,D0     * neuer Defaultstart
         JSR @PUTORG
         RTS
ENDE:    NOP
```

Wenn man das Programm startet, so passiert noch gar nichts. Wenn man aber dann den Assembler erneut startet, so legt er den neuen Programmcode nicht ab Adresse $9C00 ab, sondern direkt hinter den alten Programmcode, ab Adresse $9C0E. Wenn man dann das Programm nochmals startet, so wird bei der nächsten Übersetzung der Code wieder dahinter abgelegt.

PRINT8D 70 erst ab Version 4.0

(alle Befehle mit einem Index >= 70 sind erst ab Version 4.0 verfügbar)
Damit kann eine 32-Bit-Größe, die im Register D0.L steht, dezimal ausgegeben werden. Die Ausgabe erfolgt dabei ohne Vorzeichen.
Im Register A0 steht die Adresse des Buffers, in den die Zahl, in Form von ASCII-Zeichen abgelegt wird.
A0 zeigt anschließend auf das nächste freie Byte. Dort wird auch eine 0 (Code 00) als Endekennung abgelegt.

Beispiel:

```
START:   LEA BUFFER,A0        * Adresse Zielbuffer
         MOVE.L #$FFFFFFFF,D0
         JSR @PRINT8D
         LEA BUFFER,A0        * Dann Wert ausgeben
         MOVE #$22,D0         * Zeichengroesse.
         MOVE #2,D1           * X-Koordinate
         MOVE #120,D2         * Y-Koordinate
         JSR @WRITE           * und ausgeben
         RTS
BUFFER:  DS.B 12
```

Die Zahl 4294967295 wird auf dem Bildschirm ausgegeben.

PRINTV8D 71

Damit kann eine 32-Bit-Größe, die im Register D0.L steht, dezimal ausgegeben werden. Die Ausgabe erfolgt dabei mit Vorzeichen.
Im Register A0 steht die Adresse des Buffers, in den die Zahl, in Form von ASCII-Zeichen abgelegt wird.
A0 zeigt anschließend auf das nächste freie Byte. Dort wird auch eine 0 (Code 00) als Endekennung abgelegt.

Beispiel:

```
START:   LEA BUFFER,A0        * Adresse Zielbuffer
         MOVE.L #80000000,D0
         JSR @PRINTV8D
         LEA BUFFER,A0        * Dann Wert ausgeben
         MOVE #$22,D0         * Zeichengroesse.
         MOVE #2,D1           * X-Koordinate
         MOVE #120,D2         * Y-Koordinate
         JSR @WRITE           * und ausgeben
         RTS
BUFFER:  DS.B 12
```

Die Zahl -2147483648 wird auf dem Bildschirm ausgegeben.

Beispiel Taschenrechner:

```
START:     LEA BUFFER,A0        * Eingabebuffer
           MOVE #$22,d0         * Groesse
           MOVE #2,D1           * X-Koordinate
           MOVE #100,D2         * Y-Koordinate
           MOVE #30,D3          * Anzahl der Zeichen
           JSR @READ            * und einlesen
           LEA BUFFER,A0        * dann ausrechnen
           JSR @WERT            * in D0.L Ergebnis
           LEA BUFFER,A0        * Dort ausgeben
           JSR @PRINTV8D        * dezimale Zahl
           MOVE #20,D3          * Rest auffuellen
SCHLEIFE:  MOVE.B #' ',(A0)+    * mit Leerzeichen
           DBRA D3,SCHLEIFE     * sonst wird eine Zahl
           CLR.B (A0)           * nicht wieder ganz geloescht.
           LEA BUFFER,A0        * Dann Wert auf Bildschirm.
           MOVE #$22,D0         * Zeichengroesse
           MOVE #2,D1           * X-Koordinate
           MOVE #120,D2         * Y-Koordinate
           JSR @WRITE           * und ausgeben
           BRA START            * wieder zurueck
BUFFER:    DS.B 12
```

Nun kann man Rechenaufgaben, sogar mit Klammerechnung eingeben und sie werden dann nach der Eingabe von CR auf dem Bildschirm ausgerechnet und angezeigt. Dabei erfolgt die Rechnung natürlich nur mit ganzen Zahlen (siehe WERT-Befehl).

MULS32 72

Der 68000/8 besitzt eine eingebaute Multiplikation, die aber nur mit 16–Bit Größen arbeiten kann. Da man für bestimmte Rechnungen aber auch eine 32–Bit–Multiplikation benötigt, wurde diese Routine hinzugefügt. Der Multiplikand wird im Register D0.L übergeben, der Multiplikator im Register D2.L. Das Ergebnis erscheint nach dem Aufruf im Register D0.L und D1.L als Überlaufstelle. Dabei wird die Arithmetik vorzeichenbehaftet ausgeführt, die Zahlendarstellung erfolgt also im Zweierkomplement. Die Überlaufstelle wird nicht angeglichen.

Beispiel:

```
START:  MOVE.L #$3FFFFFFF,D0    * erster Operand
        MOVE.L #$2,D2           * zweiter Operand
        JSR @MULS32             * Ausrechnen
        RTS                     * Ergebnis in D0.L
```

Im Register D0.L erscheint nach dem Aufruf die Zahl $7FFFFFFE. Man kann das Ergebnis z.B. im Einzelschritt überprüfen.

Ab Version 4.0 wird auch die Auswertung der Assembleroperationen und aller Ausdrücke mit 32–Bit vorgenommen, dazu wird dieses Unterprogramm intern verwendet.

DIVS32 73

Der 68000/8 besitzt eine eingebaute Division, die aber nur mit 16 Bit arbeitet. Daher wurde diese Routine dazugenommen, die auch mit 32-Bit-Werten arbeiten kann.
Dazu wird im Register D0.L der Divisior abgelegt, im Register D1.L die Zahl 0 (ist eigentlich Zusatzstelle aber vorzeichenlos), und im Register D2.L steht der Divident. Das Ergebnis der Division steht anschließend im Register D0.L und der Rest der Division im Register D1.L
Die Division wird vorzeichenbehaftet durchgeführt.

Operation: D0.L (D1.L) / D2.L -> D0.L REST D1.L

Beispiel:

```
START:  MOVE.L #$7FFFFFFF,D0     * positive max. Zahl
        CLR.L D1                 * kein Ueberlauf
        MOVE.L #2,D2             * div 2
        JSR @DIVS32              * und ausfuehren
        RTS                      * Ergebnis D0.L, REST D1.L
```

Wenn man das Programm im Einzelschritt durchläuft, so steht nach dem Aufruf im Register D0.L der Wert $3FFFFFFF und im Register D1.L der Wert 1.

Da die Division mit DIVS32 langsamer erfolgt als mit der eingebauten 68000/8 Division (DIVU, DIVS) sollte man bei zeitkritschen Problemen versuchen mit den 16-Bit-Operationen auszukommen.
Die DIVS32-Routine wird mit etwa 1800 Operationen pro Sekunde ausgeführt. Also ca. 552 Mikrosekunden pro Division und die eingebauten Division arbeitet mit etwa 26000 Divisionen pro Sekunde, oder 38 Mikrosekunden pro Befehl.

FLINIT 74

Dieser Befehl ist den Anschluß eine Floppy gedacht. Dazu wird die FLO2-Baugruppe benötigt.
Der Befehl setzt die Steprate des Laufwerks auf den langsamsten Wert und initialisiert eine Spurtabelle, die besagt, daß kein Laufwerk gültig ist. Damit wird beim ersten Zugriff auf das Laufwerk sichergestellt, daß die Routinen keinen Fehler melden.
Die Routine FLINIT wird auch beim Start des Grundprogramms automatisch aufgerufen, so daß man sie normalerweise nicht benötigt, wenn man mit der Floppy arbeiten will.

Beispiel:

```
START:      JSR @FLINIT     * Voreinstellung durchfuehren
            RTS             * Ende
```

Achtung! Der Befehl FLINIT ist nicht zur Formatierung der Diskette verwendbar. Zur Formatierung von frischen Disketten benötigt man ein eigenes Programm, das nicht im Grundprogramm (aus Platzmangel) enthalten ist.

FLOPPY 75

Diese Routine ist für den Betrieb mit der FLO2-Baugruppe bestimmt. Damit kann man die Operationen: Steprate einstellen, Lesen und Schreiben durchführen.

Registerbelegung:

```
A0      Ist die Adresse fuer Ziel oder Quelle
D1.W    Befehlscode:   0=Steprate in D3 setzen
                       1=Lesen
                       2=Schreiben
D2.B    Sektor:
                1...n  (n=16,18,26 je nach Laufwerk)

D3.B    Spur:
                0...k  (k=34,39,76,79 je nach Laufwerk)

        Bei Befehl=0 (Steprate setzen) steht hier die
        Steprate 0=schnellste Rate, 3=Langsamste Rate und
        Bit7=SSO Mode, wenn=1, dann wird das SSO-Bit gesetzt,
        wenn man auf die Rueckseite zugreift. Damit wird
        auch die Seite vom Controller auf 1 geprueft.

D4.B    Laufwerk und Dichtecode.
        7   6   5   4   3   2   1   0
        sso mot min dens d   c   b   a
        1=Seite 1 anwaehlen (mit oder ohne Seitenbit)
            0=Motor ein
                1=Minilaufwerk (Mikrolaufwerk)
                    1=Einfache Dichte (single dense)
                        x   x   x   x
                        1=Laufwerk an, binaere
                          Codierung ist aber moeglich.
```

Nach dem Aufruf enthält das Register D0.W einen Fehlercode.
D0.W = 0, bedeutet kein Fehler. Gleichzeitig ist das Carry-Flag zurueckgesetzt.
D0.W = FFFF, bedeutet Fehler aufgetreten. Gleichzeitig ist das Carry-Flag gesetzt. Ein Fehler tritt auch auf, wenn ein Schreibschutz auf der Diskette vorhanden ist und ein Schreibzugriff stattfindet.

Bevor der FLOPPY-Befehl benutzt werden kann, muß die Floppy-Diskette formatiert werden, falls sie nicht schon vom Werk aus formatiert wurde. Dazu wird eine Formtierprogramm verwendet, das nicht im Grundprogramm enthalten ist.

Nach dem RESET des Grundprogramms ist die langsamste Steprate eingestellt, die uns für die ersten Versuche genügen soll.

Beispiel

```
START:   LEA BUFFER,A0        * Zieladresse festlegen
         MOVE #1,D1           * Lesen
         MOVE #2,D2           * Sektor 2
         MOVE #0,D3           * Spur 0
         MOVE #$11,D4         * 8"-Laufwerk, einfache Dichte
         JSR @FLOPPY          * Laufwerk A
         RTS
BUFFER:  DS.B 128
```

Der Dichtecode muß dem jeweiligen Laufwerk angepasst werden. Der Buffer ist bei einfacher Dichte 128 Bytes groß, bei doppelter Dichte 256 oder 1024, je nach Formatierung. Das Programm FLOPPY stellt die Größe automatisch anhand der Formatierung auf der Diskette fest und überträgt die richtige Anzahl von Bytes.

Beispiel:

```
START:   LEA BUFFER,A0        * Zieladresse festlegen
         MOVE #2,D1           * Schreiben
         MOVE #2,D2           * Sektor 2
         MOVE #0,D3           * Spur 0
         MOVE #$11,D4         * 8"-Laufwerk, einfache Dichte
         JSR @FLOPPY          * Laufwerk A
         RTS
BUFFER:  DS.B 128
```

Der Dateninhalt aus Buffer wird auf die Diskette geschrieben.

Abschließend noch eine Zusammenstellung gebräuchlicher Dichtecodes für Laufwerk A. Auf dem Laufwerk muß die Laufwerksnummer eingestellt werden. A entspricht auch der Nr 1, D der Nr 4.

```
8 Zoll einfache Dichte:      $11
8 Zoll doppelte Dichte:      $01
5 1/4 Zoll einfache D.:      $31
5 1/4 Zoll doppelte D.:      $21
```

Für 3 Zoll und 3 1/2-Zoll-Laufwerke gibt es jeweils aus dem oberen Beispiel einen gültigen Code.

Wenn man die Rückseite eines Doppelkopflaufwerks ansprechen will, so addiert man einfach den Wert $80 auf den Laufwerks-Dichtecode.

GETFLOP 76

Damit ist es möglich bei einer formatierten Diskette den Dichtecode automatisch zu ermitteln. Dazu wird nur im Register D4.B der Laufwerkscode angegeben und nach dem Aufruf steht im Register D4.B der Laufwerkscode und der Dichtecode, so wie er vom FLOPPY-Befehl verlangt wird. Die Bestimmung erfolgt immer auf Spur 0, weshalb es in manchen Fällen, bei denen die inneren Spuren in einer anderen Dichte beschrieben sind, trotzdem eine manuelle Bestimmung nötig ist. Jedoch ist der Befehl vorwiegend zum automatischen Start eines Programms auf der Spur 0 gedacht.

Beispiel:

```
START:   MOVE #1,D4      * Laufwerk A (Nr 1)
         JSR @GETFLOP    * bestimmen
         LEA BUFFER,A0   * Dann Lesen
         MOVE #1,D1      * Lesen
         MOVE #1,D2      * Sektor 1
         MOVE #0,D3      * Spur 0
         JSR @FLOPPY     * D4 ist gueltig
         RTS
BUFFER:  DS.B 1024       * Sicherheitshalber gross
```

Mit diesem Programm ist es möglich den ersten Sektor einer Diskette zu lesen, unabhängig davon, ob Mini- oder Maxilaufwerk ob einfache oder doppelte Dichte.

Da GETFLOP unter Umständen eine Weile (< 1 Sekunde) braucht, um die Dichte zu ermitteln, sollte man GETFLOP nicht vor jedem FLOPPY-Befehl verwenden, sondern nur einmal am Anfang eines Programms.

Beispiel für Codierung der Laufwerknummern:

Laufwerk A Seite 0	$01
Laufwerk B Seite 1	$82
Laufwerk C Seite 1	$84
Laufwerk D Seite 0	$08

SETXOR 77

Dies ist ein Befehl, der für eine GDP-Erweiterung gedacht ist. Damit ist es möglich die zweite Hälfte des Seitenregisters auf der GDP-Karte zu setzen. Dazu enthält das Register D0.B einen Wert zwischen 0 und F. Beim Aufruf wird auch die aktuelle Lese- und Schreibseite gesetzt.
Der Wert 0 ist die normale Einstellung.

Beispiel:

```
START:  MOVE #1,D0      * XOR-Mode setzen
        JSR @SETXOR     * und ausfuehren
        RTS
```

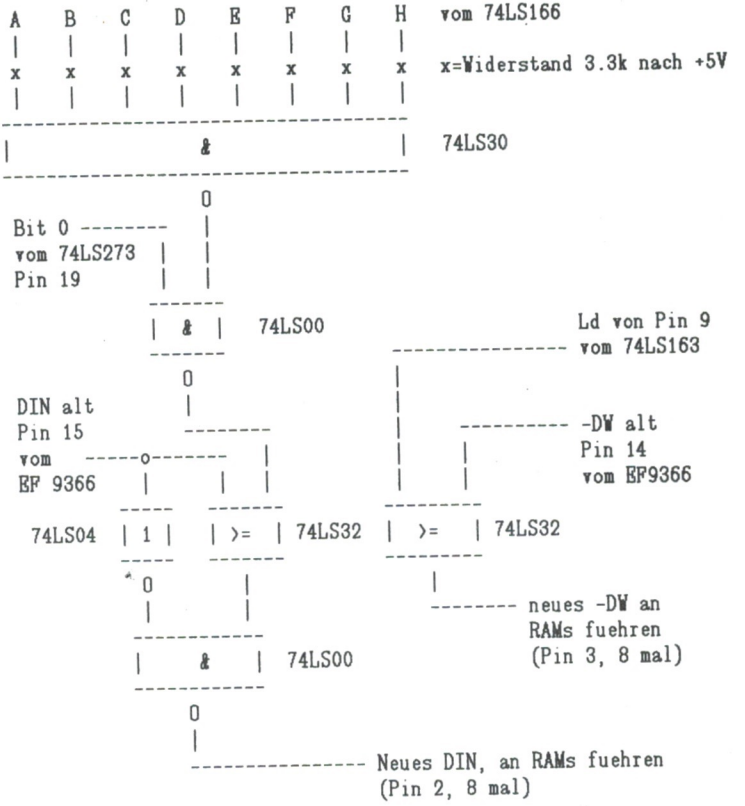
Beim XOR-Mode werden bei Schreibvorgängen alle Punkte komplementiert. Wenn z.B. ein Punkt gesetzt war, so wird er gelöscht, und wenn er nicht gesetzt war, so wird er eingeschrieben. Damit kann man auch einfach bewegte Figuren erzeugen, ohne mit dem AUTOFLIP arbeiten zu müssen. Der Vorteil ist, man erhält ein flimmerfreies Bild. Man muß aber bei der Konstruktion besser aufpassen, da ein doppelt geschriebener Punkt wieder verschwindet und das gilt z.B. für Teile von Linien, die mit DRAWTO oder FIGUR geschrieben werden.

Beispiel:

```
START:  CLR D1          * X=0
        CLR D2          * Y=0
        JSR @MOVETO     * Positionieren
        MOVE #100,D1    * dann Zeichnen
        MOVE #100,D2    * X=100,Y=100
        JSR @DRAWTO     * Linie
        MOVE #1,D0      * Dann XOR-Mode
        JSR @SETXOR     * setzen
        CLR D1          * nun darueber Schreiben
        CLR D2          * X=0, Y=0
        JSR @MOVETO     *
        JSR @WAIT       * Zeichengroesse setzen
        MOVE.B #0,$FFFFFF73 * max Groesse
        MOVE.B #'A',D0 * Buchstabe "A"
        JSR @CMD        * ausgeben
        CLR D0          * Mode zuruecksetzen
        JSR @SETXOR     * und
        RTS             * dann fertig.
```

Der Buchstabe "A" erscheint auf dem Bildschirm. An der Stelle, an der die Linie den Buchstaben durchquert ist sie als dunkle Linie auf hellem Grund sichtbar.

Um den XOR-Mode zu verwenden bedarf es einer Schaltungserweiterung:



Die Leitungen DIN und DW die vom EF9366 sind direkt am IC aufzutrennen, den die Schaltung erzeugt neue Signale, die aus den ursprünglichen gebildet werden. Der Nachbau empfiehlt sich nur für erfahrene Nachbauer. Den 74LS04 kann man auch durch ein freies Gatter des 74LS00 ersetzen.

GETXOR 78

Damit kann man den Wert des eingestellten XOR-Modes wieder auslesen. Der Wert erscheint im Register D0.L und hat einen Bereich von 0 bis F. Siehe auch SETXOR.

Beispiel:

```
START:   JSR @GETXOR     * Wert lesen
         RTS             * Ergebnis im Register D0.B
```

GETXOR und SETXOR arbeiten auch ohne, daß die Schaltungsänderung in der GDP-Karte eingebaut ist, jedoch erfolgt keine Reaktion auf dem Bildschirm. Nur die Werte werden behalten. Da ein Wertebereich von 0 bis F vorliegt, kann man den Portausgang am 74LS273 auch für andere Zwecke verwenden. GETXOR und SETXOR beziehen sich nur auf die unteren vier Bits dieses Ports.

SETCOLOR 79

Ebenfalls für eine Erweiterung ist dieser Befehl gedacht. Ein Wert zwischen 0 und $FF wird im Register D0.B übergeben.
Dieser Wert wird auf den Port $FFFFFFA0 ausgegeben. Die Ausgabe erfolgt erneute bei jedem internen AKTPAGE-Aufruf Dabei ist der Wert 0 als Farbe Weiß gedacht, und so wird er vom Grundprogramm auch nach dem RESET eingestellt. Wenn man selbst eine Farberweiterung bauen will, so empfiehlt sich folgende Konvention, um kompatibel zu bleiben:

Bitnummern des Ports:

```
7     6     5     4     3     2     1     0
ton   wrt   ton   wrt   ton   wrt   ton   wrt
Hintergrund --Blau---   --Gruen--   ---Rot---
```

Wenn das Bit "wrt" auf 0 liegt, so ist die jeweilige Farbe aktiviert.
Wenn das Bit "ton" auf 0 liegt, so wird die Farbe geschrieben, wenn es auf 1 liegt, so wird sie gelöscht. Dabei muß aber SETPEN eingeschaltet sein.
In der Realisation wird "ton" über ein Oder–Glied zum Freischalten des DIN–Eingangs bei den RAMs verwendet und "wrt" zum Freischalten des "DW"–(R/–W)Eingangs der RAMs.
Damit ist es möglich transparente oder deckende Farben zu schreiben. Die Hintergrundfarbe kann z.B. Weiß sein, um die anderen Farbebenen zu überdecken, oder eine Graustufe. Oder es können alle Farbebenen über eine Farbtabelle geführt werden, um beliebige Zuordnungen zu ermöglichen.

Beispiel:

```
START:    MOVE #$FC,D0    * Farbe Rot waehlen
          JSR @SETCOLOR
          RTS
```

GETCOLOR 80

Damit kann der aktuelle Inhalt des Farbregisters ausgelesen werden. Er erscheint im Register D0.L mit dem Wertebereich 0 bis $FF.

Beispiel:

```
START:    JSR @GETCOLOR  * Farbcode holen
          RTS
```

Siehe auch SETCOLOR.
Auch ohne Farberweiterung wird hier der zuletzt gesetzte Farbwert abgeliefert.

CURON 81

Damit wird die interne Variable "CURON" auf 1 gesetzt. Von nun an erscheint ein Cursor bei der Ausgabe durch CO, CO2.
Die Variable "CURON" wird normalerweise auf 1 gesetzt, wenn man CLRSCREEN aufruft.

Beispiel:

```
START:
        MOVE #10,D0     * Flip einschalten
        CLR D1
        JSR @SETFLIP
        JSR @CURON      * Cursor einschalten
        RTS             * wenn er einmal aus war.
```

Auf dem Bildschirm erscheint ein blinkender Cursor. Allerdings ist die Position nicht definiert und in Sonderfällen kann er auch unsichtbar bleiben, den CLRSCREEN wurde in userem Beispiel nicht aufgerufen. Jedoch verwendet der Assembler auch CLRSCREEN und so sind die meisten Variablen schon ok.
CURON wird meist dazu verwendet, um einen ausgeschalteten Cursor wieder einzuschalten.

CUROFF 82

Damit kann der Cursor ausgeschaltet werden, wenn man z.B. bei einer Eingabe keinen Cursor auf dem Bildschirm sehen will. Dazu wird intern die Variable "CURON" auf 0 gesetzt.

Beispiel:

```
START:  JSR @CLRSCREEN      * Bildschirm loeschen
        MOVE.B #'A',D0      * Buchstabe "A" ausgeben
        JSR @CO2
        JSR @CUROFF         * Cursor ausschalten
        RTS
```

Der Buchstabe "A" erscheint links oben, jedoch blinkt der Cursor nicht.
Wenn man CUROFF wegläßt, so blinkt der Cursor rechts neben dem Zeichen.
Man kann die Wirkung von CUROFF auch durch Aufruf von CURON aufheben.

ADJ360 83

Mit diesem Befehl wird ein Wert, der im Register D0.W steht in den Bereich 0 bis 359 gebracht. Die Ausgabe erfolgt über das Register D0.W. Die Eingabe kann dabei auch negative Werte in der Zweierkomplementdarstellung verarbeiten. Dabei ist an die Anwendung von Winkelfunktionen gedacht. Dieser Befehl wird intern verwendet, um die Werte für SIN und COS aufzubereiten.

Beispiel:

```
START:   MOVE #361,D0    * Wert umrechnen
         JSR @ADJ360     * in bereich 0..359
         RTS
```

Im Register D0.W steht nach dem Aufruf der Wert 1.

Die Umrechnung derfolgt durch fortgesetzte Subraktion (bzw. Addition bei negativen Werten), weshalb man nicht allzu große Zahlen als Eingabe verwenden sollte.

PRTSYM 84

Ausgabe der aktuellen Symboltabelle. Die Symboltabelle wird über die CO2-Schnittstelle ausgegeben.

Beispiel:

```
START:   JSR @CLRSCREEN     * CO2 vorbereiten
         JSR @PRTSYM        * Tabelle ausgeben
         RTS                * Ende
```

Nach der Ausgabe wartet PRTSYM auf die Eingabe von "M", erst dann wird im Programm fortgefahren. Daher erscheint der text "F=Flip" und "M=Menue" zweimal.

SYMCLR 85

Damit kann man die Symboltabelle löschen. Dieser Befehl ist wichtig, z.B. für automatische Übersetzer, um Platz für neue Symbole zu schaffen.

Beispiel:

```
START:   JSR SYMCLR      * Symbole loeschen
         RTS
```

Wenn man anschließend das Menue "4 = Symbole" aufruft, so erscheinen keine Symbole mehr auf dem Bildschirm.

GETSYM 86

Im Register A0 und D0.L wird die Adresse des Symboltabellenstarts übergeben. Wenn man keine Symbole verwendet, so kann man z.B. in eigenen Programmen dorthine seine Variable legen. Danach sollte man aber sicherheitshalber mit SYMCLR die Symboltabelle löschen, um keine Fehlinformation zu hinterlassen.
Eine andere Anwendung ist die Manipulation der Symboltabelle, z.B. Werte verändern etc., doch dabei sollte man sehr vorsichtig sein, um nicht die Struktur zu verändern.

Beispiel:

```
VAR1    EQU 0           * Wort
VAR2    EQU VAR1+2      * Langwort
VAR3    EQU VAR2+4      * Wort

START:  JSR GETSYM      * A0, D0.L definieren
        CLR VAR1(A0)    * z.B. erste Variable loeschen
        MOVE.L #$1111AAAA,VAR2(A0)
        MOVE.W #1,VAR3(A0)  *
        ADD.L #1,VAR2(A0)
        .....
        JSR @SYMCLR     * Loeschen, da ungueltig.
        RTS
```

Mit name(A0) wird auf die jeweilige Variable zugegriffen. Bei der Definition der Variablen muß man sehr sorgfältig vorgehen. Die nachfolgende Variable erhält als EQU-Wert den Wert der vorgehenden Variablen plus der Wortbreite in Bytes, ebenfalls von der vorgehenden Variablen.
Achtung! SYMCLR auf keinen Fall weglassen, sonst ergibt sich ggf. ein ADRESSERROR beim nächsten Start des Assemblers, da die Symboltabelle verändert wird.

GETNEXT 87

Damit wird die aktuelle Länge der Symboltabelle ins Register D0.L geladen. Die Länge ist aber auf den Bereich 0..$FFFF begrenzt.
Dieser Befehl kann z.B. zusammen mit dem GETSYM-Befehl dazu verwendet werden einen freien Platz für eigene lokale Variable zu bestimmen, ohne die Symboltabelle zu gefährden.

Beispiel:

```
VAR1    EQU 0
VAR2    EQU VAR1+2

START:  JSR @GETSYM     * A0, D0.L Symboltabellenstart
        JSR @GETNEXT    * D0.L Laenge
        ADDA.L D0,A0    * A0 zeigt auf freien Eintrag
        CLR.W VAR1(A0) * Nun Variable verwendbar
        MOVE.L #1,VAR2(A0)
        ....
        RTS             * ohne SYMCLR ok.
```

Die Variablen werden mit EQUs festgelegt und erhalten als Wert den Abstand zur Basisadresse. Dazu muß man jeweils die nächste Variable mit dem Wert der vorhergehenden plus die Anzahl der durch die vorhergehende Variable belegten Bytes bilden.

PUTNEXT 88

Damit kann man die interne Variable NEXTSYM belegen. Dieser Befehl ist für die Anwendung von Übersetzern gedacht und sollte mit großer Sorgfalt verwendet werden. Im Register D0.W steht der neue Wert, der als Offset zum Symboltabellenanfang verwendet wird. Wenn man diesen Wert verändert, so wird beim Erzeugen von neuen Symbolen von da an angefangen neue Symbole abzulegen.

Beispiel:

```
START:  JSR @GETNEXT    * alten Wert holen nach D0.L
        ADD #10,D0      * 10 Bytes freihalten
        JSR @PUTNEXT    * und neu ablegen
        RTS
```

GETBASIS 89

Da man das Grundprogramm ab Version 4.0 verschieben kann, gibt es einen Befehl, mit dem man die aktuelle Anfangsadresse des Grundprogramms ermitteln kann. Das Ergebnis erscheint in Register A0 und D0.L.
Der Wert ist normalerweise = 0, wenn die ROMs auf Adresse 0 anfangen.

Beispiel:

```
START:    JSR @GETBASIS
          RTS
```

Im Register A0 und D0.L wird die Anfangsadresse des Grundprogramms übergeben. Dies ist nicht mit der Startadresse des Programms zu verwechseln. Denn auf der Anfangsadresse befindet sich der erste Eintrag des ROM-Satzes.

Die Anwendung des GETBASIS-Befehl ist z.B. in CP/M68K-Umgebung interessant.

GETVAR 90

Damit kann man die vom Grundprogramm verwendete RAM-Startadresse ermitteln. Dabei wird der Wert im Register A0 und D0.L übergeben.
Der Wert ist normalerweise $8000. Nur wenn das Grundprogramm auf einer anderen Adresse sitzt, wird der Wert davon abweichen. Dies ist für die CP/M68K-Umgebung interessant.

Beispiel:

```
START:    JSR @GETVAR     * Adresse laden
          RTS
```

Nach dem Aufruf steht im Register A0 und D0.L die verwendete RAM-Anfangsadresse. Dort werden alle Variable des Grundprogramms abgelegt.

SETA5 91

Im Grundprogram der Version größer 4.0 ist alles verschiebbar. Im Register A5 wird normalerweise die Startadresse des vom Grundprogramm verwendeten RAM-Bereichs abgelegt. Man darf daher das Register A5 normalerweise nicht für eigene Zwecke verwenden. Wenn man mit dem TRAP-Befehl arbeitet, so wird das Register A5 automatisch zurückgestellt, wenn man aber JSR verwendet, so verläßt sich das Grundprogramm darauf, daß der Inhalt von A5 gültig ist. Wenn man A5 verändert, so kann man mit dem Aufruf SETA5 den Wert wieder restaurieren.

Beispiel:

```
START:   JSR @SETA5      * Register A5 belegen
         RTS
```

AUFXY 92

(Schildkrötengraphik)
Damit kann man auf eine absolute Koordinate positionieren. Dabei wird eine Linie zur neuen Koordinate gezogen, wenn SENKE aktiv ist.
Im Register D1.W steht die X-Koordinate im Bereich von 0..511 und im Register D2.W die Y-Koordinate ebenfalls im Bereich von 0..511 (nicht wie bei MOVETO).
Die Schildkröte wird am Endpunkt eingeblendet, die Richtung der Schildkröte wird nicht verändert.

Beispiel:

```
START:   MOVE #10,D1     * X=10
         MOVE #100,D2    * Y=100
         JSR @AUFXY
         RTS
```

Auf dem Bildschirm erscheint eine Linie, die nach links unten geht. Die Schildkröte zeigt aber nach oben.

KORXY 93

(Schildkrötengraphik)
Damit kann man den aktuellen Standpunkt der Schildkröte ermitteln. Im Register D1.W wird die X-Koordinate geliefert, um im Register D2.W die Y-Koordinate. Dabei wird beidesmal der Wertebereich 0..511 verwendet.

Beispiel:

```
START:   JSR @FIRSTTIME      * Schildkroete definieren
         JSR @KORXY          * Startpunkt
         ADD #100,D1         * X+100
         ADD #100,D2         * Y+100
         JSR @AUFXY          * neue Position
         RTS
```

Der Aufruf von FIRSTTIME ist wichtig, denn sonst sind die Koordinaten nicht definiert. es genügt natürlich auch ein Aufruf eines anderen Schildkrötenbefehls, z.B. SCHREITE, DREHE.

AUFK 94

(Schildkrötengraphik)
Die Schildkröte wird auf die angegebene Richtung im Register D0.W gestellt. Dabei wird der absolute Winkel in Grad angegeben.
Bei 0 Grad zeigt die Schildkröte nach rechts und bei 90 Grad zeigt sie nach oben.

Beispiel:

```
START:  MOVE #90,D0
        JSR @DREHE      * Dummy zum Test
        MOVE #180,D0
        JSR @AUFK       * nun absolut stellen
        RTS
```

Nach dem Aufruf zeigt die Schildkröte nach rechts.

komplexes Beispiel:

```
START:    CLR D5          * Winkelzaehler
          MOVE #240-1,D6  * Zaehler
SCHLEIFE: MOVE D5,D0      * Winkel 1
          MULS #10,D0     * * 10 Grad
          JSR @AUFK       * stellen
          MOVE #20,D0     * Schreite 20
          JSR @SCHREITE
          MOVE D5,D0      * Winkel 2
          MULS #-15,D0    * * -15 Grad
          JSR @AUFK       * stellen
          MOVE #20,D0     * Schreite 20
          JSR @SCHREITE
          ADD #1,D5       *
          DBRA D6,SCHLEIFE
          RTS
```

Nach dem Start ergibt sich eine recht komplexe Figur. Durch Variation der Winkel und Schrittzahlen kann man sehr unterschiedliche Gebilde erzeugen.

GETK 95

(Schildkrötengraphik)
Damit läßt sich die aktuelle Bildrichtung der Schildkröte ermitteln.
Im Register D0.W wird der Wert in Grad übergeben. Der Bereich ist dabei 0..359 Grad.

Beispiel:

```
START:  MOVE #10,D0     * Schildkroete definieren
        JSR @DREHE
        MOVE #15,D0
        JSR @DREHE
        JSR @GETK       *
        RTS
```

Im Register D0.W steht der dezimale Wert 115 oder sedezimal $73. Denn der Winkel der Endposition setzt sich aus der Grundposition (90 Grad, Schildkröte zeigt nach oben) und den beiden Werten des DREHE-Befehls (10 Grad und 15 Grad) zusammen.

RND 96

Ein Zufallszahlengenerator. Im Register D0.W übergibt man den Bereich. Es wird als Ergebnis eine Zahl zwischen 0 und n-1 ebenfalls im Register D0.W zurückgeliefert. Wenn z.B. Im Register D0 der Wert 6 übergeben wurde, so erhält man eine Zufallszahl zwischen 0 und 5.

Beispiel:

```
START:  MOVE #512,D0    * 0..511 ist Bereich
        JSR @RND        * des Ergebnis
        MOVE D0,D1      * als X-Koordinate ablegen
        MOVE #256,D0    * 0..255 ist Bereich
        JSR @RND        * des Ergebnis
        MOVE D0,D2      * als Y-Koordinate ablegen
        JSR @MOVETO     * Dorthin bewegen
        MOVE.B #$80,D0  * Punkt-Setz-Befehl
        JSR @CMD        * und ausfuehren
        JSR @CSTS       * bis Taste gedrueckt wird
        BEQ START       * wiederholen
        RTS             * dann Ende
```

Nach Start des Programms erhält man ein gleichförmiges Punktraster auf dem Bildschirm.
Nach und nach werden alle Lücken des Rasters gefüllt.

GETVERS 97

Damit wird die auf Adresse $40C enthaltene Versionsnummer des Grundprogramms im Register D0.L übergeben. Version 4.2 liefert zum Beispiel den Wert $420.
Dieser Befehl ist wichtig, wenn man Übersetzer konstruiert, die von Spezialitäten des Systems Gebrauch machen. Damit kann man den Änderungszustand feststellen.

Beispiel:

```
START:  JSR @GETVERS    * Versionsnummer nach D0.L
        RTS
```

GETSN 98

Damit wird der Wert auf Adresse $410 ausgelesen und im Register D0.L abgelegt. Die Seriennummer ist für kommerzielle Anwendungen gedacht, bei der Software geschützt werden soll. Für den normalen Betrieb ist dieser Aufruf ohne Bedeutung.

Beispiel:

```
START:  JSR @GETSN      * Seriennummer nach D0.L
        RTS
```

CRLF 99

Damit wird über die Routine CO2 ein Zeichen CR (Code $d) und ein Zeichen LF (Code $a) ausgegeben. Dadurch wird der Cursor an den nächsten Zeilenanfang gestellt. Wenn der Cursor auf der untersten Zeile stand, so wird der Bildschirminhalt um eins nach oben verschoben (Scroll).

Beispiel:

```
START:  JSR @CLRSCREEN      * vorbereiten fuer CO2-Ausgabe
        MOVE.B #'A',D0      * Buchstabe "A" ausgeben
        JSR @CO2            *
        JSR @CRLF           * dann Vorschub
        MOVE.B #'B',D0      * und in naechste Zeile
        JSR @CO2            * den Buchstaben "B"
        RTS
```

Der Buchstabe "B" steht direkt unter dem Buchstaben "A".

GETLINE 100

Damit ist es möglich auf den Text in einer Zeile des Bildschirmspeichers zuzugreifen. Dazu wird im D0.B die Zeilennummer von 0 bis 23 gewählt. 0 ist dabei die oberste Zeile. Die aktuelle Cursorposition in X-Richtung bestimmt dann noch eine Zeichenstelle.
Im Register A0 steht die Adresse des Zeilenanfangs. Im Register A1 steht die Adresse, die auf eine Speicherzelle zeigt, die die aktuelle Zeilenlänge enthält (.B-Wert). Im Register A2 steht die Adresse des Zeichens, das auf der Spalte des Cursors sitzt.
Mit diesem Befehl kann man z.B. einfach eine Bildschirm orientierte Eingabe für Programmiersprachen, Formularen oder Steuerungen realisieren.

Beispiel:

```
START:    JSR @CLRSCREEN       * CO2 vorbereiten
WDH1:     JSR @CI              * Zeichen holen
          CMP.B #3,D0          * CTRL-C bricht ab
          BEQ ENDE
          CMP.B #4,D0          * CTRL-D macht Hardcopy
          BEQ HARD             * und Start des Progs.
          JSR @CO2             * Sonst ausgeben
          BRA WDH1             * und wiederholen
HARD:     CLR D4               * Zaehler fuer X
SCHLEIFE: MOVE D4,D0           * 0..23
          JSR @GETLINE         * A0 verwenden
          CLR D3               * Zaehler
          MOVE.B (A1),D3       * aufbereiten
          BRA.S SCHL3          * wenn =0, dann nicht
SCHL2:    MOVE.B (A0)+,D0      * Zeichen holen
          JSR @LO              * auf Drucker ausgeben
SCHL3:    DBRA D3,SCHL2        * bis Zeile gedruckt
          MOVE.B #$D,D0        * CR LF ausgeben
          JSR @LO
          MOVE.B #$A,D0
          JSR @LO
          ADDQ #1,D4           * naechste Zeile
          CMP #24,D4           * 0..23
          BNE SCHLEIFE         * wiederholen,
          BRA START            * dann zurueck.
ENDE:     RTS
```

Nach Start des Programms erscheint der Cursor links oben. Nun kann man beliebig auf dem Bildschirm Texte schreiben. Dazu kann man auch die Sonderfunktionen des CO2-Befehls (siehe CO2) verwenden.
Am Schluß gibt man CTRL-D (Erst die Taste CTRL drücken, dann dazu die Taste D) ein. Der Bildschirminhalt wird auf den Drucker ausgegeben.

GETCURXY 101

Im Register D1.L steht der Wert der X-Position (0..79) des Cursors und im Register D2.L der Wert der Y-Position (0..23). Die Koordinate 0,0 ist dabei links oben auf dem Bildschirm.

Beispiel:

```
START:  JSR @CLRSCREEN      * Bildschirm loeschen
WDH1:   JSR @CI             * Zeichen holen
        CMP.B #3,D0         * CTRL-C bricht ab
        BEQ ENDE
        CMP.B #4,D0         * CTRL-D macht Hardcopy
        BEQ HARD            * und Start des Progs.
        JSR @CO2            * Sonst ausgeben
        BRA WDH1            * und wiederholen
HARD:   JSR @GETCURXY       * D1=X, D2=Y
        MOVE D2,D0          * aktuelle Zeile
        JSR @GETLINE        * Nur diese Zeile lesen
        CLR.B 79(A0)        * Ende-Kennung fuer WRITE
        CLR D1              * X=0  | zerstoert aber letztes
        CLR D2              * Y=0  | Zeichen auf Pos. 79
        MOVE.B #$21,D0      * Schriftgroesse
        JSR @WRITE          * Ausgaben in unterster Zeile
        BRA WDH1            * Zurueck dann
ENDE:   RTS
```

Wenn man die Tasten CTRL und D drückt, so erscheint die aktuelle Zeile, auf der man sich gerade befindet auf der untersten (25 ten) Zeile, blinkend. Das Blinken kommt daher, weil die Zeile nur in eine Bildseite eingeschrieben wurde.

SETCURXY 102

Damit kann man den Cursor auf eine beliebige Position des Bildschirms setzen. Dazu wird im Register D1.B die X-Position (0 .. 79) übergeben und im Register D2.B die Y-Position (0 .. 23).
Dabei ist die Koordinate 0,0 links oben.

Beispiel:

```
START:  JSR @CLRSCREEN      * Bildschirm loeschen
        MOVE #19,D1         * X=19
        MOVE #3,D2          * Y=3
        JSR @SETCURXY       * und einstellen
        MOVE.B #'A',D0      * Zeichen "A"
        JSR @CO2            * dort ausgeben
        RTS
```

Auf dem Bildschirm erscheint der Buchstabe "A" auf der vierten Zeichen-Zeile von oben nach unten und auf der 20sten Position von links nach rechts, da die Koordinaten von 0 bis n gezählt werden.

GETXY 103

Nach Aufruf der Routine steht im Register D1.W die X-Koordinate und im Register D2.W die Y-Koordinate, der aktuellen Zeichenposition des GDPs. Damit wird direkt der Registerinhalt des Graphikprozessors ausgelesen und das Vorzeichen angeglichen.

Beispiel:

```
START:    JSR @WAIT       * Schriftgroesse setzen
          MOVE.B #$55,$FFFFFF73
          MOVE #6-1,D3    * Anzahl der Zeichen
          CLR D1          * erst mal positonieren
          CLR D2          * X=0,Y=0
SCHLEIFE: JSR @MOVETO     * Position einstellen
          MOVE.B #'A',D0  * Zeichen ausgeben
          JSR @CMD        * mit aktueller Schriftgroesse
          JSR @GETXY      * Position feststellen
          SUB #5*6,D1     * X-Position
          ADD #5*8,D2     * Y-Position
          DBRA D3,SCHLEIFE
          RTS
```

Nach dem Start erscheinen 6 große Buchstaben "A" übereinanderstehend.

SI 104

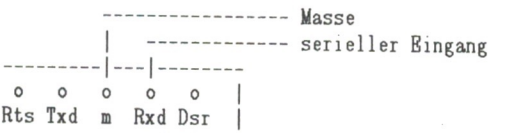
Eingabe eines Zeichens über die serielle Schnittstelle. Es wird ein Zeichen von der seriellen Schnittstelle SER in das Register D0.B gelesen. Das Zeichen kann einen Wert zwischen 0 und $FF haben. Die oberen Bits des Registers D0.L sind auf 0 gesetzt.
Es wird solange gewartet bis ein Zeichen angekommen ist.

Beispiel:

```
START:     JSR @CLRSCREEN      * Bildschirm loeschen
           MOVE.B #$1E,D0      * 9600 Baud, 1Stop, 8Bit
           MOVE.B #$0B,D0      * keine Paritaet
           JSR @SIINIT         * Werte einstellen
SCHLEIFE:  JSR @SI             * Zeichen abholen
           JSR @CO2            * Zeichen ausgeben
           BRA SCHLEIFE        * usw.
```

Wenn man das Programm startet, werden alle ankommenden Zeichen auf dem Bildschirm ausgegeben. Dabei wurde hier eine Baudrate von 9600 Baud eingestellt. Wenn man andere Baudraten haben will, so kann man sie mit dem SIINIT-Befehl einstellen (siehe SIINIT fuer Codierung).
Achtung! Der Cursor blinkt nicht bei diesem Programm. Dazu müßte man die Routinen CURSEIN, CURSAUS und AUTOFLIP verwenden.

Belegung der Stifte:



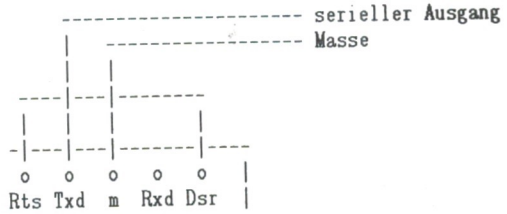
SO 105

Ausgabe eines Zeichens über die serielle Schnittstelle. Das Zeichen steht zuvor im Register D0.B und wird mit Hilfe der SER-Baugruppe ausgegeben. Die Ausgabe erfolgt aber nur, wenn der Eingang DSR auf +12V liegt, sonst wird die Ausgabe gesperrt und das Programm wartet bis die Bedingung erfüllt ist. Der Eingang DSR ist an die Stiftleiste geführt.
Die +12V findet man normalerweise am Ausgang RTS, dessen Pegel man mit SIINIT programmieren kann.

Beispiel:

```
START:     MOVE.B #$1E,D0      * 9600 Baud, 1Stop, 8Bit
           MOVE.B #$0B,D0      * keine Paritaet
           JSR @SIINIT         * Werte einstellen
SCHLEIFE:  MOVE.B #$5A,D0      * Testmuster
           JSR @SO             * ausgeben
           BRA SCHLEIFE
```

Wenn der Eingang Dsr nicht beschaltet wird, so wird die Ausgabe gesperrt. Daher hier die Belegung:



Der Ausgang Rts liefert die +12V, die zur Freigabe des Dsr-Eingangs nötig sind.

SISTS 106

Damit kann man prüfen, ob ein Zeichen an der seriellen Schnittstelle angekommen ist. Die Routine liefert den Wert $FFFFFFFF im Register D0.L, wenn ein Zeichen bereit ist, sonst den Wert 0.
Achtung! Der Status ist kurzzeitig nach einer Ausgabe über SO ungültig.

Beispiel:

```
START:     JSR @CLRSCREEN      * Bildschirm loeschen
           MOVE.B #$1E,D0      * 9600 Baud, 1Stop, 8Bit
           MOVE.B #$0B,D0      * keine Paritaet
           JSR @SIINIT         * Werte einstellen
TERMINAL:  JSR @CURSEIN        * Cursor soll blinken
           JSR @SISTS          * Zeichen da ?
           BEQ TERM1           * nein, dann weiter
           JSR @CURSAUS        * Cursor ausblenden
           JSR @SI             * ja, dann abholen
           JSR @CO2            * und ausgeben
           JSR @CURSEIN        * Cursor wieder darstellen
TERM1:     JSR @CSTS           * Zeichen von der Tastatur da?
           BEQ TERM2           * nein, dann weiter
           JSR @CI             * ja, dann abholen
           JSR @SO             * und ausgeben
TERM2:     JSR @AUTOFLIP       * und Cursor blinken lassen
           BRA TERMINAL        * und so weiter
```

Mit diesem Programm arbeitet der Computer wie ein Terminal. Zeichen die auf der Tastatur eingegeben werden, werden über die serielle Schnittstelle ausgegeben und Zeichen die von der seriellen Schnittstelle kommen, werden auf dem Bildschirm ausgegeben.
Wenn man die Leitungen Rxd mit TxD verbindet und Rts mit Dsr, so kann man auf dem Bildschirm schreiben, denn die Zeichen die man eintippt, erscheinen auf dem Bildschirm. Dies ist auch der beste Test für die serielle Schnittstelle.

Im Register D0.L erscheint der Wert $FFFFFFFF, wenn der Ausgabebuffer des seriellen Bausteins auf der SER-Karte leer ist. Dann ist die serielle Schnittstelle bereit, ein neues Zeichen auszugeben. Sonst erscheint der Wert 0 im Register D0.L.
Dieser Befehl ist wichtig, wenn man eine serielle Ausgabe im Parallelbetrieb zu anderen Operationen ausführen will, ohne den Computer warten zu lassen, bis das vorhergehende Zeichen ausgegeben wurde.

Beispiel:

```
START:     MOVE.B #$18,D0      * 1200 Baud, 1Stop, 8Bit
           MOVE.B #$0B,D0      * keine Paritaet
           JSR @SIINIT         * Werte einstellen
SCHLEIFE:  MOVE #1,D0          * Kreis zeichnen
           JSR @SCHREITE       * mit Schildkroetengraphik
           MOVE #1,D0          * und parallel dazu
           JSR @DREHE          * Zeichen ueber die
           JSR @SOSTS          * SER-Karte ausgeben
           BEQ SCHLEIFE        * ohne Zeitverlust
           MOVE.B #$5A,D0      * Zeichen
           JSR @SO             * ausgeben
           BRA SCHLEIFE        * und wiederholen
```

Wenn man die Befehle "JSR @SOSTS" und "BEQ SCHLEIFE" wegläßt, so wird der Kreis nur sehr langsam durchlaufen, denn dann wartet der Computer jedesmal auf die Beendigung der Ausgabe.
Diesen Effekt kann man aber z.B. bei 9600 Baud nicht mehr so deutlich sehen, denn sobald der Computer zum Durchlauf der Schleife eine ähnlich große Zeit, wie für Übertragungsdauer eines Zeichens, benötigt, arbeitet "SO" wieder verlustfrei. Denn in SO wird zunächst abgefragt, ob das vorhergehende Zeichen abgesendet wurde und dann das nächste Zeichen ausgegeben. Danach wartet die Routine aber nicht, bis das Zeichen auch ausgegeben wurde.

SIINIT 108

Damit wird der Baustein 6551 auf der seriellen Schnittstelle SER programmiert. Man kann so Baudrate, Stopbits, Parität etc. einstellen. Dazu werden zwei Parameter übergeben. Im Register D0.B steht der Wert für Control-Register und im Register D1.B steht der Wert für das Command-Register.
Belegung:

```
Control-Register (D0.B):
     7    6    5    4    3    2    1    0
     -
     0=1 Stop Bit
     1=2 Stop Bits oder
       1 1/2 Stop Bits, wenn 5Bits ohne Paritaet
          ------
          0    0  = 8 Datenbits
          0    1  = 7 Datenbits
          1    0  = 6 Datenbits
          1    1  = 5 Datenbits
                    -
                    0=externer Takt, 1=interner Takt
                         ----------------
                         0000    16x externer Takt
                         0001    50 Baud
                         0010    75 Baud
                         0011   109.92 Baud
                         0100   134.58 Baud
                         0101   150 Baud
                         0110   300 Baud
                         0111   600 Baud
                         1000  1200 Baud
                         1001  1800 Baud
                         1010  2400 Baud
                         1011  3600 Baud
                         1100  4800 Baud
                         1101  7200 Baud
                         1110  9600 Baud
                         1111 19200 Baud
```

```
Command-Register (D1.B):
     7    6    5    4    3    2    1    0
     -----------
     xx0 keine Paritaet
     001 ungerade Paritaet (odd)
     011 gerade Paritaet (even)
     101 gesetzte Paritaet (mark)
     111 ungesetzte Paritaet (space)
                    -
                    0 = normal, 1 = Echo
                         -------
                         00 Interrupt gesperrt, -RTS 1-Signal
                         01 Interrupt frei,     -RTS 0-Signal
                         10 Interrupt gesperrt, -RTS 0-Signal
                         11 Interrupt gesperrt, -RTS 0-Signal
                              und Break auf der Txd-Leitung
                                   -
                                   1=Interrupt sperren
                                        -
                                        0=-DTR auf 1-Signal
                                        1=Rxd und Txd freigeben.
```

Mit diesen beiden Tabellen kann der serielle Baustein programmiert werden.
Dazu werden aus den binären Muster die gewünschten Bytes zusammengesetzt.
Hier noch ein paar gebräuchliche Kombinationen:

```
D0.B      D1.B      Bemerkung
$1E       $B        9600 Baud, 8 Bit, keine Paritaet 1 Stop
$1C       $B        4800 Baud,    "
$1A       $B        2400 Baud,    "
$18       $B        1200 Baud,    "
$98       $B        1200 Baud, 8 Bit, keine Paritaet, 2 Stop
```

Wann immer möglich, sollte man diese Kombinationen bevorzugen, um Kompatibilität unter den NDR-Klein-Computer-Benutzern zu erreichen.

Beispiel:

```
START:    MOVE.B #$1E,D0      * 9600 Baud ...
          MOVE.B #$0B,D1      * Freigabe
          JSR @SIINIT         * und Aufruf
          RTS
```

Im Register D0.B wird die Kanalnummer im Bereich von 0 bis $F angegeben. Damit wird ein Kanal des AD-Umsetzers auf der Baugruppe AD 8 x 16 angesprochen und die Wandlung gestartet. Nach der Wandlung wird das Ergebnis im Register D0.L als 8-Bit-Größe abgeliefert.
Die Wandlung dauert ca. 100 Mikrosekunden.

Beispiel:

```
START:    CLR D6           * Schreibseite
          MOVE #1,D7       * Leseseite
WDH:      CLR D4           * Start bei X=0
          MOVE #16-1,D5    * Y-Achse und Kanalnr.
SCHLEIFE: MOVE D6,D0       * Schreibseite
          MOVE D7,D1       * Leseseite
          JSR @NEWPAGE     * auswaehlen
          MOVE D4,D1       * X-Achse
          JSR @ERAPEN      * Loeschmode
          MOVE #255,D2     * erst alte Linie loeschen
          JSR @MOVETO      * von oben
          CLR D2           * nach unten
          JSR @DRAWTO      * zeichnen, damit bleibt Pos.
          MOVE D5,D0       * Kanal 15..0
          JSR @GETAD8      * anwaehlen
          MOVE D0,D2       * neue Y-Koordinate
          JSR @SETPEN      * diesmal schreiben
          JSR @DRAWTO      * neue Linie
          ADD #20,D4       * neue X-Koordinate
          DBRA D5,SCHLEIFE * alle Kanaele
          EOR #1,D6        * neue Schreibseite
          EOR #1,D7        * neue Leseseite
          BRA WDH          * dann von vorne.
```

Auf dem Bildschirm wird für jeden Kanal eine Linie dargestellt. Es erscheinen 16 Linien mit der jeweiligen Amplitude des Meßwertes.

GETAD10 110

Damit wird ein 10-Bit-Wert in das Register D0.L gelesen. Dazu wird der AD-Umsetzer der Baugruppe AD10 x 1 gestartet.
Da die Baugruppe nur einen Kanal besitzt entfällt die Verwendung einer Kanalnummer. Der Baustein wandelt in ca. 20 Mikrosekunden einen Wert. Bei sehr schnellen Vorgängen empfiehlt sich die Verwendung eines Sample- und Hold-Elementes, um sich schnell änderne Vorgänge während der Meßdauer konstant zu halten.
Achtung! Ggf. sind drei Wait-Zyklen beim 68008 einzuschalten, um die Startpulsbreite zu garantieren, sonst ergeben sich Meßfehler.

Beispiel:

```
START:     JSR @GETAD10    * Wert holen
           LEA BUFFER,A0   * Wert ausgeben
           JSR @PRINT4D    * dezimal
           MOVE #10-1,D3   * Rest auffuellen
SCHLEIFE:  MOVE.B #' ',(A0)+
           DBRA D3,SCHLEIFE
           CLR.B (A0)      * Ende markieren
           LEA BUFFER,A0   * und dann ausgeben
           MOVE.B #$33,D0  * Schriftgroesse
           MOVE #10,D1     * X-Koordinate
           MOVE #120,D2    * Y-Koordinate
           JSR @WRITE      * und ausgeben
           MOVE.L #1,D0    * Warten 1/100 Sek
           JSR @DELAY      * wegen flimmern
           BRA START       * alles von vorn
BUFFER:    DS.B 20         * dorthin Text
```

Damit kann man die Baugruppe DA8 x 2 ansprechen, die zwei Digital/Analog-Umsetzer beinhaltet. Dazu wird im Register D1.B der Wert für den Kanal 0 angegeben und im Register D2.B der Wert für den Kanal 1. Jeder der Umsetzer kann eine 8Bit-Größe verarbeiten. Die Wandlung erfolgt in ca. 800 ns (Nanosekunden).

Beispiel:

```
START:     CLR D1           * Kanal 0
           CLR D2           * Kanal 1
           LEA TAB,A0       * Add-Tabelle
           CLR D3           * Startwert
WDH:       MOVE #255-1,D4   * Zaehler
SCHLEIFE:  JSR @SETDA       * D/A setzen
           ADD 0(A0,D3),D1  * Wert Kanal 0
           ADD 2(A0,D3),D2  * Wert Kanal 1
           DBRA D4,SCHLEIFE
           ADD #4,D3        * Neuer Index
           AND #$F,D3       * 0,4,8,12
           BRA WDH

TAB:       DC.W 1,0,0,1,-1,0,0,-1
```

Wenn man den Kanal 0 auf die X-Ablenkung eines Skops legt und den Kanal 1 auf die Y-Ablenkung, so erhält man ein Quadrat auf dem Bildschirm. Dieses Quadrat entsteht durch die sägezahnförmigen Schwingungen am Ausgang der beiden Kanäle.

komplexes Beispiel:

```
START:     CLR D1           * Kanal 0
           CLR D2           * Kanal 1
           LEA TAB,A0       * Add-Tabelle
           CLR D3           * Startwert
WDH:       MOVE #63-1,D4    * Zaehler
SCHLEIFE:  JSR @SETDA       * D/A setzen
           ADD 0(A0,D3),D1  * Wert Kanal 0
           ADD 2(A0,D3),D2  * Wert Kanal 1
           DBRA D4,SCHLEIFE
           ADD #4,D3        * Neuer Index
           CMP #TABE-TAB,D3 * bis fertig
           BCS WDH
           BRA START        * und neu

TAB:       DC.W 0,2,0,2,2,0,2,0,0,-2,-2,0,-2,0,2,-1,2,-1
TABE:
```

Nach Start des Programms erscheint der Buchstabe "R" auf dem Skop-Bild, wenn man Kanal 0 mit der X-Ablenkung und Kanal 1 mit der Y-Ablenkung verbindet.
Diese Art der Zeichenerzeugung nennt man Vektorscan.
Daneben gibt es aber auch ein Rasterscan-Verfahren, so wie es bei der Erzeugung des Bildes durch den GDP (Graphik Display Prozessor) passiert.
Ein Beispiel für das Rasterscan-Verfahren auf dem Skop sei hier mal als Anregung gezeigt.
Beispiel:

```
START:    LEA TAB,A0        * Start Bildraster
          MOVE #130,D7      * X-Wegwerf.
          MOVE #127,D2      * Y-Position
          MOVE #16-1,D4     * Y-Anzahl
WDH:      CLR D1            * X-Position
          MOVE.W (A0)+,D5   * Wert holen
          MOVE #16-1,D6     * Punkte
SCHLEIFE: BTST #15,D5       * =0, dann dunkel
          BEQ SKIP1         * sonst Punkt darstellen
          JSR @SETDA
          BRA SKIP
SKIP1:    EXG D1,D7
          JSR @SETDA        * Punkt wegwerfen
          EXG D1,D7
SKIP:     ROL #1,D5         * Wert Bitweise verschieben
          ADD #8,D1         * neues X
          DBRA D6,SCHLEIFE  * X-Achse Punkte
          SUB #8,D2         * neues Y
          DBRA D4,WDH       * Y-Achse
          BRA START         * und nochmals
TAB:      DC.W $F3C9        * Bildwiederholspeicher
          DC.W $892A        * Text als Punktraster
          DC.W $892C
          DC.W $F128
          DC.W $A12C
          DC.W $912A
          DC.W $8BC9
          DC.W 0,0,0,0
          DC.W $FFFF
          DC.W $8001,$8001,$8001
          DC.W $FFFF
```

Wenn man das Programm startet, so erscheint ein Text auf dem Skop (Kanal 0 an X und Kanal 1 an Y).

SPEAK 112

Mit diesem Befehl kann die Baugruppe "SPRACHE" bedient werden, die einen Sprachsynthese-IC mit der Bezeichnung SSI 263 verwendet. Bei dem IC handelt es sich um einen sogenannten Phonem-Synthesizer.
Bei diesem Befehl können alle 5 Parameter des Bausteins angegeben werden. Das Register A0 erhält dazu die Adresse des Parameterblocks. Das erste Wort des Parameterblocks enthält die Anzahl der Phoneme. Dann folgend pro Phonem jeweils fünf Bytes mit den Daten.

Beispiel:

```
START:  LEA TAB,A0      * Startadresse laden
        JSR @SPEAK      * und sprechen
        RTS
TAB:    DC.W (ENDE-ANF)/5
ANF:    DC.B 0,$58,$a8,$5c,$ea
        DC.B 0,$58,$a8,$5c,$ea
        DC.B $0a,$68,$d8,$50,$ea
        DC.B $2c,$50,$a8,$5c,$ea
        DC.B $0f,$40,$a8,$52,$ea
        DC.B $0f,$28,$a8,$5a,$ea
        DC.B $e2,$20,$a8,$5c,$ea
        DC.B $22,$20,$a8,$5c,$ea
        DC.B $11,$40,$a8,$5c,$ea
        DC.B $12,$40,$a8,$5c,$ea
        DC.B $d6,$30,$a8,$5c,$ea
        DC.B $00,$38,$a8,$5c,$ea
ENDE:
```

Nach Start des Programms gibt die Sprachkarte das Wort "HALLO" aus. Durch ändern der Parameter kann unterschiedliche Klangfarbe und Betonung erreicht werden (siehe Beschreibung SSI 263).

SPEAK1 113

Mit diesem Befehl kann die Baugruppe "SPRACHE" bedient werden, die einen Sprachsynthese-IC mit der Bezeichnung SSI 263 verwendet. Bei dem IC handelt es sich um einen sogenannten Phonem-Synthesizer.
Bei diesem Befehl kann nur 1 Parameter, das Phonem, angegeben werden. Dadurch ergibt sich eine grobere Aussprache, jedoch benötigt man nur 1 Byte pro Phonem. Die restlichen Parameter werden vom Programm auf Standard-Werte eingestellt.
Das Register A0 erhält beim Aufruf die Adresse des Parameterblocks. Das erste Wort des Parameterblocks enthält die Anzahl der Phoneme. Dann folgend pro Phonem jeweils ein Byte.

Beispiel:

```
START:   LEA TAB,A0      * Startadresse
         JSR @SPEAK1     * ausgeben
         RTS
TAB:     DC.W ENDE-ANF   * Anzahl der Phoneme
ANF:     DC.B 0,0,$2f,6,6,$24,2,$38,0,0
ENDE:
```

Nach Start des Programm spricht der Rechner das Wort "SIEBEN".

SOUND 114

Damit kann man die Baugruppe SOUND bedienen. Dazu wird im Register A0 die Adresse des Parameterblocks übergeben. Dort stehen die 16 Werte, die an das IC AY-3-8912 übertragen werden.

Beispiel:

```
START:   LEA TAB,A0      * Startadresse
         JSR @SOUND      * und ausgeben
         RTS
TAB:     DC.B $DE,1,$DD,1,$BE,0
         DC.B 0,$F8,$10,$10,$10,0,$a,8,0,0
```

Nach Start des Programms erscheint ein rhytmisches Klanggebilde am Ausgang des Lautsprechers.

GETUHR 115

Die Baugruppe UHR kann damit angesprochen werden. Diese Baugruppe enthält den Baustein E050-16, ein Uhren-IC, das nicht nur die Uhrzeit liefert, sondern auch das Datum. Dabei wird das IC durch eine kleine Batterie auf der Baugruppe auch nach dem Spannungsabschalten am Laufen gehalten.
Mit dem Befehl GETUHR wird die Uhrzeit gelesen.
Im Register A0 wird die Adresse eines Buffers übergeben.
Nach dem Aufruf stehen im Buffer die Daten in einem BCD-Format, so daß man die Werte direkt ausgeben kann.

Das Format des Buffers lautet (mit Ausgabebeispiel)

0	1	2	3	4	5	6
Stunden	Minuten	Datum	Monat	Jahr	Tag	Sekunde
$18	$15	$23	$09	$84	$01	$00

Beispiel:

```
START:   LEA BUFFER,A0  * Adresse laden
         JSR @GETUHR    * Uhrinformation laden
         LEA AUSBUF,A0  * Ziel fuer Textausgabe
         MOVE.B BUFFER,D0     * Stunden
         JSR @PRINT2X         * wegen BCD-Darstellung
         MOVE.B #' ',(A0)+    * Leerstelle
         MOVE.B BUFFER+1,D0   * Minuten
         JSR @PRINT2X         * ausgeben
         MOVE.B #' ',(A0)+    * Leerstelle
         MOVE.B BUFFER+6,D0   * Sekunde
         JSR @PRINT2X         * ausgeben
         LEA AUSBUF,A0        * dann ausgeben
         MOVE #$33,D0         * Schriftgroesse
         MOVE #10,D1          * X-Koordinate
         MOVE #120,D2         * Y-Koordinate
         JSR @WRITE           * und ausgeben auf Bildschirm
         JSR @CSTS            * solange bis Taste gedrueckt
         BEQ START            * erst dann aufhoehren
         RTS
BUFFER:  DS.B 7
AUSBUF:  DS.B 20
```

Auf dem Bildschirm wird die Zeit in Stunden, Minuten und Sekunden ausgegeben.
Dabei kann man das Programm natürlich auch ergänzen, um Datum, Jahr und Wochentag auszugeben.

SETUHR 116

Die Baugruppe UHR kann damit angesprochen werden. Diese Baugruppe enthält den Baustein E050-16, ein Uhren-IC, das nicht nur die Uhrzeit liefert, sondern auch Datum. Dabei wird das IC durch eine kleine Batterie auf der Baugruppe auch nach dem Spannungsabschalten am Laufen gehalten.
Mit dem Befehl SETUHR wird die Uhrzeit gestellt.
Im Register A0 wird die Adresse eines Buffers übergeben.
Die Daten werden in dem Buffer übergeben.

Das Format des Buffers lautet:

0	1	2	3	4	5	6
Stunden	Minuten	Datum	Monat	Jahr	Tag	Sekunde
$18	$15	$23	$09	$84	$01	$00

Beispiel:

```
START:   LEA BUFFER,A0        * Adresse laden
         JSR @SETUHR          * Uhr stellen
         RTS
BUFFER:  DC.B $18,$15,$23,$09,$84,$01,$00
```

Nach dem Aufruf, wird der Uhrenbaustein gestellt. Die Zeile bei BUFFER muß natürlich mit den aktuellen Daten gefüllt werden. Anschließend kann man das Ergebnis mit dem Programm des GETUHR-Befehls testen.

Damit kann man pruefen, ob der Drucker das letzte Zeichen schon ausgegeben hat und bereit für den Empfang eines neuen Zeichens ist. Mit diesem Befehl ist es möglich den Drucker im Hintergrundbetrieb laufen zu lassen, und in einem Hauptprogramm etwas anderes zu tun. Wenn der Drucker bereit ist, so wird im Register D0.L der Wert $FF übergeben, sonst der Wert 0.

Beispiel:

```
START:    LEA BUFFER,A0  * Fuer Hintergrundbetrieb
SCHLEIFE: MOVE #1,D0     * Hauptprogramm
          JSR @SCHREITE  * Kreis zeichnen
          MOVE #1,D0     * und drehen
          JSR @DREHE     * dazu
          JSR @LSTS      * Drucker bereit ?
          BEQ SCHLEIFE   * nein, dann zurueck
          MOVE.B (A0)+,D0     * Zeichen holen
          BEQ START      * =0, dann Buffer von vorne
          JSR @LO        * sonst ausgeben
          BRA SCHLEIFE
BUFFER:   DC.B 'Hallo Hintergrundbetrieb des Druckers'
          DC.B ' druckt diesen Text dauernd aus',$D,0
```

Während der Kreis gezeichnet wird, wird auch die Druckausgabe durchgeführt. Wenn man den Drucker anhält, so wird das Kreisprogramm dennoch nicht unterbrochen. Da kein LF (Code: $A) ausgeben wird, erfolgt die Ausgabe immer auf einer Zeile, um Papier zu sparen.

RELAN 118

Damit kann das Relais (falls vorhanden) auf der CAS2-Baugruppe einschalten. Diese Relais kann den Kassettenmotor bedienen. Es wird auch automatisch vom Grundprogramm eingeschaltet, wenn man über die Menues Daten aufzeichnet oder lädt.

Beispiel:

```
START:   JSR @RELAN
         RTS
```

RELAUS 119

Damit kann man das Relais (falls vorhanden) auf der CAS2-Baugruppe ausschalten. Dieses Relais kann den Kassettenmotor bedienen.

Beispiel:

```
START:   JSR @RELAUS
         RTS
```

ASSERR 120

Die Anzahl der Fehler, die bei einer Übersetzung durch den Assembler auftraten werden im Register D0.L übergeben. Die Maximalzahl ist dabei $FFFF. 0 bedeutet, es trat kein Fehler auf. Dieser Befehl wird von höheren Programmiersprachen verwendet, um festzustellen ob Fehler bei einem Assemblerlauf aufgetreten sind und dann das Programm nicht zu starten.

Beispiel:

```
            ORG $9400              * neuer Bereich
START:      JSR @GETSTX            * alter Buffer
            MOVE.L D0,-(A7)        * retten
            MOVE.L #BUFFER,D0      * Textbuffer
            JSR @PUTSTX            * fuers editieren
SCHLEIFE:   JSR @EDIT              * Text eingeben
            JSR @ASSEMBLE          * uebersetzme
            JSR @ASSERR            * wenn Fehler, dann zum
            BNE SCHLEIFE           * Editor zurueck
            MOVE.L (A7)+,D0        *
            JSR @PUTSTX            * alter Textstart
            RTS
BUFFER:     DC.B 0                 * Dorthin neue Quelle
```

Wenn man das Programm startet, so meldet sich der Texteditor mit einem leeren Feld. Man kann nun ein Assemblerprogramm eingeben. Wenn man CTRL-K und dann X betätigt, so wird automatisch der Assembler gestartet, wenn man einen Fehler im Assemblerprogramm hat, so wird automatisch wieder der Texteditor aufgerufen und man kann den Fehler beseitigen. Es meldet sich erst dann wieder das Menue, wenn man einen korrekten Assemblertext eingibt. Wichtig ist die ORG-Adresse am Anfang, man darf sie nicht vergessen, sonst überschreibt das neue Programm, das alte und es gibt Fehler.

```
..A68K 4.0a 06/13/83   ...Run on 01/22/85
                        1.
                        2.
                        3.          LLEN 120
                        4.
                        5. ************************************************
                        6. * 68000 GRUNDPROGRAMM - TEIL A                 *
                        7. * von Rolf-Dieter Klein (C) 1984     V 4.2      *
                        8. * rev 2.0  vom 3.12.1983          RDK          *
                        9. * rev 3.0  vom 29.1.1984          RDK erweiterungen *
                       10. * rev 3.1  vom 28.2.1984          ae,oe,ue in texte.*
                       11. *                                 echosperre bei get*
                       12. *                                 trap erweiterung  *
                       13. *                                 Centronix adr $48 *
                       14. * rev 4.0  vom 3.9.1984           RDK versch. CPUs  *
                       15. *                                 %.figur Korr      *
                       16. *                                 Tabelle an Anfang *
                       17. *                                 turtle era.set    *
                       18. * rev 4.0b vom 8.9.1984           Relokative Version*
                       19. *                                aller Speicher     *
                       20. *                                ueber A5 adressiert*
                       21. *                                FLO2-Routinen      *
                       22. * rev 4.1 840924                                    *
                       23. * rev 4.2 841015     korr CI.CO22B.CTRL-V ...       *
                       24. *                    Freigabe 29.10.1984 V 4.2      *
                       25. * rev 4.3 850117     korr ae,oe ue in Einzelschr. *
                       26. *                    speichern, laden von Daten.  *
                       27. *                    Freigabe 17.1.1985 V 4.3     *
                       28. ************************************************
                       29.
                       30. * Achtung: Hinweis zum Editor. Die Texte
                       31. * koennen maximal 64K lang sein, groessere
                       32. * Texte muessen aufgeteilt werden, da
                       33. * der Editor Transportschleifen mit dem DBRA-Befehl
                       34. * realisiert. Zum Aufteilen verwendet man das
                       35. * Optionen-Menue (TEXTSTART).
                       36.
                       37.                          * CPU-Definition durchfuehren
                       38.  include cou
                       39: ********************************
                       40: * CPU Include File   840930    *
                       41: * wird von GRUND.FLO2.UTIL.CRT *
                       42: * benoetigt um IO-Adressen     *
                       43: * festzulegen                 *
                       44: ********************************
                       45:
00000001               46: cou     equ     1
                       47:
                       48: * 1= CPU 68008       8-Bit-Datenbus
                       49: * 2= CPU 68000/10   16-Bit-Datenbus
                       50: * 4= CPU 68020      32-Bit-Datenbus (32er Mode)
                       51:
                       52:
                       53: *
                       54: * Ende CPU include
                       55: * Achtung, bei Aendern der Datei, alle vier
                       56: * betroffenen Programme neu uebersetzen
                       57: *
                       58:
                       59:
                       60:
                       61:
                       62:
                       63:
00008000               64. ram  equ         $8000    * Ram liegt $8000 ueber Programm
                       65.                           * Dieser Wert liegt als
                       66.                           * Konstante auf der zweiten Zelle
                       67.                           * beginnend bei $400+basis
                       68.                           * Aenderungen sind daher im RAM
                       69.                           * oder EPROM einfach moeglich
                       70.                           * ACHTUNG es aendern sich ggf.
                       71.                           * USER-Trap und Vektoren
                       72. * RAM Speicher std. von 8000 bis 9fff ueber dem
                       73. * Grundprogrammstart. Grundprogramm ist
                       74. * relokativ.
                       75.
                       76. ****** Externe Proceduren
                       77.
                       78.          xref assemble.edit
                       79.          xref flinit.floppy,getfloo
```

```
                80.
                81.
                82.             xref cursorein.cursoraus.co
                83.             xref clrscreen,prtline,getline
                84.             xref aktcur.charhandler.progzge
                85.             xref getcode,putchar
                86.
                87.  ****** Interne Proceduren und Speicherzellen
                88.
                89.             xdef pagecnt.pagebuf,print4d
                90.
                91.             xdef wrtpage,viewpage,curon
                92.             xdef groesse,curx,cury
                93.             xdef lineptr,linecnt.screen
                94.             xdef color,xormode,optflag
                95.             xdef neuzeich.escmerker.tempzeich
                96.             xdef flip1,flip1cnt,flip,flipcnt
                97.  *
                98.             xdef aktpage.setpen,erapen
                99.             xdef cmd.wait,moveto.clrall
               100.             xdef getxy
               101.
               102.             xdef ci.csts
               103.             xdef print6x.textein.textaus.moveto.drawto
               104.             xdef setpen.erapen.aktpage
               105.             xdef carset.carres
               106.             xdef einbuf.ausbuf
               107.             xdef stxtxt,akttxt.etxtxt.eottxt
               108.             xdef insl
               109.
               110.             xdef ci2.co2.ciinit2.prtco2
               111.             xdef expr
               112.             xdef errart.errpoi.errcnt.errflag.pcstand.anfstand
               113.             xdef passflag.befcode
               114.             xdef attcode.wordbyte.iostat.zeilen
               115.             xdef setupname
               116.             xdef bucheck.setval.getval.newval
               117.             xdef igbn
               118.             xdef symtab.symnext.pcorg
               119.             xdef auszahl.auspoi
               120.             xdef clrinvis
               121.
               122.             xdef optflag
               123.
               124.             xdef debug,debugst.debugak
               125.
               126.             xdef offset
               127.
               128.             xdef steprate.fehza.ccode.drvat.drvtab.srsto
               129.
               130.             xref sists.sosts.si.so.siinit
               131.             xref getuhr.setuhr.speak.speak1
               132.             xref sound.getad8.getad10.setda
               133.
               134.  ************************************************
               135.
               136.             include speicher        * Speicherdefinitionen.und stack
               137:
               138: ****************************************
               139: * Insert fuer GRUND.A68 rev 2.0  831203 *
               140: * rev 4.0b  Relokative Grundversion     *
               141: * Speicher befindet sich 8000h Bytes    *
               142: * ueber Grundprogrammstart und wird     *
               143: * indirekt ueber Register A5 adressiert *
               144: * rev 4.2 841029   RDK                  *
               145: ****************************************
               146:
               147: * Speichergebiet fuer Variable etc.
               148:
               149:         offset 0        * start RAM-Gebiet
               150:                         * relokativ
               151:
               152: ****** byte,wort,lang-variable *** User-Verwendbar
00000000       153: vekdest:                 * INIT-STart  fuer transport
               154:                         * dort Sprunge eintragen JMP xxxx
00000000       155: intlv2:        ds.b 6    * Auto-Interrupt LV2 (ltg nmi)
00000006       156: intlv5:        ds.b 6    * Auto-Interrupt LV5 (ltg int)
0000000C       157: intlv7:        ds.b 6    * Auto-Interrupt LV7 (ltg nmi+int)
00000012       158: trap0:         ds.b 6    * TRAP0-Vektor
               159:
```

```
00000018                160: userci:        ds.b 6          * Vektor USER CI. 1=erster Zugriff
0000001E                161: usercsts:      ds.b 6          * Vektor USER CSTS
00000024                162: userco:        ds.b 6          * Vektor USER CO
                        163:
                        164: **** gebiet oberhalb nicht veraendern ohne init: zu beachten
                        165:
0000002A                166: iostat:        ds.b 1          * =1 dann co2 Ausgabe auf CRT
0000002B                167: iostatb:       ds.b 1          * fuer ci2.Umschalter
0000002C                168: iodir:         ds.b 1          * fuer CTRL-A Umschaltung
0002D                   169:                ds.b 1          * Angleich
                        170:
0000002E                171: flio:          ds.b 1          * fuer autoflio
0000002F                172: fliocnt:       ds.b 1
00000030                173: flio1:         ds.b 1
00000031                174: flio1cnt:      ds.b 1
                        175:
                        176:
00000032                177: passflag:      ds.b 2          * Aktueller PASS beim Assembler
00000034                178: errflag:       ds.b 2          * Fehlerflag 0=kein Fehler bei Ass
                        179:
                        180: * editor
                        181:
00000036                182: stxtxt:        ds.b 4          * Zeiger auf Textanfang
0000003A                183: akttxt:        ds.b 4          * Zeiger auf Startzeile des Bildsc
0000003E                184: etxtxt:        ds.b 4          * Zeiger auf Textende
00000042                185: eottxt:        ds.b 4          * Zeiger auf das Ende aller Texte
                        186:
                        187:
                        188: *** lokale Variable und Nicht-Benutzer-Zellen
                        189:
                        190: ****** wort.lang-variable
                        191:
                        192:
00000046                193: turx:           ds.b 2          *x-pos turtle *16
00000048                194: tury:           ds.b 2          *y-pos turtle *32
0000004A                195: turphi:         ds.b 2          *winkel
0000004C                196: turlx:          ds.b 2          *merker
0000004E                197: turly:          ds.b 2
00000050                198: turlphi:        ds.b 2
00000052                199: pcstand:        ds.b 4          *Programmzaehlerstand ASSEM
00000056                200: anfstand:       ds.b 4          *Startadr
0000005A                201: auspoi:         ds.b 4          *Pointer Ausgabe PUTWORD..
0000005E                202: einpoi:         ds.b 4          * Eingabepointer. fuer GET. ci
00000062                203: auszahl:        ds.b 2          *Anzahl ausgegebener Bytes
                        204:
00000064                205: txtmerk:        ds.b 4          * Im Fehlerfall adresse text
                        206:
00000068                207: nametab:        ds.b 8          * Name fuer Symboltabelle
00000070                208: ausbuf:         ds.b 132        * ausgabebuffer :even fuer schnell
000000F4                209: einbuf:         ds.b 132        * eingabebuffer :laden
                        210:
                        211: * trace tabellen
00000178                212: regsave:        ds.b 4*16       * d0-d7 a0-a7
000001B8                213: qdpsave:        ds.b 16         * qdp0..qdp15
000001C8                214: srsave:         ds.b 2          * SR-Register
000001CA                215: pcsave:         ds.b 4          * PC next instruktion
000001CE                216: uspsave:        ds.b 4          * USP
000001D2                217: sspsave:        ds.b 4          * SSP
                        218:
000001D6                219: tracbis:        ds.b 4          * bis merker
000001DA                220: tracnmal:       ds.b 4          * n mal tracen
                        221:
000001DE                222: modber:         ds.b 16         * Modbereich fuer Trace TRAP.JSR
                        223:
                        224: * assembler
                        225:
000001EE                226: offset:         ds.b 4          * fuer verschobene Programme
000001F2                227: pcorg:          ds.b 4          * Default PC Start wenn kein ORG d
000001F6                228: befcode:        ds.b 2          * Obcodespeicher
000001F8                229: attcode:        ds.b 2          * Attribute aus Befehle
000001FA                230: wordbyte:       ds.b 2          * Byte Word Long-Code
                        231:
000001FC                232: errcnt:         ds.b 2          * Anzahl der Fehler
000001FE                233: errart:         ds.b 2          * Fehlerart
00000200                234: errpoi:         ds.b 4          * Zeiger auf Fehlerquelle
                        235:
00000204                236: debugst:        ds.b 4          * Start Debugger-Info
00000208                237: debugak:        ds.b 4          * Naechste freie Zelle
                        238:
0000020C                239: oldx:           ds.b 2          * Fuer figur-Befehl
```

```
0000020E                240: oldy:          ds.b 2          * x,y -Koordinate
00000210                241: oldadr:        ds.b 4          * Adresse der Figur
                        242:
                        243: ****** byte-variable
                        244:
00000214                245: oldsize:       ds.b 1          * Bytewert Adresse
                        246:
                        247:
                        248:
00000215                249: fliosave:      ds.b 1          * trace merker
00000216                250: fliolsave:     ds.b 1
                        251:
00000217                252: viewpage:      ds.b 1          * aktuelle leseseite 0..3
00000218                253: wrtpage:       ds.b 1          * aktuelle schreibseite 0..3
                        254:
00000219                255: synstate:      ds.b 1          * fuer sync aufruf
0000021A                256: synccnt:       ds.b 1          * fuer textein
                        257:
0000021B                258: groesse:       ds.b 1          * Default Groesse bei CO-Ausgabe
0000021C                259: sizesto:       ds.b 1          * fuer textein
                        260:
0000021D                261: first:         ds.b 1          * fuer logo-graphic 1=first
0000021E                262: turdo:         ds.b 1          * turtle down,up=0
                        263:
                        264: ** assembler
                        265:
0000021F                266: zeilen:        ds.b 1          * Zeilenzaehler
00000220                267: debug:         ds.b 1          * Debug flag =1 dann gueltige Info
                        268:
                        269: ** editor
                        270:
00000221                271: insl:          ds.b 1          * =0 std mode kein insert =1 inser
                        272:
                        273: ** crt verwaltung
                        274:
00000222                275: curon:          ds.b 1          *=1 dann crt cursor aktiv
                        276:                                 *wird von ci gesteuert
                        277:
00000223                278: curx:          ds.b 1          * cursor position 0..79
00000224                279: cury:          ds.b 1          * 0..23
                        280:
00000225                281: escmerker:     ds.b 1          *escape flag
                        282:
00000226                283: neuzeich:      ds.b 1
00000227                284: tempzeich:     ds.b 1          *zeichen in charhand
00000228                285: optflag:       ds.b 1          * =0 dann amerikan. Zge
                        286:
00000229                287: linecnt:       ds.b 24         *zeichen pro zeile
00000241                288: lineptr:       ds.b 24         *pointer auf reale zeile
00000259                289: screen:        ds.b 80*24      *bildschirmbuffer
                        290:
                        291: * zwischen screen und stack keine Eintraege
                        292: * Symboltabelle  auf gerader Adresse un letzter Eintrag
                        293: * da dynamisch verwaltet
                        294:
009DA                   295:                 ds      0       *auf even setzen
                        296:
009DA                   297:
                        298:                 ds.b 40         * laeuft  kurzzeitig in
                        299:                                 * screenbereich. dann
                        300:                                 * wird er jedoch umdefinier[illegible]
                        301:                                 * bevor dieser Bereich verw[illegible]t
                        302:                                 * wird.
                        303:                                 * daher nicht kritisch
00000A02                304: stack:           ds.b 2          * auf even
                        305: ********************************************************************
00000A04                306: trap2:          ds.b 6          * Vektor fuer User 2-Trap
00000A0A                307: trap3:          ds.b 6          * Vektor fuer User 3-Trap
00000A10                308: trap6:          ds.b 6          * Vektor fuer User 6-Trap
00000A16                309: trap7:          ds.b 6          * Vektor fuer User 7-Trap
                        310:                                 * dieser Bereich wird nicht
                        311:                                 * initialisiert. um frei stehen
                        312:                                 * zu bleiben. nach RESET etc.
                        313: ********************************************************************
                        314:
                        315:                                 * Davor keine Speicheraenderung
                        316:                                 * seit rev 3.1 FIX
                        317:                                 * Start rev 4.0 /4.2 Variable
                        318:                                 * Bereich von hier ab nicht stabil
                        319:                                 * Verschiebungen koennen sich bei
                                                             * Versionsaenderungen ergeben.
```

```
                                320:                                    * daher nie direkt zugreifen
                                321:                                    * *** Variable fuer FCO2
      00000A1C                  322: srsto:           ds.b 2            * temp Merker fuer Status
      00000A1E                  323: steprate:        ds.b 1            * Merker fuer Steprate
      00000A1F                  324: fehza:           ds.b 1            * Fehlerzaehler
      00000A20                  325: ccode:           ds.b 1            * Merker fuer aktuelles Laufwerk
      00000A21                  326: drvat:           ds.b 1            * Merker fuer altes Laufwerk
      00000A22                  327: drvtab:          ds.b 16           * Sourtabelle Laufwerke
000A32                          328:                  ds 0
      00000A32                  329: rnd1var:         ds.b 2            * RANDOM Indizes
      00000A34                  330: rnd2var:         ds.b 2            * beide zufaellig belegt, Wortgros
      00000A36                  331: rnd3var:         ds.b 2            * fuer Knuth-Algorithmus
      00000A38                  332: pagecnt:         ds.b 2            * Seitenzaehler bei Assembler
      00000A3A                  333: pagebuf:         ds.b 10           * Buffer fuer Seitenummer
                                334:                                    *
                                335:                                    * *** Byte-Variable fuer Erweiteru
      00000A44                  336: color:           ds.b 1            * Aktuelle Farbe       GDP64-C1
      00000A45                  337: xormode:         ds.b 1            * 0..15 XOR-Flags fuer GDP64-C2
      00000A46                  338: ci2flag:         ds.b 1            * 1=erster CI2-Zugriff bei USERCI
                                339:
000A48                          340:                  ds 0
                                341: ********************************************************************
                                342: *** Platz fuer weitere Variable
                                343:                                    * Bereich als Block fest, aber
                                344:                                    * verschiebbar als Ganzes
      000A48                    345: poweron:         ds.b 4            * Poweron Erkennung
      00000A4C                  346: svmnext:         ds.b 2
      00000A4E                  347: svmtab:          ds.b 2            *Bereich beliebig
                                348: ************************************** Keine Eintraege mehr
                                349:
                                350:
                                351:
                                352: **
                                353:
                                354:
                                355:
                                356: ***************************************************
                                357: * ABSOLUTES GEBIET
                                358: * Auf Adresse 0. entweder Grundprogramm selbst auf
                                359: * Adresse 0. oder in einem Boot-Prom + Startoffset
                                360: * des Grundprogramms.
                                361:
000000                          362:          section 1             * Alles Relokativ von hier an
                                363:                                *
000000                          364: basis:                         * Start BASIS des Programms
                                365:                                * Tabelle nur bei 0 gueltig
000000  00008FFE              366:          dc.l $8ffe            * Dummy Stackpointer, muss nicht RAM sein
000004  000033F4              367:          dc.l start-basis      *
                                368:
000008  0000333E              369:          dc.l buserr-basis
00000C  00003346              370:          dc.l adrerr-basis
000010  0000334E              371:          dc.l illins-basis
000014  00003356              372:          dc.l zerdiv-basis
000018  0000335E              373:          dc.l chkins-basis
00001C  00003366              374:          dc.l trapvins-basis
000020  0000336E              375:          dc.l priv-basis
000024  00001FB6              376:          dc.l tracein-basis
000028  00003376              377:          dc.l linea-basis
00002C  0000337E              378:          dc.l linef-basis
000030                        379:          ds.l 24-12
                                380:                                * ab Adresse $60
00003336                        381:          dc.l intlvx-basis     * sourious int
00003336                        382:          dc.l intlvx-basis     * level 1 68000
00008000                        383:          dc.l intlv2+ram       * level 2 68008
00003336                        384:          dc.l intlvx-basis     * level 3 68000
00003336                        385:          dc.l intlvx-basis     * level 4 68000
00008006                        386:          dc.l intlv5+ram       * level 5 68008
00003336                        387:          dc.l intlvx-basis     * level 6 68000
0000800C                        388:          dc.l intlv7+ram       * level 7 68008
                                389:
                                390:                                * TRAP-Befehle
00008012                        391:          dc.l trap0+ram        *0  user trap 0
00003200                        392:          dc.l trap1a-basis     *1  grundprg trap
00008A04                        393:          dc.l trap2+ram        *2  fuer cp/m reserviert
00008A0A                        394:          dc.l trap3+ram        *3  fuer cp/m reserviert
00003200                        395:          dc.l trap1a-basis     *4  grundprg trap unter cp/m
00003200                        396:          dc.l trap1a-basis     *5  reserve.
00008A10                        397:          dc.l trap6+ram        *6  user trap 6
00008A16                        398:          dc.l trap7+ram        *7  user trap 7
00008012                        399:          dc.l trap0+ram        *8 -- frei fuer Anwender --
```

VECTORS

```
00000A4   00008012          400.          dc.l trap0+ram          *9
00000A8   00008012          401.          dc.l trap0+ram          *10
00000AC   00008012          402.          dc.l trap0+ram          *11
00000B0   00008012          403.          dc.l trap0+ram          *12
00000B4   00008012          404.          dc.l trap0+ram          *13
00000B8   00008012          405.          dc.l trap0+ram          *14
00000BC   00008012          406.          dc.l trap0+ram          *15 --      *       --
00000C0                     407.          ds.l 64-48
                            408.
000100   00003336           409.          dc.l intlvx-basis       * Interruots mit Vektor
000104                      410.          ds.l 256-65             * usw. bis 3ff
                            411.                                  * bis 400h
                            412. ************************************************
                            413.
                            414. * Definitionen von IO-Ports etc
                            415. * Multiplikation mit CPU und dadurch auch fuer
                            416. * unterschiedliche Wortbreiten geeignet.
                            417. * beim 68000 z.B. liegen dann alle Ports
                            418. * auf geraden Adressen. beim 68020 auf jeweils der vierten Adresse
                            419. * ACHTUNG. speziell bei GDP sind auch im
                            420. * restlichen Hauptprogramm Aenderungen durchgefuehrt.
                            421. * da gdp mehrere Register besitzt.
                            422.
FFFFFF60                    423. page     equ      cou*$ffffff60   * Seitenanwahl Graphic-Proz[illegible]-mo
FFFFFF70                    424. gdp      equ      cou*$ffffff70   * Graphik Prozessor
FFFFFFA0                    425. colport  equ      cou*$ffffffa0   * z.Z. Farbregister dort la[illegible] re
                            426.
FFFFFF68                    427. kevd     equ      cou*$ffffff68   * daten tastatur + strobe
FFFFFF69                    428. kevs     equ      cou*$ffffff69   * schalter + ruecksetzen bit 7 auf
                            429.
FFFFFFC8                    430. bankboot equ      cou*$ffffffc8   * Bank-Boot-Karte fuer CP/M68K
                            431.                                   * wahlweise auch Boot-Karte
                            432.                                   * verwendbar.
                            433.
                            434. * Prom Programmierports
                            435.
FFFFFF80                    436. promd    equ      cou*$ffffff80
FFFFFF81                    437. proma1   equ      cou*$ffffff81
FFFFFF82                    438. proma2   equ      cou*$ffffff82
                            439.
                            440. * proma2
                            441. * 7    6    5    4    3    2    1    0
                            442. * ena  led  trg  a12  a11  a10  a9   a8
                            443. *
                            444.
                            445. * bei read
                            446. * 0    1    0
                            447. *
                            448. * bei write
                            449. * 1    0    0
                            450. * 1    0    1
                            451. * 1    0    0
                            452. *
                            453. *
                            454.
                            455. * proma1 lesen
                            456. * x    x    x    x    x    x    x    bsv
                            457. * bsv=1 bei programmieren sonst 0
                            458. *
                            459.
FFFFFFCA                    460. cmdcas   equ cou*$ffffffca
FFFFFFCB                    461. datcas   equ cou*$ffffffcb
                            462.
                            463.
                            464. * rev 3.1 Aenderung von 30 auf 48
FFFFFF48                    465. centdaten equ cou*$ffffff48      * Datenport
FFFFFF49                    466. centin equ cou*$ffffff49         * Bit 0 = Busy-Status
FFFFFF49                    467. centstb equ cou*$ffffff49        * Bit 0 = -Strobe Ausgang
                            468.
                            469.
                            470. ** Bis hierher ABS-Werte **
                            471. ************************************************
                            472. * Tabelle alle Befehle die fuer Anwenderprogramme
                            473. * verwendet werden koennen.
                            474. * koennen Indirekt oder ueber Trap 1 angesprungen
                            475. * werden. Zahl ist dabei Index. Uebergabe in Register D7
                            476. * Aufruf TRAP 1
                            477. * alle Adressen Relativ zum Anfang traptab
                            478.                                   * Startadresse basis+400
10400   5AA58001            479.          dc.l $5aa58001          * Suchstring fuer Anfang
```

```
000404  00008000
000408  000033F4
00040C  00000420
000410  00000001
000414  00000001
000418  00000000
        00000000

COMMANDS

000420  6000 2DB4
000424  6000 2FCE
000428  6000 0000
00042C  6000 0000

000430
000430  00000E58
000434  00000EC6
000[redacted]  00000C5A
000[redacted]  00000C60
000[redacted]  00002E02
000444  00002E02
000448  00000EEA
00044C  00000AB2
000450  00000B1C
000454  00000F56
000458  00001056
00045C  000005D0
000460  000005A8
000464  000008C6
000468  000008DA
00046C  00000970
000470  00000990
000474  0000092C
000478  00000E5A
00047C  FFFFFBD0
000480  FFFFFBD0
000484  000008F8
000488  00000C04
00048C  00000C00
000490  00001AE8
000494  00000936
000498  000009DC
00049C  00000A08
0004A0  000012FC
0004A4  0000129C
0004A8  00000638
0004AC  00000646
0004B0  000006B0
0004B4  00001AD6
0004B8  00000A2A
0004BC  00000F12
0004C0  0000094C
0004C4  0000093E
0004C8  00000CDA
000[redacted]  00000F48
000[redacted]  00001238
000[redacted]  00001230
0004D8  00001228
0004DC  0000121A
0004E0  000011F6
0004E4  000011CC
0004E8  00000CE2
0004EC  00000CFE
0004F0  00002D70
0004F4  00002D78
0004F8  00002D80
0004FC  00002D88
000500  00002D94
000504  00002D9A
000508  00002DA0
00050C  FFFFFBD0
000510  00002E5C
000514  00002E56
000518  00001904
```

```
480. ramstart: dc.l ram                 * relativer RAMSTART (basis+ramsta
481.           dc.l start-basis         * Pointer auf Startadresse
482. versnum:  dc.l $00000420           * Release 4.2
483. snnum:    dc.l $00000001           * SN-Nummer, z.Z. nicht genutzt
484.           dc.l cpu                 * CPU 1=08 2=00/10 4=20
485.           dc.l 0,0                 * Reserve

486.
487. * Sprungtabelle auf Basisadresse $420 + basis
488.
489.           bra traprts              * Start Trap mit RTS-Abschluss
490.           bra start                * Kalt-Start
491.           bra 0                    * Reserve
492.           bra 0                    * Reserve
493.                                    *
494.                                    * Dann Befehlstabelle.
495.                                    * auch als Pointer nutzbar
496.                                    * jedoch muss A5 auf
497.                                    * Variablengebiet zeigen.
498.                                    * Besser ueber TRAP ansprungen
499. traptab:                           * Befehlstabelle, relativ zu trapt
500.          dc.l tschreite-traptab    * Befehl 1
501.          dc.l tdrehe-traptab       * Befehl 2
502.          dc.l thoch-traptab        * Befehl 3
503.          dc.l trunter-traptab      * Befehl 4
504.          dc.l tnotimo-traptab      * Befehl 5
505.          dc.l tnotimo-traptab      * Befehl 6
506.          dc.l tmove-traptab        * Befehl 7
507.          dc.l moveto-traptab       * Befehl 8
508.          dc.l drawto-traptab       * Befehl 9
509.          dc.l textaus-traptab      * Befehl 10
510.          dc.l textein-traptab      * Befehl 11
511.          dc.l ci-traptab           * Befehl 12
512.          dc.l csts-traptab         * Befehl 13
513.          dc.l ri-traptab           * Befehl 14
514.          dc.l ooo-traptab          * Befehl 15
515.          dc.l clrall-traptab       * Befehl 16
516.          dc.l clrinvis-traptab     * Befehl 17
517.          dc.l wait-traptab         * Befehl 18
518.          dc.l tschr16tel-traptab   * Befehl 19
519.          dc.l clrscreen-traptab    * Befehl 20
520.          dc.l co-traptab           * Befehl 21
521.          dc.l lo-traptab           * Befehl 22
522.          dc.l sin-traptab          * Befehl 23
523.          dc.l cos-traptab          * Befehl 24
524.          dc.l setgroe-traptab      * Befehl 25
525.          dc.l cmd-traptab          * Befehl 26
526.          dc.l newpage-traptab      * Befehl 27
527.          dc.l sync-traptab         * Befehl 28
528.          dc.l wert-traptab         * Befehl 29
529.          dc.l zuweis-traptab       * Befehl 30
530.          dc.l ciinit2-traptab      * Befehl 31
531.          dc.l ci2-traptab          * Befehl 32
532.          dc.l co2-traptab          * Befehl 33
533.          dc.l setflio-traptab      * Befehl 34
534.          dc.l delay-traptab        * Befehl 35
535.          dc.l firsttime-traptab    * Befehl 36
536.          dc.l setpen-traptab       * Befehl 37
537.          dc.l erapen-traptab       * Befehl 38
538.          dc.l graoff-traptab       * Befehl 39
539.          dc.l print-traptab        * Befehl 40
540.          dc.l print2x-traptab      * Befehl 41
541.          dc.l print4x-traptab      * Befehl 42
542.          dc.l print6x-traptab      * Befehl 43
543.          dc.l print8x-traptab      * Befehl 44
544.          dc.l print8b-traptab      * Befehl 45
545.          dc.l print4d-traptab      * Befehl 46
546.          dc.l hide-traptab         * Befehl 47
547.          dc.l show-traptab         * Befehl 48
548.          dc.l crtex-traptab        * Befehl 49
549.          dc.l lstex-traptab        * Befehl 50
550.          dc.l usrex-traptab        * Befehl 51
551.          dc.l nilex-traptab        * Befehl 52
552.          dc.l seterr-traptab       * Befehl 53
553.          dc.l geterr-traptab       * Befehl 54
554.          dc.l setpass-traptab      * Befehl 55
555.          dc.l edit-traptab         * Befehl 56
556.          dc.l figur-traptab        * Befehl 57
557.          dc.l setfig-traptab       * Befehl 58
558.          dc.l getram-traptab       * Befehl 59
```

```
00051C  00000A4C
00520  FFFFFBD0
00524  FFFFFBD0
00528  FFFFFBD0
0052C  FFFFFBD0
00530  FFFFFBD0
00534  00002CD2
00538  00002CD8
0053C  00002D64
00540  00002D6A
00544  0000118E
00548  00001184
0054C  00001442
00550  0000147E
00554  FFFFFBD0
00558  FFFFFBD0
0055C  FFFFFBD0
00560  00002CF2
00564  00002CFE
00568  00002D06
0056C  00002D0E
00570  00002D16
00574  00002D1E
00578  00000B8C
0057C  00002B66
00580  00002CBA
00584  00002D42
00588  00002D4A
0058C  00002D52
00590  00002D24
00594  00002D2C
00598  00002D38
0059C  00000DFE
005A0  00000E46
005A4  00000DEE
005A8  00000DF6
005AC  00000D9E
005B0  00002D58
005B4  00002D5E
005B8  00001268
005BC  FFFFFBD0
005C0  00001280
005C4  0000128E
005C8  00000ADC
005CC  FFFFFBD0
005D0  FFFFFBD0
005D4  FFFFFBD0
005D8  FFFFFBD0
005DC  FFFFFBD0
005E0  FFFFFBD0
005E4  FFFFFBD0
005E8  FFFFFBD0
005EC  FFFFFBD0
005F0  FFFFFBD0
005F4  FFFFFBD0
005F8  FFFFFBD0
005FC  FFFFFBD0
00600  00000918
00604  000008BE
00608  000008B6
0060C  00001278

00610

     00000078

00610
00610  53 43 48 52
       45 49 54 45
00618  44 52 45 48
       45 20 20 20
00620  48 45 42 45
       20 20 20 20
00628  53 45 4E 4B
       45 20 20 20
00630  3A 3A 3A 20
       20 20 20 20
00638  3A 3A 3A 20
       20 20 20 20
```

```
559.            dc.l autoflip-traptab    * Befehl 60
560.            dc.l cursorein-traptab   * Befehl 61
561.            dc.l cursoraus-traptab   * Befehl 62
562.            dc.l charhandler-traptab * Befehl 63
563.            dc.l progzge-traptab     * Befehl 64
564.            dc.l assemble-traptab    * Befehl 65
565.            dc.l getstx-traptab      * Befehl 66
566.            dc.l outstx-traptab      * Befehl 67
567.            dc.l getorg-traptab      * Befehl 68
568.            dc.l outorg-traptab      * Befehl 69
569.            dc.l print8d-traptab     * Befehl 70  rev 4.0
570.            dc.l printv8d-traptab    * Befehl 71
571.            dc.l muls32-traptab      * Befehl 72
572.            dc.l divs32-traptab      * Befehl 73
573.            dc.l flinit-traptab      * Befehl 74
574.            dc.l floppy-traptab      * Befehl 75
575.            dc.l getflop-traptab     * Befehl 76
576.            dc.l setxor-traptab      * Befehl 77
577.            dc.l getxor-traptab      * Befehl 78
578.            dc.l setcolor-traptab    * Befehl 79
579.            dc.l getcolor-traptab    * Befehl 80
580.            dc.l curonein-traptab    * Befehl 81
581.            dc.l curonaus-traptab    * Befehl 82
582.            dc.l adi360-traptab      * Befehl 83
583.            dc.l symbolaus-traptab   * Befehl 84
584.            dc.l symloesche-traptab  * Befehl 85
585.            dc.l getsymtab-traptab   * Befehl 86
586.            dc.l getnext-traptab     * Befehl 87
587.            dc.l outnext-traptab     * Befehl 88
588.            dc.l getbasis-traptab    * Befehl 89
589.            dc.l getvar-traptab      * Befehl 90
590.            dc.l seta5-traptab       * Befehl 91
591.            dc.l aufxy-traptab       * Befehl 92
592.            dc.l korxy-traptab       * Befehl 93
593.            dc.l aufk-traptab        * Befehl 94
594.            dc.l getk-traptab        * Befehl 95
595.            dc.l rnd-traptab         * Befehl 96
596.            dc.l getvers-traptab     * Befehl 97
597.            dc.l getsn-traptab       * Befehl 98
598.            dc.l crlfe-traptab       * Befehl 99
599.            dc.l getline-traptab     * Befehl 100
600.            dc.l getcurxy-traptab    * Befehl 101
601.            dc.l setcurxy-traptab    * Befehl 102
602.            dc.l getxy-traptab       * Befehl 103
603.            dc.l si-traptab          * Befehl 104
604.            dc.l so-traptab          * Befehl 105
605.            dc.l sists-traptab       * Befehl 106
606.            dc.l sosts-traptab       * Befehl 107
607.            dc.l siinit-traptab      * Befehl 108
608.            dc.l getad8-traptab      * Befehl 109
609.            dc.l getad10-traptab     * Befehl 110
610.            dc.l setda-traptab       * Befehl 111
611.            dc.l speak-traptab       * Befehl 112
612.            dc.l speak1-traptab      * Befehl 113
613.            dc.l sound-traptab       * Befehl 114
614.            dc.l getuhr-traptab      * Befehl 115
615.            dc.l setuhr-traptab      * Befehl 116
616.            dc.l lsts-traptab        * Befehl 117 rev 4.2
617.            dc.l relaisan-traptab    * Befehl 118
618.            dc.l relaisaus-traptab   * Befehl 119
619.            dc.l asserr-traptab      * Befehl 120
620.
621. trapende:
622.
623. maxtraps         equ   (trapende-traptab)/4       * anzahl befehle
624.                                                    .
625.                                                    .
626. trapsym:                    * Symbole fuer EXPR linear suchen
627.            dc.l 'SCHR','EITE'       * 1
628.            dc.l 'DREH','E   '       * 2
629.            dc.l 'HEBE','    '       * 3
630.            dc.l 'SENK','E   '       * 4
631.            dc.l ':::','.    '       * 5
632.            dc.l ':::','.    '       * 6
```

```
000640  53 45 54 20     633.        dc.l 'SET '.'    '      * 7
        20 20 20 20
000648  4D 4F 56 45     634.        dc.l 'MOVE'.'TO  '      * 8
        54 4F 20 20
000650  44 52 41 57     635.        dc.l 'DRAW'.'TO  '      * 9
        54 4F 20 20
000658  57 52 49 54     636.        dc.l 'WRIT'.'E   '      * 10
        45 20 20 20
000660  52 45 41 44     637.        dc.l 'READ'.'    '      * 11
        20 20 20 20
000668  43 49 20 20     638.        dc.l 'CI  '.'    '      * 12
        20 20 20 20
000670  43 53 54 53     639.        dc.l 'CSTS'.'    '      * 13
        20 20 20 20
000678  52 49 20 20     640.        dc.l 'RI  '.'    '      * 14
        20 20 20 20
000680  50 4F 20 20     641.        dc.l 'PO  '.'    '      * 15
        20 20 20 20
000688  43 4C 52 20     642.        dc.l 'CLR '.'    '      * 16
        20 20 20 20
000690  43 4C 50 47     643.        dc.l 'CLPG'.'    '      * 17
        20 20 20 20
000698  57 41 49 54     644.        dc.l 'WAIT'.'    '      * 18
        20 20 20 20
0006A0  53 43 48 52     645.        dc.l 'SCHR'.'16TE'      * 19
        31 36 54 45
0006A8  43 4C 52 53     646.        dc.l 'CLRS'.'CREE'      * 20
        43 52 45 45
0006B0  43 4F 20 20     647.        dc.l 'CO  '.'    '      * 21
        20 20 20 20
0006B8  4C 4F 20 20     648.        dc.l 'LO  '.'    '      * 22
        20 20 20 20
0006C0  53 49 4E 20     649.        dc.l 'SIN '.'    '      * 23
        20 20 20 20
0006C8  43 4F 53 20     650.        dc.l 'COS '.'    '      * 24
        20 20 20 20
0006D0  53 49 5A 45     651.        dc.l 'SIZE'.'    '      * 25
        20 20 20 20
0006D8  43 4D 44 20     652.        dc.l 'CMD '.'    '      * 26
        20 20 20 20
0006E0  4E 45 57 50     653.        dc.l 'NEWP'.'AGE '      * 27
        41 47 45 20
0006E8  53 59 4E 43     654.        dc.l 'SYNC'.'    '      * 28
        20 20 20 20
0006F0  57 45 52 54     655.        dc.l 'WERT'.'    '      * 29
        20 20 20 20
0006F8  5A 55 57 45     656.        dc.l 'ZUWE'.'IS  '      * 30 / rev 4.2 korr
        49 53 20 20
000700  43 49 49 4E     657.        dc.l 'CIIN'.'IT2 '      * 31
        49 54 32 20
000708  43 49 32 20     658.        dc.l 'CI2 '.'    '      * 32
        20 20 20 20
000710  43 4F 32 20     659.        dc.l 'CO2 '.'    '      * 33
        20 20 20 20
000718  53 45 54 46     660.        dc.l 'SETF'.'LIP '      * 34
        4C 49 50 20
000720  44 45 4C 41     661.        dc.l 'DELA'.'Y   '      * 35
        59 20 20 20
000728  46 49 52 53     662.        dc.l 'FIRS'.'TTIM'      * 36
        54 54 49 4D
000730  53 45 54 50     663.        dc.l 'SETP'.'EN  '      * 37
        45 4E 20 20
000738  45 52 41 50     664.        dc.l 'ERAP'.'EN  '      * 38
        45 4E 20 20
000740  47 52 41 50     665.        dc.l 'GRAP'.'OFF '      * 39
        4F 46 46 20
000748  43 4D 44 50     666.        dc.l 'CMDP'.'RINT'      * 40
        52 49 4E 54
000750  50 52 49 4E     667.        dc.l 'PRIN'.'T2X '      * 41
        54 32 58 20
000758  50 52 49 4E     668.        dc.l 'PRIN'.'T4X '      * 42
        54 34 58 20
000760  50 52 49 4E     669.        dc.l 'PRIN'.'T6X '      * 43
        54 36 58 20
000768  50 52 49 4E     670.        dc.l 'PRIN'.'T8X '      * 44
        54 38 58 20
000770  50 52 49 4E     671.        dc.l 'PRIN'.'T8B '      * 45
        54 38 42 20
000778  50 52 49 4E     672.        dc.l 'PRIN'.'T4D '      * 46
        54 34 44 20
```

```
000780  48 49 44 45 20 20 20 20    673.    dc.l 'HIDE','    ' * 47
000788  53 48 4F 57 20 20 20 20    674.    dc.l 'SHOW','    ' * 48
000790  43 52 54 20 20 20 20 20    675.    dc.l 'CRT ','    ' * 49
000798  4C 53 54 20 20 20 20 20    676.    dc.l 'LST ','    ' * 50
0007A0  55 53 52 20 20 20 20 20    677.    dc.l 'USR ','    ' * 51
0007A8  4E 49 4C 20 20 20 20 20    678.    dc.l 'NIL ','    ' * 52
0007B0  53 45 54 45 52 52 20 20    679.    dc.l 'SETE','RR  ' * 53
0007B8  47 45 54 45 52 52 20 20    680.    dc.l 'GETE','RR  ' * 54
0007C0  53 45 54 50 41 53 53 20    681.    dc.l 'SETP','ASS ' * 55
0007C8  45 44 49 54 20 20 20 20    682.    dc.l 'EDIT','    ' * 56
0007D0  46 49 47 55 52 20 20 20    683.    dc.l 'FIGU','R   ' * 57
0007D8  53 45 54 46 49 47 20 20    684.    dc.l 'SETF','IG  ' * 58
0007E0  47 45 54 52 41 4D 20 20    685.    dc.l 'GETR','AM  ' * 59
0007E8  41 55 54 4F 46 4C 49 50    686.    dc.l 'AUTO','FLIP' * 60
0007F0  43 55 52 53 45 49 4E 20    687.    dc.l 'CURS','EIN ' * 61
0007F8  43 55 52 53 41 55 53 20    688.    dc.l 'CURS','AUS ' * 62
000800  43 48 41 52 20 20 20 20    689.    dc.l 'CHAR','    ' * 63
000808  50 52 4F 47 5A 47 45 20    690.    dc.l 'PROG','ZGE ' * 64
000810  41 53 53 45 4D 42 4C 45    691.    dc.l 'ASSE','MBLE' * 65
000818  47 45 54 53 54 58 20 20    692.    dc.l 'GETS','TX  ' * 66
000820  50 55 54 53 54 58 20 20    693.    dc.l 'PUTS','TX  ' * 67
000828  47 45 54 4F 52 47 20 20    694.    dc.l 'GETO','RG  ' * 68
000830  50 55 54 4F 52 47 20 20    695.    dc.l 'PUTO','RG  ' * 69
000838  50 52 49 4E 54 38 44 20    696.    dc.l 'PRIN','T8D ' * 70
000840  50 52 49 4E 54 56 38 44    697.    dc.l 'PRIN','TV8D' * 71
000848  4D 55 4C 53 33 32 20 20    698.    dc.l 'MULS','32  ' * 72
000850  44 49 56 53 33 32 20 20    699.    dc.l 'DIVS','32  ' * 73
000858  46 4C 49 4E 49 54 20 20    700.    dc.l 'FLIN','IT  ' * 74
000860  46 4C 4F 50 50 59 20 20    701.    dc.l 'FLOP','PY  ' * 75
000868  47 45 54 46 4C 4F 50 20    702.    dc.l 'GETF','LOP ' * 76
000870  53 45 54 58 4F 52 20 20    703.    dc.l 'SETX','OR  ' * 77
000878  47 45 54 58 4F 52 20 20    704.    dc.l 'GETX','OR  ' * 78
000880  53 45 54 43 4F 4C 4F 52    705.    dc.l 'SETC','OLOR' * 79
000888  47 45 54 43 4F 4C 4F 52    706.    dc.l 'GETC','OLOR' * 80
000890  43 55 52 4F 4E 20 20 20    707.    dc.l 'CURO','N   ' * 81
000898  43 55 52 4F 46 46 20 20    708.    dc.l 'CURO','FF  ' * 82
0008A0  41 44 4A 33 36 30 20 20    709.    dc.l 'ADJ3','60  ' * 83
0008A8  50 52 54 53 59 4D 20 20    710.    dc.l 'PRTS','YM  ' * 84
0008B0  53 59 4D 43 4C 52 20 20    711.    dc.l 'SYMC','LR  ' * 85
0008B8  47 45 54 53 59 4D 20 20    712.    dc.l 'GETS','YM  ' * 86
```

```
0008C0  47 45 54 4E    713.          dc.l 'GETN','EXT '      * 87
        45 58 54 20
0008C8  50 55 54 4E    714.          dc.l 'PUTN','EXT '      * 88
        45 58 54 20
0008D0  47 45 54 42    715.          dc.l 'GETB','ASIS'      * 89
        41 53 49 53
0008D8  47 45 54 56    716.          dc.l 'GETV','AR  '      * 90
        41 52 20 20
0008E0  53 45 54 41    717.          dc.l 'SETA','5   '      * 91
        35 20 20 20
0008E8  41 55 46 58    718.          dc.l 'AUFX','Y   '      * 92
        59 20 20 20
0008F0  4B 4F 52 58    719.          dc.l 'KORX','Y   '      * 93
        59 20 20 20
0008F8  41 55 46 4B    720.          dc.l 'AUFK','    '      * 94
        20 20 20 20
000900  47 45 54 4B    721.          dc.l 'GETK','    '      * 95
        20 20 20 20
000908  52 4E 44 20    722.          dc.l 'RND ','    '      * 96
        20 20 20 20
000910  47 45 54 56    723.          dc.l 'GETV','ERS '      * 97
        45 52 53 20
000918  47 45 54 53    724.          dc.l 'GETS','N   '      * 98
        4E 20 20 20
000920  43 52 4C 46    725.          dc.l 'CRLF','    '      * 99
        20 20 20 20
000928  47 45 54 4C    726.          dc.l 'GETL','INE '      * 100
        49 4E 45 20
000930  47 45 54 43    727.          dc.l 'GETC','URXY'      * 101
        55 52 58 59
000938  53 45 54 43    728.          dc.l 'SETC','URXY'      * 102
        55 52 58 59
000940  47 45 54 58    729.          dc.l 'GETX','Y   '      * 103
        59 20 20 20
000948  53 49 20 20    730.          dc.l 'SI  ','    '      * 104
        20 20 20 20
000950  53 4F 20 20    731.          dc.l 'SO  ','    '      * 105
        20 20 20 20
000958  53 49 53 54    732.          dc.l 'SIST','S   '      * 106
        53 20 20 20
000960  53 4F 53 54    733.          dc.l 'SOST','S   '      * 107
        53 20 20 20
000968  53 49 49 4E    734.          dc.l 'SIIN','IT  '      * 108
        49 54 20 20
000970  47 45 54 41    735.          dc.l 'GETA','D8  '      * 109
        44 38 20 20
000978  47 45 54 41    736.          dc.l 'GETA','D10 '      * 110
        44 31 30 20
000980  53 45 54 44    737.          dc.l 'SETD','A   '      * 111
        41 20 20 20
000988  53 50 45 41    738.          dc.l 'SPEA','K   '      * 112
        4B 20 20 20
000990  53 50 45 41    739.          dc.l 'SPEA','K1  '      * 113
        4B 31 20 20
000998  53 4F 55 4E    740.          dc.l 'SOUN','D   '      * 114
        44 20 20 20
0009A0  47 45 54 55    741.          dc.l 'GETU','HR  '      * 115
        48 52 20 20
0009A8  53 45 54 55    742.          dc.l 'SETU','HR  '      * 116
        48 52 20 20
0009B0  4C 53 54 53    743.          dc.l 'LSTS','    '      * 117
        20 20 20 20
0009B8  52 45 4C 41    744.          dc.l 'RELA','N   '      * 118
        4E 20 20 20
0009C0  52 45 4C 41    745.          dc.l 'RELA','US  '      * 119
        55 53 20 20
0009C8  41 53 53 45    746.          dc.l 'ASSE','RR  '      * 120
        52 52 20 20
                       747.
0009D0  00000000       748.          dc.l 0.0
        00000000
                       749.
                       750.
                       751. * Unterprogramme Tastatur
                       752. * Achtung CSTS gibt keinen Cursor aus.
                       753. * wenn man einen will, so muß man mit CURSEIN.CURSAUS
                       754. * arbeiten.
                       755.
0009D8  0C2D 0006      756. csts:    cmp.b #6,iostatb(a5)
        002B
```

```
0009DE   6714                  beq.s uscsts     * Userstatus
0009E0   1038 FF68             move.b kevd.d0   * status testen
0009E4   4A00                  tst.b d0         * bit 7 ist Vorzeichen
0009E6   6B08                  bmi.s cstsi
0009E8   203C 000000FF         move.l #$FF.d0   * $FF in D0.L rev 4.2
0009EE   4E75                  rts
0009F0   4280          cstsi:  clr.l d0         * 0 wenn kein zeichen da
0009F2   4E75                  rts

0009F4
0009F4   7002          uscsts:
0009F6   4EED 001E             moveq #2.d0      * Flag fuer CSTS von CI-Seite
                               jmp usercsts(a5) * 0=frei ff=belegt.als Ergebnis
                                                * in d0.l liefern.

                       ** IODIR
                       * Bit 2 =1 dann last char in ci gelesen
                       * Bit 1 =1 dann ci von einbuf holen. ueber einpoi
                       * Bit 0 =1 dann ausgabe auf ausbuf ueber auspoi
                       *
                       * CI-Umsteuerung mit IOSTATB
                       * =5. dann ci2 Umsteuerung.
                       * =6. dann ueber USERCI  als Kennung d0.w=2
                       * um es vom CI2 zu unterscheiden.

0009FA
0009FA   7002          ciusext:
                               moveq #2.d0      * Kennung an USERCI. dass Data an
                                                * CI geht und nicht an CI2
0009FC   4EED 0018             jmp userci(a5)   * Aufrufen. Achtung. keine Register veraen
                                                * ausser d0.l und ganzer Register d0.l
                                                * laden wegen co/m68k etc.

000A00
000A00   0C2D 0006     ci:     cmp.b #6.iostatb(a5)     * USER CI ?
         002B
000A06   67F2                  beq ciusext
000A08   08AD 0002             bclr.b #2.iodir(a5)      * Last char freigabe
         002C
000A0E   082D 0001             btst.b #1.iodir(a5)      * Test ob Daten aus Buffer da sind
         002C
000A14   6600 0022             bne ci11                 * ja dann aus Buffer holen
000A18   6100 F5E6             bsr cursorein
000A1C
000A1C   6100 045E     ci1:    bsr autoflio     * flio-mode
000A20   1038 FF68             move.b kevd.d0   * test ob daten da
000A24   4A00                  tst.b d0         * bit 7 bestimmt dies
000A26   6BF4                  bmi.s ci1        * warten bis daten da
000A28   4A38 FF69             tst.b kevs       * ruecksetzten bit dadurch
000A2C   6100 F5D2             bsr cursoraus
000A30   C0BC 0000007F         and.l #$7f.d0    * d0.w d0.l gueltig wegen com68k
000A36   4E75                  rts              * in d0.b ist zeichen

000A38   48E7 0080     ci11:   movem.l a0.-(a7)
000A3C   206D 005E             movea.l einpoi(a5).a0    * Quelle
000A40   1018                  move.b (a0)+.d0
000A42   2B48 005E             move.l a0.einpoi(a5)
000A46   B03C 0000             cmp.b #0.d0      * Terminator ist 0. Leerz. wird ausgegeben
000A4A   6610                  bne.s ci12
000A4C   08AD 0001             bclr #1.iodir(a5)
         002C
000A52   08ED 0002             bset #2.iodir(a5)        * Last char flag
         002C
000A58   103C 0020             move.b #' '.d0
000A5C   4CDF 0100     ci12:   movem.l (a7)+.a0
000A60   4E75                  rts


000A62   619C          ki:     bsr ci           * mit grossbu-konverter
000A64   6000 10C8             bra bucheck      * umwandlung in grossbuchstabe


000A68
000A68   2B6D 0036     ciinit2:                        * init vorbereiten
         003A                  move.l stxtxt(a5).akttxt(a5)
000A6E   1B7C 0001             move.b #1.ci2flag(a5)    * flag first aktiv
         0A46
000A74   4E75                  rts

000A76                 ci2:                     * Zeichen holen von Quelle
                               * ggf ueber schalter anwaehlbar
                               * sonst AKTTXT
```

```
                                830.                            * wert = 0 wenn ende erreicht
                                831.                            * ETXTXT
000A76  0C2D 0005               832.            cmp.b #5.iostatb(a5)    * USERCI
        002B
000A7C  6700 0026               833.            beq ci2ext
000A80  2F08                    834.            move.l a0.-(a7)
000A82  206D 003A               835.            movea.l akttxt(a5).a0
000A86  1010                    836.            move.b (a0).d0          * =0 dann ende
000A88  670C                    837.            beq.s ci2a
000A8A  5288                    838.            addq.l #1.a0            * erste wenn <>0 erhoehen
000A8C  2B48 003A               839.            move.l a0.akttxt(a5)    * neuer Zeiger
000A90  205F                    840.            movea.l (a7)+.a0
000A92  6000 102E               841.            bra carres              * CARRY NUR BEI FEHLER
000A96                          842. ci2a:
000A96  3B7C 0002               843.            move #2.errflag(a5)     * Ende uebersetzung
        0034
000A9C  4200                    844.            clr.b d0
000A9E  205F                    845.            movea.l (a7)+.a0
000AA0  6000 101A               846.            bra carset      * CARRY DA FEHLER
                                847.
                                848.
                                849.
000AA4                          850. ci2ext:                    * d0 muss ergebnis sein
000AA[illegible]  48E7 7FFE     851.            movem.l d1-d7/a0-a6.-(a7)
000A[illegible]  4240           852.            clr d0                  * wort ist gueltig  rev 4.2
000AA[illegible]  102D 0A46     853.            move.b ci2flag(a5).d0   * uebergeben 1=erstes Zeichen hole
000AAE  422D 0A46               854.            clr.b ci2flag(a5)       * nun erstes Zeichen gelesen ok=0
000AB2                          855. ci21ext:
000AB2  4EAD 0018               856.            jsr userci(a5)  * Sprung dorthin mit Parameter in d0.w
000AB6  4CDF 7FFE               857.            movem.l (a7)+.d1-d7/a0-a6 * d0=0 dann errflag
000ABA                          858. ci2final:
000ABA  B03C 0000               859.            cmp.b #$0.d0
000ABE  6600 1002               860.            bne carres
000AC2  3B7C 0002               861.            move #2.errflag(a5)
        0034
000AC8  6000 0FF2               862.            bra carset
                                863.
                                864.
000ACC                          865. prtco2:                    * Ausgabe eines Textstrings von (a0) an
000ACC  1018                    866.            move.b (a0)+.d0
000ACE  6700 0FF2               867.            beq carres
000AD2  48E7 0080               868.            movem.l a0.-(a7)
000AD6  6100 0008               869.            bsr co2
000ADA  4CDF 0100               870.            movem.l (a7)+.a0
000ADE  60EC                    871.            bra prtco2
                                872.
000AE0                          873. co2:                       * Zeichen ausgeben  nur
                                874.                            * wenn pass = 2
000AE0  0C6D 0002               875.            cmp #2.passflag(a5)     * wenn = 2 dann ok
        0032
000AE6  6600 0FD4               876.            bne carset      * nein keine ausgabe
000AEA                          877. co3:                       * wie co2 . jedoch ohne passabfrage
                                878.                            * Mit CTRL-S. CTRL-Q Erkennung
000AEA  3F00                    879.            move d0.-(a7)
000AEC  1038 FF68               880.            move.b kevd.d0
000AF0  4A00                    881.            tst.b d0
000AF2  6B1C                    882.            bmi.s co2w1     * Bit7=1 dann frei
000AF4  B03C 0013               883.            cmp.b #$13.d0   * CTRL-S Stopot
000AF8  6616                    884.            bne.s co2w1     * Sonst Zeichen ignorieren. Status lassen
000AFA  4A38 FF69               885. co212:     tst.b kevs      * loeschen Status
000AF[illegible]  1038 FF68     886. co211:     move.b kevd.d0  *
000B0[illegible]  4A00          887.            tst.b d0
000B04  6BF8                    888.            bmi.s co211     * warten auf Eingabe
000B06  B03C 0011               889.            cmp.b #$11.d0   * CTRL-Q startet wieder
000B0A  66EE                    890.            bne co212       * loeschen und warten
000B0C  4A38 FF69               891.            tst.b kevs      * und loeschen Status
                                892.                            * dann weiter
000B10                          893. co2w1:                     * Tastatur Neutral
000B10  301F                    894.            move (a7)+.d0   * Zeichen fuer Ausgabe
000B12  B03C 0001               895.            cmp.b #1.d0     * CTRL-A ist escape fuer Sonderfunktion
000B16  6700 00B2               896.            beq co22b
000B1A  082D 0000               897.            btst.b #0.iodir(a5)
        002C
000B20  6600 00C2               898.            bne co22a       * Mode ist an  nun aufspeichern
                                899.                            * sonst normal
                                900.                            * REV 3.1  Echosperre
000B24  082D 0002               901.            btst.b #2.iodir(a5)     * Last char
        002C
000B2A  6710                    902.            beq.s co2w1a
000B2C  08AD 0002               903.            bclr.b #2.iodir(a5)     * nur einmal last char
```

```
                002C
000B32  08AD 0001       904.            bclr.b #1.iodir(a5)     * savetv
                002C
000B38  6000 0F88       905.            bra carres      * auch nicht bei letzem Leerzeichen
000B3C                  906. co2w1a:
000B3C  082D 0001       907.            btst.b #1.iodir(a5)
                002C
000B42  6600 0F7E       908.            bne carres      * KEINE AUSGABE wenn Get-Befehl
                        909.
000B46  B03C 000A       910.            cmo.b #$a.d0
000B4A  6604            911.            bne.s co2a1
000B4C  522D 021F       912.            addo.b #1.zeilen(a5)    * Zeilenzaehler
000B50                  913. co2a1:
000B50  0C2D 0001       914.            cmo.b #1.iostat(a5)     * IO-Kanal
                002A
000B56  660C            915.            bne.s co21
000B58  4A6D 0034       916.            tst errflag(a5) * = 1.2 dann ausgabe auf Console
000B5C  6700 0F64       917.            beq carres      *= 0 dann nicht
000B60  6000 000A       918.            bra cocrt       * nur waehrend Fehlerfalls
                        919.
000B64                  920. co21:
000B64  0C2D 0002       921.            cmo.b #2.iostat(a5)     * crt
                002A
000B6A  6618            922.            bne.s co22
000B6C                  923. cocrt:
000B6C  48E7 FFFE       924.            movem.l d0-d7/a0-a6.-(a7)
000B70  6100 F48E       925.            bsr co          * ja ausgeben
000B74  422D 0218       926.            clr.b wrtoage(a5)       * wichtig fuer nachfolgende Prge.
000B78  6100 029A       927.            bsr aktoage             * Schreibseite neu definieren auf
000B7C  4CDF 7FFF       928.            movem.l (a7)+.d0-d7/a0-a6
000B80  6000 0F40       929.            bra carres
                        930.
000B84  0C2D 0003       931. co22:      cmo.b #3.iostat(a5)     * LST
                002A
000B8A  6610            932.            bne.s co23
000B8C  48E7 FFFE       933.            movem.l d0-d7/a0-a6.-(a7)
000B90  6100 0196       934.            bsr lo          * ja ausgeben
000B94  4CDF 7FFF       935.            movem.l (a7)+.d0-d7/a0-a6
000B98  6000 0F28       936.            bra carres
                        937.
000B9C                  938. co23:
000B9C  0C2D 0004       939.            cmo.b #4.iostat(a5)     * LST ohne LF
                002A
000BA2  6610            940.            bne.s co24              * Auch Fehler
000BA4  48E7 FFFE       941.            movem.l d0-d7/a0-a6.-(a7)
000BA8  6100 0176       942.            bsr lolf                * ja ausgeben
000BAC  4CDF 7FFF       943.            movem.l (a7)+.d0-d7/a0-a6
000BB0  6000 0F10       944.            bra carres
                        945.
000BB4  0C2D 0005       946. co24:      cmo.b #5.iostat(a5)     * USERCO
                002A
000BBA  66B0            947.            bne cocrt
000BBC  48E7 FFFE       948.            movem.l d0-d7/a0-a6.-(a7)
000BC0  4EAD 0024       949.            jsr userco(a5)          * Sprung dorthin
000BC4  4CDF 7FFF       950.            movem.l (a7)+.d0-d7/a0-a6
000BC8  4E75            951.            rts
                        952.
000BCA                  953. co22b:                             * CTRL-A
000BCA  08ED 0000       954.            bset #0.iodir(a5)       * setzen
                002C
000BD0  48E7 0080       955.            movem.l a0.-(a7)
000BD4  41ED 0070       956.            lea ausbuf(a5).a0
000BD8  4210            957.            clr.b (a0)              * Terminator
000BDA  2B48 005A       958.            move.l a0.ausooi(a5)    * Ziel auf Anfang
000BDE  4CDF 0100       959.            movem.l (a7)+.a0        * rev 4.2 korr auf a0
000BE2  4E75            960.            rts
                        961.
                        962.
000BE4                  963. co22a:
000BE4  B03C 000A       964.            cmo.b #$a.d0            * Ende Zeichen ist LF
000BE8  6716            965.            beq.s co222a            * dort weiter wenn ja
000BEA  48E7 0080       966.            movem.l a0.-(a7)
000BEE  206D 005A       967.            movea.l ausooi(a5).a0
000BF2  10C0            968.            move.b d0.(a0)+         * Zeichen absoeichern
000BF4  4210            969.            clr.b (a0)              * Ende markieren
000BF6  2B48 005A       970.            move.l a0.ausooi(a5)
000BFA  4CDF 0100       971.            movem.l (a7)+.a0
000BFE  4E75            972.            rts
                        973.
000C00                  974. co222a:
```

```
000C00  08AD 0000      975.          bclr #0.iodir(a5)        * Ende erkannt
        002C
000C06  48E7 FFFE      976.          movem.l d0-d7/a0-a6.-(a7)
000C0A  41ED 0070      977.          lea ausbuf(a5).a0
                       978.          repeat
000C0E  1018           979.             move.b (a0)+.d0
000C10  B03C 0020      980.          until.b d0 <ne> #' '
        67F8
000C16  B03C 0045      981.          if.b d0 <eq> #'E' then  * Execute
        6600 005E
000C1E  6100 0B0C      982.             bsr wert        * E adr val val val
                       983.                             *       d0..d7      KEINE ADRESSEN
000C22  B27C 0000      984.             if d1 <ne> #0 then
        6700 004E
000C2A  B27C 0005      985.                if d1 <ne> #5 then
        6700 0046
000C32  2F00           986.                   move.l d0.-(a7)       * retten Adresse
                       987.                                         * Nun Parameter
                       988.
000C34  6100 0AF6      989.                   bsr wert              * alle .l  d0
000C38  2F00           990.                   move.l d0.-(a7)
000C3A  6100 0AF0      991.                   bsr wert              * alle .l
000C3E  2F00           992.                   move.l d0.-(a7)
000C42  6100 0AEA      993.                   bsr wert              * alle .l
000C46  2F00           994.                   move.l d0.-(a7)
000C4A  6100 0AE4      995.                   bsr wert              * alle .l
000C4A  2F00           996.                   move.l d0.-(a7)
000C4C  6100 0ADE      997.                   bsr wert              * alle .l
000C50  2F00           998.                   move.l d0.-(a7)
000C52  6100 0AD8      999.                   bsr wert              * alle .l
000C56  2F00          1000.                   move.l d0.-(a7)
000C58  6100 0AD2     1001.                   bsr wert              * alle .l
000C5C  2F00          1002.                   move.l d0.-(a7)
000C5E  6100 0ACC     1003.                   bsr wert              * alle .l d7
000C62  2E00          1004.                   move.l d0.d7
000C64  2C1F          1005.                   move.l (a7)+.d6
000C66  2A1F          1006.                   move.l (a7)+.d5
000C68  281F          1007.                   move.l (a7)+.d4
000C6A  261F          1008.                   move.l (a7)+.d3
000C6C  241F          1009.                   move.l (a7)+.d2
000C6E  221F          1010.                   move.l (a7)+.d1
000C70  201F          1011.                   move.l (a7)+.d0
000C72  2C5F          1012.                   movea.l (a7)+.a6      * Ziel
000C74  4E96          1013.                   jsr (a6)
                      1014.                endi
                      1015.             endi
000C76  6000 005A     1016.          else if.b d0 <eq> #'G' then      * Get
        B03C 0047 6600
        0026
000C82  6100 0AA8     1017.             bsr wert        * d1=0 fehler 5=undef
000C86  2040          1018.             movea.l d0.a0   * auch bei Fehler
000C88  1010          1019.             move.b (a0).d0 * Wert
000C8A  C07C 00FF     1020.             and.w #$ff.d0
000C8E  41ED 00F4     1021.             lea einbuf(a5).a0
000C92  2B48 005E     1022.             move.l a0.einpoi(a5)
000C96  6100 0964     1023.             bsr print4d     * Als Zahl 0..255
000C9A  4218          1024.             clr.b (a0)+     * Ende
000C9C  08ED 0001     1025.             bset #1.iodir(a5)              * Umschalten CI
        002C
000CA2  6000 002E     1026.          else if.b d0 <eq> #'P' then      * Poke
        B03C 0050 6600
        0026
000CAE  6100 0A7C     1027.             bsr wert
000CB2  B27C 0000     1028.             if d1 <ne> #0 then
        6700 001A
000CBA  2F00          1029.                move.l d0.-(a7)
000CBC  6100 0A6E     1030.                bsr wert
000CC0  B27C 0000     1031.                if d1 <ne> #0 then
        6700 000A
000CC8  205F          1032.                   movea.l (a7)+.a0      * P adr byte
000CCA  1080          1033.                   move.b d0.(a0)
000CCC  6000 0004     1034.                else
000CD0  201F          1035.                   move.l (a7)+.d0
                      1036.                endi
                      1037.             endi
                      1038.          endi
000CD2  4CDF 7FFF     1039.          movem.l (a7)+.d0-d7/a0-a6
000CD6  4E75          1040.          rts
                      1041.
                      1042. * Kassettenio
```

CAS

```
000CD8                          1043. * Unterprogramme fuer die Kassettenbedienung
                                1044.
000CD8                          1045. casinit:
000CD8  11FC 0053 FFCA          1046.           move.b #$53.cmdcas
000CDE  11FC 0050 FFCA          1047.           move.b #$50.cmdcas      * Relais anschalten
000CE4  4E75                    1048.           rts
                                1049.
000CE6                          1050. relaisaus:
000CE6  11FC 0010 FFCA          1051.           move.b #$10.cmdcas      * Relais aus.
000CEC  4E75                    1052.           rts
                                1053.
000CEE                          1054. relaisan:
000CEE  11FC 0050 FFCA          1055.           move.b #$50.cmdcas      * Relais an.
000CF4  4E75                    1056.           rts
                                1057.
                                1058.
000CF6                          1059. ri:
000CF6  0838 0000 FFCA          1060.           btst.b #0.cmdcas
000CFC  67F8                    1061.           beq ri
000CFE  1038 FFCB               1062.           move.b datcas.d0
000D02  C0BC 000000FF           1063.           and.l #$ff.d0    * Byte innerhalb Langwort
000D08  4E75                    1064.           rts              * gueltig. achtung frueher nur byte
                                1065.
000D0A                          1066. ooo:
000D0A  0838 0001 FFCA          1067.           btst.b #1.cmdcas
000D10  67F8                    1068.           beq ooo
000D12  11C0 FFCB               1069.           move.b d0.datcas
000D16  4E75                    1070.           rts
                                1071.
                                1072.
```

CENT

```
                                1073. * Unterprogramme Drucker. Centronix
                                1074.
                                1075.
000D18                          1076. loinit:
000D18  11FC 0001 FF49          1077.           move.b #1.centstb
000D1E  4E75                    1078.           rts
                                1079.
000D20                          1080. lolf:
000D20  B03C 000A               1081.           cmp.b #$a.d0
000D24  6700 0D9C               1082.           beq carres       * Ignorieren wenn lf
000D28                          1083. lo:                        * Einsprung wenn nicht krit.
000D28  11FC 0001 FF49          1084.           move.b #1.centstb
000D2E  0838 0000 FF49          1085.           btst.b #0.centin
000D34  66F2                    1086.           bne.s lo         * Warten bis BUSY = Low-Signal
000D36  11C0 FF48               1087.           move.b d0.centdaten      * Ausgabe auf Datenport
000D3A  11FC 0000 FF49          1088.           move.b #0.centstb
000D40  11FC 0001 FF49          1089.           move.b #1.centstb * ---____--- Strobeform
000D46  4E75                    1090.           rts
                                1091.
000D48  0838 0000 FF49          1092. lsts:     btst.b #0.centin
000D4E  6704                    1093.           beq.s lstsirdv
000D50  4280                    1094.           clr.l d0
000D52  4E75                    1095.           rts
000D54                          1096. lstsirdv:
000D54  203C 000000FF           1097.           move.l #$ff.d0  * 0ffh=readv
000D5A  4E75                    1098.           rts
                                1099.
                                1100.
```

GDP

```
                                1101. * Unterprogramme GDP 9366
                                1102.
                                1103. * warten bis gdp fertig mit Befausfuehrung
                                1104.
000D5C  0838 0002 FF70          1105. wait:     btst.b #2.gdp    * bit 2 pruefen
000D62  67F8                    1106.           beq wait         * warten bis auf 1
000D64  4E75                    1107.           rts
                                1108.
                                1109. * Befehl an gdp ausgeben . in D0 ist CMD
```

```
                                1110.
000D66  61F4                    1111. cmd:    bsr wait
000D68  11C0 FF70               1112.         move.b d0,gdo   * und befehl ausgeben
000D6C  4E75                    1113.         rts
                                1114.
000D6E  3F00                    1115. eraoen: move d0.-(a7)
000D70  7001                    1116.         moveq #1.d0
000D72  61F2                    1117. era1:   bsr cmd
000D74  7002                    1118.         moveq #2.d0
000D76  61EE                    1119.         bsr cmd
000D78  301F                    1120.         move (a7)+.d0
000D7A  4E75                    1121.         rts
                                1122.
000D7C  3F00                    1123. setoen: move d0.-(a7)
000D7E  4240                    1124.         clr d0
000D80  60F0                    1125.         bra era1
                                1126.
                                1127. * Loeschen einer bildschirmseite in d0
                                1128. * Die Bildschirmseite muss auch die aktuelle
                                1129. * Leseseite sein.. XOR-Mode aus. rev 4.0
                                1130. * Farbart belassen . rev 4.0
                                1131.
000D82  61D8                    1132. clear:  bsr wait
000D84  C03C 00F0               1133.         and.b #$f0.d0
00[redacted]  11C0 FF60         1134.         move.b d0,oage
00[redacted]  11FC 0007         1135.         move.b #$7.gdp+1*cou * blank
        FF71
000D92  7006                    1136.         moveq #6.d0       * clear screen
000D94  61D0                    1137.         bsr cmd
000D96  61C4                    1138.         bsr wait
000D98  11FC 0003               1139.         move.b #$3.gdo+1*cou * sichtbar
        FF71
000D9E  4E75                    1140.         rts
                                1141.
                                1142. * Loeschen aller vier Bildschirmseiten.aktuelle Farbe
                                1143. * a5-> Speichergebiet
                                1144.
000DA0  7000                    1145. clrall: moveq #%00000000.d0
000DA2  61DE                    1146.         bsr clear
000DA4  7050                    1147.         moveq #%01010000.d0
000DA6  61DA                    1148.         bsr clear
000DA8  103C 00A0               1149.         move.b #%10100000.d0
000DAC  61D4                    1150.         bsr clear
000DAE  103C 00F0               1151.         move.b #%11110000.d0
000DB2  61CE                    1152.         bsr clear
000DB4  422D 0217               1153.         clr.b viewoage(a5)
000DB8  422D 0218               1154.         clr.b wrtoage(a5)
000DBC  6156                    1155.         bsr.s aktoage
000DBE  4E75                    1156.         rts
                                1157.
                                1158. * Damit koennen auch unsichtbare Bildseiten geloscht
                                1159. * werden. also Leseseite<>Schreibseite
                                1160.
000DC0                          1161. clrinvis:               * Loeschen auch unsichbare Seiten
000DC0  48E7 FE00               1162.         movem.l d0-d6.-(a7)
000DC4  61A8                    1163.         bsr eraoen      * eraser funktion
000DC6  6194                    1164.         bsr wait
000DC8  4238 FF73               1165.         clr.b gdo+3*cou * size = max
000DCC  4238 FF72               1166.         clr.b gdo+2*cou * line = 0
000DD0  343C 0000               1167.         move #0.d2      * v:=0
000DD4  3A3C 0004               1168.         move #5-1.d5    * counter
00[redacted]  323C 0000         1169. clr1:   move #0.d1      * x:=0
00[redacted]  6100 0104         1170.         bsr moveto      * setzen
000DE0  3C3C 0007               1171.         move #8-1.d6    *
000DE4  303C 000B               1172. clr2:   move #$B.d0     * befehl square
000DE8  6100 FF7C               1173.         bsr cmd
000DEC  51CE FFF6               1174.         dbra d6.clr2    *bis alle
000DF0  D47C 0040               1175.         add #64.d2      *v:=v+16*4
000DF4  51CD FFE2               1176.         dbra d5.clr1
000DF8  6100 FF62               1177.         bsr wait        *dann size normal
000DFC  11FC 0011               1178.         move.b #%00010001.gdo+3*cou
        FF73
000E02  6100 FF78               1179.         bsr setoen      *wieder setzmode
000E06  4CDF 007F               1180.         movem.l (a7)+.d0-d6
000E0A  4E75                    1181.         rts
                                1182.
                                1183. * Anwahl einer neuen Bildseite.
                                1184. * dabei ist d0 die Schreibseite und d1 die Leseseite
                                1185.
```

```
000E0C
000E0C  1B40 0218       1186. newpage:                  * d0=wrtpage d1=viewpage
000E10  1B41 0217       1187.           move.b d0.wrtpage(a5)
                        1188.           move.b d1.viewpage(a5)
                        1189.
000E14  102D 0218       1190. aktpage: move.b wrtpage(a5).d0  * 0..3
000E18  E518            1191.           rol.b #2.d0
000E1A  802D 0217       1192.           or.b viewpage(a5).d0    * 0..3
000E1E  E918            1193.           rol.b #4.d0             * xxxx0000
                        1194.                                   * dann setzen
000E20  6100 FF3A       1195. setpage: bsr wait                 * ausgabe d0 wwrr0000
000E24  C03C 00F0       1196.           and.b #$f0.d0           * als pagecode wrt.view
000E28  802D 0A45       1197.           or.b xormode(a5).d0 * rev 4.0 XOR-Mode anwaehlen
000E2C  11C0 FF60       1198.           move.b d0.page
000E30  11ED 0A44       1199.           move.b color(a5).coloort
        FFA0
000E36  4E75            1200.           rts
                        1201.
                        1202.
                        1203. * Wenn svnc dann 0ff als ergebnis sonst 0 im Register D0
                        1204. * Damit koennen Einschreibvorgaenge zum Bildwechsel
                        1205. * synchronisiert werden. Ebenfalls als genaue
                        1206. * Verzoegerung verwendbar. da Quarzgesteuert.
                        1207.
000E38  0838 0001       1208. svnc:    btst #1.gdp      * gdp syncbit
        FF70
000E3E  6712            1209.           beq.s svn1
000E40  4A2D 0219       1210.           tst.b svnstate(a5)
000E44  6610            1211.           bne.s svn2      * sync schon abgefragt
000E46  303C FFFF       1212.           move #$FFFF.d0  * flag ok
000E4A  1B40 0219       1213.           move.b d0.svnstate(a5) * auch dorthin
000E4E  4A40            1214.           tst d0          *setze flags
000E50  4E75            1215.           rts             * flag non zero
                        1216.                           *
000E52  422D 0219       1217. svn1:    clr.b svnstate(a5)      * auf 0  kein sync
000E56  4240            1218. svn2:    clr d0           * d0=0 als flag
000E58  4E75            1219.           rts             * flag zero
                        1220.
                        1221.
                        1222. * Verzoegerungsschleife mit Quarzgenauigkeit
                        1223. * da der GDP64 als Zeitbasis verwendet wird
                        1224.
000E5A                  1225. delav:                      * d0 in .1 sec
000E5A  4A80            1226.           tst.l d0
000E5C  671C            1227.           beq.s delfin
000E5E  2F00            1228.           move.l d0.-(a7)
000E60  303C 0004       1229.           move #5-1.d0
000E64                  1230. del2av:
000E64  3F00            1231.           move d0.-(a7)
000E66                  1232. del1av:
000E66  61D0            1233.           bsr svnc        * warten bis svnc
000E68  67FC            1234.           beq del1av
000E6A  6100 0016       1235.           bsr autoja      * ggf. autoflip aufrufen
000E6E  301F            1236.           move (a7)+.d0
000E70  51C8 FFF2       1237.           dbra d0.del2av
000E74  201F            1238.           move.l (a7)+.d0
000E76  5380            1239.           subq.l #1.d0
000E78  66E0            1240.           bne delav
000E7A                  1241. delfin:
000E7A  4E75            1242.           rts
                        1243.
                        1244. * Die Routine sollte in allen Warteschleifen aufgerufen
                        1245. * werden, wenn man mehrere Bildebenen Quasisimulan anze
                        1246. * will. oder bei Verwendung eines Cursor. z.B. bei CI.CO
                        1247.
                        1248.
000E7C                  1249. autoflip:
000E7C  61BA            1250.           bsr svnc
000E7E  6700 0060       1251.           beq auto2       * kein sync da
000E82                  1252. autoja:                   * einsprung delay
000E82  4A2D 0030       1253.           tst.b flip1(a5) * 4fach swap
000E86  6728            1254.           beq.s auto1
000E88  6100 02C8       1255.           bsr setturt     * nur wenn first
000E8C  102D 0031       1256.           move.b flip1cnt(a5).d0
000E90  B03C 0001       1257.           cmp.b #1.d0
000E94  6616            1258.           bne.s auto11
000E96  1B6D 0030       1259.           move.b flip1(a5).flip1cnt(a5)
        0031
000E9C  522D 0217       1260.           add.b #1.viewpage(a5)
000EA0  022D 0003       1261.           and.b #3.viewpage(a5)
        0217
```

```
000EA6  6100 FF6C      1262.           bsr aktoage     * neue seite anzeigen
000EAA  6004            1263.           bra.s auto1
000EAC  532D 0031       1264. auto11: sub.b #1.flip1cnt(a5)   * veringern
000EB0  4A2D 002E       1265. auto1:  tst.b flip(a5)  * 2fach swap
000EB4  672A            1266.           beq.s auto2
000EB6  6100 029A       1267.           bsr setturt     * nur wenn first
000EBA  102D 002F       1268.           move.b flipcnt(a5).d0
000EBE  B03C 0001       1269.           cmp.b #1.d0
000EC2  6618            1270.           bne.s auto22
000EC4  1B6D 002E       1271.           move.b flip(a5).flipcnt(a5)
        002F
000ECA  0A2D 0001       1272.           eor.b #1.viewpage(a5)
        0217
000ED0  022D 0003       1273.           and.b #3.viewpage(a5)
        0217
000ED6  6100 FF3C       1274.           bsr aktoage
000EDA  4E75            1275.           rts
000EDC  532D 002F       1276. auto22: sub.b #1.flipcnt(a5)
000EE0  4E75            1277. auto2:  rts
                        1278.
                        1279.
                        1280. *****************
                        1281.
                        1282.
00[illegible]  48E7 8080      1283. moveto: movem.l d0/a0.-(a7)
00[illegible]  6100 FE74      1284.           bsr wait        * d1.w = x  d2.w = v
000EEA  41F8 FF70       1285.           lea gdo.a0      * Basisadresse gdo
000EEE  3001            1286.           move d1.d0
000EF0  1140 0009       1287.           move.b d0.cou*9(a0)     * lsb oort x
000EF4  E058            1288.           ror.w #8.d0
000EF6  1140 0008       1289.           move.b d0.cou*8(a0)     * msb oort x
000EFA  3002            1290.           move d2.d0      * lsb oort v
000EFC  1140 000B       1291.           move.b d0.cou*$b(a0)
000F00  E058            1292.           ror.w #8.d0     * msb oort v
000F02  1140 000A       1293.           move.b d0,cou*$a(a0)
000F06  4CDF 0101       1294.           movem.l (a7)+.d0/a0
000F0A  4E75            1295.           rts
                        1296.
000F0C  48E7 0080       1297. getxv:  movem.l a0.-(a7)
000F10  6100 FE4A       1298.           bsr wait        * holt nach d1.w=x und d2.w=v
000F14  207C FFFFFF70   1299.           movea.l #gdo.a0
000F1A  1228 0008       1300.           move.b cou*8(a0).d1     * msb x
000F1E  E159            1301.           rol.w #8.d1
000F20  1228 0009       1302.           move.b cou*9(a0).d1     * lsb x
000F24  1428 000A       1303.           move.b cou*$a(a0).d2
000F28  E15A            1304.           rol.w #8.d2     * msb v
000F2A  1428 000B       1305.           move.b cou*$b(a0).d2
000F2E  E941            1306.           asl #4.d1       * angleich vorzeichen
000F30  48C1            1307.           ext.l d1
000F32  E841            1308.           asr #4.d1
000F34  E942            1309.           asl #4.d2
000F36  48C2            1310.           ext.l d2
000F38  E842            1311.           asr #4.d2       *da gdo nur 12 bits hat
000F3A  4CDF 0100       1312.           movem.l (a7)+.a0
000F3E  4E75            1313.           rts             * lsb v  ergebnis d1.d2
                        1314.
                        1315.
                        1316. * procedure drawto(x.v)
                        1317. * begin
                        1318. *  dx := x-xalt
                        1319. *  dv := v-valt
                        1320. *  if abs(dx)>255 or abs(dv)>255
                        1321. *   drawto(xalt+dx/2,valt+dv/2)
                        1322. *   drawto(x.v)
                        1323. *  else
                        1324. *   draw1(dx.dv)
                        1325. *  endif
                        1326. *
                        1327.
                        1328. * d1.w = x     d2.w = v
                        1329.
000F40                  1330. calcdx:
000F40  3601            1331.           move d1.d3      * xneu
000F42  3802            1332.           move d2.d4      * vneu
000F44  61C6            1333.           bsr getxv       * d1.d2 mit aktuellem wert
000F46  9641            1334.           sub d1.d3       * xneu-xalt  -> dx
000F48  9842            1335.           sub d2.d4       * vneu-valt  -> dv
000F4A  4E75            1336.           rts
                        1337.
000F4C  48E7 FF80       1338. drawto: movem.l d0-d7/a0.-(a7)
```

```
000F50  6106          1339.          bsr.s drawint
000F52  4CDF 01FF     1340.          movem.l (a7)+.d0-d7/a0
000F56  4E75          1341.          rts
                      1342.
000F58                1343. drawint:                    * interne routine
000F58  7E11          1344.          moveq #%00010001.d7      *vz code
000F5A  3A01          1345.          move d1.d5         * x merken
000F5C  3C02          1346.          move d2.d6         * y merken
000F5E  61E0          1347.          bsr calcdx         *d1.d2.d3.d4
000F60  4A43          1348.          tst.w d3           *dx<0
000F62  6A06          1349.          bpl.s dr1
000F64  08C7 0001     1350.          bset #1.d7         *bit 1 setzen
000F68  4443          1351.          neg.w d3
000F6A  4A44          1352. dr1:     tst.w d4           *dy<0
000F6C  6A06          1353.          bpl.s dr2
000F6E  08C7 0002     1354.          bset #2.d7         *bit 2 setzen
000F72  4444          1355.          neg.w d4
000F74  B67C 0100     1356. dr2:     cmp.w #256.d3
000F78  6406          1357.          bcc.s drteil       *teilen noetig
000F7A  B87C 0100     1358.          cmp.w #256.d4
000F7E  6528          1359.          bcs.s noteil       *auch kleiner dann ok
000F80  3205          1360. drteil:  move d5.d1
000F82  3406          1361.          move d6.d2         *d1.d2 sind parameter
000F84  61BA          1362.          bsr calcdx         *d3.d4 sind dann dx.dy
000F86  48C3          1363.          ext.l d3
000F88  48C4          1364.          ext.l d4
000F8A  87FC 0002     1365.          divs #2.d3         *asr nicht gut fuer neg werte
000F8E  89FC 0002     1366.          divs #2.d4         *hier symetrisches ergebnis
000F92  D243          1367.          add.w d3.d1        *xold + dx/2
000F94  D444          1368.          add.w d4.d2        *yold + dy/2
000F96  48A7 0600     1369.          movem.w d5/d6.-(a7)
000F9A  61BC          1370.          bsr drawint
000F9C  4C9F 0060     1371.          movem.w (a7)+.d5/d6
000FA0  3205          1372.          move.w d5.d1
000FA2  3406          1373.          move.w d6.d2       *x.y
000FA4  61B2          1374.          bsr drawint        *recursiv
000FA6  4E75          1375.          rts
                      1376.
000FA8  6100 FDB2     1377. noteil:  bsr wait           *bis ready
000FAC  11C3 FF75     1378.          move.b d3.gdp+5*cpu      *dx
000FB0  11C4 FF77     1379.          move.b d4.gdp+7*cpu      *dy
000FB4  3007          1380.          move d7.d0         *cmd
000FB6  6100 FDAE     1381.          bsr cmd
000FBA  4E75          1382.          rts
                      1383.
                      1384.
                      1385. * Unterprogramme fuer sin.cos
                      1386.
                      1387. * bereich in 0..359 wandeln
                      1388.
000FBC  4A40          1389. adj360:  tst d0             * d0 = 0..359 nachher
000FBE  6B0C          1390.          bmi.s adj1
000FC0  B07C 0168     1391. adjlp:   cmp #360.d0       * wenn kleiner dann ok
000FC4  6B0C          1392.          bmi.s adjfi        * fertig wenn <360
000FC6  907C 0168     1393.          sub #360.d0
000FCA  60F4          1394.          bra adjlp
000FCC  D07C 0168     1395. adj1:    add #360.d0
000FD0  6BFA          1396.          bmi adj1
000FD2  4E75          1397. adjfi:   rts
                      1398.
                      1399.
000FD4                1400. sintab:
000FD4  00 04 09 0D   1401.          dc.b 0.4.9.13.18.22.27.31.36
        12 16 1B 1F 24
000FDD  28 2C 31 35   1402.          dc.b 40.44.49.53.58.62.66.71.75
        3A 3E 42 47 4B
000FE6  4F 53 58 5C   1403.          dc.b 79.83.88.92.96.100.104.108.112
        60 64 68 6C 70
000FEF  74 78 7C 80   1404.          dc.b 116.120.124.128.132.136.139.143.147
        84 88 8B 8F 93
000FF8  96 9A 9E A1   1405.          dc.b 150.154.158.161.165.168.171.175.178
        A5 A8 AB AF B2
001001  B5 B8 BB BE   1406.          dc.b 181.184.187.190.193.196.199.202.204
        C1 C4 C7 CA CC
00100A  CF D2 D4 D7   1407.          dc.b 207.210.212.215.217.219.222.224.226
        D9 DB DE E0 E2
001013  E4 E6 E8 EA  1408.          dc.b 228.230.232.234.236.237.239.241.242
        EC ED EF F1 F2
00101C  F3 F5 F6 F7   1409.          dc.b 243.245.246.247.248.249.250.251.252
        F8 F9 FA FB FC
```

```
001025  FD FE FE FF      1410.          dc.b 253.254.254.255,255.255.0,0,0
        FF FF 00 00 00
00102E  00 00            1411.          dc.b 0.0
                         1412.
                         1413.
                         1414. * werte in d0  bereich in grad
                         1415. * ergebnis auch in d0
                         1416.
001030  D07C 005A        1417. cos:     add #90.d0
001034  6186             1418. sin:     bsr adj360       *abbilden
001036  48E7 4080        1419.          movem.l d1/a0.-(a7)
00103A  4241             1420.          clr d1           *flag
00103C  B07C 005A        1421.          cmp #90.d0
001040  6B24             1422.          bmi.s sin1       *0..89
001042  B07C 00B4        1423.          cmp #180.d0
001046  6B18             1424.          bmi.s sin11      *90..179
001048  08C1 0000        1425.          bset #0.d1       *flag negativ
00104C  B07C 010E        1426.          cmp #270.d0
001050  6B08             1427.          bmi.s sin111     *180..269
001052  907C 0168        1428.          sub #360.d0
001056  4440             1429.          neg d0
001058  600C             1430.          bra.s sin1
00105A  907C 00B4        1431. sin111:  sub #180.d0
00105E  6006             1432.          bra.s sin1
001060  907C 00B4        1433. sin11:   sub #180.d0      *90..179
001064  4440             1434.          neg d0           *89..0
001066  4A40             1435. sin1:    tst d0
001068  671A             1436.          beq.s sin2       *sin(0)=0 grad
00106A  41FA FF68        1437.          lea sintab(pc).a0
00106E  1030 0000        1438.          move.b 0(a0.d0).d0 *neuer wert
001072  C07C 00FF        1439.          and #$ff.d0      * nur lsb gueltig
001076  6604             1440.          bne.s sin12
001078  303C 0100        1441.          move #256.d0     * sin(90) grad
00107C  0801 0000        1442. sin12:   btst #0.d1       * flag
001080  6702             1443.          beq.s sin2
001082  4440             1444.          neg d0           * ergebnis negativ
001084  4CDF 0102        1445. sin2:    movem.l (a7)+.d1/a0
001088  4E75             1446.          rts              * d0 ist ergebnis
                         1447.
                         1448.
                         1449. *********  grap-Unterprogramme
                         1450.
                         1451.
                         1452.
00108A  422D 021E        1453. thoch:   clr.b turdo(a5)
00108E  4E75             1454.          rts
                         1455.
001090                   1456. trunter:
001090  1B7C 0001        1457.          move.b #1.turdo(a5)
        021E
001096  4E75             1458.          rts
                         1459.
001098                   1460. turtab:
001098  FA FD FF FA      1461.          dc.b %11111010.%11111101.%11111111.%11111010.0.0.0.0
        00 00 00 00
0010A0  D3 D0 D0 D4      1462.          dc.b %11010011.%11010000.%11010000.%11010100.%11010100.%11
        D4 D3 00 00
0010A8  FE F9 FD FE      1463.          dc.b %11111110.%11111001.%11111101.%11111110.0.0.0.0
        00 00 00 00
0010B0  D7 D2 D2 D0      1464.          dc.b %11010111.%11010010.%11010010.%11010000.%11010000.%11
        00 D7 00 00
0010B8  FC FB F9 FC      1465.          dc.b %11111100.%11111011.%11111001.%11111100.0.0.0.0
        00 00 00 00
0010C0  D5 D6 D6 D2      1466.          dc.b %11010101.%11010110.%11010110.%11010010.%11010010.%11
        D2 D5 00 00
0010C8  F8 FF FB F8      1467.          dc.b %11111000.%11111111.%11111011.%11111000.0.0.0.0
        00 00 00 00
0010D0  D1 D4 D4 D6      1468.          dc.b %11010001.%11010100.%11010100.%11010110.%11010110.%11
        D6 D1 00 00
                         1469.
0010D8  48E7 E080        1470. turtle:  movem.l d0/d1/d2/a0.-(a7)
0010DC  302D 0050        1471.          move turlohi(a5).d0 * alter wert nehmen
0010E0  48C0             1472.          ext.l d0         * angleich auf 32 bits
0010E2  80FC 002D        1473.          divu #45.d0      * rest in msb.l quotient in d0.lsb
0010E6  C07C 0007        1474.          and #7.d0        * sicherheitshalber
0010EA  E740             1475.          asl #3.d0        * mal 8
0010EC  3400             1476.          move d0.d2
0010EE  41FA FFA8        1477.          lea turtab(pc).a0
0010F2  7207             1478.          moveq #8-1.d1    * counter
0010F4  1030 2000        1479. turlo:   move.b 0(a0.d2).d0
```

```
0010F8  6704          1480.          beq.s tusko
0010FA  6100 FC6A     1481.          bsr cmd          * nur wenn ungleich 0 ausgeben
0010FE  5242          1482. tusko:   addq #1.d2       * displacement
001100  51C9 FFF2     1483.          dbra d1.turlp    * bis ende
001104  4CDF 0107     1484.          movem.l (a7)+.d0/d1/d2/a0
001108  4E75          1485.          rts
                      1486.
00110A                1487. grapoff:
00110A  1B7C 0001     1488.          move.b #1.first(a5)     * keine turtle mehr sichtbar
        021D
001110  4E75          1489.          rts
                      1490.
001112                1491. hide:                    * Turtle aus
001112  4A2D 021D     1492.          tst.b first(a5)
001116  6704          1493.          beq.s hide1
001118  6100 0228     1494.          bsr firsttime   *register bleiben
00111C                1495. hide1:
00111C  422D 002E     1496.          clr.b flip(a5)
001120  422D 0030     1497.          clr.b flip1(a5)
001124  422D 0217     1498.          clr.b viewpage(a5)      * Seite 0 defaulten
001128  6100 FCEA     1499.          bsr aktpage             * als Leseseite
00112C  4E75          1500.          rts
                      1501.
00112E                1502. show:
00112E  4A2D 021D     1503.          tst.b first(a5)
001132  6704          1504.          beq.s show1
001134  6100 020C     1505.          bsr firsttime   *register bleiben
001138                1506. show1:
001138  1B7C 0001     1507.          move.b #1.flip(a5)
        002E
00113E  1B7C 0001     1508.          move.b #1.flipcnt(a5)
        002F
001144  422D 0217     1509.          clr.b viewpage(a5)
001148  422D 0218     1510.          clr.b wrtpage(a5)
00114C  6100 FCC6     1511.          bsr aktpage     * STD-Einstellung
001150  4E75          1512.          rts
                      1513.
                      1514.                          * rev 4.0 Penstatus merken
001152                1515. setturt:
001152  4A2D 021D     1516.          tst.b first(a5)
001156  6600 0964     1517.          bne carset      *aus geschaltet
00115A  48E7 E000     1518.          movem.l d0-d2.-(a7)
00115E  6100 FBFC     1519.          bsr wait
001162  1038 FF71     1520.          move.b gdp+1*cou.d0
001166  3F00          1521.          move.w d0.-(a7) * Penstatus ablegen
001168  322D 004C     1522.          move turlx(a5).d1
00116C  342D 004E     1523.          move turlv(a5).d2
001170  E841          1524.          asr #4.d1
001172  EA42          1525.          asr #5.d2       *normieren
001174  6100 FD6C     1526.          bsr moveto      *gdp position
001178  08ED 0000     1527.          bset #0.wrtpage(a5) *seite 1.3 waehlen
        0218
00117E  6100 FC94     1528.          bsr aktpage
001182  6100 FBEA     1529.          bsr eragen
001186  6100 FF50     1530.          bsr turtle
00118A  3B6D 004A     1531.          move turohi(a5).turlohi(a5)
        0050
001190  3B6D 0046     1532.          move turx(a5).turlx(a5)
        004C
001196  3B6D 0048     1533.          move turv(a5).turlv(a5)
        004E
00119C  322D 0046     1534.          move turx(a5).d1
0011A0  342D 0048     1535.          move turv(a5).d2
0011A4  E841          1536.          asr #4.d1       */16
0011A6  EA42          1537.          asr #5.d2       */32
0011A8  6100 FD38     1538.          bsr moveto
0011AC  6100 FBCE     1539.          bsr setpen
0011B0  6100 FF26     1540.          bsr turtle
0011B4  08AD 0000     1541.          bclr #0.wrtpage(a5)
        0218
0011BA  6100 FC58     1542.          bsr aktpage     * wieder 0.2-seite verwenden
0011BE  6100 FB9C     1543.          bsr wait
0011C2  301F          1544.          move.w (a7)+.d0
0011C4  11C0 FF71     1545.          move.b d0.gdp+1*cou * Und Pen zurueck
0011C8  4CDF 0007     1546.          movem.l (a7)+.d0-d2
0011CC  4E75          1547.          rts
                      1548.
                      1549. * rev 4.0
                      1550.
                      1551.
```

RND

```
0011CE                                  1552. rnd:              * Fuer Experimenten mit Zufallszahlen
                                        1553.                   * liefert 0..ffff in d0.l
                                        1554.                   * rev 4.2 zus. Knuth Algorithmus.
                                        1555.                   * d1.w gibt den Bereich n+1 an. damit
                                        1556.                   * limit festlegbar. Beispiel RND 6
                                        1557.                   * liefert Zahlen zwischen 0 und 5
                                        1558.                   * Wenn D1.L =$10000 also D1.W=0. dann max
                                        1559.                   * Bereich verfuegbar.
0011CE  48E7 6000                       1560.           movem.l d1/d2.-(a7)
0011D2  3200                            1561.           move d0.d1
0011D4  342D 0A36                       1562.           move rnd3var(a5).d2     * zus. RND fuer grosse Bereiche
0011D8  C4FC 7AB5                       1563.           mulu #31413.d2  * mod 8 = 5 ist wichtig hier
0011DC  D47C 6A2E                       1564.           add #27182.d2   * fuer Startwert. nur msb-Teil ist RND
0011E0  3B42 0A36                       1565.           move d2.rnd3var(a5)     * wieder als neuer Startwert
                                        1566.                           * ok. dann RDK-Algorithmus
0011E4  526D 0A32                       1567.           addq.w #1.rnd1var(a5)
0011E8  302D 0A34                       1568.           move rnd2var(a5).d0
0011EC  D06D 0A32                       1569.           add rnd1var(a5).d0
0011F0  6100 FE42                       1570.           bsr sin
0011F4  D06D 0A34                       1571.           add rnd2var(a5).d0
0011F8  E758                            1572.           rol #3.d0
0011FA  B340                            1573.           eor d1.d0
0011FC  3B40 0A34                       1574.           move d0.rnd2var(a5)
001200  B540                            1575.           eor d2.d0               * Beide RND-Zahlen EXOR gibt
00120  C0BC 0000FFFF                    1576.           and.l #$ffff.d0 * immer Zuf. wenn eine von beiden zuf. ist
001  4A41                               1577.           tst.w d1                * Divisor=0. dann max Zufall liefern
00120  6700 0006                        1578.           beq rnd1                * und D0 einfach stehen lassen.
00120E  80C1                            1579.           divu d1.d0              * dann Bereich abbilden
001210  4840                           1580.           swap d0                 * Modulo Rest ist Ergebnis
001212  C0BC 0000FFFF                   1581. rnd1:     and.l #$ffff.d0 * Langwort ist gueltig.
001218  4CDF 0006                       1582.           movem.l (a7)+.d1/d2
00121C  4E75                            1583.           rts
                                        1584.
                                        1585.
00121E                                  1586. aufk:             * Aufkurs Winkel. d0.l
00121E  426D 004A                       1587.           clr.w turphi(a5)
001222  6000 00D2                       1588.           bra tdrehe      * Wie relatives drehen. nur Startwinkel=0
                                        1589.
001226                                  1590. getk:
001226  4240                            1591.           clr.w d0                * d0.l ist dann Winkel
001228  302D 004A                       1592.           move.w turphi(a5).d0
00122C  4E75                            1593.           rts
                                        1594.
00122E                                  1595. aufxv:            * d1=x neu, d2= vneu
00122E  48E7 6000                       1596.           movem.l d1/d2.-(a7)
001232  4A2D 021D                       1597.           tst.b first(a5)
001236  6704                            1598.           beq.s auf1xv
001238  6100 0108                       1599.           bsr firsttime   *register bleiben
00123C                                  1600. auf1xv:
00123C  322D 0046                       1601.           move turx(a5).d1
001240  342D 0048                       1602.           move turv(a5).d2
001244  E841                            1603.           asr #4.d1               */16
001246  EA42                            1604.           asr #5.d2               */32
001248  6100 FC98                       1605.           bsr moveto              *fuer drawto alte pos setzen
00124C  4CDF 0006                       1606.           movem.l (a7)+.d1/d2
001250  4A2D 021E                       1607.           tst.b turdo(a5)
001254  670E                            1608.           beq.s auf1exe
001256  48E7 6000                       1609.           movem.l d1/d2.-(a7)
00125A  E282                            1610.           asr.l #1.d2             * v/2
00125C  6100 FCEE                       1611.           bsr drawto              *linie zeichnen
001260  4CDF 0006                       1612.           movem.l (a7)+.d1/d2
001  E941                               1613. auf1exe:
001266  3B41 0046                       1614.           asl #4.d1
00126A  E942                            1615.           move.w d1.turx(a5)
00126C  3B42 0048                       1616.           asl #4.d2
001270  6100 FC0A                       1617.           move.w d2.turv(a5)
001274  4E75                            1618.           bsr autoflip
001276                                  1619.           rts
001276  322D 0046                       1620. korxv:                    * Absol. Position
00127A  342D 0048                       1621.           move.w turx(a5).d1
00127E  E841                            1622.           move.w turv(a5).d2
001280  E842                            1623.           asr #4.d1
001282  48C1                            1624.           asr #4.d2
001284  48C2                            1625.           ext.l d1
001286  4E75                            1626.           ext.l d2
                                        1627.           rts
                                        1628.
                                        1629.
                                        1630. * ende rev 4.0
                                        1631.
```

```
001288
001288  E940
00128A
00128A  48A7 8000
00128E  4A2D 021D
001292  6704
001294  6100 00AC
001298  322D 0046
00129C  342D 0048
0012A0  E841
0012A2  EA42
0012A4  6100 FC3C
0012A8  4C9F 0001
0012AC  3200
0012AE  302D 004A
0012B2  6100 FD7C
0012B6  C1C1
0012B8  81FC 0100
0012BC  D06D 0046
0012C0  3B40 0046
0012C4  302D 004A
0012C8  6100 FD6A
0012CC  C1C1
0012CE  81FC 0100

0012D2  D06D 0048
0012D6  3B40 0048
0012DA  4A2D 021E
0012DE  6710
0012E0  322D 0046
0012E4  342D 0048
0012E8  E841
0012EA  EA42
0012EC  6100 FC5E
0012F0  6100 FB8A
0012F4  4E75

0012F6  48A7 8000
0012FA  4A2D 021D
0012FE  6704
001300  6100 0040
001304  4C9F 0001
001308  D06D 004A
00130C  6100 FCAE
001310  3B40 004A
001314  6100 FB66
001318  4E75


00131A  E941
00131C  E942
00131E  3B41 0046
001322  3B42 0048
001326  3003
001328  6100 FC92
00132C  3B40 004A
001330  1B7C 0001
        002E
001336  1B7C 0001
        002F
00133C  422D 021D
001340  4E75


001342
001342  48E7 F000
001346  323C 0100
00134A  343C 0100
00134E  363C 005A
001352  61C6
001354  422D 0218
001358  422D 0217
00135C  6100 FAB6
001360  4CDF 000F
001364  4E75
```

```
1632. * setturt nur in autoflip verwendet
1633.
1634. tschreite:                *d0 = delta w
1635.           asl.w #4.d0
1636. tschr16tel:               *fein Schritt
1637.           movem d0.-(a7)
1638.           tst.b first(a5)
1639.           beq.s tsch1
1640.           bsr firsttime   *register bleiben
1641. tsch1:   move turx(a5).d1
1642.           move turv(a5).d2
1643.           asr #4.d1       */16
1644.           asr #5.d2       */32
1645.           bsr moveto      *fuer drawto alte pos setzen
1646.           movem (a7)+.d0  *parameter zurueck
1647.           move d0.d1      *dort merken
1648.           move turphi(a5).d0      *winkel
1649.           bsr cos         *cos256(phi)
1650.           muls d1.d0      *cos256(phi)*dw
1651.           divs #256.d0    *nicht asr verwenden
1652.           add turx(a5).d0 *cos256(phi)*dw/256+xold
1653.           move d0.turx(a5)        *neuer wert x
1654.           move turphi(a5).d0
1655.           bsr sin         *sin256(phi)
1656.           muls d1.d0      *sin256(phi)*dw
1657.           divs #256.d0    *sin256(phi)*dw/256 nicht asr.l
1658.                           *wegen rundungsfehler
1659.           add turv(a5).d0 *sin256(phi)*dw/256+yold
1660.           move d0.turv(a5)        *neuer wert y
1661.           tst.b turdo(a5)
1662.           beq.s tsch2
1663.           move turx(a5).d1
1664.           move turv(a5).d2
1665.           asr #4.d1       */16 in x-richtung
1666.           asr #5.d2       */32 in y-richtung
1667.           bsr drawto      *linie zeichnen
1668. tsch2:   bsr autoflip     *flip-mode
1669.           rts             *ok ende
1670.
1671. tdrehe:  movem d0.-(a7)
1672.           tst.b first(a5) *d0 = deltaphi
1673.           beq.s tdre1
1674.           bsr firsttime   *register bleiben
1675. tdre1:   movem (a7)+.d0
1676.           add turphi(a5).d0       *phi := phi + deltaphi
1677.           bsr adj360
1678.           move d0.turphi(a5)      *ergebnis der drehung
1679.           bsr autoflip
1680.           rts
1681.
1682.
1683. * d1=x d2=y d3=phi fuer direktes setzen
1684.
1685. tmove:   asl.w #4.d1     *x * 16
1686.           asl.w #4.d2     *y * 16
1687.           move d1.turx(a5)
1688.           move d2.turv(a5)        *x.y ablegen
1689.           move d3.d0
1690.           bsr adj360              *normieren
1691.           move d0.turphi(a5)      *ok
1692.           move.b #1.flip(a5)      *schnellste rate

1693.           move.b #1.flipcnt(a5)

1694.           clr.b first(a5)         *nicht mehr first
1695.           rts
1696.
1697.
1698.
1699. firsttime:                *register unveraendert
1700.           movem.l d0-d3.-(a7)
1701.           move #256.d1    * x=256
1702.           move #256.d2    * y=256
1703.           move #90.d3     * phi=90 grad
1704.           bsr tmove
1705.           clr.b wrtpage(a5)       * SEITE 0.1 ist std
1706.           clr.b viewpage(a5)
1707.           bsr aktpage     * STD-Einstellung
1708.           movem.l (a7)+.d0-d3
1709.           rts
```

TEXT10

```
                                1710.
                                1711.
                                1712. ************************************************
                                1713.
                                1714.
                                1715. * print etc
                                1716.
                                1717. * print    d0=size d1=xpos d2=ypos a0=adresse text endet bei 0
                                1718.
001366  6100 FB7A               1719. setort: bsr moveto       * erst mal positionieren
00136A  6100 F9F0               1720.         bsr wait
00136E  4238 FF72               1721.         clr.b gdp+2*cpu * zeichen gerade aufrecht
001372  11C0 FF73               1722.         move.b d0,gdp+3*cpu * scalierung x,y
001376  4E75                    1723.         rts
                                1724.
001378  61EC                    1725. print:  bsr setort
00137A                          1726. printlp:
00137A  1018                    1727.         move.b (a0)+,d0 * wert holen
00137C  6706                    1728.         beq.s printfi
00137E  6100 F9E6               1729.         bsr cmd           * an gdp ausgeben. hier binaere Daten
001382  60F6                    1730.         bra printlp       * erlaubt, also auch Kurzvektoren.
                                1731.
001384  4E75                    1732. printfi: rts
                                1733.
[illegible]86                   1734.
                                1735. textaus:                 * fuer texte loeschen der
                                1736.                         * rev 4.3 mi ae oe ue etc.
                                1737.                         * zeichen
                                1738.                         * d0=size d1=x d2=y a0=quelle
                                1739.                         * 0= ende
001386  48E7 E080               1740.         movem.l d0-d2/a0,-(a7)
00138A  61DA                    1741.         bsr setort        * size etc setzen d0,d1,d2 erhalten
00138C  4A10                    1742. textlp: tst.b (a0)
00138E  6700 0028               1743.         beq textf1        * abbruch wenn = 0
001392  6100 F9DA               1744.         bsr erapen
001396  11FC 000A               1745.         move.b #%1010,gdp * befehl rechteck
        FF70
00139C  6100 FB44               1746.         bsr moveto        * und zurueck
0013A0  6100 F9DA               1747.         bsr setpen
0013A4  1018                    1748.         move.b (a0)+,d0 * wert holen
0013A6  6A06                    1749.         bpl.s textllp     * rev 4.3
0013A8  6100 EC56               1750.         bsr putchar       * wenn Sonderzeichen
0013AC  6004                    1751.         bra.s text2lp
0013AE                          1752. textllp:
0013AE  6100 F9B6               1753.         bsr cmd           *und ausgeben
0013B2                          1754. text2lp:
0013B2  6100 FB58               1755.         bsr getxy         *neue position
0013B6  60D4                    1756.         bra textlp        * bis alle zeichen aus
                                1757.                           * auch fuer textprint
0013B8  4CDF 0107               1758. textf1: movem.l (a7)+,d0-d2/a0
0013BC  4E75                    1759.         rts
                                1760.
                                1761.
0013BE                          1762. textprint:                * wie print aber mit 0ah = CRLF
                                1763.                           * parameter d1=x d2=y d0=size
                                1764.                           * a=0 textblock
                                1765.                           * ohne vorloeschen
                                1766.                           * mit Sonderzeichen, rev 4.3
0013BE  48E7 E080               1767.         movem.l d0-d2/a0,-(a7)
0013C2  61A2                    1768.         bsr setort        * werte definieren
[illegible]C4  3600             1769.         move d0,d3        * size merken in d0.b
[illegible]C6  1018             1770. testp1: move.b (a0)+,d0 * holen data
0013C8  67EE                    1771.         beq textf1        *abbruch wenn 0 erkannt
0013CA  B03C 000A               1772.         cmp.b #$0a,d0     *wenn lf dann CRLF ausfuehren
0013CE  6612                    1773.         bne.s testp2      *sonst y * size
0013D0  3003                    1774.         move d3,d0
0013D2  C07C 000F               1775.         and #$F,d0        *y-size maskieren
0013D6  C0FC 000A               1776.         mulu #10,d0       *Y*10
0013DA  9440                    1777.         sub d0,d2         *ykoor = ykoor - ysize*10
0013DC  6100 FB04               1778.         bsr moveto        *wieder setzen
0013E0  60E4                    1779.         bra testp1
                                1780.
0013E2  4A00                    1781. testp2: tst.b d0
0013E4  6A06                    1782.         bpl.s testlp1     * rev 4.3
0013E6  6100 EC18               1783.         bsr putchar       * wenn Sonderzeichen
0013EA  6004                    1784.         bra.s test2p1
0013EC                          1785. testlp1:
0013EC  6100 F978               1786.         bsr cmd           * einfach ausgeben
0013F0                          1787. test2p1:
0013F0  60D4                    1788.         bra testp1
```

```
                                1789.
0013F2                          1790. menueio:                  *a0=ausgabetext
0013F2  422D 0222               1791.           clr.b curon(a5) *cursor off alpha-mode
0013F6  323C 0032               1792.           move #50.d1
0013FA  343C 00B4               1793.           move #180.d2
0013FE  303C 0033               1794.           move #$33.d0    * a0 gegeben
001402  61BA                    1795.           bsr textorint
001404  323C 0032               1796. menlo:    move #50.d1
001408  343C 000A               1797.           move #10.d2
00140C  303C 0033               1798.           move #$33.d0
001410  363C 0002               1799.           move #2.d3      *anzahl
001414  41ED 00F4               1800.           lea einbuf(a5).a0
001418  6100 006C               1801.           bsr textein     *eingabe
00141C  B87C 0001               1802.           cmp #1.d4       *wenn 1 gelesen ok
001420  66E2                    1803.           bne menlo       *terminator in d5
001422  6100 F97C               1804.           bsr clrall      *register d4.d5 ok
001426  102D 00F4               1805.           move.b einbuf(a5).d0 * zeichen holen
00142A  6100 0702               1806.           bsr bucheck     * in Grossbu konvertieren
00142E  4E75                    1807.           rts
                                1808.
                                1809. ** Texteingabe Unterprogramme
                                1810.
001430                          1811. umrande:                  * Zeichnen Textfeld
                                1812.                           * d1=x d2=y d3=anzahl zeichenmax
                                1813.                           * d0=size
                                1814.
001430  48E7 FC00               1815.           movem.l d0-d5.-(a7)
001434  3A00                    1816.           move d0.d5
001436  E858                    1817.           ror #4.d0
001438  C07C 000F               1818.           and #$F.d0      * size x
00143C  C0C3                    1819.           mulu d3.d0      * size x * anzahl
00143E  C0FC 0006               1820.           mulu #6.d0      * size x * anzahl * 6
001442  5440                    1821.           add #2.d0       * size x * anzahl * 6 + 2
001444  3800                    1822.           move d0.d4      * d4 := x
001446  3005                    1823.           move d5.d0
001448  C07C 000F               1824.           and #$F.d0      * size y
00144C  C0FC 0008               1825.           mulu #$8.d0     * size y * 8
001450  5640                    1826.           add #3.d0       * size y * 8 + 3
001452  3A00                    1827.           move d0.d5      * d5 := y
001454  6100 FA8C               1828.           bsr moveto      * setzen position
001458  103C 00D7               1829.           move.b #%11010111.d0
00145C  6100 F908               1830.           bsr cmd         * richtung links unten
001460  6100 FAAA               1831.           bsr getxy
001464  D244                    1832.           add d4.d1       * x + xneu
001466  6100 FAE4               1833.           bsr drawto
00146A  D445                    1834.           add d5.d2
00146C  6100 FADE               1835.           bsr drawto
001470  9244                    1836.           sub d4.d1
001472  6100 FAD8               1837.           bsr drawto
001476  9445                    1838.           sub d5.d2
001478  6100 FAD2               1839.           bsr drawto      *rand ok
00147C  4CDF 003F               1840.           movem.l (a7)+.d0-d5
001480  6100 FA60               1841.           bsr moveto      * alte position
001484  4E75                    1842.           rts
                                1843.
                                1844. * aktueller setpen oder erapen mode wirksam
                                1845. * buffer muss ein zeichen mehr haben als anzahl
                                1846. * rev 3.1 auch fuer ae oe ue  gueltig
                                1847. * je nach optflag
                                1848.
001486                          1849. textein:                  * d0= size  d1=x d2=y d3=anzahl max
                                1850.                           * d4= ergebnis anzahl
                                1851.                           * d5= terminator zeichen
                                1852.                           * a0= ziel
001486  422D 0222               1853.           clr.b curon(a5) *alpha cursor off
00148A  1B40 021C               1854.           move.b d0.sizesto(a5)   *merker
00148E  6100 F8EC               1855.           bsr setpen      * setzend
001492  619C                    1856.           bsr umrande     * rand
001494  6100 FED0               1857.           bsr setort      * parameter in gdp laden
001498  6100 F8C2               1858.           bsr wait        * komolementaer
00149C  6100 F8D0               1859.           bsr erapen
0014A0  3803                    1860.           move d3.d4      *temp
0014A2  303C 000A               1861. texto1:   move #%1010.d0  *block
0014A6  6100 F8BE               1862.           bsr cmd         *ausgeben
0014AA  5304                    1863.           subq.b #1.d4
0014AC  66F4                    1864.           bne texto1
0014AE  6100 FA32               1865.           bsr moveto      *und erneut positionieren
0014B2  4210                    1866.           clr.b (a0)      *zielpointer =0 endezeichen
0014B4  422D 021A               1867.           clr.b syncont(a5)       *20ms zaehler
0014B8  4244                    1868.           clr d4          *anzahl gelesener = 0
```

```
0014BA  48E7 6000
0014BE  6100 F518
0014C2  6700 0082
0014C6  6100 F538
0014CA  3A00
0014CC  B03C 0008
0014D0  6700 000A
0014D4  B03C 007F
0014D8  6600 0024
0014DC
0014DC  6100 00C0
0014E0  4A44
0014E2  67DA
0014E4  5344
0014E6  4220
0014E8  102D 021C
0014EC  E858
0014EE  C07C 000F
0014F2  C0FC 0006
0014F6  9240
0014F8  6100 00A4
0014FC  60C0

    FE  B03C 0020
    02  6A00 000A
001506  6100 0096
00150A  6000 007C
00150E
00150E  4A2D 0228
001512  6704
001514  6100'EAEA
001518
001518  1080
00151A  6100 0082
00151E  6100 F9C2
001522  6100 F858
001526  1018
001528  6A06
00152A  6100'EAD4
00152E  6004
001530
001530  6100 FB34
001534
001534  4210
001536  6100 F9D4
00153A  5244
00153C  B843
00153E  6700 004B
001542  6000 FF7A

001546  6100 F8F0
00154A  6700 FF72
00154E  102D 021A
001552  5200
001554  1B40 021A
001558  B03C 000D
00155C  6600 0016
001560  6100 F9B0
001564  6100 F816
    68  303C 000A
    6C  6100 F7FB
001570  6000 FF4C
001574  B03C 001A
001578  6600 FF44
00157C  6100 0020
001580  422D 021A
001584  6000 FF3B

001588
001588  4CDF 0006
00158C  6100 F7E0
001590  102D 021C
001594  6100 FE9A
001598  6100 F7E2
00159C  4E75

00159E
00159E  6100 F942
0015A2  6100 F7CA
```

```
1869.           movem.l d1/d2,-(a7)
1870. telo2:    bsr csts            *zeichen da ?
1871.           beq teski           *zeichen holen
1872.           bsr ci              *merken terminator
1873.           move d0.d5
1874.           cmp.b #8.d0
1875.           beq teski1
1876.           cmp.b #$7f.d0       *DEL.BS
1877.           bne teski2          *DEL.BS
1878. teski1:
1879.           bsr deloos
1880.           tst d4              *gelesene zeichen
1881.           beq telo2           *nichts mehr da
1882.           subq #1.d4          *-1
1883.           clr.b -(a0)         *loeschen vorgaenger
1884.           move.b sizesto(a5).d0
1885.           ror #4.d0
1886.           and #$F.d0
1887.           mulu #6.d0          *size x * 6
1888.           sub d0.d1           *x-richtung neu
1889.           bsr deloos          *loeschen zeichen dort
1890.           bra telo2           *fertig
1891.
1892.
1893. teski2: cmp.b #32.d0         *<32 dann stop terminator
1894.           bol teski3
1895.           bsr deloos          *cursor off
1896.           bra texfin1
1897. teski3:                      * zeichen lesbare form
1898.           tst.b ootflag(a5)           *=0 dann amerikan zge
1899.           beq.s test13a       * rev 3.1
1900.           bsr getcode         *gof. umkodieren bit 7 setzen
1901. test13a:
1902.           move.b d0.(a0)      * textausgabe
1903.           bsr deloos
1904.           bsr moveto
1905.           bsr setpen
1906.           move.b (a0)+.d0     * data
1907.           bol.s test13b       * rev 3.1
1908.           bsr putchar         * bit 7 gesetzt
1909.           bra.s test13c
1910. test13b:
1911.           bsr cmd             * ausgeben
1912. test13c:
1913.           clr.b (a0)          * ende zeichen
1914.           bsr getxy           * neue koordinate
1915.           addq #1.d4          * aktuelle anzahl
1916.           cmp d3.d4
1917.           beq texfin1         * ende
1918.           bra telo2           * sonst weiter
1919.
1920. teski:   bsr sync            *sync abwarten
1921.           beq telo2           *nein dann ok
1922.           move.b syncent(a5).d0
1923.           addq.b #1.d0
1924.           move.b d0.syncent(a5)
1925.           cmp.b #13.d0
1926.           bne teski2
1927.           bsr moveto
1928.           bsr setpen          *setzen cursor
1929.           move #%1010.d0      *block
1930.           bsr cmd
1931.           bra telo2
1932. teski2:  cmp.b #26.d0
1933.           bne telo2
1934.           bsr deloos
1935.           clr.b syncent(a5)
1936.           bra telo2
1937.
1938. texfin1:
1939.           movem.l (a7)+.d1/d2
1940.           bsr eraoen
1941.           move.b sizesto(a5).d0
1942.           bsr umrande
1943.           bsr setpen
1944.           rts
1945.
1946. deloos:
1947.           bsr moveto
1948.           bsr eraoen
```

```
0015A6  103C 000A        1949.           move.b #%1010.d0
0015AA  6100 F7BA        1950.           bsr cmd
0015AE  6100 F7CC        1951.           bsr setoen
0015B2  4E75             1952.           rts
                         1953.
                         1954. ******
                         1955.
                         1956. * rev 4.0 auch mit 8 Stellen
                         1957.
0015B4                   1958. printv8d:         * a0-> Ziel, d0.l Wert. Ausgabe dezimal
                         1959.                   * mit Vorzeichen
0015B4  4A80             1960.           tst.l d0
0015B6  6A06             1961.           bpl.s print8d
0015B8  10FC 002D        1962.           move.b #'-'.(a0)+ * Minus-Zeichen ablegen
0015BC  4480             1963.           neg.l d0
                         1964.                   * Weiter mit print 8d
0015BE                   1965. print8d:          * Ziel ist Speicher bei Adresse A0
                         1966.                   * Wert ist D0.L
                         1967.                   * Ausgabe ohne Vorzeichen
0015BE  48E7 6000        1968.           movem.l d1/d2.-(a7)
0015C2  2200             1969.           move.l d0.d1
0015C4  4267             1970.           clr.w -(a7)
0015C6  343C 001F        1971. prt8d2: move #32-1.d2
0015CA  4280             1972.           clr.l d0
0015CC  E381             1973. prt8d3: asl.l #1.d1
0015CE  E390             1974.           roxl.l #1.d0
0015D0  B07C 000A        1975.           cmp.w #10.d0
0015D4  6508             1976.           bcs.s prt8d4
0015D6  907C 000A        1977.           sub.w #10.d0
0015DA  08C1 0000        1978.           bset #0.d1
0015DE  51CA FFEC        1979. prt8d4: dbra d2,prt8d3
0015E2  803C 0030        1980.           or.b #'0'.d0
0015E6  3F00             1981.           move.w d0.-(a7)
0015E8  4A81             1982.           tst.l d1
0015EA  66DA             1983.           bne.s prt8d2
0015EC  301F             1984. prt8d5: move.w (a7)+.d0
0015EE  6704             1985.           beq.s prt8d6
0015F0  10C0             1986.           move.b d0.(a0)+
0015F2  60F8             1987.           bra.s prt8d5
0015F4  4CDF 0006        1988. prt8d6: movem.l (a7)+.d1/d2
0015F8  4210             1989.           clr.b (a0)
0015FA  4E75             1990.           rts
                         1991.
0015FC                   1992. print4d:                  * Ausgabe 4 sed-Stellen
                         1993.                           * in dezimal nach (a0)
                         1994.                           * wert in d0.w
                         1995.                           * Ausgabe ohne Vorzeichen
0015FC  1F3C 0000        1996.           move.b #0.-(a7) * merker fuer Ausgabeteil
001600  4840             1997. pr4dlp: swap d0
001602  4240             1998.           clr d0
001604  4840             1999.           swap d0         * msb loeschen
001606  80FC 000A        2000.           divu #10.d0     * /10 -> rest + wert
00160A  4840             2001.           swap d0         * rest ausgeben
00160C  803C 0030        2002.           or.b #'0'.d0
001610  1F00             2003.           move.b d0.-(a7) * auf den Stack
001612  4840             2004.           swap d0
001614  4A40             2005.           tst d0
001616  66E8             2006.           bne pr4dlp      * weiter bis division = 0 ist
001618  101F             2007. pr4d1:  move.b (a7)+.d0
00161A  6700 0006        2008.           beq pr4d2
00161E  10C0             2009.           move.b d0.(a0)+ * ablegen
001620  60F6             2010.           bra pr4d1
                         2011.
001622  4210             2012. pr4d2:  clr.b (a0)      * ende
001624  4E75             2013.           rts
                         2014.
                         2015.
                         2016.
                         2017. ** ausgabe bin in buffer (a0)
                         2018.
001626                   2019. print8b:                  * 1 byte ausgeben d0.b
001626  2F01             2020.           move.l d1.-(a7)
001628  323C 0007        2021.           move #8-1.d1
00162C  0800 0007        2022. prtb1:  btst #7.d0      *=0 dann 0
001630  6600 0008        2023.           bne prtb2
001634  10FC 0030        2024.           move.b #'0'.(a0)+
001638  6004             2025.           bra.s prtb3
00163A  10FC 0031        2026. prtb2:  move.b #'1'.(a0)+
00163E  E318             2027. prtb3:  rol.b #1.d0
001640  51C9 FFEA        2028.           dbra d1,prtb1
```

```
001644  4210                2029.           clr.b (a0)      * last = 0
001646  221F                2030.           move.l (a7)+,d1
001648  4E75                2031.           rts
                            2032.
                            2033.
                            2034.
                            2035. ** ausgabe hex in buffer (a0)
                            2036.
                            2037.
00164A                      2038. print8x:                  * ziel (a0)  0 terminator
                            2039.                           * d0 ist wert
00164A  4840                2040.           swap d0         * untere bits ausgeben
00164C  6100 0012           2041.           bsr print4x
001650  4840                2042.           swap d0
001652  6100 000C           2043.           bsr print4x     *
001656  4E75                2044.           rts
                            2045.
001658                      2046. print6x:
001658  4840                2047.           swap d0
00165A  6100 000C           2048.           bsr print2x
00165E  4840                2049.           swap d0
001660                      2050. print4x:
001660  E098                2051.           ror.l #8,d0
001662  6100 0004           2052.           bsr print2x
001666  E198                2053.           rol.l #8,d0
001668                      2054. print2x:
001668  E898                2055.           ror.l #4,d0
00166A  6100 0004           2056.           bsr print1x
00166E  E998                2057.           rol.l #4,d0
001670                      2058. print1x:                  * lsb ausgeben
                            2059.                           * untere 4 bits
001670  48E7 8000           2060.           movem.l d0,-(a7)
001674  C03C 000F           2061.           and.b #$F,d0
001678  B03C 000A           2062.           cmp.b #10,d0
00167C  6B0A                2063.           bmi.s prt11x
00167E  903C 000A           2064.           sub.b #10,d0
001682  D03C 0041           2065.           add.b #'A',d0
001686  6004                2066.           bra.s prt12x
001688  D03C 0030           2067. prt11x: add.b #'0',d0
00168C  10C0                2068. prt12x: move.b d0,(a0)+
00168E  10BC 0000           2069.           move.b #0,(a0)  *ende zeichen
001692  4CDF 0001           2070.           movem.l (a7)+,d0
001696  4E75                2071.           rts
                            2072.
                            2073.
                            2074. ************************************************
                            2075.
                            2076. * Bildschirm Handling fuer seriellen und
                            2077. * Editor Betrieb
                            2078.
                            2079. * rev 4.1 Routinen fuer Bibo
                            2080.
001698                      2081. crlfe:                    * Ausgabe eines CR LF
001698  103C 000D           2082.           move.b #$d,d0   * auf die CO2-Schnittstelle
00169C  6100 F442           2083.           bsr co2
0016A0  103C 000A           2084.           move.b #$a,d0
0016A4  6000 F43A           2085.           bra co2
                            2086.
0016A8  4280                2087. asserr: clr.l d0          * Fehleranzahl -> d0
0016AA  302D 01FC           2088.           move.w errcnt(a5),d0  * rev 4.2
0016AE  4E75                2089.           rts             * z.B. nach Assembleruebersetzung
                            2090.
0016B0                      2091. getcurxy:       * Danach D1=Curx-Position,D2=Curv-Position
0016B0  4281                2092.           clr.l d1
0016B2  4282                2093.           clr.l d2
0016B4  122D 0223           2094.           move.b curx(a5),d1
0016B8  142D 0224           2095.           move.b curv(a5),d2
0016BC  4E75                2096.           rts
                            2097.
0016BE                      2098. setcurxy:       * Neuer Cursor, D1=Curx, D2=Curv
0016BE  1B41 0223           2099.           move.b d1,curx(a5)
0016C2  1B42 0224           2100.           move.b d2,curv(a5)
0016C6  6100 E938           2101.           bsr aktcur      * Und neu Einstellen
0016CA  4E75                2102.           rts
                            2103.
                            2104.
                            2105. ************************************************
                            2106. *
                            2107. * expression Auswertung fuer Menues
                            2108. * Damit kann man arithmetische Ausdruecke in denen
```

EXPRESS

```
                 * auch Symbole vorkommen auswerten und das Ergebnis
                 * berechnen lassen.
                 ***************************************************

                 ****************************************
                 * a0 ist der Zeiger auf den Buffer     *
                 * d0.l ist anschliessend das ergebnis  *
                 * d1.w = 0   Fehler                    *
                 * d1.w = 1   Bytewert                  *
                 * d1.w = 2   Wortwert                  *
                 * d1.w = 3   Langwort                  *
                 * d1.w = 4   reserviert fuer getea     *
                 * d1.w = 5   Referenz ohne Definition  *
                 * d1.w = FFFF kein Fehler jedoch intern*
                 ****************************************

                 ** rev 3.1  IGBN   streng verwaltet
                 ** zerstoeren ggf. Register
                 ** wegen Geschwindigkeit nicht alles gerettet

0016CC           zuweis:                  * a0 zeigt auf Text
                                          * Zuweisung
                                          * Carry wenn nicht ok
0016CC  6100 00E8        bsr igbn         *leerzeichen ignorieren
0016D0  48E7 0080        movem.l a0,-(a7)        *retten
0016D4  6100 0480        bsr setupname    *name definieren
0016D8  6500 004A        bcs zuwf
0016DC  6100 00D8        bsr igbn         *check nun :=
0016E0  1018             move.b (a0)+,d0
0016E2  B03C 003A        cmp.b #':',d0
0016E6  663C             bne.s zuwf
0016E8  1018             move.b (a0)+,d0
0016EA  B03C 003D        cmp.b #'=',d0
0016EE  6634             bne.s zuwf
0016F0  6100 00C4        bsr igbn
0016F4  6100 0036        bsr wert         *wert holen
0016F8  652A             bcs.s zuwf       *syntax Fehler
                                          *d0.d1 nun definiert
0016FA  2248             move.l a0,a1     *retten dort
0016FC  4CDF 0100        movem.l (a7)+,a0
001700  48E7 00C0        movem.l a0/a1,-(a7)
001704  6100 0616        bsr newval       *Carry = Fehler
001708  640C             bcc.s zuw1
00170A  4CDF 0300        movem.l (a7)+,a0/a1      *altes a0 holen
00170E  48E7 00C0        movem.l a0/a1,-(a7)
001712  6100 0600        bsr setval       *war schon da, einfach setzen
001716  4CDF 0300  zuw1:  movem.l (a7)+,a0/a1

00171A  6500 03A0        bcs carset       *Fehlerausgang a0=start
00171E  2049             move.l a1,a0     *neuer Zeiger hinter Namen
001720  6000 03A0        bra carres       *OK ZUgewiesen


001724  4CDF 0100  zuwf:  movem.l (a7)+,a0
001728  6000 0392        bra carset       *Error no assignement




                 * d0.d1.d2 a0 destroyed, a0 zeigt auf naechstes byte
                 * Carry wenn Syntaxfehler
                 * rev 4.0 auch .b.w.l zugelassen

00172C  323C FFFF  wert:  move #$FFFF,d1   *
001730  6100 0084        bsr igbn         * rev 3.1  HIER IGBN ERLAUBT
001734  6100 008C        bsr expr         *expression holen
001738  4A01             tst.b d1
00173A  6700 0380        beq carset       *fehlerausgang
                                          * hier immer umdefinierbar durch .x
00173E  B27C 0005        cmp #5,d1        * undef dann bleibt es so
001742  6700 006E        beq wertf        *FFFF nicht ausreichend zZ.
001746  0C10 002E        cmp.b #'.',(a0)  *wenn punkt dann .L etc ggf.
00174A  672A             beq.s wert1      *sonst nicht wertabh entscheiden
                                          *laenge dynamisch
00174C  4840       wertwh: swap d0         *eingang wenn nicht punkt W,B,L
00174E  4A40             tst d0           *=0 dann wort oder bytes
001750  670A             beq.s wert12
001752  4840             swap d0
001754  323C 0003  wertlo: move #3,d1      *langwort
001758  6000 0058        bra wertf
```

```
00175C  4840            2189. wert12: swap d0
00175E  2200            2190.         move.l d0,d1
001760  C07C FF00       2191.         and #$FF00,d0    *=0 dann byte
001764  6708            2192.         beq.s wert13
001766  2001            2193.         move.l d1,d0
001768  323C 0002        2194. wertwo: move #2,d1       *Wort
00176C  6044            2195.         bra.s wertf
00176E  2001            2196. wert13: move.l d1,d0
001770  323C 0001        2197. wertbv: move #1,d1       *byte
001774  603C            2198.         bra.s wertf
001776  0C28 006C       2199. wert1:  cmp.b #'l',1(a0)
        0001
00177C  6708            2200.         beq.s wert1a
00177E  0C28 004C       2201.         cmp.b #'L',1(a0)        *,1
        0001
001784  6604            2202.         bne.s wert21
001786                  2203. wert1a:
001786  5488            2204.         addq.l #2,a0
001788  60CA            2205.         bra wertlo
00178A  0C28 0077       2206. wert21: cmp.b #'w',1(a0)
        0001
001790  6708            2207.         beq.s wert21a
001792  0C28 0057       2208.         cmp.b #'W',1(a0)
        0001
00?798  6604            2209.         bne.s wert22
00179A                  2210. wert21a:
00179A  5488            2211.         addq.l #2,a0
00179C  60CA            2212.         bra wertwo
00179E  0C28 0062       2213. wert22: cmp.b #'b',1(a0)
        0001
0017A4  6708            2214.         beq.s wert22a
0017A6  0C28 0042       2215.         cmp.b #'B',1(a0)
        0001
0017AC  669E            2216.         bne.s wertwh     *zurueck ,n nicht gueltig
0017AE                  2217. wert22a:
0017AE  5488            2218.         addq.l #2,a0
0017B0  60BE            2219.         bra wertbv
0017B2  6000 030E       2220. wertf:  bra carres
                        2221.
                        2222.
0017B6                  2223. iqbn:                    *leerzeichen inorieren
0017B6  0C10 0020       2224.         cmp.b #' ',(a0)
0017BA  6604            2225.         bne.s iqbnf
0017BC  5288            2226.         addq.l #1,a0
0017BE  60F6            2227.         bra iqbn
                        2228.
0017C0  4E75            2229. iqbnf:  rts
                        2230.
                        2231.
0017C2  6100 003C       2232. expr:   bsr term
0017C6                  2233. exprwh:                  * rev 3.1 iqbn gestrichen
0017C6  0C10 002B       2234.         cmp.b #'+',(a0) *wenn + folgt
0017CA  670E            2235.         beq.s expradd
0017CC  0C10 002D       2236.         cmp.b #'-',(a0)
0017D0  6714            2237.         beq.s exprsub
0017D2  0C10 0021       2238.         cmp.b #'!',(a0) * rev 4.0 OR
0017D6  671C            2239.         beq.s expror
0017D8  4E75            2240.         rts
                        2241.
0017DA                  2242. expradd:
0017DA  5288            2243.         addq.l #1,a0
0017DC  2F00            2244.         move.l d0,-(a7)
0017DE  6100 0020       2245.         bsr term
0017E2  D09F            2246.         add.l (a7)+,d0
0017E4  60E0            2247.         bra exprwh
                        2248.
0017E6                  2249. exprsub:
0017E6  5288            2250.         addq.l #1,a0
0017E8  2F00            2251.         move.l d0,-(a7)
0017EA  6100 0014       2252.         bsr term
0017EE  909F            2253.         sub.l (a7)+,d0
0017F0  4480            2254.         neg.l d0
0017F2  60D2            2255.         bra exprwh
                        2256.
0017F4                  2257. exproor:
0017F4  5288            2258.         addq.l #1,a0
0017F6  2F00            2259.         move.l d0,-(a7)
0017F8  6100 0006       2260.         bsr term
0017FC  809F            2261.         or.l (a7)+,d0
0017FE  60C6            2262.         bra exprwh
```

```
001800  6100 00F8
001804
001804  0C10 002A
001808  6716
00180A  0C10 002F
00180E  6724
001810  0C10 005C
001814  6740
001816  0C10 0026
00181A  6700 00D0
00181E  4E75

001820
001820  5288
001822  2F00
001824  6100 00D4
001828  241F
00182A  2F01
00182C  6100 0044
001830  221F
001832  60D0


001834
001834  5288
001836  2F00
001838  6100 00C0
00183C  241F
00183E  4A80
001840  670C
001842  C142
001844  2F01
001846  6100 0066
00184A  221F
00184C  60B6
00184E
00184E  203C 7FFFFFFF
001854  60AE

001856
001856  5288
001858  2F00
00185A  6100 009E
00185E  241F
001860  4A80
001862  67EA
001864  C142
001866  2F01
001868  6100 0044
00186C  2001
00186E  221F
001870  6092


001872
001872  48E7 1800
001876  4244
001878  4A80
00187A  6A04
00187C  4480
00187E  7801

001880  4A82
001882  6A06
001884  4482
001886  0A44 0001

00188A  4281
00188C  363C 001F
001890
001890  0800 0000
001894  6702
001896  D282
001898
001898  E281
00189A  E290
00189C  51CB FFF2
0018A0  B87C 0000
        6702
```

```
2263.
2264. term:   bsr faktor
2265. termwh:                   * rev 3.1 iobn getstrichen
2266.         cmp.b #'*'.(a0)
2267.         beq.s termmul
2268.         cmp.b #'/'.(a0)
2269.         beq.s termdiv
2270.         cmp.b #'\'.(a0) * rev 4.0 Modulo
2271.         beq.s termmod
2272.         cmp.b #'&'.(a0)
2273.         beq termand       * rev 4.0
2274.         rts
2275.
2276. termmul:
2277.         addq.l #1,a0
2278.         move.l d0,-(a7)
2279.         bsr faktor
2280.         move.l (a7)+,d2 *rev 4.0 32 Bit
2281.         move.l d1,-(a7) * d0 * d2 -> d0
2282.         bsr muls32      * d0.l ist Ergebnis
2283.         move.l (a7)+,d1
2284.         bra termwh      *
2285.
2286.
2287. termdiv:
2288.         addq.l #1,a0
2289.         move.l d0,-(a7)
2290.         bsr faktor
2291.         move.l (a7)+,d2
2292.         tst.l d0
2293.         beq.s term1div  * rev 4.0 32 Bits
2294.         exg d0,d2       * d0 / d2 -> d0
2295.         move.l d1,-(a7)
2296.         bsr divs32      * d0.l Ergebnis
2297.         move.l (a7)+,d1
2298.         bra termwh      *aber vorzeichenbehaftet
2299. term1div:               * /0   -----
2300.         move.l #$7fffffff,d0
2301.         bra termwh
2302.
2303. termmod:
2304.         addq.l #1,a0
2305.         move.l d0,-(a7)
2306.         bsr faktor
2307.         move.l (a7)+,d2
2308.         tst.l d0
2309.         beq.s term1div  * undef
2310.         exg d0,d2
2311.         move.l d1,-(a7)
2312.         bsr divs32      * d1.l ist REST
2313.         move.l d1,d0
2314.         move.l (a7)+,d1
2315.         bra termwh
2316.
2317.
2318. muls32:   * d0 * d2 -> D0  (d1 ueberlauf)
2319.         movem.l d3/d4,-(a7)
2320.         clr d4    * 0=pos
2321.         tst.l d0
2322.         if <mi> then.s
2323.            neg.l d0
2324.            moveq #1,d4
2325.         endi
2326.         tst.l d2
2327.         if <mi> then.s
2328.            neg.l d2
2329.            eor.w #1,d4
2330.         endi
2331.         clr.l d1
2332.         move #32-1,d3
2333. mulslp:
2334.         btst #0,d0
2335.         beq.s mulsll
2336.         add.l d2,d1
2337. mulsll:
2338.         asr.l #1,d1
2339.         roxr.l #1,d0
2340.         dbra d3,mulslp
2341.         if d4 <ne> #0 then.s
```

```
0018A6  4480                2342.            neg.l d0
                            2343.         endi
0018A8  4CDF 0018           2344.         movem.l (a7)+,d3/d4
0018AC  4E75                2345.         rts
                            2346.
                            2347.
0018AE                      2348. divs32:   * d0 (d1) / d2 -> d1 rest  d0 Ergebnis
0018AE  48E7 1800           2349.         movem.l d3/d4,-(a7)
0018B2  4244                2350.         clr d4    * 0=pos
0018B4  4A80                2351.         tst.l d0
0018B6  6A04                2352.         if <mi> then.s
0018B8  4480                2353.            neg.l d0
0018BA  7801                2354.            moveq #1,d4
                            2355.         endi
0018BC  4A82                2356.         tst.l d2
0018BE  6A06                2357.         if <mi> then.s
0018C0  4482                2358.            neg.l d2
0018C2  0A44 0001           2359.            eor.w #1,d4
                            2360.         endi
0018C6  4281                2361.         clr.l d1
0018C8  363C 001F           2362.         move #32-1,d3
0018CC                      2363. divslp:
0018CC  E380                2364.         asl.l #1,d0
0018CE  E391                2365.         roxl.l #1,d1
0018D0  B282                2366.         cmp.l d2,d1
0018D2  6506                2367.         bcs.s divsll
0018D4  9282                2368.         sub.l d2,d1
0018D6  08C0 0000           2369.         bset #0,d0
0018DA                      2370. divsll:
0018DA  51CB FFF0           2371.         dbra d3,divslp
0018DE  B87C 0000           2372.         if d4 <ne> #0 then.s
        6702
0018E4  4480                2373.            neg.l d0
                            2374.         endi
0018E6  4CDF 0018           2375.         movem.l (a7)+,d3/d4
0018EA  4E75                2376.         rts
                            2377.
                            2378.
                            2379.
                            2380.
0018EC                      2381. termand:
0018EC  5288                2382.         addq.l #1,a0
0018EE  2F00                2383.         move.l d0,-(a7)
0018F0  6100 0008           2384.         bsr faktor
0018F4  C09F                2385.         and.l (a7)+,d0
0018F6  6000 FF0C           2386.         bra termwh
                            2387.
0018FA                      2388. faktor:                 *wenn name rev 3.1 igbn gestrichen
0018FA  0C10 0027           2389.         cmp.b #$27.(a0) * TEXTTEIL
0018FE  6626                2390.         bne.s fakto11
001900  5288                2391.         addq.l #1,a0    * bis Ende
001902  4280                2392.         clr.l d0        * mit 0en fuellen
001904                      2393. fakto13:
001904  0C10 0027           2394.         cmp.b #$27.(a0) * ENDE
001908  670C                2395.         beq.s fakto12
00190A  0C10 0020           2396.         cmp.b #' ',(a0) * < 20h rev 4.1 auch So-Zeichen
00190E  6508                2397.         bcs.s fakto14   * auch Endezeichen aber Fehlerfall
001910  E198                2398.         rol.l #8,d0     * von rechts nach links  fuellen
001912  1018                2399.         move.b (a0)+,d0 * Maskieren in D0
001914  60EE                2400.         bra fakto13
                            2401.
001916                      2402. fakto12:
001916  5288                2403.         addq.l #1,a0
001918                      2404. fakto14:
001918  B27C 0005           2405.         cmp #5,d1
00191C  6700 01A4           2406.         beq carres
001920  323C 0003           2407.         move #3,d1      * LONG annehmen
001924  4E75                2408.         rts
                            2409.
001926                      2410. fakto11:
001926  0C10 002A           2411.         cmp.b #'*',(a0)
00192A  6614                2412.         bne.s faktok1   * ANFSTAND hierfuer verwenden
00192C  5288                2413.         addq.l #1,a0    * rev 4.0 .1
00192E  202D 0056           2414.         move.l anfstand(a5),d0
001932  B27C 0005           2415.         cmp #5,d1
001936  6700 018A           2416.         beq carres      * auf undef lassen
00193A  323C 0003           2417.         move #3,d1      * als Langw definiert
00193E  4E75                2418.         rts
                            2419.
001940                      2420. faktok1:                * REV 3.1  STXTXT Erreichbar
```

```
001940  0C10 003F      2421.           cmp.b #'?',(a0)
001944  6614           2422.           bne.s fakto1
001946  5288           2423.           addq.l #1,a0    * Next
001948  202D 0036      2424.           move.l stxtxt(a5),d0
00194C  B27C 0005      2425.           cmp #5,d1
001950  6700 0170      2426.           beq carres      * auf undef lassen
001954  323C 0003      2427.           move #3,d1      * als Langw definiert
001958  4E75           2428.           rts
                       2429.
                       2430.
00195A                 2431. fakto1:
00195A  1010           2432.           move.b (a0),d0  *A..Z zugelassen nur
00195C  6100 01D0      2433.           bsr bucheck     *00..nn sind Zahlen $xx - ( etc.
001960  6500 003C      2434.           bcs fakt10      *Nur A..Z und a..z als Namensanfang ok
001964  48E7 0080      2435.           movem.l a0,-(a7)        * retten temp
001968  B27C 0005      2436.           cmp #5,d1
00196C  670E           2437.           beq.s fakto01
00196E  6100 039C      2438.           bsr getval      *a0 zeigt dann hinter den Namen
001972  4A42           2439.           tst d2          *=0 dann Name gefunden
001974  6618           2440.           bne.s fakt101
001976  4CDF 0200      2441.           movem.l (a7)+,a1        * entfernen a0 bleibt queltio
00197A  4E75           2442.           rts             *carry wenn syntax
                       2443.                           *oder d0=wert d1=attribut
                       2444.                           *d1=0 dann fehler
00197C                 2445. fakto01:
00197C  6100 038E      2446.           bsr getval      *a0 zeigt dann hinter den Namen
001980  4A42           2447.           tst d2          *=0 dann Name gefunden
001982  660A           2448.           bne.s fakt101
001984  4CDF 0200      2449.           movem.l (a7)+,a1        * entfernen a0 bleibt queltio
001988  323C 0005      2450.           move #5,d1      * wenn undef vorher dann bleibt es undef
00198C  4E75           2451.           rts             *carry wenn syntax
                       2452.
                       2453.
00198E                 2454. fakt101:                 *Name nicht eingetragen
00198E  4CDF 0100      2455.           movem.l (a7)+,a0        * a0 zurueck auf Anfang
001992  4280           2456.           clr.l d0        * WERT = 0 eintragen
001994  323C 0005      2457.           move #5,d1      * Attribut = 5  Referenz ohne Definition
001998  6100 0382      2458.           bsr newval      * und definieren, danach a0 hinter namen
00199C  4E75           2459.           rts             * name auch nicht definieren
                       2460.                           * carry = fehler system
                       2461.
00199E                 2462. fakt10:                  *kein name
00199E  0C10 0028      2463.           cmp.b #'(',(a0)
0019A2  6600 0016      2464.           bne fakt1
0019A6  5288           2465.           addq.l #1,a0    *ok weiter
0019A8  6100 FE18      2466.           bsr expr        *recursiv
                       2467.                           *rev 3.1 jabn gestrichen
0019AC  0C10 0029      2468.           cmp.b #')',(a0) *muss schliessen sonst fehler
0019B0  6604           2469.           bne.s fakerr
0019B2  5288           2470.           addq.l #1,a0
0019B4  4E75           2471.           rts
                       2472.
0019B6  4241           2473. fakerr: clr d1
0019B8  4E75           2474.           rts
                       2475.
0019BA                 2476. fakt1:
0019BA  0C10 007E      2477.           cmp.b #'~',(a0) * rev 4.0 NOT
0019BE  660A           2478.           bne.s faktoa11
0019C0  5288           2479.           addq.l #1,a0
0019C2  6100 FF36      2480.           bsr faktor
0019C6  4680           2481.           not.l d0        * NOT ausfuehren
0019C8  4E75           2482.           rts
                       2483.
0019CA                 2484. faktoa11:
0019CA  0C10 002D      2485.           cmp.b #'-',(a0)
0019CE  6600 000C      2486.           bne fakt2
0019D2  5288           2487.           addq.l #1,a0
0019D4  6100 FF24      2488.           bsr faktor      *rekursiv
0019D8  4480           2489.           neg.l d0        *negativ machen
0019DA  4E75           2490.           rts
                       2491.
0019DC  0C10 0024      2492. fakt2:  cmp.b #'$',(a0)
0019E0  6600 0036      2493.           bne fakt3       *
0019E4  5288           2494.           addq.l #1,a0    *$xxxx muss sedezimal sein
0019E6  1410           2495.           move.b (a0),d2  *neue zahl
0019E8  6100 00F0      2496.           bsr sedcheck
0019EC  65C8           2497.           bcs fakerr      *fehler
0019EE  5288           2498.           addq.l #1,a0
0019F0  4280           2499.           clr.l d0
0019F2                 2500. fakt32:
```

```
0019F2  E980            2501.           asl.l #4,d0     *mal 16
0019F4  C4BC 000000FF   2502.           and.l #$FF,d2
0019FA  D082            2503.           add.l d2,d0
0019FC  1410            2504.           move.b (a0),d2
0019FE  6100 00DA       2505.           bsr sedcheck
001A02  6500 0006       2506.           bcs fakt34
001A06  5288            2507.           addq.l #1,a0
001A08  60E8            2508.           bra fakt32
                        2509.
001A0A  B27C 0005       2510. fakt34:   cmp #5,d1
001A0E  6700 00B2       2511.           beq carres
001A12  323C 0003       2512.           move #3,d1      *Long annehmen
001A16  4E75            2513.           rts
                        2514.
                        2515.
001A18                  2516. fakt3:
001A18  0C10 0025       2517.           cmp.b #'%',(a0)
001A1C  6600 0032       2518.           bne fakt3a      * Binaere Eingabe
001A20  5288            2519.           addq.l #1,a0
001A22  1410            2520.           move.b (a0),d2  * wert holen
001A24  B43C 0031       2521.           cmp.b #'1',d2
001A28  6706            2522.           beq.s fakt310
001A2A  B43C 0030       2523.           cmp.b #'0',d2
001A2E  6686            2524.           bne fakerr      * Fehler wenn nicht 0,1
001A30                  2525. fakt310:
001A30  5288            2526.           addq.l #1,a0
001A32  4280            2527.           clr.l d0        * Ergebnis
001A34                  2528. fakt320:
001A34  E380            2529.           asl.l #1,d0
001A36  C4BC 00000001   2530.           and.l #$1,d2    * rev 4.0 korrektur .1
001A3C  D082            2531.           add.l d2,d0     * d0 := d0*2 + d2
001A3E  1410            2532.           move.b (a0),d2
001A40  B43C 0031       2533.           cmp.b #'1',d2
001A44  6706            2534.           beq.s fakt330
001A46  B43C 0030       2535.           cmp.b #'0',d2
001A4A  66BE            2536.           bne fakt34      * OK gueltig
001A4C                  2537. fakt330:
001A4C  5288            2538.           addq.l #1,a0
001A4E  60E4            2539.           bra fakt320
                        2540.
                        2541.
001A50                  2542. fakt3a:
001A50  0C10 0040       2543.           cmp.b #'@',(a0)
001A54  6600 000E       2544.           bne fakt3b      * Adresse von trapsymbol
001A58  5288            2545.           addq.l #1,a0
001A5A  6100 17D8       2546.           bsr gettrap     * a0-> ende d0=wertadresse d2=index
001A5E  6500 001A       2547.           bcs fakundef
001A62  60A6            2548.           bra fakt34      * wie wert
                        2549.
001A64                  2550. fakt3b:
001A64  0C10 0021       2551.           cmp.b #'!',(a0)
001A68  6600 0016       2552.           bne fakt3c      * Index von Trapsymbol
001A6C  5288            2553.           addq.l #1,a0
001A6E  6100 17C4       2554.           bsr gettrap
001A72  6500 0006       2555.           bcs fakundef
001A76  3002            2556.           move d2,d0
001A78  6090            2557.           bra fakt34
                        2558.
001A7A                  2559. fakundef:
001A7A  323C 0005       2560.           move #5,d1
001A7E  4E75            2561.           rts             * undefiniertes Symbol
                        2562.
001A80                  2563. fakt3c:
001A80  1410            2564.           move.b (a0),d2
001A82  6100 0044       2565.           bsr dezcheck    *check ob dezimal
001A86  6500 FF2E       2566.           bcs fakerr      *nein dann fehler
001A8A  5288            2567.           addq.l #1,a0    *next
001A8C  4280            2568.           clr.l d0        *ergebnis = 0 setzen
001A8E                  2569. fakt21:                   *schleife eingang
001A8E  2600            2570.           move.l d0,d3
001A90  D080            2571.           add.l d0,d0
001A92  D080            2572.           add.l d0,d0
001A94  D083            2573.           add.l d3,d0
001A96  D080            2574.           add.l d0,d0     *d0.l * 10
001A98  C4BC 000000FF   2575.           and.l #$FF,d2
001A9E  D082            2576.           add.l d2,d0     *d0.l * 10 + d2.b
001AA0  1410            2577.           move.b (a0),d2
001AA2  6100 0024       2578.           bsr dezcheck
001AA6  6500 0006       2579.           bcs fakt22
001AAA  5288            2580.           addq.l #1,a0    *next
```

```
001AAC  60E0                2581.           bra fakt21
                            2582.
001AAE  B27C 0005           2583. fakt22:   cmp #5,d1
001AB2  6700 000E           2584.           beq carres
001AB6  323C 0003           2585.           move #3,d1      *Long annehmen
001ABA  4E75                2586.           rts
                            2587.
                            2588.
                            2589.
                            2590. ****************** Globale Routinen
001ABC  003C 0001           2591. carset:   or #1,ccr
001AC0  4E75                2592.           rts
                            2593.
001AC2  023C 0000           2594. carres:   and #0,ccr
001AC6  4E75                2595.           rts
                            2596. ******************
                            2597.
                            2598.
                            2599.
001AC8                      2600. dezcheck:                 *wert ascii in d2
                            2601.                           *carry wenn fehler sonst conv
001AC8  B43C 0030           2602.           cmp.b #'0',d2
001ACC  65EE                2603.           bcs carset      * rev 4.1 cs.cc wegen So-Zeichen
001ACE  B43C 003A           2604.           cmp.b #'9'+1,d2
001AD2  64EB                2605.           bcc carset
001AD4  943C 0030           2606.           sub.b #'0',d2
001AD8  60EB                2607.           bra carres
                            2608.
001ADA                      2609. sedcheck:                 *wert ascii in d2
                            2610.                           *carry wenn fehler sonst conv
001ADA  B43C 0030           2611.           cmp.b #'0',d2
001ADE  65DC                2612.           bcs carset      * rev 4.1
001AE0  B43C 003A           2613.           cmp.b #'9'+1,d2
001AE4  6400 0008           2614.           bcc sed1        * rev 4.1
001AE8  943C 0030           2615.           sub.b #'0',d2
001AEC  60D4                2616.           bra carres
001AEE  B43C 0041           2617. sed1:     cmp.b #'A',d2
001AF2  65C8                2618.           bcs carset
001AF4  B43C 0047           2619.           cmp.b #'F'+1,d2
001AF8  6400 000C           2620.           bcc sed2
001AFC  943C 0041           2621.           sub.b #'A',d2
001B00  D43C 000A           2622.           add.b #10,d2
001B04  60BC                2623.           bra carres
001B06  B43C 0061           2624. sed2:     cmp.b #'a',d2
001B0A  65B0                2625.           bcs carset
001B0C  B43C 0067           2626.           cmp.b #'f'+1,d2
001B10  64AA                2627.           bcc carset
001B12  943C 0061           2628.           sub.b #'a',d2
001B16  D43C 000A           2629.           add.b #10,d2
001B1A  60A6                2630.           bra carres
                            2631.
```

```
2632. ** TEIL B
2633.
2634.          include grundb
2635: ****************************************
2636: * Grundprogramm 68k 2.0    TEIL B      *
2637: * RDK 3.12.1983     Rolf-Dieter Klein  *
2638: * rev 831211        figur             *
2639: * rev 840129  rev 3.0 Erweiterungen    *
2640: * rev 840228  rev 3.1                  *
2641: * rev 840903  rev 4.0 CPUs             *
2642: * rev 840908  rev 4.0b relokativ       *
2643: * rev 840929  rev 4.1  Autoload        *
2644: * rev 841016  rev 4.2 div.             *
2645: * rev 850117  rev 4.3 ri in cas laden *
2646: ****************************************
2647:
2648: *** Symboltabellenverwaltung
2649:
2650:
2651: * Die Symboltabelle ist mit festen Recordlaengen
2652: * aufgebaut um umdefinieren von Symbolen wie
2653: * die Bearbeitung zu erleichtern
2654: * Die Symboltabelle muss auf einer Wortgrenze
2655: * anfangen
2656: * A1 ist der Globale Zeiger auf den Start der Tabelle
2657: *
2658: *  Record
2659: *   symtab =
2660: *            kleiner : wort:  Zeiger auf Record der Kleiner ist
2661: *            groesser: wort:  Zeiger auf Record der Groesser ist
2662: *            datenwert: wort32: Daten
2663: *            attribut: wort:  Attribute des Datenwertes
2664: *            name:  wort64:  Name in ASCII mit 8 Zeichen
2665: *                            Leerzeichen fuellend den Namen auf
2666: *
2667: *  kleiner,groesser = 0 dann NIL
2668: *  sonst additiver Faktor auf den Anfang der Symboltabelle
2669: *
2670:
00000000  2671: kleiner          equ 0   *record indizes
00000002  2672: groesser         equ 2   *pointer
00000004  2673: datenwert        equ 4   *wert 32 Bit
00000008  2674: attribut          equ 8   *attr 16 Bit
0000000A  2675: name              equ $A  *64 Bit
          2676:
00000012  2677: symlen            equ 18  *laenge der symboltabelle
          2678:
          2679:
          2680:
001B1C    2681: namecheck:
001B1C B03C 0030  2682:         cmp.b #'0',d0
001B20 659A       2683:         bcs carset      * rev 4.1 So-Zeichen
001B22 B03C 003A  2684:         cmp.b #'9'+1,d0 *
001B26 659A       2685:         bcs carres
001B28 B03C 005F  2686:         cmp.b #'_',d0
001B2C 6794       2687:         beq carres
001B2E            2688: bucheck:
001B2E B03C 0041  2689:         cmp.b #'A',d0
001B32 6588       2690:         bcs carset
001B34 B03C 005B  2691:         cmp.b #'Z'+1,d0
001B38 6588       2692:         bcs carres
001B3A B03C 0061  2693:         cmp.b #'a',d0
001B3E 6500 FF7C  2694:         bcs carset
001B42 B03C 007B  2695:         cmp.b #'z'+1,d0
001B46 6400 FF74  2696:         bcc carset
001B4A 903C 0061  2697:         sub.b #'a',d0
001B4E D03C 0041  2698:         add.b #'A',d0
001B52 6000 FF6E  2699:         bra carres
                  2700:
                  2701:
                  2702:
001B56            2703: setupname:              * name in register d2,d3
                  2704:                         * aufbereiten von a0 an
                  2705:                         * a0 zeigt auf das Ende des
                  2706:                         * Namens
                  2707:                         * Carry wenn Fehler, kein gueltiger
                  2708:                         * Namenswert
                  2709:                         *
```

```
001B56  1010
001B58  61D4
001B5A  6500 FF60

001B5E  45ED 0068
001B62  257C 20202020
        0000
001B6A  257C 20202020
        0004
001B72  323C 0007
001B76
001B76  14C0
001B78  5288
001B7A  1010
001B7C  619E
001B7E  6500 000E
001B82  51C9 FFF2
001B86  5288
001B88  1010
001B8A  6190
001B8C  64F8
001B8E
001B8E  242D 0068
001B92  262D 006C
001B96  6000 FF2A


001B9A
001B9A  2649


01B9C  61B8
01B9E  6500 FF1C
01BA2
01BA2  B4AB 000A
01BA6  6B00 001E
01BAA  6200 002E
01BAE  B6AB 000E
01BB2  6B00 0012
01BB6  6200 0022
01BBA  202B 0004
01BBE  322B 0008
01BC2  6000 002A

01BC6
01BC6  4A6B 0000
01BCA  6700 0028
01BCE  4280
01BD0  302B 0000
01BD4  D089
01BD6  2640
01BD8  60C8

01BDA
01BDA  4A6B 0002
01BDE  6700 001C
01BE2  4280
01BE4  302B 0002
01BE8  D089
01BEA  2640
01BEC  60B4


01BEE  4242
01BF0  6000 FED0

01BF4
1BF4  343C 0001
1BF8  6000 FEC8
```

```
2710:           move.b (a0).d0  *zeichen holen A..Z  erlaubt an erster
2711:           bsr bucheck
2712:           bcs carset      *kein Buchstabe A..Z. a..z
2713: *
2714:           lea nametab(a5).a2
2715:           move.l #'    '.0(a2)
2716:           move.l #'    '.4(a2)    *geloescht
2717:           move #8-1.d1       *max zaehler
2718: setul2:                      *Schleife Name ablegen MAX 8 Zeichen
2719:           move.b d0.(a2)+ *Zeichen dorthin
2720:           addq.l #1.a0       *ok zeichen akzeptiert
2721:           move.b (a0).d0  *neues Zeichen pruefen
2722:           bsr namecheck
2723:           bcs setu3          *kein name
2724: setu4:    dbra d1.setul2  *bis max dann weiterlesen
2725: setu5:    addq.l #1.a0       *weiter da ok
2726:           move.b (a0).d0
2727:           bsr namecheck
2728:           bcc setu5          *bis ende erreicht
2729: setu3:                      *fertig a0=next zeichen
2730:           move.l nametab(a5).d2
2731:           move.l nametab+4(a5).d3
2732:           bra carres         *ohne carry
2733:
2734:
2735:
2736: sucheeintrag:              * Hauptprogramm
2737:           movea.l a1.a3    * Basis und erster Suchteil
2738:                            * a0 zeigt auf fraglichen Text
2739:                            * a1 zeigt auf Symboltabellenstart
2740:                            * Basis fest ***
2741:                            * Ergebnis
2742:                            * carry wenn Fehler
2743:                            * sonst d2=0 wenn gefunden
2744:                            * d2 = 1 wenn nicht gefunden
2745:                            *      name kleiner last symbol
2746:                            * d2 = 2 wenn nicht gefunden
2747:                            *      name groesser last symbol
2748:                            * sonst d0.l=datenwert d1.w=attributwert
2749:                            * a3 zeigt auf record nach der ausfuenrung
2750:           bsr setupname    *carry dann fehler
2751:           bcs carset       *stoppen name ungueltig
2752: suche:                     *name nur einmal festlegen
2753:           cmp.l name(a3).d2      *
2754:           bmi suchklei     *kleiner teil suchen
2755:           bhi suchgroe     *groesser
2756:           cmp.l name+4(a3).d3    *
2757:           bmi suchklei
2758:           bhi suchgroe     *nun muss gleich sein
2759:           move.l datenwert(a3).d0 * Langwort
2760:           move attribut(a3).d1    * nur Wort
2761:           bra found        *ok symbol in Ordnung
2762:
2763: suchklei:
2764:           tst kleiner(a3) *kleiner Teil
2765:           beq notkfound    *ende Fehler nicht gefunden
2766:           clr.l d0
2767:           move kleiner(a3).d0
2768:           add.l a1.d0      *pointer
2769:           move.l d0.a3     *neuer pointer
2770:           bra suche        *weiter suchen
2771:
2772: suchgroe:
2773:           tst groesser(a3)
2774:           beq notgfound    *nicht gefunden
2775:           clr.l d0
2776:           move groesser(a3).d0
2777:           add.l a1.d0      *pointer
2778:           movea.l d0.a3
2779:           bra suche        *weiter suchen
2780:
2781:
2782: found:    clr d2           *gefunden
2783:           bra carres
2784:
2785: notkfound:
2786:           move #1.d2       *merker kleiner Teil fehlt
2787:           bra carres
```

```
001BFC
001BFC  343C 0002       2788:
001C00  6000 FEC0       2789: notgfound:
                        2790:           move #2.d2      *merker groesser Teil fehlt
                        2791:           bra carres
                        2792:
                        2793:
001C04                  2794: ueberschreibe:            * Eintragen von Datenwerten
                        2795:                           * wenn ein Name schon da ist
                        2796:                           * a0=name a1=symboltabelle
                        2797:                           * d0.l = Datenwert
                        2798:                           * d1.w = Attribut
001C04  48E7 C000       2799:           movem.l d0/d1,-(a7)     * retten der neuen Parameter
001C08  6190            2800:           bsr sucheeintrag        * a0.d0,d1.d2 nun definiert
001C0A  4CDF 0003       2801:           movem.l (a7)+,d0/d1
001C0E  6500 FEAC       2802:           bcs carset      * Fehler Name falsch
001C12  4A42            2803:           tst d2          * = 0 dann gefunden
001C14  6600 FEA6       2804:           bne carset      * sonst Eintrag nicht moeglich
001C18  2740 0004       2805:           move.l d0,datenwert(a3) * Langwort
001C1C  3741 0008       2806:           move d1,attribut(a3)    * Wort
001C20  6000 FEA0       2807:           bra carres
                        2808:
                        2809:
                        2810:
00[illegible]           2811: eintrage:                 * eintragen eines namens
                        2812:                           * wenn er noch nicht existiert
                        2813:                           * a0 zeigt auf namen
                        2814:                           * a1 zeigt auf symboltabelle
                        2815:                           * d3 gibt freien Platz an
                        2816:                           * wird erneuert
                        2817:                           * d0 = datenwert d1 = attributwert
                        2818:                           * Carry wenn Fehler Name schon da
001C24  48E7 D020       2819:           movem.l d0/d1/d3/a2,-(a7)
001C28  6100 FF70       2820:           bsr sucheeintrag
001C2C  4CDF 040B       2821:           movem.l (a7)+,d0/d1/d3/a2
001C30  6500 FE8A       2822:           bcs carset      *fehler
001C34  4A42            2823:           tst d2          *=0 dann schon da
001C36  6700 FE84       2824:           beq carset      *fehler hier
001C3A  B47C 0001       2825:           cmp #1.d2       *kleiner
001C3E  6600 000A       2826:           bne eint1
001C42  3743 0000       2827:           move d3,kleiner(a3)     *abspeichern
001C46  6000 000E       2828:           bra eint2
                        2829: *
001C4A  B47C 0002       2830: eint1:    cmp #2.d2
001C4E  6600 FE6C       2831:           bne carset      *Fatal Fehler
001C52  3743 0002       2832:           move d3,groesser(a3)
                        2833: *
001C56  2409            2834: eint2:    move.l a1,d2    *berechnen adresse absolut
001C58  C6BC 0000FFFF   2835:           and.l #$FFFF,d3 *rest 0 setzen
001C5E  D483            2836:           add.l d3,d2     *neue adresse
001C60  2642            2837:           movea.l d2,a3   *aktueller pointer
001C62  D67C 0012       2838:           add #symlen,d3  *laenge des Symbolrekords
001C66  426B 0000       2839:           clr kleiner(a3)
001C6A  426B 0002       2840:           clr groesser(a3)
001C6E  2740 0004       2841:           move.l d0,datenwert(a3)
001C72  3741 0008       2842:           move d1,attribut(a3)
001C76  276D 0068       2843:           move.l nametab(a5),name(a3)
        000A
001C7C  276D 006C       2844:           move.l nametab+4(a5),name+4(a3)
        000E
001C82  6000 FE3E       2845:           bra carres      *Fertig
                        2846:
                        2847:
                        2848:
                        2849: * ausgabe recursiv
                        2850: *     procedure symbaus(adresse):
                        2851: *       if adresse.left <> nil
                        2852: *         symbaus(leftadresse)
                        2853: *       endif
                        2854: *       ausgabe name
                        2855: *       if adresse.right <> nil
                        2856: *         symbaus(rightadresse)
                        2857: *       endif
                        2858: *     endproc
                        2859:
                        2860:
                        2861:
                        2862:
                        2863: * ausgabe erfolgt ueber CO2. IOSTAT bestimmt Daten-Ziel
                        2864:
001C86                  2865: prtsymb:                        *Tabelle max 32K
```

```
001C86  43ED 0A4E      2866:           lea svmtab(a5),a1       *start der Tabelle
001C8A  363C 0000       2867:           move #0,d3              *Basisadresse
                        2868:                                   *d3 ist Parameter
001C8E                  2869: prtrecur:                         *recursive Ausgaberoutine
001C8E  4A71 3000       2870:           tst kleiner(a1,d3)      *=NIL
001C92  670A            2871:           beq.s prt1re
001C94  3F03            2872:           move d3,-(a7)           *merken
001C96  3631 3000       2873:           move kleiner(a1,d3),d3  *auf Kleiner Linken
001C9A  61F2            2874:           bsr prtrecur
001C9C  361F            2875:           move (a7)+,d3
001C9E                  2876: prt1re:                           *Ausgabe eines Symboles
001C9E  48E7 5040       2877:           movem.l d1/d3/a1,-(a7)
001CA2  41ED 0070       2878:           lea ausbuf(a5),a0       *Ziel Buffer
001CA6  20F1 300A       2879:           move.l name(a1,d3),(a0)+
001CAA  20F1 300E       2880:           move.l name+4(a1,d3),(a0)+
001CAE  30FC 2020       2881:           move.w #'  ',(a0)+      *name  wert  attribut
001CB2  2031 3004       2882:           move.l datenwert(a1,d3),d0
001CB6  6100 F992       2883:           bsr print8x
001CBA  10FC 0020       2884:           move.b #' ',(a0)+       *ggf ungerade
001CBE  10FC 0020       2885:           move.b #' ',(a0)+
001CC2  3031 3008       2886:           move.w attribut(a1,d3),d0
001CC6  6100 F998       2887:           bsr print4x             *ok nun ausgeben
001CCA  10FC 000D       2888:           move.b #$0d,(a0)+
001CCE  10FC 000A       2889:           move.b #$0a,(a0)+
001CD2  4210            2890:           clr.b (a0)              *last symbol
001CD4  41ED 0070       2891:           lea ausbuf(a5),a0
001CD8  6100 EDF2       2892:           bsr prtco2              *a1,d3 erhalten IOSTAT wichtig
001CDC  4CDF 020A       2893:           movem.l (a7)+,d1/d3/a1
001CE0  6100 ECF6       2894:           bsr csts        * wenn ctrl-s dann warten
001CE4  6714            2895:           beq.s prt1aa
001CE6  6100 ED18       2896:           bsr ci
001CEA  B03C 0013       2897:           cmp.b #$13,d0
001CEE  660A            2898:           bne.s prt1aa
001CF0                  2899: prt1ab:
001CF0  6100 ED0E       2900:           bsr ci
001CF4  B03C 0011       2901:           cmp.b #$11,d0
001CF8  66F6            2902:           bne.s prt1ab
001CFA                  2903: prt1aa:
001CFA  4A71 3002       2904:           tst groesser(a1,d3)
001CFE  670A            2905:           beq.s prt2re
001D00  3F03            2906:           move d3,-(a7)
001D02  3631 3002       2907:           move groesser(a1,d3),d3
001D06  6186            2908:           bsr prtrecur
001D08  361F            2909:           move (a7)+,d3
001D0A  4E75            2910: prt2re: rts
                        2911:
                        2912:
                        2913: ** Routinen mit Voreinstellungen
                        2914:
                        2915:
                        2916: ** setupname a0=name   carry = Syntaxfehler bei Namen
                        2917:
                        2918:
001D0C                  2919: getval:          *holt Wert aus der Symboltabelle
                        2920:                 * a0= name start
                        2921:                 * nachher          d0.l = wert
                        2922:                 *                  d1.w = attribut
001D0C  43ED 0A4E       2923:           lea svmtab(a5),a1       *tabellen start
001D10  6000 FE88       2924:           bra sucheeintrag        *Carry wenn Fehler SYNTAX Name
                        2925:                                   *a0 zeigt hinter den Namen
                        2926:                                   *d2.w = 0 wenn gefunden
                        2927:                                   *d2.w = 1 wenn kleiner Teil
                        2928:                                   *d2.w = 2 wenn groesser Teil Nil
                        2929:                                   *a3 = pointer auf Record
                        2930:
001D14                  2931: setval:          *setzt Wert neu Parameter wie oben
001D14  43ED 0A4E       2932:           lea svmtab(a5),a1       *carry wenn Syntax oder
001D18  6000 FEEA       2933:           bra ueberschreibe       *Name noch nicht da
                        2934:
                        2935:
001D1C                  2936: newval:          *traegt Symbol neu ein  Parameter wie oben
001D1C  43ED 0A4E       2937:           lea svmtab(a5),a1
001D20  362D 0A4C       2938:           move svmnext(a5),d3     *start index
001D24  6100 FEFE       2939:           bsr eintrage
001D28  6500 FD92       2940:           bcs carset      *carry=fehler Name schon da. oder Syntax
001D2C  3B43 0A4C       2941:           move d3,svmnext(a5)     *sonst neuer Pointer
001D30  6000 FD90       2942:           bra carres              *ohne carry ok
                        2943:
                        2944: ******** Diverses Utilitys
                        2945:
```

MEMORY

```
                                2946: * Suchen eines zusammenhaengenden Speicherbereiches
                                2947:
001D34                          2948: getram:                  * RAM-Bereich suchen von A0 an
                                2949:                          * Bereich auf 1MB - 64K begrenzt
                                2950:                          * A0 -> naechste Nicht-Ram-Zelle
                                2951:                          * A1 -> erste RAM Zelle
                                2952:                          * Carry dann kein Bereich mehr da
001D34  2008                    2953:          move.l a0,d0
001D36  C0BC FFFFFC00           2954:          and.l #$fffffc00,d0     * Auf Seitengrenze bringen
001D3C  2040                    2955:          movea.l d0,a0
001D3E                          2956: getlaram
001D3E  B1FC 00FF0000           2957:          cmpa.l #$ff0000,a0      * ueberlauf
001D44  6700 FD76               2958:          beq carset      * Fehler
001D48  6100 003C               2959:          bsr memchk
001D4C  6500 001E               2960:          bcs get2ram
001D50  2248                    2961:          movea.l a0,a1   * Anfang merken in A1
001D52                          2962: getlram:                 * Nun nach Ende suchen
001D52  B1FC 00FF0000           2963:          cmpa.l #$ff0000,a0 *
001D58  6700 001A               2964:          beq get3ram     * Ende erreicht ist OK
001D5C  6100 0028               2965:          bsr memchk
001D60  6500 0012               2966:          bcs get3ram     * a0 -> erste Nicht-Ram-Zelle
001D64  D1FC 00000400           2967:          adda.l #$400,a0 * 1K - Weise suchen sonst zu langsam
001D6A  60E6                    2968:          bra getlram
                                2969:
001[illegible]                  2970: get2ram:
001D[illegible]  D1FC 00000400  2971:          adda.l #$400,a0 * start auf 1K Genau finden reicht aus
001D72  60CA                    2972:          bra getlaram
                                2973:
001D74                          2974: get3ram:                 * ende gefunden nun nochmals genauer suche
001D74  91FC 00000400           2975:          suba.l #$400,a0 * zurueck
001D7A                          2976: get4ram:
001D7A  6100 000A               2977:          bsr memchk
001D7E  6500 FD42               2978:          bcs carres      * ok fertig a0-> erste Nicht-Ramzelle
001D82  5288                    2979:          addq.l #1,a0    * nun ueber ende hinaus suchen
001D84  60F4                    2980:          bra get4ram
                                2981:
001D86                          2982: memchk:                  * carry dann kein RAM auf (a0)
001D86  48E7 E000               2983:          movem.l d0-d2,-(a7)
001D8A  1010                    2984:          move.b (a0),d0
001D8C  1228 0001               2985:          move.b 1(a0),d1 * zwei Zellen nehmen
001D90  1428 FFFF               2986:          move.b -1(a0),d2 * um virtuelle Zellen nicht mit
001D94  4600                    2987:          not.b d0        * RAM-Zellen zu verwechseln
001D96  1080                    2988:          move.b d0,(a0)
001D98  4600                    2989:          not.b d0        * komplement zum nachbarn
001D9A  1140 0001               2990:          move.b d0,1(a0) * eine danach und davor um
001D9E  1140 FFFF               2991:          move.b d0,-1(a0) * bei Grenzen richtig zu entscheiden
001DA2  4600                    2992:          not.b d0
001DA4  B010                    2993:          cmp.b (a0),d0
001DA6  6600 0028               2994:          bne memfault
001DAA  4600                    2995:          not.b d0
001DAC  1080                    2996:          move.b d0,(a0)
001DAE  4600                    2997:          not.b d0
001DB0  1140 0001               2998:          move.b d0,1(a0)
001DB4  1140 FFFF               2999:          move.b d0,-1(a0)
001DB8  4600                    3000:          not.b d0
001DBA  B010                    3001:          cmp.b (a0),d0
001DBC  6600 0012               3002:          bne memfault
001DC0  1141 0001               3003:          move.b d1,1(a0)
001DC4  1142 FFFF               3004:          move.b d2,-1(a0)
001DC8  4CDF 0007               3005:          movem.l (a7)+,d0-d2
001[illegible]  6000 FCF4       3006:          bra carres
                                3007:
001DD0                          3008: memfault:
001DD0  1141 0001               3009:          move.b d1,1(a0) * Hilfszelle zurueck
001DD4  1142 FFFF               3010:          move.b d2,-1(a0)
001DD8  4CDF 0007               3011:          movem.l (a7)+,d0-d2
001DDC  6000 FCDE               3012:          bra carset
                                3013:
                                3014:
                                3015: ***************************************************
                                3016:
                                3017: * Befehlsimplementierung der Menues
                                3018:
                                3019:
001DE0                          3020: aussober:                * Speicherbereiche ausgeben
                                3021:                          * Graphisch
001DE0  41FA 1D7A               3022:          lea sober1(pc),a0
001DE4  6100 110C               3023:          bsr headaus     * Ueberschrift
001DE8  41FA 1D7F               3024:          lea sober2(pc),a0
001DEC  103C 0022               3025:          move.b #$22,d0
```

```
001DF0  323C 0000
001DF4  343C 00C8
001DF8  6100 F58C
001DFC  41FA 1D4F
001E00  6100 FF32
001E04  2409
001E06  2608
001E08  41ED 0070
001E0C  2002
001E0E  6100 F848
001E12  10FC 002D
001E16  2003
001E18  5380
001E1A  6100 F83C
001E1E  41ED 0070
001E22  103C 0022
001E26  323C 00FA
001E2A  343C 00C8
001E2E  6100 F556
001E32  41FA 1D46
001E36  103C 0021
001E3A  323C 0042
001E3E  343C 00AC
001E42  6100 F542
001E46  7000
001E48  323C 0000
001E4C  343C 00A2
001E50  41ED 0070
001E54  6100 F80A
001E58  48E7 E000
001E5C  41ED 0070
001E60  103C 0021
001E64  6100 F520
001E68  4CDF 0007
001E6C  D0BC 00001000
001E72  947C 000A
001E76  B0BC 00010000
001E7C  66D2
001E7E  6100 EEFC
001E82  323C 003C
001E86  343C 00AA
001E8A  6100 F056
001E8E  323C 01BC
001E92  6100 F0B8
001E96  343C 000A
001E9A  6100 F0B0
001E9E  323C 003C
001EA2  6100 F0A8
001EA6  343C 00AA
001EAA  6100 F0A0

001EAE  207C 00000000
001EB4  363C 000E
001EB8  323C 003C
001EBC
001EBC  383C 001F
001EC0  343C 00AA
001EC4
001EC4  48E7 F800
001EC8  6100 FEBC
001ECC  6500 001C

001ED0  363C 0004

001ED4
001ED4  6100 F00C
001ED8  D27C 0016
001EDC  6100 F06E
001EE0  927C 0016
001EE4  5342
001EE6  51CB FFEC
001EEA
001EEA  4CDF 001F
001EEE  D1FC 00000800
001EF4  5B42
001EF6  51CC FFCC
001EFA  D27C 0018
001EFE  51CB FFBC
001F02  6000 091E
```

```
3026:           move #0.d1
3027:           move #200.d2
3028:           bsr textaus       * Erster Bereich:
3029:           lea fimsg(pc).a0          * ERST VON SCHLUSS AN SUCHEN
3030:           bsr getram        * a1-> Anfang a0-> Ende
3031:           move.l a1.d2      * Merken dort
3032:           move.l a0.d3
3033:           lea ausbuf(a5).a0
3034:           move.l d2.d0
3035:           bsr print6x
3036:           move.b #'-'.(a0)+
3037:           move.l d3.d0
3038:           sub.l #1.d0       * -1 da letzte Zelle
3039:           bsr print6x       * Ok
3040:           lea ausbuf(a5).a0
3041:           move.b #$22.d0
3042:           move #250.d1
3043:           move #200.d2      * Text dort starten
3044:           bsr textaus       * Nun Speichermap ausgeben
3045:           lea spber3(pc).a0
3046:           move.b #$21.d0
3047:           move #66.d1
3048:           move #172.d2
3049:           bsr textaus
3050:           move.l #0.d0      * Zaehler fuer Daten
3051:           move #0.d1        * X
3052:           move #170-8.d2    * Y
3053: a1us:     lea ausbuf(a5).a0
3054:           bsr print4x
3055:           movem.l d0-d2.-(a7)
3056:           lea ausbuf(a5).a0
3057:           move.b #$21.d0    * Hoehe 1
3058:           bsr textaus       * ausgeben xxxx
3059:           movem.l (a7)+.d0-d2
3060:           add.l #$1000.d0 * Counter
3061:           sub #10.d2        * Y := Y - 10
3062:           cmp.l #$10000.d0          * ende
3063:           bne a1us          * wenn ereicht
3064:           bsr setpen
3065:           move #60.d1
3066:           move #170.d2
3067:           bsr moveto
3068:           move #60+16*24.d1
3069:           bsr drawto
3070:           move #170-16*10.d2
3071:           bsr drawto
3072:           move #60.d1
3073:           bsr drawto
3074:           move #170.d2
3075:           bsr drawto        * Rahmen gezeichnet
3076: *
3077:           movea.l #0.a0     * Start RAM-Test
3078:           move #15-1.d3     * Bank-Loop ohne Periferie-Bank
3079:           move #60.d1       * X
3080: abank:
3081:           move #32-1.d4     * Reihen-Loop 32 da 2K
3082:           move #170.d2      * Y
3083: areihe:
3084:           movem.l d0-d4.-(a7)
3085:           bsr memchk
3086:           bcs a1reihe       * nein dann kein rambereich dort
3087:                             * OK RAM dort dann markieren mit 2
3088:           move #5-1.d3
3089:
3090: a1rhlp:
3091:           bsr moveto        * d1.d2 queltig
3092:           add #22.d1
3093:           bsr drawto        * linie
3094:           sub #22.d1
3095:           sub #1.d2         * Y:=Y-1
3096:           dbra d3.a1rhlp
3097: a1reihe:
3098:           movem.l (a7)+.d0-d4
3099:           adda.l #$800.a0 * IN 2K Bloecken testen
3100:           sub #5.d2         * Y := Y - 5
3101:           dbra d4.areihe
3102:           add #24.d1        * X := X + 24
3103:           dbra d3.abank
3104:           bra finmenue
3105:
```

Bloc

GETSET

```
                        3106:
001F06                  3107: setflip:                  * d0=flip, d1=flip1
001F06  1B40 002E       3108:           move.b d0,flip(a5)      * doppelseite
001F0A  1B40 002F       3109:           move.b d0,flipcnt(a5)
001F0E  1B41 0030       3110:           move.b d1,flip1(a5)     * viererseite
001F12  1B41 0031       3111:           move.b d1,flip1cnt(a5)
001F16  4E75            3112:           rts
                        3113:
001F18                  3114: setgroe:                  * setzt zeichengroesse
001F18  1B40 021B       3115:           move.b d0,groesse(a5)   * nur 11, 21 sinnvoll
001F1C  4E75            3116:           rts                     * andere aber denkbar
                        3117:
001F1E                  3118: edittext:
001F1E  6100 E0E0       3119:           bsr edit        * aufruf texteditor
001F22  4E75            3120:           rts
                        3121:                                   EDIT
                        3122:
                        3123: ******* Einzelschrittbearbeitung *******
                        3124: * TRAP-Befehle und JSR xx.L werden nicht im
                        3125: * Einzelschritt durchlaufen sondern als
                        3126: * Ganzes ausgefuehrt
                        3127: * Einzel ist kein Unterprogramm
                        3128: * rev 4.0 ACHTUNG A5 ist Adresse lokales Speichergebiet
                        3129:
001F                    3130: einzel:
001F    1B7C 0001       3131:           move.b #1,first(a5)     * grap first aufruf
        021D
001F2A  1B7C 0001       3132:           move.b #1,turdo(a5)     * turtle down
        021E
001F30  41FA 1F67       3133:           lea txtreg5(pc),a0
001F34  6100 0FBC       3134:           bsr headaus
001F38  41FA 1F1C       3135:           lea txtreg3(pc),a0
001F3C  103C 0022       3136:           move.b #$22,d0
001F40  323C 0000       3137:           move #0,d1
001F44  343C 000A       3138:           move #10,d2
001F48  6100 F43C       3139:           bsr textaus     *HINWEIS AUF MOEGL. BEFs
001F4C  6100 0ECA       3140:           bsr getadr
001F50  4A41            3141:           tst d1
001F52  6700 FB68       3142:           beq carset      *D0.L = STARTADRESSE INITIAL
001F56  2B40 01CA       3143:           move.l d0,pcsave(a5)    * AKTUELLER STAND
001F5A  3B7C A700       3144:           move #$A700,srsave(a5)  *TRACE ON, SYSTEM-MODE ON
        01C8
001F60  41FA 031E       3145:           lea tracende(pc),a0     * rev 4.0
001F64  2F08            3146:           move.l a0,-(a7)         * Rueckkehradresse
001F66  5D8F            3147:           sub.l #6,a7             * STACK VERRINGERN
001F68  3F6D 01C8       3148:           move srsave(a5),0(a7)
        0000
001F6E  2F6D 01CA       3149:           move.l pcsave(a5),2(a7) * fuer RTE vorbereitet
        0002
001F74  6100 EE2A       3150:           bsr clrall              * loeschen aller Seiten
001F78  6100 EDE2       3151:           bsr wait                * warten bis gdp fertig
001F7C  41F8 FF71       3152:           lea gdp+1*cpu,a0        * retten der GDP-Register
001F80  43ED 01B9       3153:           lea gdpsave+1(a5),a1
001F84  323C 000A       3154:           move #15-4-1,d1         * NUR CMD und resv nicht retten
001F88                  3155: tracillo:
001F88  12D0            3156:           move.b (a0),(a1)+
001F8A  5288            3157:           addq.l #cpu,a0          * rev 4.0 SIZE addieren
001F8C  51C9 FFFA       3158:           dbra d1,tracillo
001F90  422D 0215       3159:           clr.b flipsave(a5)
001F94  422D 0216       3160:           clr.b flip1save(a5)     * neutral stellen
001F98  42AD 01D6       3161:           clr.l tracbis(a5)
001F9C  42AD 01DA       3162:           clr.l tracnmal(a5)      * NEUTRAL wenn =0
001F                    3163:           move.l a7,uspsave(a5)   * Aktueller Stand fuer Start
001FA4  4E67            3164:           move.l a7,usp           * und auch USP als merker
001FA6  2B4F 01D2       3165:           move.l a7,sspsave(a5)   * beide auf default setzen
001FAA  2B4F 01B4       3166:           move.l a7,regsave+4*15(a5) * letztes Register ist A7
001FAE  2B4D 01AC       3167:           move.l a5,regsave+4*13(a5) * und POINTER A5 defaulten
001FB2  6000 008E       3168:           bra intrace                * auf Variablengebiet
                        3169:
001FB6                  3170: tracein:                  * exception durch Trace-Bit
                        3171:                           * A7->SR  2(A7)->PC
001FB6  48E7 0004       3172:           movem.l a5,-(a7)        * Temp benoetigt
001FBA  4BFA E044       3173:           lea basis(pc),a5
001FBE  DBFA E444       3174:           adda.l ramstart(pc),a5          * A5 neu belegt REV 4.0b
001FC2  48ED FFFF       3175:           movem.l d0-d7/a0-a7,regsave(a5) * Register retten
        0178
001FC8  4CDF 0100       3176:           movem.l (a7)+,a0
001FCC  2B4F 01B4       3177:           move.l a7,regsave+4*15(a5)  * A7 erst hier aktuell
001FD0  2B48 01AC       3178:           move.l a0,regsave+4*13(a5)  * A5 ueberschreiben damit
001FD4  3B6F 0000       3179:           move 0(a7),srsave(a5)   * SR retten der vorher vorlag nc
```

```
          01C8
001FDA    4E68
001FDC    2B48 01CE
001FE0    2B4F 01D2
001FE4    2B6F 0002
          01CA

001FEA    1B6D 002E
          0215
001FF0    1B6D 0030
          0216

001FF6    6100 ED64
001FFA    41F8 FF71
001FFE    43ED 01B9
002002    323C 000A
002006
002006    12D0
002008    5288
00200A    51C9 FFFA

00200E    41FA 0270
002012    B1ED 01CA
002016    6700 0268

00201A    4AAD 01DA
00201E    6700 000A
002022    53AD 01DA
002026    6000 018C

00202A    4AAD 01D6
00202E    6700 0012
002032    202D 01CA
002036    B0AD 01D6
00203A    6600 0178
00203E    42AD 01D6


002042
002042    1F2D 0217
002046    1F2D 0218
00204A    422D 0218
00204E    6100 EDC4
002052    6100 025C
002056    1B7C 0001
          0218
00205C    6100 EDB6
002060    6100 024E
002064    1B5F 0218
002068    1B5F 0217
00206C    6100 EDA6

002070
002070    6100 E9F0
002074    B03C 004D
002078    6700 0206
00207C    B03C 004E
002080    6600 0048
002084    1F2D 0217
002088    1F2D 0218
00208C    6100 0100
002090    41ED 00F4
002094    103C 0011
002098    323C 0002
00209C    343C 00F5
0020A0    363C 0014
0020A4    6100 F3E0
0020A8    1B5F 0218
0020AC    1B5F 0217
0020B0    41ED 00F4
0020B4    6100 F676
0020B8    4A41
0020BA    67B6
0020BC    4A80
0020BE    6782
0020C0    5380
0020C2    2B40 01DA
0020C6    6000 00EC

0020CA    B03C 0053
```

```
3180:           move.l usp,a0
3181:           move.l a0,uspsave(a5)
3182:           move.l a7,sspsave(a5)    * auch gerettet
3183:           move.l 2(a7),pcsave(a5) * und naechster Befehl

3184: *
3185:           move.b flip(a5),flipsave(a5)

3186:           move.b flip1(a5),flip1save(a5)

3187: *
3188:           bsr wait                 * warten bis gdp fertig
3189:           lea gdp+1*cpu,a0         * retten der GDP-Register
3190:           lea gdpsave+1(a5),a1
3191:           move #15-4-1,d1          * NUR CMD und resv. nicht retten
3192: traclp:
3193:           move.b (a0),(a1)+
3194:           addq.l #cpu,a0           * rev 4.0 CPU Size addieren
3195:           dbra d1,traclp
3196: *
3197:           lea tracende(pc),a0      * rev 4.0
3198:           cmpa.l pcsave(a5),a0     * wenn pcstand auf ende
3199:           beq tracende             * STOP TRACE
3200: *
3201:           tst.l tracnmal(a5)       * =0
3202:           beq trac2a
3203:           subq.l #1,tracnmal(a5)   *-1
3204:           bra trac5m               * DURCHSTARTEN
3205:
3206: trac2a: tst.l tracbis(a5)
3207:           beq intrace              * JA AUSGABE
3208:           move.l pcsave(a5),d0
3209:           cmp.l tracbis(a5),d0
3210:           bne trac5m
3211:           clr.l tracbis(a5)        * loeschen
3212:
3213:
3214: intrace:                          * registerausgabe etc
3215:           move.b viewpage(a5),-(a7)
3216:           move.b wrtpage(a5),-(a7)
3217:           clr.b wrtpage(a5)
3218:           bsr aktpage
3219:           bsr regdump
3220:           move.b #1,wrtpage(a5)

3221:           bsr aktpage
3222:           bsr regdump              * nur seiten 0,1
3223:           move.b (a7)+,wrtpage(a5)
3224:           move.b (a7)+,viewpage(a5)
3225:           bsr aktpage              * wieder zurueck
3226: *
3227: tracmen:
3228:           bsr ki                   * Befehlsverteiler
3229:           cmp.b #'M',d0            * ENDE wenn 0 erreicht
3230:           beq tracende
3231:           cmp.b #'N',d0            * N-Mal
3232:           bne traclm
3233:           move.b viewpage(a5),-(a7)
3234:           move.b wrtpage(a5),-(a7)
3235:           bsr wrtupdate            * sichtbare Seite nehmen
3236:           lea einbuf(a5),a0
3237:           move.b #$11,d0
3238:           move #2,d1
3239:           move #255-10,d2
3240:           move #20,d3
3241:           bsr textein              * BIS
3242:           move.b (a7)+,wrtpage(a5)
3243:           move.b (a7)+,viewpage(a5)
3244:           lea einbuf(a5),a0
3245:           bsr wert
3246:           tst d1
3247:           beq intrace              * IGNORE
3248:           tst.l d0
3249:           beq intrace              * 0 mal
3250:           subq.l #1,d0             * 0..n-1
3251:           move.l d0,tracnmal(a5)   * N-Mal ohne ausgabe
3252:           bra trac5m
3253:
3254: traclm: cmp.b #'S',d0             * Seite
```

```
0020CE  6600 0074      3255:           bne trac2m
0020D2  1F2D 0217      3256:           move.b viewpage(a5),-(a7)
0020D6  1F2D 0218      3257:           move.b wrtpage(a5),-(a7)
0020DA  6100 00B2      3258:           bsr wrtupdate
0020DE  41ED 00F4      3259:           lea einbuf(a5),a0
0020E2  103C 0011      3260:           move.b #$11,d0
0020E6  323C 0002      3261:           move #2,d1
0020EA  343C 00F5      3262:           move #255-10,d2
0020EE  363C 0014      3263:           move #20,d3
0020F2  6100 F392      3264:           bsr textein             * BIS
0020F6  1B5F 0218      3265:           move.b (a7)+,wrtpage(a5)
0020FA  1B5F 0217      3266:           move.b (a7)+,viewpage(a5)
0020FE  41ED 00F4      3267:           lea einbuf(a5),a0
002102  6100 F628      3268:           bsr wert
002106  4A41           3269:           tst d1
002108  6700 001E      3270:           beq trac12m             * IGNORE flip on
00210C  B03C 0004      3271:           cmp.b #4,d0             * 0..3 erlaubt
002110  6400 0016      3272:           bcc trac12m
002114  422D 002E      3273:           clr.b flip(a5)
002118  422D 0030      3274:           clr.b flip1(a5)         * loeschen flips
00211C  1B40 0217      3275:           move.b d0,viewpage(a5)
002120  6100 ECF2      3276:           bsr aktpage
002124  6000 FF1C      3277:           bra intrace
                       3278:
00                     3279: trac12m:
002128  1B6D 0215      3280:           move.b flipsave(a5),flip(a5)
        002E
00212E  1B6D 0215      3281:           move.b flipsave(a5),flipcnt(a5)
        002F
002134  1B6D 0216      3282:           move.b flip1save(a5),flip1(a5)
        0030
00213A  1B6D 0216      3283:           move.b flip1save(a5),flip1cnt(a5)
        0031
002140  6000 FF00      3284:           bra intrace
                       3285:
002144  B03C 0042      3286: trac2m:   cmp.b #'B',d0           * Bis
002148  6600 0052      3287:           bne trac3m
00214C  1F2D 0217      3288:           move.b viewpage(a5),-(a7)
002150  1F2D 0218      3289:           move.b wrtpage(a5),-(a7)
002154  6100 0038      3290:           bsr wrtupdate
002158  41ED 00F4      3291:           lea einbuf(a5),a0
00215C  103C 0011      3292:           move.b #$11,d0
002160  323C 0002      3293:           move #2,d1
002164  343C 00F5      3294:           move #255-10,d2
002168  363C 0014      3295:           move #20,d3
00216C  6100 F318      3296:           bsr textein             * BIS
002170  1B5F 0218      3297:           move.b (a7)+,wrtpage(a5)
002174  1B5F 0217      3298:           move.b (a7)+,viewpage(a5)
002178  41ED 00F4      3299:           lea einbuf(a5),a0
00217C  6100 F5AE      3300:           bsr wert
002180  4A41           3301:           tst d1
002182  6700 FEBE      3302:           beq intrace             * IGNORE
002186  2B40 01D6      3303:           move.l d0,tracbis(a5)
00218A  6000 0028      3304:           bra trac5m
                       3305:
00218E                 3306: wrtupdate:                * Seite 0 als Auswahlseite
00218E  422D 0218      3307:           clr.b wrtpage(a5)
002192  422D 0217      3308:           clr.b viewpage(a5)
002196  6100 EC7C      3309:           bsr aktpage
00219A  4E75           3310:           rts
                       3311:
002   B03C 004C        3312: trac3m:   cmp.b #'L',d0           * Loeschen
0021A0  6600 000A      3313:           bne trac4m
0021A4  6100 EBFA      3314:           bsr clrall              * alle seiten loeschen
0021A8  6000 FE98      3315:           bra intrace
                       3316:
0021AC  B03C 000D      3317: trac4m:   cmp.b #$d,d0
0021B0  6600 FE90      3318:           bne intrace             * SONST WEITER
                       3319:
0021B4                 3320: trac5m:                   * Zurueck
0021B4  6100 EBA6      3321:           bsr wait
0021B8  41F8 FF71      3322:           lea gdp+1*cpu,a0        * CMD NICHT
0021BC  43ED 01B9      3323:           lea gdpsave+1(a5),a1
0021C0  323C 000A      3324:           move #15-4-1,d1         * alle register zurueck bis auf
0021C4                 3325: trac21p:
0021C4  1099           3326:           move.b (a1)+,(a0)
0021C6  5288           3327:           addq.l #cpu,a0          * rev 4.0 CPU Size addieren
0021C8  51C9 FFFA      3328:           dbra d1,trac21p         * 0(a7) alter SR queltig mit tra
0021CC  6100 EB8E      3329:           bsr wait
0021D0  302F 0000      3330:           move 0(a7),d0           * SR
```

```
0021D4  807C 8000          3331:          or #$8000,d0             * Trace-Bit immer erneut setzen
0021D8  3F40 0000          3332:          move d0,0(a7)
                           3333: * test ob jsr oder trap Befehl vorliegt.
                           3334: * wenn ja, dann direkt ausfuehren
0021DC  206F 0002          3335:          movea.l 2(a7),a0         * PC holen
0021E0  3010               3336:          move (a0),d0             * naechster Befehl
0021E2  B07C 4EB9          3337:          cmp #$4eb9,d0            * JSR xxxx.l
0021E6  6700 0016          3338:          beq jsrlex
0021EA  C07C FFF0          3339:          and #$FFF0,d0            * TRAP BEFEHL
0021EE  B07C 4E40          3340:          cmp #$4e40,d0
0021F2  6700 004E          3341:          beq trapex
0021F6  4CED FFFF          3342:          movem.l regsave(a5),d0-d7/a0-a7
        0178
0021FC  4E73               3343:          rte                      * NEUSTART
                           3344:
                           3345:
0021FE                     3346: jsrlex:                           * JSR xxx.l ausfuehrung
                           3347:                                   * ACHTUNG PARAMETER sind
                           3348:                                   * nicht erlaubt,wenn sie
                           3349:                                   * hinter dem Befehl stehen
0021FE  3B50 01DE          3350:          move (a0),modber(a5)     * Befehlscode
002202  2B68 0002         3351:          move.l 2(a0),modber+2(a5)        * ADressteil
        01E0
002208  3B7C 4E75          3352:          move #$4e75,modber+6(a5)         * RTS-Befehl
        01E4
00220E  5CAF 0002          3353:          addo.l #6,2(a7)          * PC + 6 ablegen
002212  302F 0000          3354:          move 0(a7),d0            * SR
002216  C07C 7FFF          3355:          and #$7fff,d0            * ohne trace
00221A  46C0               3356:          move d0,sr               * status register setzen
00221C  4CED FFFF          3357:          movem.l regsave(a5),d0-d7/a0-a7
        0178
002222  4BFA DDDC          3358:          lea basis(pc),a5         * nochmal zurueck
002226  DBFA E1DC          3359:          adda.l ramstart(pc),a5
00222A  487A 000E          3360:          pea jsrlex(pc)
00222E  486D 01DE          3361:          pea modber(a5)
002232  4CED 2000          3362:          movem.l regsave+4*13(a5),a5 * Neuer Wert A5
        01AC
002238  4E75              3363:          rts                      * Sprung
                           3364:
00223A  40EF 0000          3365: jsrlex: move sr,0(a7)            * als neuer Status
00223E  6000 FD76          3366:          bra tracein              * rev 4.0
                           3367:
                           3368:
002242                     3369:
                           3370: trapex:                          * trapbefehl direkt ausfuehren
                           3371:                                   * Parameteruebergabe damit
                           3372:                                   * nicht testbar !!!
                           3373:                                   * wenn Parameter direkt hinter TRA
002242  3B50 01DE          3374:          move (a0),modber(a5)     * Trap-Befehl
002246  3B7C 4E75          3375:          move #$4e75,modber+2(a5)         * RTS-Befehl
        01E0
00224C  54AF 0002          3376:          addo.l #2,2(a7)          * PC + 2 ablegen
002250  302F 0000          3377:          move 0(a7),d0            * SR
002254  C07C 7FFF          3378:          and #$7fff,d0            * ohne trace
002258  46C0               3379:          move d0,sr               * status register setzen
00225A  4CED FFFF          3380:          movem.l regsave(a5),d0-d7/a0-a7
        0178
002260  4BFA DD9E          3381:          lea basis(pc),a5         * nochmal zurueck
002264  DBFA E19E          3382:          adda.l ramstart(pc),a5   * aber condition codes behalten
002268  487A 000E          3383:          pea traplex(pc)
00226C  486D 01DE          3384:          pea modber(a5)
002270  4CED 2000          3385:          movem.l regsave+4*13(a5),a5 * Neuer Wert A5
        01AC
002276  4E75              3386:          rts                      * Sprung
002278                     3387: traplex:
002278  40EF 0000          3388:          move sr,0(a7)            * neuer Status rev 4.0
00227C  6000 FD38          3389:          bra tracein              * ok register retten
                           3390:
                           3391:
002280                     3392: tracende:       * ENDE TRACE IMMER MIT NEUER STACKDEFINITION BEGIN
                           3393:                 * rev 4.0
002280  4BFA DD7E          3394:          lea basis(pc),a5
002284  DBFA E17E          3395:          adda.l ramstart(pc),a5
002288  4FED 0A02          3396:          lea stack(a5),a7         * Damit neuer Stackanfang da
00228C  1B7C 0001          3397:          move.b #1,first(a5)      * grap first aufruf
        021D
002292  1B7C 0001          3398:          move.b #1,turdo(a5)      * turtle down
        021E
002298  422D 0217          3399:          clr.b viewpage(a5)
00229C  422D 0218          3400:          clr.b wrtpage(a5)
```

```
0022A0  422D 002E       3401:           clr.b flip(a5)
0022A4  422D 0030       3402:           clr.b flip1(a5)
0022A8  6100 EB6A       3403:           bsr aktpage       * zurueck auf seite 0
0022AC  6000 1190       3404:           bra rstart                *
                        3405:
                        3406:
0022B0                  3407: regdump:                    * Register auf dem Schirm ausgeben
                        3408:                             * rev 4.1/ ausgabe (pc) wenn keine
                        3409:                             * Debug-Info da ist
                        3410:                             * aktuelle Seite --
0022B0  6100 EACA       3411:           bsr setpen        * Setzend Schreiben
0022B4  41ED 0070       3412:           lea ausbuf(a5),a0       *
0022B8  302D 01C8       3413:           move srsave(a5),d0
0022BC  6100 F3A2       3414:           bsr print4x
0022C0  10FC 0020       3415:           move.b #' ',(a0)+
0022C4  202D 01CE       3416:           move.l uspsave(a5),d0
0022C8  6100 F380       3417:           bsr print8x
0022CC  10FC 0020       3418:           move.b #' ',(a0)+
0022D0  202D 01D2       3419:           move.l sspsave(a5),d0
0022D4  6100 F374       3420:           bsr print8x
0022D8  10FC 0020       3421:           move.b #' ',(a0)+
0022DC  202D 01CA       3422:           move.l pcsave(a5),d0
0022E0  6100 F368       3423:           bsr print8x
0022E4  41ED 0070       3424:           lea ausbuf(a5),a0
0022E8  103C 0011       3425:           move.b #$11,d0
0022EC  323C 0000       3426:           move #0,d1
0022F0  343C 0000       3427:           move #0,d2
0022F4  6100 F090       3428:           bsr textaus       * erste Zeile
0022F8  41FA 1BB5       3429:           lea txtreg4(pc),a0
0022FC  103C 0011       3430:           move.b #$11,d0
002300  323C 0000       3431:           move #0,d1
002304  343C 000A       3432:           move #10,d2
002308  6100 F07C       3433:           bsr textaus       * text SR ...
00230C  41ED 0070       3434:           lea ausbuf(a5),a0
002310  43ED 0198       3435:           lea regsave+4*8(a5),a1
002314  323C 0007       3436:           move #8-1,d1
002318                  3437: reg1d:
002318  2019            3438:           move.l (a1)+,d0
00231A  6100 F32E       3439:           bsr print8x
00231E  10FC 0020       3440:           move.b #' ',(a0)+
002322  51C9 FFF4       3441:           dbra d1,reg1d
002326  4210            3442:           clr.b (a0)
002328  41ED 0070       3443:           lea ausbuf(a5),a0
00232C  103C 0011       3444:           move.b #$11,d0
002330  323C 0000       3445:           move #0,d1
002334  343C 0014       3446:           move #20,d2
002338  6100 F04C       3447:           bsr textaus
00233C  41FA 1AD6       3448:           lea txtreg2(pc),a0
002340  103C 0011       3449:           move.b #$11,d0
002344  323C 0000       3450:           move #0,d1
002348  343C 001E       3451:           move #30,d2
00234C  6100 F038       3452:           bsr textaus       * text A0-A7
002350  41ED 0070       3453:           lea ausbuf(a5),a0
002354  43ED 0178       3454:           lea regsave(a5),a1
002358  323C 0007       3455:           move #8-1,d1
00235C                  3456: reg2d:
00235C  2019            3457:           move.l (a1)+,d0
00235E  6100 F2EA       3458:           bsr print8x
002362  10FC 0020       3459:           move.b #' ',(a0)+
002366  51C9 FFF4       3460:           dbra d1,reg2d
00236A  4210            3461:           clr.b (a0)
00236C  41ED 0070       3462:           lea ausbuf(a5),a0
002370  103C 0011       3463:           move.b #$11,d0
002374  323C 0000       3464:           move #0,d1
002378  343C 0028       3465:           move #40,d2
00237C  6100 F008       3466:           bsr textaus
002380  41FA 1A50       3467:           lea txtreg1(pc),a0
002384  103C 0011       3468:           move.b #$11,d0
002388  323C 0000       3469:           move #0,d1
00238C  343C 0032       3470:           move #50,d2
002390  6100 EFF4       3471:           bsr textaus       * text D0-D7
                        3472: *** ausgabe in hoehe 60   Quell-Zeile
002394  4A2D 0220       3473:           tst.b debug(a5)
002398  6700 0062       3474:           beq nodebugi      * kein debug Info
00239C  206D 0204       3475:           movea.l debugst(a5),a0  * start dort     PC,LINE
0023A0  226D 01CA       3476:           movea.l pcsave(a5),a1   * suchen PC SAVE
0023A4                  3477: regdsu:
0023A4  2010            3478:           move.l (a0),d0  * vergleichen
0023A6  6700 F71A       3479:           beq carres        * ende, nicht gefunden
0023AA  B089            3480:           cmp.l a1,d0       * wenn gleich ok
```

```
0023AC  6700 0006          3481:           beq reg1su
0023B0  5088               3482:           addq.l #8,a0
0023B2  60F0               3483:           bra reqdsu        * weiter beim naechsten Eintrag
                           3484:
0023B4                     3485: reg1su:                     * gefunden, also bis 0d oder 0 ausgeben
0023B4  2268 0004          3486:           move.l 4(a0),a1   * Neuer Wert A0 -> Textstart
0023B8  41ED 0070          3487:           lea ausbuf(a5),a0
0023BC  363C 004F          3488:           move #80-1,d3     * Zaehler MAX ZEICHEN
0023C0  1019               3489: reg2su:   move.b (a1)+,d0   *=0d dann ende
0023C2  6700 0018          3490:           beq reg3su        * auch ende
0023C6  B03C 000D          3491:           cmp.b #$d,d0
0023CA  6700 0010          3492:           beq reg3su
0023CE  B03C 000A          3493:           cmp.b #$a,d0
0023D2  67EC               3494:           beq reg2su        * LFs ignorieren
0023D4  10C0               3495:           move.b d0,(a0)+
0023D6  51CB FFE8          3496:           dbra d3,reg2su    * Transfer
0023DA  6008               3497:           bra.s reg4su
0023DC  10FC 0020          3498: reg3su:   move.b #' ',(a0)+  * EINGANG NODEBUGI rev 4.1
0023E0  51CB FFFA          3499:           dbra d3,reg3su    * auffuellen Rest
0023E4  4210               3500: reg4su:   clr.b (a0)        * Ende Kennung
0023E6  41ED 0070          3501:           lea ausbuf(a5),a0
0023EA  103C 0011          3502:           move.b #$11,d0
0023EE  323C 00C0          3503:           move #192,d1
0023F2  343C 0000          3504:           move #0,d2
0023F6  6100 EF8E          3505:           bsr textaus       * Ausgeben neben PC-Meldung
0023FA  4E75               3506:           rts
                           3507:
0023FC                     3508: nodebugi:                   * rev 4.1, ausgabe (pc)
0023FC  41ED 0070          3509:           lea ausbuf(a5),a0
002400  226D 01CA          3510:           movea.l pcsave(a5),a1
002404  3011               3511:           move.w (a1),d0    * Befehlscode ausgeben
002406  6100 F258          3512:           bsr print4x
00240A  363C 0028          3513:           move #40,d3       * Rest fuellen
00240E  60CC               3514:           bra reg3su        * Und Weiter wie gehabt
                           3515:
                           3516:
                           3517: ******* IO lesen/schreiben
                           3518:
002410                     3519: ioread:                     * auch Speicher zulaessig
002410  41FA 1979          3520:           lea iomsg(pc),a0
002414  6100 0ADC          3521:           bsr headaus
002418  6100 E962          3522:           bsr setpen
00241C  41FA 1976          3523:           lea io1msg(pc),a0
002420  103C 0033          3524:           move.b #$33,d0
002424  323C 000A          3525:           move #10,d1
002428  343C 000A          3526:           move #10,d2
00242C  6100 EF58          3527:           bsr textaus
002430  6100 09E6          3528:           bsr getadr        * quelladresse
002434  4A41               3529:           tst d1
002436  6700 F684          3530:           beq carset
00243A  2640               3531:           movea.l d0,a3     * merken dort
00243C  6100 0048          3532: io1rd:    bsr ioprint       * Ausgabe texte
002440  6100 E620          3533:           bsr ki
002444  B03C 0044          3534:           cmp.b #'D',d0
002448  6600 002C          3535:           bne io2rd
00244C  6100 E9EA          3536: io11rd:   bsr sync
002450  67FA               3537:           beq io11rd
002452  6100 0032          3538:           bsr ioprint
002456  6100 E580          3539:           bsr csts
00245A  67F0               3540:           beq io11rd
00245C  6100 E604          3541:           bsr ki
002460  B03C 004D          3542:           cmp.b #'M',d0
002464  6700 F65C          3543:           beq carres
002468  B03C 0053          3544:           cmp.b #'S',d0
00246C  67CE               3545:           beq io1rd
00246E  B03C 0052          3546:           cmp.b #'R',d0
002472  679C               3547:           beq ioread
002474  60D6               3548:           bra io11rd        * weiter sonst
                           3549:
002476  B03C 004D          3550: io2rd:    cmp.b #'M',d0
00247A  6700 F646          3551:           beq carres        * FINAL
00247E  B03C 0052          3552:           cmp.b #'R',d0
002482  67BC               3553:           beq ioread        * nochmals
002484  60B6               3554:           bra io1rd         * zurueck sonst
                           3555:
                           3556:
002486                     3557: ioprint:
002486  41ED 0070          3558:           lea ausbuf(a5),a0
00248A  1013               3559:           move.b (a3),d0
00248C  6100 F1DA          3560:           bsr print2x
```

PORT10

```
002490  41ED 0070      3561:          lea ausbuf(a5),a0
002494  103C 0022      3562:          move.b #$22,d0
002498  323C 00C8      3563:          move #200,d1
00249C  343C 0096      3564:          move #150,d2
0024A0  6100 EEE4      3565:          bsr textaus
0024A4  41ED 0070      3566:          lea ausbuf(a5),a0
0024A8  1013           3567:          move.b (a3),d0
0024AA  6100 F17A      3568:          bsr print8b
0024AE  41ED 0070      3569:          lea ausbuf(a5),a0
0024B2  103C 0022      3570:          move.b #$22,d0
0024B6  323C 0096      3571:          move #150,d1
0024BA  343C 0064      3572:          move #100,d2
0024BE  6100 EEC6      3573:          bsr textaus
0024C2  4E75           3574:          rts
                       3575:
0024C4                 3576: iowrite:
0024C4  41FA 18EB      3577:          lea iomsg(pc),a0
0024C8  6100 0A28      3578:          bsr headaus
0024CC  6100 094A      3579:          bsr getadr       * d0
0024D0  4A41           3580:          tst d1
0024D2  6700 F5E8      3581:          beq carset       * not valid
0024D6  2640           3582:          movea.l d0,a3    * dort merken
0024D8  41FA 18E1      3583:          lea datamsg(pc),a0
0024[illegible]  103C 0022      3584:          move.b #$22,d0
002[illegible]  323C 0014      3585:          move #20,d1
0024[illegible]  343C 0096      3586:          move #150,d2
0024E8  6100 EE9C      3587:          bsr textaus
0024EC  41ED 00F4      3588:          lea einbuf(a5),a0
0024F0  103C 0022      3589:          move.b #$22,d0
0024F4  323C 005A      3590:          move #90,d1
0024F8  343C 0096      3591:          move #150,d2
0024FC  363C 001E      3592:          move #30,d3
002500  6100 EF84      3593:          bsr textein
002504  41ED 00F4      3594:          lea einbuf(a5),a0
002508  6100 F222      3595:          bsr wert
00250C  4A41           3596:          tst d1
00250E  6700 F5AC      3597:          beq carset
002512  1680           3598:          move.b d0,(a3)  * dort ausgeben
002514  41FA 18AE      3599:          lea iop1msg(pc),a0
002518  103C 0022      3600:          move.b #$22,d0
00251C  323C 000A      3601:          move #10,d1
002520  343C 0014      3602:          move #20,d2
002524  6100 EE60      3603:          bsr textaus
002528                 3604: iopw1:
002528  6100 E538      3605:          bsr ki
00252C  B03C 0052      3606:          cmp.b #'R',d0
002530  6792           3607:          beq iowrite      * nochmals
002532  B03C 004D      3608:          cmp.b #'M',d0
002536  66F0           3609:          bne iopw1
002538  6100 E866      3610:          bsr clrall
00253C  4E75           3611:          rts
                       3612:
                       3613:
                       3614: *
                       3615: * Promprogrammierer
                       3616: *
                       3617:
00253E                 3618: promread:
00253E  11FC 0040      3619:          move.b #%01000000,proma2        * neutral stellen
        FF82
002[illegible]  41FA 17F8      3620:          lea txtp2(pc),a0
002[illegible]  6100 09A8      3621:          bsr headaus
00254C  41FA 17FC      3622:          lea txtp3(pc),a0
002550  6100 0472      3623:          bsr get3par      * a0->text. d6.l =wert1 VON  d7.l =wert2
002554  6500 F566      3624:          bcs carset       * fehler
002558  41ED 00F4      3625:          lea einbuf(a5),a0
00255C  6100 F1CE      3626:          bsr wert         * d0=wert3 NACH
002560  4A41           3627:          tst d1
002562  6700 F558      3628:          beq carset       * bei Fehler zurueck
002566  2040           3629:          movea.l d0,a0    * A0 NACH
002568                 3630: loprom:          * Lese-Schleife
002568  11C6 FF81      3631:          move.b d6,proma1
00256C  E05E           3632:          ror #8,d6
00256E  1006           3633:          move.b d6,d0
002570  E15E           3634:          rol #8,d6
002572  C03C 001F      3635:          and.b #%00011111,d0
002576  803C 0040      3636:          or.b  #%01000000,d0
00257A  11C0 FF82      3637:          move.b d0,proma2
00257E  10F8 FF80      3638:          move.b promd,(a0)+
002582  BE86           3639:          cmp.l d6,d7      * bis beide gleich
```

PROMER

```
002584  6700 0006          3640:           beq prmrdfin     * Fertig
002588  5286               3641:           addq.l #1,d6
00258A  60DC               3642:           bra loprom
                           3643:
00258C                     3644: prmrdfin:
00258C  6100 E812          3645:           bsr clrall
002590  4E75               3646:           rts
                           3647: *
                           3648:
                           3649:
002592                     3650: promwrite:
002592  41FA 179E          3651:           lea txtp1(pc),a0
002596  6100 095A          3652:           bsr headaus
00259A  41FA 17AE          3653:           lea txtp3(pc),a0
00259E  6100 0424          3654:           bsr get3par
0025A2  6500 F518          3655:           bcs carset
0025A6  41ED 00F4          3656:           lea einbuf(a5),a0
0025AA  6100 F180          3657:           bsr wert
0025AE  4A41               3658:           tst d1
0025B0  6700 F50A          3659:           beq carset       * d0=nach d6=von d7=bis
0025B4  2A00               3660:           move.l d0,d5     * d5=nach
0025B6  41FA 17A1          3661:           lea txtp4(pc),a0
0025BA  103C 0022          3662:           move.b #$22,d0
0025BE  323C 0050          3663:           move #80,d1
0025C2  343C 0050          3664:           move #80,d2
0025C6  6100 EDBE          3665:           bsr textaus      * Bereit ?
0025CA  11FC 0040          3666:           move.b #%01000000,proma2
        FF82
0025D0  6100 E490          3667:           bsr ki
0025D4  B03C 0042          3668:           cmp.b #'B',d0
0025D8  6600 F4E2          3669:           bne carset       * Abbruch wenn nicht
0025DC  2046               3670:           movea.l d6,a0    * VON in A0
0025DE  11FC 0080          3671:           move.b #%10000000,proma2
        FF82
0025E4  48E7 0580          3672:           movem.l d5/d7/a0,-(a7)  ***** ALLE WERTE MERKEN
0025E8                     3673: prlp1:
0025E8  11D0 FF80          3674:           move.b (a0),promd        * Datenwert
0025EC  11C5 FF81          3675:           move.b d5,proma1         * nach
0025F0  E05D               3676:           ror #8,d5
0025F2  1005               3677:           move.b d5,d0
0025F4  E15D               3678:           rol #8,d5
0025F6  C03C 001F          3679:           and.b #%00011111,d0
0025FA  803C 0080          3680:           or.b #%10000000,d0
0025FE  11C0 FF82          3681:           move.b d0,proma2
002602  803C 0020          3682:           or.b #%00100000,d0
002606  11C0 FF82          3683:           move.b d0,proma2
00260A  C03C 00DF          3684:           and.b #%11011111,d0
00260E  11C0 FF82          3685:           move.b d0,proma2
002612  48E7 0580          3686:           movem.l d5/d7/a0,-(a7)
                           3687: *
002616                     3688: prsyn1:
002616  6100 E820          3689:           bsr sync
00261A  67FA               3690:           beq prsyn1       * max 20ms
00261C  41ED 0070          3691:           lea ausbuf(a5),a0
002620  2005               3692:           move.l d5,d0
002622  6100 F026          3693:           bsr print8x
002626  41ED 0070          3694:           lea ausbuf(a5),a0
00262A  103C 0033          3695:           move.b #$33,d0
00262E  323C 00FA          3696:           move #250,d1
002632  343C 0050          3697:           move #80,d2
002636  6100 ED4E          3698:           bsr textaus      * LOOP
                           3699: *
00263A                     3700: prsyn2:
00263A  6100 E7FC          3701:           bsr sync
00263E  67FA               3702:           beq prsyn2       * Gesamt 60ms
002640                     3703: prsyn3:
002640  6100 E7F6          3704:           bsr sync
002644  67FA               3705:           beq prsyn3
                           3706: *
002646  4CDF 01A0          3707:           movem.l (a7)+,d5/d7/a0
00264A  BE88               3708:           cmp.l a0,d7
00264C  6700 0008          3709:           beq prgfin
002650  5288               3710:           addq.l #1,a0
002652  5285               3711:           addq.l #1,d5
002654  6092               3712:           bra prlp1
                           3713:
002656                     3714: prgfin:
002656  11FC 0040          3715:           move.b #%01000000,proma2        * neutral
        FF82
00265C  4CDF 01A0          3716:           movem.l (a7)+,d5/d7/a0  * d5=ziel a0=von d7=bis
```

```
                                3717: *
002660                          3718: prglp4:
002660  11C5 FF81               3719:           move.b d5,proma1
002664  E05D                    3720:           ror #8,d5
002666  1005                    3721:           move.b d5,d0
002668  E15D                    3722:           rol #8,d5
00266A  C03C 001F               3723:           and.b #%00011111,d0
00266E  803C 0040               3724:           or.b #%01000000,d0
002672  11C0 FF82               3725:           move.b d0,proma2
002676  1038 FF80               3726:           move.b promd,d0
00267A  B010                    3727:           cmp.b (a0),d0    * mit ziel vgl
00267C  6600 000E               3728:           bne prgnfin      * FEHLER
002680  BE88                    3729:           cmp.l a0,d7      * wenn gleich ende
002682  6700 0010               3730:           beq okfin
002686  5288                    3731:           addq.l #1,a0
002688  5285                    3732:           addq.l #1,d5
00268A  60D4                    3733:           bra prglp4
                                3734:
00268C                          3735: prgnfin:
00268C  41FA 16E8               3736:           lea txtp6(pc),a0
002690  6000 0006               3737:           bra prskip1
                                3738: *
002694                          3739: okfin:
0026[illegible]  41FA 16CE      3740:           lea txtp5(pc),a0
0026[illegible]                 3741: prskip1:
0026[illegible]  103C 0022      3742:           move.b #$22,d0
00269C  323C 0050               3743:           move #80,d1
0026A0  343C 0032               3744:           move #50,d2
0026A4  6100 ECE0               3745:           bsr textaus
0026A8  6000 0178               3746:           bra finmenue
                                3747:
                                3748:
                                3749: * Bibliotheksprogramm
                                3750: * Format
                                3751: *
                                3752: *0 - 55 AA 01 80                Kopf fuer Erkennung alle 2K moeglich
                                3753: *4 - n  n  n  n  n  n  n  n    Name des Programmes
                                3754: *12- s  s  s  s                Startadresse abs oder relativ
                                3755: *16- l  l  l  l                Programmlaenge
                                3756: *20- r                         =0 dann absolut, =1 dann relokativ
                                3757: * x  x  x                    reserve
                                3758: * x  x  x  x  x  x  x  x     reserve
                                3759: *
                                3760: *
                                3761: * 55 AA 01 80                  Naechster Eintrag kann dort folgen
                                3762:
00000020                        3763: biblen equ 32              * Laenge der Eintraege
                                3764:
                                3765:
0026AC                          3766: bibsuch:                   * a0 -> Start des Suchteils
                                3767:                            * Carry wenn kein gueltiger Eintrag
                                3768:                            *
                                3769:                            *
                                3770:                            * a1-> startadresse absolut
                                3771:                            * a2-> Start des Namens
                                3772:                            * d0.l Laenge des Moduls
                                3773:
0026AC  0C90 55AA01B0           3774:           cmp.l #$55aa0180,(a0)    * nur auf Wortgrenze ok
0026B2  6600 F408               3775:           bne carset       * Fehler,kein Eintrag
0026B6  45E8 0004               3776:           lea 4(a0),a2             * Name
0026[illegible]  2028 0010      3777:           move.l 16(a0),d0         * Laenge
0026[illegible]  2268 000C      3778:           movea.l 12(a0),a1        * Start rel
0026[illegible]  0C28 0000      3779:           cmp.b #0,20(a0)
        0014
0026C8  6700 000E               3780:           beq bibis                * wenn abs dann ok
0026CC  0C28 0001               3781:           cmp.b #1,20(a0)
        0014
0026D2  6600 F3E8               3782:           bne carset               * FEHLER
0026D6  D3C8                    3783:           adda.l a0,a1             *+ BASISADRESSE bei rel
0026D8                          3784: bibis:                             *
0026D8  6000 F3E8               3785:           bra carres
                                3786:
                                3787:
0026DC                          3788: prtbuf:                    * a0,nach ausbuf bis 0
0026DC  2F09                    3789:           move.l a1,-(a7)
0026DE  43ED 0070               3790:           lea ausbuf(a5),a1
0026E2                          3791: prt1buf:
0026E2  12D8                    3792:           move.b (a0)+,(a1)+
0026E4  66FC                    3793:           bne prt1buf
0026E6  5389                    3794:           subq.l #1,a1     * a1-> letzte Stelle
```

```
0026E8  2049                    movea.l a1,a0    * a0 als ergebnis -> in ausbuf
0026EA  225F                    movea.l (a7)+,a1
0026EC  4E75                    rts


0026EE                  bibaus:                 * Ausgabe aller vorhandenen Module
0026EE  207C 00000000           movea.l #$0,a0  * suche alle 2K von 0 anfangend
0026F4
0026F4  2F08            bib1aus:
0026F6                          move.l a0,-(a7)
0026F6  61B4            bib3aus:
0026F8  6500 00BC               bsr bibsuch
                                bcs bib2aus     * nicht gefunden
                                                * d0.l=laenge a1=startadr a2=namadr
                        * ja gefunden ausgabe durchfuehren
0026FC  2F08                    move.l a0,-(a7)
0026FE  48E7 8060               movem.l d0/a1/a2,-(a7)
002702  41FA 15EA               lea txtbi2(pc),a0       * quelle
002706  61D4                    bsr prtbuf      * start ausbuf. a0->freie Stelle
002708  303C 0007               move #8-1,d0
00270C                  bib4au:
00270C  10DA                    move.b (a2)+,(a0)+
00270E  51C8 FFFC               dbra d0,bib4au
002712  4210                    clr.b (a0)
002714  103C 0033               move.b #$33,d0
002718  41ED 0070               lea ausbuf(a5),a0
00271C  323C 000A               move #10,d1
002720  343C 00A0               move #160,d2
002724  6100 EC60               bsr textaus
002728  4CDF 0601               movem.l (a7)+,d0/a1/a2
00272C  48E7 8060               movem.l d0/a1/a2,-(a7)
002730  41FA 15C6               lea txtbi3(pc),a0
002734  61A6                    bsr prtbuf      * start ausbuf. a0->freie Stelle
002736  2009                    move.l a1,d0
002738  6100 EF1E               bsr print6x     * Startadresse in d0 ausgeben
00273C  4210                    clr.b (a0)
00273E  103C 0033               move.b #$33,d0
002742  41ED 0070               lea ausbuf(a5),a0
002746  323C 000A               move #10,d1
00274A  343C 0082               move #130,d2
00274E  6100 EC36               bsr textaus
002752  4CDF 0601               movem.l (a7)+,d0/a1/a2
002756  48E7 8060               movem.l d0/a1/a2,-(a7)
00275A  41FA 15A6               lea txtbi4(pc),a0
00275E  6100 FF7C               bsr prtbuf      * start ausbuf. a0->freie Stelle
002762  6100 EEF4               bsr print6x     * Laenge
002766  4210                    clr.b (a0)
002768  103C 0033               move.b #$33,d0
00276C  41ED 0070               lea ausbuf(a5),a0
002770  323C 000A               move #10,d1
002774  343C 0064               move #100,d2
002778  6100 EC0C               bsr textaus
00277C  4CDF 0601               movem.l (a7)+,d0/a1/a2
002780  6100 E2E0               bsr ki          * EINGABE M.cr.J moeglich
002784  B03C 004D               cmp.b #'M',d0
002788  6606                    bne.s bib5au
00278A  205F                    movea.l (a7)+,a0        * 2 mal dummy
00278C  205F                    movea.l (a7)+,a0
00278E  4E75                    rts

002790  B03C 004A       bib5au: cmp.b #'J',d0
002794  6614                    bne.s bibcaus
002796  205F                    movea.l (a7)+,a0        * 2 mal dummy
002798  205F                    movea.l (a7)+,a0
00279A  6100 E604               bsr clrall      * alles Loeschen
00279E  4E91                    jsr (a1)        * start Programm
0027A0  4BFA D85E               lea basis(pc),a5
0027A4  DBFA DC5E               adda.l ramstart(pc),a5  * rev 4.0b
0027A8  4E75                    rts                     * wieder definieren


0027AA                  bibcaus:
0027AA  205F                    movea.l (a7)+,a0
0027AC  D1FC 00000020           adda.l #biblen,a0       * suche dahinter
0027B2  6000 FF42               bra bib3aus

                        bib2aus:                * Nur bei 68008
0027B6  205F                    movea.l (a7)+,a0
0027B8  D1FC 00000400           adda.l #$400,a0         * 1K Blockabstand moeglich
0027BE  B1FC 000F0000           cmp.l #$0f0000,a0       * ENDE 1MB-64K
```

```
0027C4  6600 FF2E       3875:           bne biblaus
0027C8  4E75            3876:           rts
                        3877:
                        3878:
0027CA                  3879: bibo:                     * Ausgabe aller Bibliotheken
0027CA  41FA 1517       3880:           lea txtbi1(pc),a0
0027CE  6100 0722       3881:           bsr headaus
0027D2  41FA 1538       3882:           lea txtbi5(pc),a0
0027D6  103C 0022       3883:           move.b #$22,d0
0027DA  323C 000A       3884:           move #10,d1
0027DE  343C 0032       3885:           move #50,d2
0027E2  6100 EBDA       3886:           bsr textorint
0027E6  323C 0005       3887:           move #10-5,d1
0027EA  343C 005F       3888:           move #100-5,d2
0027EE  6100 E6F2       3889:           bsr moveto
0027F2  343C 00C3       3890:           move #190+5,d2
0027F6  6100 E754       3891:           bsr drawto
0027FA  323C 01FA       3892:           move #511-5,d1
0027FE  6100 E74C       3893:           bsr drawto
002802  343C 005F       3894:           move #100-5,d2
002806  6100 E744       3895:           bsr drawto
00280A  323C 0005       3896:           move #10-5,d1
00280E  6100 E73C       3897:           bsr drawto
002812  6100 FEDA       3898:           bsr bibaus      * Ausgabe und Start
0028[illegible]  6000 000A       3899:           bra finmenue    * ENDE mit M=Menue-Eingabe
                        3900:
00281A                  3901:
00281A  6100 D7E4       3902: asstext:
00281E  6000 0002       3903:           bsr assemble    * aufruf assembler
                        3904:           bra finmenue    * warten und ruecksetzen dort
002822                  3905:
                        3906: finmenue:
002822  422D 0217       3907:
002826  1B7C 0001       3908:           clr.b viewpage(a5)
        0218            3909:           move.b #1,wrtpage(a5)
00282C  6100 E5E6       3910:           bsr aktpage
002830  41FA 131B       3911:           lea fimsg(pc),a0
002834  303C 0011       3912:           move #$11,d0
002838  323C 0190       3913:           move #400,d1
00283C  343C 0001       3914:           move #1,d2      * nicht 245
002840  6100 EB44       3915:           bsr textaus
002844  422D 0218       3916:           clr.b wrtpage(a5)
002848  6100 E5CA       3917:           bsr aktpage
00284C  41FA 12FF       3918:           lea fimsg(pc),a0
002850  303C 0011       3919:           move #$11,d0
002854  323C 0190       3920:           move #400,d1
002858  343C 0001       3921:           move #1,d2      * nicht 245 sondern bott
00285C  6100 EB28       3922:           bsr textaus
002860                  3923: fin1m:
002860  6100 E200       3924:           bsr ki
002864  B03C 0046       3925:           cmp.b #'F',d0
002868  661C            3926:           bne.s fin2m
00286A  4A2D 002E       3927:           tst.b flip(a5)
00286E  6608            3928:           bne.s fin12m
002870  1B7C 0001       3929:           move.b #1,flip(a5)
        002E
002876  60E8            3930:           bra.s fin1m
002878  422D 002E       3931: fin12m:   clr.b flip(a5)  * umschalten Flip oder nicht
00287C  422D 0217       3932:           clr.b viewpage(a5)
00[illegible]0  6100 E592       3933:           bsr aktpage     * Auf Seite 0 ansehen
00[illegible]4  60DA            3934:           bra fin1m
00[illegible]6                  3935: fin2m:
002886  B03C 004D       3936:           cmp.b #'M',d0
00288A  66D4            3937:           bne fin1m
00288C  6100 E512       3938:           bsr clrall
002890  422D 0222       3939:           clr.b curon(a5)
002894  422D 002E       3940:           clr.b flip(a5)
002898  422D 0030       3941:           clr.b flip1(a5)
00289C  1B7C 0001       3942:           move.b #1,first(a5)     *grapoff
        021D
0028A2  4E75            3943:           rts
                        3944:
0028A4                  3945: aendere:                  * Aenderungsmenue
0028A4  41FA 13FA       3946:           lea txta1(pc),a0
0028A8  6100 0648       3947:           bsr headaus     * Ueberschrift ausgeben
0028AC                  3948: aend1:
0028AC  103C 0022       3949:           move.b #$22,d0
0028B0  323C 000A       3950:           move #10,d1
0028B4  343C 0032       3951:           move #50,d2
```

```
0028B8  41FA 13F7      3952:           lea txta2(pc),a0
0028BC  6100 EB00      3953:           bsr textprint
0028C0  6100 0556      3954:           bsr getadr
0028C4  4A41           3955:           tst d1
0028C6  6700 F1F4      3956:           beq carset        * Fehlerausgang Aenderungsmenue
0028CA  2240           3957:           movea.l d0,a1     * Pointer auf Zielgebiet
0028CC                 3958: aendlp:
0028CC  2449           3959:           movea.l a1,a2     * retten
0028CE  41ED 0070      3960:           lea ausbuf(a5),a0
0028D2  2009           3961:           move.l a1,d0
0028D4  6100 ED82      3962:           bsr print6x
0028D8  10FC 003A      3963:           move.b #':',(a0)+
0028DC  363C 0003      3964:           move #4-1,d3
0028E0  101A           3965: aend2:    move.b (a2)+,d0
0028E2  6100 ED84      3966:           bsr print2x
0028E6  10FC 0020      3967:           move.b #' ',(a0)+
0028EA  51CB FFF4      3968:           dbra d3,aend2
0028EE  4210           3969:           clr.b (a0)
0028F0  41ED 0070      3970:           lea ausbuf(a5),a0
0028F4  103C 0022      3971:           move.b #$22,d0
0028F8  323C 0014      3972:           move #20,d1
0028FC  343C 00A0      3973:           move #160,d2
002900  6100 EA84      3974:           bsr textaus       * ausgabe daten alt
002904                 3975: aend2a:
002904  41ED 00F4      3976:           lea einbuf(a5),a0       * Ziel ist Eingabebuffer
002908  103C 0022      3977:           move.b #$22,d0
00290C  323C 0014      3978:           move #20,d1
002910  343C 006E      3979:           move #110,d2
002914  363C 001E      3980:           move #30,d3
002918  6100 EB6C      3981:           bsr textein       * daten holen  -.M reserviert
00291C  B83C 0000      3982:           cmp.b #0,d4       * nur cr
002920  6604           3983:           bne.s aend2aa
002922  5289           3984:           addq.l #1,a1
002924  60A6           3985:           bra aendlp
002926                 3986:
002926                 3987: aend2aa:
002926  B83C 0001      3988:           cmp.b #1,d4
00292A  6600 0026      3989:           bne aend3         * nein dann weiter
00292E  102D 00F4      3990:           move.b einbuf(a5),d0
002932  6100 F1FA      3991:           bsr bucheck       * to upper
002936  B03C 002D      3992:           cmp.b #'-',d0
00293A  6600 0006      3993:           bne aend2b
00293E  5389           3994:           subq.l #1,a1
002940  608A           3995:           bra aendlp
002942                 3996:
002942                 3997: aend2b:
002942  B03C 0052      3998:           cmp.b #'R',d0
002946  6700 FF64      3999:           beq aend1         * neue Adresse eingeben
00294A  B03C 004D      4000:           cmp.b #'M',d0
00294E  6700 006E      4001:           beq aendfin       * ENDE der Eingabe
                       4002:                             * weiter bei aend3 sonst
002952                 4003: aend3:
002952  48E7 0060      4004:           movem.l a1/a2,-(a7)
002956  41ED 00F4      4005:           lea einbuf(a5),a0
00295A  6100 ED70      4006:           bsr zuweis
00295E  4CDF 0600      4007:           movem.l (a7)+,a1/a2     * carry bleibt ggf
002962  6400 FF68      4008:           bcc aendlp        * erneute ausgabe ohne aenderung
002966  41ED 00F4      4009:           lea einbuf(a5),a0
00296A  48E7 0060      4010:           movem.l a1/a2,-(a7)
00296E  6100 EDBC      4011:           bsr wert
002972  4CDF 0600      4012:           movem.l (a7)+,a1/a2
002976  4A41           4013:           tst d1
002978  6700 FF52      4014:           beq aendlp        * ignorieren
                       4015:                             * d1=1,2,3  anzahl
00297C  B23C 0001      4016:           cmp.b #1,d1
002980  6600 0008      4017:           bne aend4
002984  12C0           4018:           move.b d0,(a1)+
002986  6000 FF44      4019:           bra aendlp
                       4020:
00298A  B23C 0002      4021: aend4:    cmp.b #2,d1
00298E  6600 000E      4022:           bne aend5
002992  E058           4023:           ror #8,d0
002994  12C0           4024:           move.b d0,(a1)+
002996  E058           4025:           ror #8,d0
002998  12C0           4026:           move.b d0,(a1)+
00299A  6000 FF30      4027:           bra aendlp
                       4028:
00299E  B23C 0003      4029: aend5:    cmp.b #3,d1
0029A2  6600 FF28      4030:           bne aendlp        * IGNORE Udefs etc
0029A6  4840           4031:           swap d0
```

```
0029A8  E058            4032:           ror #8.d0
0029AA  12C0            4033:           move.b d0.(a1)+
0029AC  E058            4034:           ror #8.d0
0029AE  12C0            4035:           move.b d0.(a1)+
0029B0  4840            4036:           swap d0
0029B2  E058            4037:           ror #8.d0
0029B4  12C0            4038:           move.b d0.(a1)+
0029B6  E058            4039:           ror #8.d0
0029B8  12C0            4040:           move.b d0.(a1)+
0029BA  6000 FF10       4041:           bra aendlp
0029BE                  4042: aendfin:
0029BE  6100 E3E0       4043:           bsr clrall
0029C2  4E75            4044:           rts
                        4045:
                        4046:
                        4047:
0029C4                  4048: get3par:          * a0-> textblock
                        4049:                   * d6.1 = wert1  d7.1 = wert2  name in einbuf
0029C4  103C 0022       4050:           move.b #$22.d0
0029C8  323C 000A       4051:           move #10.d1
0029CC  343C 00B4       4052:           move #180.d2
0029D0  6100 E9EC       4053:           bsr textprint
0029D4  103C 0022       4054:           move.b #$22.d0
0029D8  323C 0096       4055:           move #150.d1
0029DC  343C 00B4       4056:           move #180.d2
0029E0  363C 001E       4057:           move #30.d3
0029E4  41ED 00F4       4058:           lea einbuf(a5).a0
0029E8  6100 EA9C       4059:           bsr textein     *d4.d5 zerstoert
0029EC  41ED 00F4       4060:           lea einbuf(a5).a0
0029F0  6100 ED3A       4061:           bsr wert        * datenwert holen
0029F4  4A41            4062:           tst d1
0029F6  6700 F0C4       4063:           beq carset      * fehler
0029FA  2C00            4064:           move.l d0.d6
0029FC                  4065: get3rest:                 * einsprung restdata
0029FC  103C 0022       4066:           move.b #$22.d0
002A00  323C 0096       4067:           move #150.d1
002A04  343C 00A0       4068:           move #160.d2
002A08  363C 001E       4069:           move #30.d3
002A0C  41ED 00F4       4070:           lea einbuf(a5).a0
002A10  6100 EA74       4071:           bsr textein     *d4.d5 zerstoert
002A14  41ED 00F4       4072:           lea einbuf(a5).a0
002A18  6100 ED12       4073:           bsr wert        * datenwert holen
002A1C  4A41            4074:           tst d1
002A1E  6700 F09C       4075:           beq carset      * fehler
002A22  2E00            4076:           move.l d0.d7    * wert 2
002A24  103C 0022       4077:           move.b #$22.d0
002A28  323C 0096       4078:           move #150.d1
002A2C  343C 008C       4079:           move #140.d2
002A30  363C 001E       4080:           move #30.d3
002A34  41ED 00F4       4081:           lea einbuf(a5).a0
002A38  6100 EA4C       4082:           bsr textein     *d4.d5 zerstoert
002A3C  6000 F084       4083:           bra carres      * ok kein fehler
                        4084:
                        4085:
002A40                  4086: get2par:          * a0 -> start
                        4087:                   * d7.1 = wert1  name in einbuf
002A40  103C 0022       4088:           move.b #$22.d0
002A44  323C 000A       4089:           move #10.d1
002A48  343C 00A0       4090:           move #160.d2
002A4C  6100 E970       4091:           bsr textprint
002A50  60AA            4092:           bra get3rest
                        4093:
                        4094:
002A52                  4095: namepoo:                  * Name ausgeben auf Kassette
002A52  363C 0027       4096:           move #40-1.d3
002A56                  4097: text1cx:
002A56  103C 00FF       4098:           move.b #$ff.d0
002A5A  6100 E2AE       4099:           bsr poo
002A5E  51CB FFF6       4100:           dbra d3,text1cx
002A62  103C 0000       4101:           move.b #0.d0
002A66  6100 E2A2       4102:           bsr poo
002A6A  103C 002F       4103:           move.b #'/'.d0  * FFFFFFFF 0 / name 0d ff ff
002A6E  6100 E29A       4104:           bsr poo
002A72  41ED 00F4       4105:           lea einbuf(a5).a0
002A76  363C 001D       4106:           move #30-1.d3   * savetv
002A7A                  4107: text2cx:
002A7A  1018            4108:           move.b (a0)+.d0
002A7C  6700 000A       4109:           beq text3cx
002A80  6100 E288       4110:           bsr poo         * ausgabe ausser terminator
002A84  51CB FFF4       4111:           dbra d3,text2cx
```

```
002A88
002A88  103C 000D      4112: text3cx:
002A8C  6100 E27C      4113:           move.b #$d,d0
002A90  363C 0014      4114:           bsr poo          * ende Zeichen
002A94                 4115:           move #20,d3
002A94  103C 00FF      4116: text4cx:
002A98  6100 E270      4117:           move.b #$ff,d0
002A9C  51CB FFF6      4118:           bsr poo
002AA0  4E75           4119:           dbra d3,text4cx
                       4120:           rts
                       4121:
002AA2                 4122:
002AA2  41FA 1133      4123: textcas:                   * text auf Kassette
002AA6  6100 044A      4124:           lea textcas1(pc),a0
002AAA  41FA 11A0      4125:           bsr headaus
002AAE  6190           4126:           lea textcas8(pc),a0      *
002AB0  6500 F00A      4127:           bsr get2par      * d7 = textstart
                       4128:           bcs carset       * bei Fehler abbruch
002AB4  6100 E238      4129:                            * sonst Anfangen
002AB8  2F00           4130:           bsr relaisan     * Starten
002ABA  7028           4131:           move.l d0,-(a7)
002ABC  6100 E39C      4132:           move.l #40,d0    * 4 Sek. Warten
002AC0  201F           4133:           bsr delay        * bis Motor eingeschaltet
002AC2  618E           4134:           move.l (a7)+,d0
                       4135:           bsr namepoo      * Name ausgeben zuerst
                       4136:                            * dann Datenbereich ohne ADRESSE
                       4137:                            * FF FF FF FF  0  daten 0 CRC CRC FF FF FF
002AC4  363C 000A      4138:           move #10,d3
002AC8  103C 00FF      4139: txt1c:    move.b #$ff,d0
002ACC  6100 E23C      4140:           bsr poo
002AD0  51CB FFF6      4141:           dbra d3,txt1c    *
002AD4  103C 0000      4142:           move.b #$0,d0
002AD8  6100 E230      4143:           bsr poo          * FF FF FF FF 0
002ADC  2047           4144:           movea.l d7,a0    * Pointer auf Text
002ADE  4245           4145:           clr d5           * CHECKSUM.W
002AE0  4240           4146:           clr d0           * msb = 0
002AE2  1018           4147: txt2c:    move.b (a0)+,d0
002AE4  6708           4148:           beq.s txt3c      *=0 dann Ende erreicht
002AE6  DA40           4149:           add d0,d5        * CHECKSUM
002AE8  6100 E220      4150:           bsr poo
002AEC  60F4           4151:           bra txt2c
002AEE  4200           4152: txt3c:    clr.b d0
02AF0  6100 E218       4153:           bsr poo          * 0 als Abschluss
02AF4  3005            4154:           move d5,d0
02AF6  E058            4155:           ror #8,d0
02AF8  6100 E210       4156:           bsr poo          * MSB ausgeben
02AFC  3005            4157:           move d5,d0
02AFE  6100 E20A       4158:           bsr poo          * LSB ausgeben
02B02  363C 0014       4159:           move #20,d3
02B06  103C 00FF       4160: txt4c:    move.b #$ff,d0
02B0A  6100 E1FE       4161:           bsr poo
02B0E  51CB FFF6       4162:           dbra d3,txt4c    * bis fertig
02B12  6100 E1D2       4163:           bsr relaisaus    * und stoppen CAS
02B16  6100 E288       4164:           bsr clrall
02B1A  4E75            4165:           rts              * ENDE CAS-AUSGABE TEXT
                       4166:
02B1C                  4167:
02B1C  41FA 10C8       4168: datacas:                   * daten auf Kassette
02B20  6100 03D0       4169:           lea textcas2(pc),a0
02B24  41FA 1102       4170:           bsr headaus
02B28  6100 FE9A       4171:           lea textcas7(pc),a0      *
02B2C  6500 EF8E       4172:           bsr get3par      * d6=start d7=ende
                       4173:           bcs carset       * bei Fehler abbruch
02B30  6100 E1BC       4174:                            * sonst Anfangen
02B34  2F00            4175:           bsr relaisan     * Starten
02B36  7028            4176:           move.l d0,-(a7)
02B38  6100 E320       4177:           move.l #40,d0    * 4 Sek. Warten
02B3C  201F            4178:           bsr delay        * bis Motor eingeschaltet
02B3E  6100 FF12       4179:           move.l (a7)+,d0
                       4180:           bsr namepoo      * Name ausgeben zuerst
                       4181:                            * dann Datenbereich
                       4182:                            * FF FF FF 00 : st4 st3 st2 st1 en4 en3 en
                       4183:                            * data check4 check3 check2 check1 FF FF F
                       4184:
02B42  363C 0013       4185:           move #20-1,d3
02B46  103C 00FF       4186: dat1c:    move.b #$ff,d0
02B4A  6100 E1BE       4187:           bsr poo
02B4E  51CB FFF6       4188:           dbra d3,dat1c
02B52  4200            4189:           clr.b d0
02B54  6100 E1B4       4190:           bsr poo
02B58  103C 003B       4191:           move.b #':',d0   * Kennung 68008-Daten-Format
```

```
002B5C  6100 E1AC       4192:           bsr poo
002B60  4285            4193:           clr.l d5            * CRC-merker      vn st bis dataende
002B62  2006            4194:           move.l d6,d0
002B64  6100 0042       4195:           bsr put8poo
002B68  DA86            4196:           add.l d6,d5
002B6A  2007            4197:           move.l d7,d0
002B6C  6100 003A       4198:           bsr put8poo         * st4..st1 en4..en1
002B70  DA87            4199:           add.l d7,d5         * und endadr
002B72  2046            4200:           movea.l d6,a0       * startpointer
002B74  2247            4201:           movea.l d7,a1       * ende  inclusive
002B76  4280            4202:           clr.l d0
002B78                  4203: dat2c:
002B78  1010            4204:           move.b (a0),d0
002B7A  DA80            4205:           add.l d0,d5         *CHECKSUM
002B7C  6100 E18C       4206:           bsr poo             * wenn beide gleich dann 1 byte aus
002B80  B3C8            4207:           cmpa.l a0,a1
002B82  6704            4208:           beq.s dat3c
002B84  5248            4209:           addq #1,a0
002B86  60F0            4210:           bra dat2c
002B88  2005            4211: dat3c:    move.l d5,d0
002B8A  6100 001C       4212:           bsr put8poo
002B8E  363C 0013       4213:           move #20-1,d3       * alle vier CHECKSUM-Bytes ausgeben
002B92  103C 00FF       4214: dat4c:    move.b #$ff,d0
002B96  6100 E172       4215:           bsr poo
002B9A  51CB FFF6       4216:           dbra d3,dat4c
002B9E  6100 E146       4217:           bsr relaisaus       * und stoppen Relais
002BA2  6100 E1FC       4218:           bsr clrall
002BA6  4E75            4219:           rts
                        4220:
002BA8                  4221: put8ooo:
002BA8  4840            4222:           swap d0
002BAA  E058            4223:           ror #8,d0
002BAC  6100 E15C       4224:           bsr poo
002BB0  E058            4225:           ror #8,d0
002BB2  6100 E156       4226:           bsr poo
002BB6  4840            4227:           swap d0
002BB8  E058            4228:           ror #8,d0
002BBA  6100 E14E       4229:           bsr poo
002BBE  E058            4230:           ror #8,d0
002BC0  6100 E148       4231:           bsr poo
002BC4  4E75            4232:           rts
                        4233:
002BC6                  4234: nameausc:                     * Ausgabe Namen von CAS
002BC6  6100 E12E       4235:           bsr ri
002BCA  B03C 00FF       4236:           cmp.b #$ff,d0       * warten bis ff
002BCE  66F6            4237:           bne nameausc
002BD0  6100 E124       4238: name1c:   bsr ri
002BD4  B03C 0000       4239:           cmp.b #$0,d0
002BD8  66F6            4240:           bne name1c
002BDA  6100 E11A       4241:           bsr ri
002BDE  B03C 002F       4242:           cmp.b #'/',d0
002BE2  66E2            4243:           bne nameausc        * nein, dann zurueck
002BE4  41ED 0070       4244:           lea ausbuf(a5),a0
002BE8  6100 E10C       4245: name2c:   bsr ri
002BEC  B03C 000D       4246:           cmp.b #$d,d0
002BF0  6704            4247:           beq.s name3c
002BF2  10C0            4248:           move.b d0,(a0)+
002BF4  60F2            4249:           bra name2c
002BF6  4210            4250: name3c:   clr.b (a0)          * ende
002BF8  41ED 0070       4251:           lea ausbuf(a5),a0
002BFC  103C 0022       4252:           move.b #$22,d0
002C00  323C 0014       4253:           move #20,d1
002C04  343C 0080       4254:           move #128,d2
002C08  6100 E77C       4255:           bsr textaus         * Ausgabe text
002C0C  4E75            4256:           rts
                        4257:
                        4258:
002C0E                  4259: castext:                      * Kassette Text in Speicher
                        4260:                               * d6=1 dann prueflesen
002C0E  41FA 0FE6       4261:           lea textcas3(pc),a0
002C12  4A46            4262:           tst d6
002C14  6700 0006       4263:           beq casitx
002C18  41FA 0FF3       4264:           lea textcas5(pc),a0
002C1C  6100 02D4       4265: casitx:   bsr headaus
002C20  6100 01F6       4266:           bsr getadr          * Startadresse holen
002C24  4A41            4267:           tst d1
002C26  6700 EE94       4268:           beq carset          * Fehler, sonst d0.l = wert
002C2A  6100 E0C2       4269:           bsr relaisan        * Relais an, rev 4.2
002C2E  2E00            4270:           move.l d0,d7        * dort merken Zieladr
002C30  6194            4271:           bsr nameausc
```

```
002C32
002C32  6100 E0C2
002C36  B03C 00FF
002C3A  66F6
002C3C
002C3C  6100 E0B8
002C40  B03C 0000
002C44  66F6
002C46  2047
002C48  4245
002C4A  4240
002C4C
002C4C  6100 E0A8
002C50  B03C 0000
002C54  6700 0016
002C58  DA40
002C5A  4A46
002C5C  6700 000A
002C60  B018
002C62  6600 0046
002C66  60E4
002C68
002C68  10C0
002C6A  60E0
002C6C
002C6C  4A46
002C6E  6602
002C70  4210
002C72
002C72  6100 E082
002C76  E05D
002C78  BA00
002C7A  6600 0034
002C7E  6100 E076
002C82  E05D
002C84  BA00
002C86  6600 002B
002C8A  4A46
002C8C  6600 0028
002C90  2B47 0036
002C94  2B47 003A
002C98  2B48 003E
002C9C  2B4B 0042
002CA0  6000 0014

002CA4
002CA4  41FA 0FE4
002CA8  6010
002CAA
002CAA  41FA 0FB8
002CAE  600A
002CB0
002CB0  41FA 0FC3
002CB4  6004
002CB6
002CB6  41FA 0FCF
002CBA
002CBA  6100 E02A
002CBE  103C 0033
002CC2  323C 000A
002CC6  343C 000A
002CCA  6100 E6BA
002CCE  6000 FB52


002CD2

002CD2  6100 E01A
002CD6  41FA 0F29
002CDA  4A46
002CDC  6700 0006
002CE0  41FA 0F38
002CE4  6100 020C
002CE8  6100 FEDC
002CEC  6100 E008
002CF0  B03C 00FF
002CF4  66F6
002CF6  6100 DFFE
002CFA  B03C 0000
002CFE  66F6
```

```
4272: cas2tx:
4273:           bsr ri
4274:           cmp.b #$ff.d0
4275:           bne cas2tx
4276: cas3tx:
4277:           bsr ri
4278:           cmp.b #$0.d0
4279:           bne cas3tx      * ff ff ff 0
4280:           movea.l d7.a0           * Ziel
4281:           clr d5          * Pruefsumme .d6=1 dann nur vgl.
4282:           clr d0
4283: cas4tx:
4284:           bsr ri
4285:           cmp.b #$0.d0
4286:           beq cas5tx      *   ende gefunden
4287:           add d0.d5       * Pruefsumme
4288:           tst d6
4289:           beq cas4atx
4290:           cmp.b (a0)+.d0
4291:           bne casvgl      * Fehler beim Vergleich
4292:           bra cas4tx
4293: cas4atx:
4294:           move.b d0.(a0)+
4295:           bra cas4tx
4296: cas5tx:                   * a0-> ende textblock
4297:           tst d6
4298:           bne.s cas6tx
4299:           clr.b (a0)      * immer 0-setzen
4300: cas6tx:
4301:           bsr ri          * MSB CHECK
4302:           ror #8.d5
4303:           cmp.b d0.d5
4304:           bne cascrc
4305:           bsr ri
4306:           ror #8.d5
4307:           cmp.b d0.d5
4308:           bne cascrc      * d7 pointer auf startext
4309:           tst d6
4310:           bne casok
4311:           move.l d7.stxtxt(a5)
4312:           move.l d7.akttxt(a5)
4313:           move.l a0.etxtxt(a5)
4314:           move.l a0.eottxt(a5)
4315:           bra casok
4316:
4317: casformat:
4318:           lea textcas12(pc).a0
4319:           bra.s casende
4320: casvgl:                    * Vergleichsfehler Kassette
4321:           lea textcas9(pc).a0
4322:           bra.s casende
4323: cascrc:                    * CRC Fehler
4324:           lea textcas10(pc).a0
4325:           bra.s casende
4326: casok:                     * OK richtig geladen
4327:           lea textcas11(pc).a0
4328: casende:
4329:           bsr relaisaus   * nur aus. wenn ok rev 4.2
4330:           move.b #$33.d0
4331:           move #10.d1
4332:           move #10.d2
4333:           bsr textaus
4334:           bra finmenue
4335:
4336:
4337: casdata:                   * Kassette Data in Speicher
4338:                           * d6=1 dann prueflesen
4339:           bsr relaisan    * Relais anschalten
4340:           lea textcas4(pc).a0
4341:           tst d6
4342:           beq cas1dx
4343:           lea textcas6(pc).a0
4344: cas1dx: bsr headaus
4345:           bsr nameausc
4346: cas2dx: bsr ri
4347:           cmp.b #$ff.d0
4348:           bne cas2dx
4349: cas3dx: bsr ri
4350:           cmp.b #$0.d0
4351:           bne cas3dx      * FF FF FF  00
```

```
002D00  6100 DFF4       4352:           bsr ri                  * kennung
002D04  B03C 003B       4353:           cmp.b #':',d0
002D08  669A            4354:           bne casformat           * Lade-Format-Fehler
002D0A  4285            4355:           clr.l d5
002D0C  6100 0046       4356:           bsr get8ri              * STARTADRESSE
002D10  2040            4357:           move.l d0,a0
002D12  DA80            4358:           add.l d0,d5
002D14  6100 003E       4359:           bsr get8ri
002D18  2240            4360:           move.l d0,a1            * ENDADRESSE
002D1A  DA80            4361:           add.l d0,d5
002D1C  B3C8            4362:           cmpa.l a0,a1            * d6(=d7 dann ok
002D1E  6590            4363:           bcs cascrc              * wie CRC-Fehler
002D20  4280            4364:           clr.l d0                * checksumrest
002D22  6100 DFD2       4365: cas4dx:   bsr ri                  * mind ein Wert
002D26  DA80            4366:           add.l d0,d5
002D28  4A46            4367:           tst d6
002D2A  6700 0010       4368:           beq cas4adx
002D2E  B010            4369:           cmp.b (a0),d0
002D30  6600 FF78       4370:           bne casvgl
002D34  B3C8            4371:           cmpa.l a0,a1
002D36  670E            4372:           beq.s cas5dx
002D38  5288            4373:           addq.l #1,a0
002D3A  60E6            4374:           bra cas4dx
002D3C                  4375: cas4adx:
002D3C  1080            4376:           move.b d0,(a0)
002D3E  B3C8            4377:           cmpa.l a0,a1
002D40  6704            4378:           beq.s cas5dx
002D42  5288            4379:           addq.l #1,a0
002D44  60DC            4380:           bra cas4dx
002D46  6100 000C       4381: cas5dx:   bsr get8ri
002D4A  BA80            4382:           cmp.l d0,d5
002D4C  6600 FF62       4383:           bne cascrc
002D50  6000 FF64       4384:           bra casok
                        4385:
002D54                  4386: get8ri:                           * rev 4.3. ri nicht mehr nur .b
002D54  2F01            4387:           move.l d1,-(a7)         * daher Hilfsregister noetig
002D56  6100 DF9E       4388:           bsr ri
002D5A  1200            4389:           move.b d0,d1            * nur Byte transportieren
002D5C  E159            4390:           rol #8,d1               * dann lsb dahinter
002D5E  6100 DF96       4391:           bsr ri
002D62  1200            4392:           move.b d0,d1
002D64  4841            4393:           swap d1                 * das ganze austauschen
002D66  6100 DF8E       4394:           bsr ri                  * und rest dazu
002D6A  1200            4395:           move.b d0,d1
002D6C  E159            4396:           rol #8,d1
002D6E  6100 DF86       4397:           bsr ri                  * letztes Byte.
002D72  1200            4398:           move.b d0,d1
002D74  2001            4399:           move.l d1,d0            * Datenwort fertig
002D76  221F            4400:           move.l (a7)+,d1         * in d0.l ist das Ergebnis
002D78  4E75            4401:           rts
                        4402:
002D7A                  4403: startex:                          * Starten eines Programms
002D7A  41FA 0DC0       4404:           lea stmsg1(pc),a0
002D7E  6100 0172       4405:           bsr headaus             * ausgabe an Ueberschriftposition
002D82  6100 0094       4406:           bsr getadr
002D86  4A41            4407:           tst d1                  * d0.l = wert d1 = flag
002D88  6700 ED32       4408:           beq carset              * fehler ende
002D8C  2F00            4409:           move.l d0,-(a7)
002D8E  6100 E010       4410:           bsr clrall              * alles loeschen
002D92  1B7C 0001       4411:           move.b #1,first(a5)     * grap first aufruf
        021D
002D98  1B7C 0001       4412:           move.b #1,turdo(a5)     * turtle down
        021E
002D9E  205F            4413:           movea.l (a7)+,a0 * Zieladresse
002DA0  4E90            4414:           jsr (a0)                * und ausfuehren
002DA2  4BFA D25C       4415:           lea basis(pc),a5        * Falls Anwenderprog
002DA6  DBFA D65C       4416:           adda.l ramstart(pc),a5  * Pointer zerstoert hat
002DAA  6100 FA76       4417:           bsr finmenue
002DAE  1B7C 0001       4418:           move.b #1,first(a5)     *erst hier down
        021D
002DB4  4E75            4419:           rts
                        4420:
                        4421:
002DB6                  4422: testex:
002DB6  41ED 00F4       4423:           lea einbuf(a5),a0
002DBA  103C 0022       4424:           move.b #$22,d0
002DBE  323C 000A       4425:           move #10,d1
002DC2  343C 0032       4426:           move #50,d2
002DC6  363C 001E       4427:           move #30,d3
002DCA  6100 E6BA       4428:           bsr textein
```

```
002DCE  41ED 00F4       4429:           lea einbuf(a5),a0
002DD2  6100 E8F8       4430:           bsr zuweis      *a0 bleibt im Fehlerfall
002DD6  6404            4431:           bcc.s testex1
002DD8  6100 E952       4432:            bsr wert       *holen datenwert
002DDC  41ED 0070       4433: testex1: lea ausbuf(a5),a0
002DE0  6100 E868       4434:           bsr print8x     *d0
002DE4  10FC 0020       4435:           move.b #' ',(a0)+
002DE8  3001            4436:           move d1,d0
002DEA  6100 E85E       4437:           bsr print8x
002DEE  10FC 0020       4438:           move.b #' ',(a0)+
002DF2  3002            4439:           move d2,d0
002DF4  6100 E854       4440:           bsr print8x
002DF8  41ED 0070       4441:           lea ausbuf(a5),a0
002DFC  103C 0022       4442:           move.b #$22,d0
002E00  323C 000A       4443:           move #10,d1
002E04  343C 0064       4444:           move #100,d2
002E08  6100 E57C       4445:           bsr textaus
002E0C  6100 DC54       4446:           bsr ki
002E10  B03C 0057       4447:           cmp.b #'W',d0
002E14  66A0            4448:           bne testex
002E16  4E75            4449:           rts
                        4450:
                        4451:
                        4452: *** Speicherinhalt ausgeben
                        4453:
002E18                  4454: getadr:                   *routine liefert in d0 wert nach wert konv
002E18  41FA 0D19       4455:           lea adrmsg(pc),a0
002E1C  323C 0014       4456:           move #20,d1
002E20  343C 00BE       4457:           move #190,d2    *text adr:
002E24  103C 0022       4458:           move.b #$22,d0
002E28  6100 E55C       4459:           bsr textaus
002E2C                  4460: getlpadr:
002E2C  41ED 00F4       4461:           lea einbuf(a5),a0
002E30  323C 0044       4462:           move #68,d1
002E34  343C 00BE       4463:           move #190,d2
002E38  103C 0022       4464:           move.b #$22,d0
002E3C  363C 001E       4465:           move #30,d3     *anzahl zeichen
002E40  6100 E644       4466:           bsr textein     *eingabe interaktiv
002E44  41ED 00F4       4467:           lea einbuf(a5),a0
002E48  6100 E8B2       4468:           bsr zuweis      * auch moeglich
002E4C  64DE            4469:           bcc getlpadr
002E4E  6100 E8DC       4470:           bsr wert        *eingabe zeichenwert
002E52  4E75            4471:           rts             *d0=wert d1=flag
                        4472:
                        4473:
                        4474:
002E54                  4475: hexdump:                  *ausgabe in sedezimaler Form
                        4476:                           *a1 = startadresse a2 = merkeradresse
002E54  323C 000A       4477:           move #10,d1     *x-start adresse
002E58  343C 00A0       4478:           move #10*16,d2  *v-start adresse
002E5C  2009            4479:           move.l a1,d0
002E5E  C0BC FFFFFFF0   4480:           and.l #$fffffff0,d0     *auf 16er Grenze bringen
002E64  2240            4481:           movea.l d0,a1
002E66  363C 0007       4482:           move #8-1,d3    *8 zeilen ausgeben
002E6A  41ED 0070       4483: hexd1:    lea ausbuf(a5),a0       *ziel fuer Ausgabe
002E6E  2009            4484:           move.l a1,d0
002E70  6100 E7D8       4485:           bsr print8x     *ausgabe adresse
002E74  383C 000F       4486:           move #16-1,d4
002E78  10FC 0020       4487: hexd2:    move.b #' ',(a0)+
002E7C  B5C9            4488:           cmp.l a1,a2     *wenn gleich dann marke
002E7E  6606            4489:           bne.s hexd3
002E80  10FC 003E       4490:           move.b #'>',(a0)+
002E84  6004            4491:           bra.s hexd4
002E86  10FC 0020       4492: hexd3:    move.b #' ',(a0)+
002E8A  1019            4493: hexd4:    move.b (a1)+,d0
002E8C  6100 E7DA       4494:           bsr print2x
002E90  51CC FFE6       4495:           dbra d4,hexd2
002E94  41ED 0070       4496:           lea ausbuf(a5),a0
002E98  103C 0011       4497:           move.b #$11,d0  *size
002E9C  6100 E4E8       449B:           bsr textaus
002EA0  93FC 00000010   4499:           suba.l #16,a1   *zurueck legen
002EA6  947C 000A       4500:           sub #10,d2      *v-richtung
002EAA  41ED 0070       4501:           lea ausbuf(a5),a0
002EAE  2009            4502:           move.l a1,d0
002EB0  6100 E798       4503:           bsr print8x
002EB4  383C 000F       4504:           move #16-1,d4   *textzeile
002EB8  10FC 0020       4505: hexd5:    move.b #' ',(a0)+
002EBC  10FC 0020       4506:           move.b #' ',(a0)+
002EC0  1019            4507:           move.b (a1)+,d0
002EC2  C03C 007F       4508:           and.b #$7f,d0
```

```
002EC6  B03C 0020      4509:           cmp.b #32,d0
002ECA  6A04           4510:           bpl.s hexd6
002ECC  103C 0020      4511:           move.b #' ',d0
002ED0  10C0           4512: hexd6:    move.b d0,(a0)+
002ED2  10FC 0020      4513:           move.b #' ',(a0)+
002ED6  51CC FFE0      4514:           dbra d4,hexd5
002EDA  4210           4515:           clr.b (a0)          *terminator zeichen
002EDC  41ED 0070      4516:           lea ausbuf(a5),a0
002EE0  103C 0011      4517:           move.b #$11,d0
002EE4  6100 E4A0      4518:           bsr textaus
002EE8  947C 000A      4519:           sub #10,d2          *zeilenabstand
002EEC  51CB FF7C      4520:           dbra d3,hexd1
002EF0  4E75           4521:           rts
                       4522:
002EF2                 4523: headaus:
002EF2  103C 0033      4524:           move.b #$33,d0
002EF6  323C 0050      4525:           move #80,d1
002EFA  343C 00DC      4526:           move #220,d2
002EFE  6100 E486      4527:           bsr textaus         *ueberschrift
002F02  4E75           4528:           rts
                       4529:
002F04                 4530: sphead:
002F04  41FA 0BD7      4531:           lea spmsg1(pc),a0
002F08  61E8           4532:           bsr headaus
002F0A  41FA 0C02      4533:           lea spmsg3(pc),a0
002F0E  103C 0021      4534:           move.b #$21,d0  *adressblock
002F12  323C 000A      4535:           move #10,d1
002F16  343C 00AE      4536:           move #174,d2
002F1A  6100 E46A      4537:           bsr textaus
002F1E  4E75           4538:           rts
                       4539:
                       4540:
002F20                 4541: speicheraus:              *Ausgabe des Speicherinhalts
002F20  61E2           4542:           bsr sphead          *header ausgeben
002F22  6100 FEF4      4543:           bsr getadr          *start adresse
002F26  4A41           4544:           tst d1
002F28  6700 006A      4545:           beq spfin           *ende bei fehler
002F2C  2240           4546:           movea.l d0,a1
002F2E  2440           4547:           movea.l d0,a2   *ziel ausgabeadresse
002F30  48E7 0060      4548: sploop:   movem.l a1/a2,-(a7)
002F34  6100 FF1E      4549:           bsr hexdump
002F38  4CDF 0600      4550:           movem.l (a7)+,a1/a2
002F3C  41FA 0BB0      4551:           lea spmsg2(pc),a0
002F40  103C 0022      4552:           move.b #$22,d0
002F44  323C 0014      4553:           move #20,d1
002F48  343C 00BE      4554:           move #190,d2
002F4C  6100 E43B      4555:           bsr textaus
002F50  6100 DB10      4556:           bsr ki
002F54  C03C 007F      4557:           and.b #$7f,d0
002F58  B03C 004D      4558:           cmp.b #'M',d0
002F5C  6700 0036      4559:           beq spfin
002F60  B03C 002B      4560:           cmp.b #'+',d0
002F64  660A           4561:           bne.s spei1
002F66  D3FC 00000040  4562:           adda.l #$40,a1
002F6C  6000 0024      4563:           bra spei3
002F70  B03C 002D      4564: spei1:    cmp.b #'-',d0
002F74  660A           4565:           bne.s spei2
002F76  93FC 00000040  4566:           suba.l #$40,a1
002F7C  6000 0014      4567:           bra spei3
002F80  B03C 0052      4568: spei2:    cmp.b #'R',d0
002F84  660C           4569:           bne.s spei3
002F86  6100 FE90      4570:           bsr getadr          *d0=neuer wert
002F8A  4A41           4571:           tst d1
002F8C  6706           4572:           beq.s spfin
002F8E  2240           4573:           movea.l d0,a1
002F90  2440           4574:           movea.l d0,a2
002F92  609C           4575: spei3:    bra sploop
                       4576:
002F94  4E75           4577: spfin:    rts
                       4578:
                       4579:
                       4580: *** Ende Speicherinhalt
                       4581:
                       4582:
                       4583: *** Symboltabelle ausgeben
                       4584:
002F96                 4585: svmbolaus:
002F96  3B7C 0002      4586:           move #2,passflag(a5)
        0032
002F9C  6100 D062      4587:           bsr clrscreen
```

```
002FA0  6100 ECE4       4588:           bsr prtsymb     *ausgabe der symboltabelle
002FA4  6000 F87C       4589:           bra finmenue    * weiter dort
                        4590:
                        4591:
                        4592: *** Ende Symboltabelle
                        4593:
                        4594:
                        4595:
                        4596: ****************************************************
                        4597: * init aller variable, hardware etc
                        4598:
                        4599:
002FA8                  4600: svmdefault:                *symboltabelle default
002FA8  0000            4601:           dc.w 0          *nil
002FAA  0000            4602:           dc.w 0          *nil
002FAC  00000000        4603:           dc.l 0          *leere daten
002FB0  0000            4604:           dc.w 0          *attribute
002FB2  00000000        4605:           dc.l 0          *name
002FB6  00000000        4606:           dc.l 0          *name
                        4607:
002FBA  4E73            4608: nmiex:    rte             * dummy
                        4609:
                        4610: ***********
002FBC                  4611: vekdef:                   * Relativ  bezogen auf 0
                        4612:                           * nur bei adr 0 gueltig
002FBC  4EF9 00003336   4613:           jmp (intlvx-basis).l    *2
002FC2  4EF9 00003336   4614:           jmp (intlvx-basis).l    *5
002FC8  4EF9 00002FBA   4615:           jmp (nmiex-basis).l     *7 ignorieren
002FCE  4EF9 00003230   4616:           jmp (trap0a-basis).l
                        4617: *
002FD4  4EF9 00000A00   4618:           jmp (ci-basis).l
002FDA  4EF9 000009D8   4619:           jmp (csts-basis).l
002FE0  4EF9 00000000   4620:           jmp (co-basis).l
                        4621: ****************
                        4622:
002FE6  4BFA D018       4623: init:     lea basis(pc).a5
002FEA  DBFA D418       4624:           adda.l ramstart(pc),a5  * Registerbasis rev 4.0
002FEE  48ED FFFF       4625:           movem.l d0-d7/a0-a7,regsave(a5) * Register retten fuer ein
        0178
                        4626:                                   *
002FF4  1038 7000       4627:           move.b $7000.d0         * Boot-Karte loeschen rev 4.2
002FF8  11FC 0080       4628:           move.b #$80.bankboot    * Bank-Boot-Karte loeschen rev 4.2
        FFC8
002FFE  11FC 0040       4629:           move.b #%01000000.proma2  * neutral Stellen des PROMMERS r
        FF82
                        4630:                                   *
003004  422D 002C       4631:           clr.b iodir(a5)         * ci2.co2 frei geben und lastchar
003008  422D 0214       4632:           clr.b oldsize(a5)       * Fuer Figur-Befehl
00300C  422D 0219       4633:           clr.b synstate(a5)      * variable initialisieren
003010  422D 002E       4634:           clr.b flip(a5)
003014  422D 0030       4635:           clr.b flip1(a5)         *autoflip variable
003018  422D 0A44       4636:           clr.b color(a5)         * WEISS
00301C  422D 0A45       4637:           clr.b xormode(a5)       * STD-Mode, kein XOR
003020  426D 004C       4638:           clr turtx(a5)           *old pos
003024  426D 004E       4639:           clr turty(a5)
003028  426D 0050       4640:           clr turtphi(a5)
00302C  426D 0046       4641:           clr turx(a5)
003030  426D 0048       4642:           clr turv(a5)
003034  426D 004A       4643:           clr turphi(a5)
003038  1B7C 0001       4644:           move.b #1.turdo(a5)     * turtle down
        021E
00303E  1B7C 0001       4645:           move.b #1.ci2flag(a5)   * 1= erstes CI2 fuer USERCI
        0A46
003044  1B7C 0002       4646:           move.b #2.iostat(a5)    * Ausgabe auf CRT
        002A
00304A  1B7C 0002       4647:           move.b #2.iostatb(a5)   * Eingabe STD (opt)
        002B
003050  3B7C 0002       4648:           move #2.passflag(a5)    * sonst co2 gesperrt
        0032
003056  422D 022B       4649:           clr.b optflag(a5)
00305A  422D 0222       4650:           clr.b curon(a5)         *cursor off
00305E  422D 0225       4651:           clr.b escmerker(a5)
003062  1B7C 0001       4652:           move.b #1.first(a5)     *grad ausgewesen
        021D
                        4653:                                   *
003068  41FA CF96       4654:           lea basis(pc).a0        * Wenn BASIS<>0,
00306C  B1FC 00000000   4655:           cmpa.l #0.a0            * dann Transport ins RAM noetig
003072  6714            4656:           beq.s inin2             * Auch nach Poweron, wegen
                        4657:                                   * Zerstoerung, bei RESET etc.
003074  43F8 0000       4658:           lea 0.a1                * Ziel ist RAM bei ADR. 0
```

```
003078  2408            4659:           move.l a0,d2                * Adresse ROM
00307A  323C 00FF       4660:           move #$100-1,d1             * Langworte transportieren
00307E  2018            4661: inin3:    move.l (a0)+,d0             * Wort von ROM holen
003080  D082            4662:           add.l d2,d0                 * Rom-Adresse addieren
003082  22C0            4663:           move.l d0,(a1)+             * Wort ins RAM uebertragen
003084  51C9 FFF8       4664:           dbra d1,inin3               * 1024 Bytes = 256 Langworte
                        4665:                                       * geht nur, wenn RAM ueber ROM
                        4666:                                       * liegt, sonst spezielle TRAPs
                        4667:                                       * getrennt aendern.
                        4668:                                       *
003088                  4669: inin2:
003088  0CAD A140557A   4670:           cmp.l #$A140557A,poweron(a5)
        0A48
003090  6700 EA30       4671:           beq carres          * FERTIG nicht neu initialisieren
                        4672:                               * Ja, erstes mal dann schon
                        4673:                               * alle weiteren befehle nur ein Mal
003094  6100 0054       4674:           bsr svmloesche
                        4675:                               *
003098  41ED 1000       4676:           lea $1000(a5),a0            * Dort Editorstart (9000 bei 0)
00309C  2B48 0036       4677:           move.l a0,stxtxt(a5)        * TEST EDIT
0030A0  2B48 003A       4678:           move.l a0,akttxt(a5)
0030A4  2B48 003E       4679:           move.l a0,etxtxt(a5)
0030A8  2B48 0042       4680:           move.l a0,eottxt(a5)
0030AC  4210            4681:           clr.b (a0)                  * TEST EDIT
0030[illegible] 41ED 1C00  4682:        lea $1c00(a5),a0
0030[illegible] 2B48 01F2  4683:        move.l a0,pcorg(a5)         * Defaultorg
                        4684:
0030B6  422D 0220       4685:           clr.b debug(a5) * kein debug-mode
0030BA  303C 0021       4686:           move #$21,d0
0030BE  6100 EE58       4687:           bsr setoroe         * Groesse festlegen
0030C2  41FA FEF8       4688:           lea  vekdef(pc),a0 *REV REV REV
0030C6  43ED 0000       4689:           lea  vekdest(a5),a1
0030CA  45FA CF34       4690:           lea basis(pc),a2
0030CE  220A            4691:           move.l a2,d1        * Adresse des ROMS als Basis
0030D0  363C 0006       4692:           move #7-1,d3        * Befehls-Anzahl
0030D4                  4693: inin1:
0030D4  32D8            4694:           move (a0)+,(a1)+            * Sprung-Befehl
0030D6  2018            4695:           move.l (a0)+,d0             * Ziel-Adresse
0030D8  D081            4696:           add.l d1,d0                 * +Adresse ROM
0030DA  22C0            4697:           move.l d0,(a1)+             * im RAM ablegen
0030DC  51CB FFF6       4698:           dbra d3,inin1
                        4699:                               * erst hier Init POWERON,bei prellen etc w
0030E0  2B7C A140557A   4700:           move.l #$A140557A,poweron(a5)   * Merker ob schon definier
        0A48
0030E8  4E75            4701:           rts
                        4702:
0030EA                  4703: svmloesche:
0030EA  426D 0A4C       4704:           clr.w svmnext(a5)       *symboltabelle leer
0030EE  41FA FEB8       4705:           lea svmdefault(pc),a0
0030F2  43ED 0A4E       4706:           lea svmtab(a5),a1
0030F6  303C 0011       4707:           move #svmlen-1,d0
0030FA  12D8            4708: inil1:    move.b (a0)+,(a1)+
0030FC  51C8 FFFC       4709:           dbra d0,inil1   *ok nun leer
003100  4E75            4710:           rts
                        4711:
                        4712: ************************************************
                        4713:
                        4714: * upgre, rev 3.1 neue routinen
                        4715:
003102  202D 0036       4716: getstx:   move.l stxtxt(a5),d0
003106  4E75            4717:           rts
                        4718:
003108  2B40 0036       4719: putstx:   move.l d0,stxtxt(a5)    * alle Zeiger gueltig setzen
00310C  2B40 003A       4720:           move.l d0,akttxt(a5)    * auf Anfang legen
003110  2040            4721:           movea.l d0,a0           *
003112                  4722: putlstx:
003112  4A18            4723:           tst.b (a0)+     * =0 dann fertig
003114  66FC            4724:           bne putlstx     * bis ende gefunden
003116  5388            4725:           subq.l #1,a0    * auf ende zeigen
003118  2B48 003E       4726:           move.l a0,etxtxt(a5)
00311C  2B48 0042       4727:           move.l a0,eottxt(a5)
003120  4E75            4728:           rts
                        4729:
                        4730: * rev 4.0 neue Unterprogramme
                        4731:
003122                  4732: setxor:          * d0 enthaelt xor-code. 0=neutral (std)
003122  C03C 000F       4733:           and.b #$f,d0
003126  1B40 0A45       4734:           move.b d0,xormode(a5)
00312A  6000 DCE8       4735:           bra aktpage     * und gleich setzen
                        4736:
```

SYMBOL

GETSET

```
00312E
00312E  4280
003130  102D 0A45
003134  4E75

003136
003136  1B40 0A44
00313A  6000 DCD8

00313E
00313E  4280
003140  102D 0A44
003144  4E75

003146
003146  1B7C 0001
        0222
00314C  4E75

00314E
00314E  422D 0222
003152  4E75

003154
003154  41FA CEAA
003158  2008
00315A  4E75
00315C
00315C  41FA CEA2
003160  D1FA D2A2
003164  2008
003166  4E75

003168
003168  4BFA CE96
00316C  DBFA D296
003170  4E75

003172
003172  41ED 0A4E
003176  2008
003178  4E75

00317A
00317A  4280
00317C  302D 0A4C
003180  4E75

003182
003182  3B40 0A4C
003186  4E75

003188
003188  203A D282
00318C  4E75

00318E  203A D280
003192  4E75




003194  202D 01F2
003198  4E75

00319A  2B40 01F2
00319E  4E75

0031A0
0031A0  1B7C 0002
        002A
0031A6  4E75
0031A8
0031A8  1B7C 0003
        002A
0031AE  4E75

0031B0
0031B0  1B7C 0005
```

```
4737: getxor:           * d0 erhaelt den xor-code
4738:           clr.l d0
4739:           move.b xormode(a5),d0
4740:           rts
4741:
4742: setcolor:         * d0 enthaelt 8 Bit Farbcode. 0=WEISS
4743:           move.b d0,color(a5)
4744:           bra aktpage     * und gleich setzen
4745:
4746: getcolor:
4747:           clr.l d0
4748:           move.b color(a5),d0
4749:           rts
4750:
4751: curonein:
4752:           move.b #1,curon(a5)

4753:           rts
4754:
4755: curonaus:
4756:           clr.b curon(a5)
4757:           rts
4758:
4759: getbasis:
4760:           lea basis(pc),a0
4761:           move.l a0,d0
4762:           rts
4763: getvar:
4764:           lea basis(pc),a0
4765:           adda.l ramstart(pc),a0
4766:           move.l a0,d0
4767:           rts
4768:
4769: seta5:
4770:           lea basis(pc),a5
4771:           adda.l ramstart(pc),a5
4772:           rts
4773:
4774: getsymtab:              * Start der Records
4775:           lea symtab(a5),a0
4776:           move.l a0,d0
4777:           rts
4778:
4779: getnext:
4780:           clr.l d0
4781:           move.w symnext(a5),d0
4782:           rts
4783:
4784: putnext:                * Symboltabelle, next setzen
4785:           move.w d0,symnext(a5)
4786:           rts
4787:
4788: getvers:
4789:           move.l versnum(pc),d0
4790:           rts
4791:
4792: getsn:    move.l snnum(pc),d0
4793:           rts
4794:
4795:
4796: * Ende rev 4.0
4797:
4798:
4799: getorg: move.l pcorg(a5),d0
4800:           rts
4801:
4802: putorg: move.l d0,pcorg(a5)
4803:           rts
4804:
4805: crtex:
4806:           move.b #2,iostat(a5)

4807:           rts
4808: lstex:
4809:           move.b #3,iostat(a5)

4810:           rts
4811:
4812: usrex:                          * 5 ist user, 4 ist spez
4813:           move.b #5,iostat(a5)
```

```
          002A
0031B6  4E75                4814:          rts
                            4815:
0031B8                      4816: nilex:                    * Nur Fehlerausgabe
0031B8  1B7C 0001           4817:          move.b #1,iostat(a5)
          002A
0031BE  426D 0034           4818:          clr errflag(a5) * Sperren, keine Fehler da
0031C2  4E75                4819:          rts
                            4820:
0031C4                      4821: seterr:                   * Fehlercode setzen, fuer BIBO
0031C4  3B40 0034           4822:          move d0,errflag(a5) * Wert in D0
0031C8  4E75                4823:          rts              * Beeinflusst NIL: - Ausgabe
                            4824:
0031CA                      4825: geterr:                   * Fehlercode lesen nach D0
0031CA  302D 0034           4826:          move errflag(a5),d0
0031CE  4E75                4827:          rts
                            4828:
0031D0                      4829: setpass:                  * PASS setzen fuer externe BIBOs
0031D0  3B40 0032           4830:          move d0,passflag(a5)
0031D4  4E75                4831:          rts
                            4832:
                            4833: * Exception Handling
                            4834: * ------------------
                            4835:
00[redacted]                4836: traprts:                  * wie Trap, jedoch ohne RTE
                            4837:                           * sondern mit RTS als Abschluss
                            4838:                           * Aufruf move #xx,d7
                            4839:                           *        trap #1
                            4840:                           * A6 geht verloren
                            4841:                           * A5 ist lokaler Pointer
0031D6  4BFA CE28           4842:          lea basis(pc),a5
0031DA  DBFA D228           4843:          adda.l ramstart(pc),a5  * Nun Variablengebiet definiert
0031DE  5347                4844:          subq #1,d7              * Bereich 0..n
0031E0  BE7C 0078           4845:          cmp #maxtraps,d7
0031E4  6400 0018           4846:          bcc traprerr     * n+1 befehl nicht da
0031E8  CEFC 0004           4847:          mulu #4,d7       * mal 4 fuer langwort
0031EC  4DFA D242           4848:          lea traptab(pc),a6
0031F0  DDC7                4849:          adda.l d7,a6     * Pointer in Tabelle
0031F2  2C56                4850:          movea.l (a6),a6 * Relative Adresse holen
0031F4  2E0E                4851:          move.l a6,d7
0031F6  4DFA D238           4852:          lea traptab(pc),a6       * Basisadr
0031FA  DDC7                4853:          adda.l d7,a6     * Da Eintrag: adr-traptab
0031FC  4E96                4854:          jsr (a6)         * und aufrufen damit
0031FE                      4855: traprerr:                 * Fehlerausgang
0031FE  4E75                4856:          rts              * Status bleibt in SR
                            4857:
                            4858:
                            4859: * rev 4.0 neue Codierung
003200                      4860: trap1a:                   * Benutzerbefehle d7.w = Befehlscode
                            4861:                           * Aufruf move #xx,d7
                            4862:                           *        trap #1
                            4863:                           * A6 geht verloren
                            4864:                           * A5 ist lokaler Pointer
003200  4BFA CDFE           4865:          lea basis(pc),a5
003204  DBFA D1FE           4866:          adda.l ramstart(pc),a5  * Nun Variablengebiet definiert
003208  5347                4867:          subq #1,d7       * Bereich 0..n
00320A  BE7C 0078           4868:          cmp #maxtraps,d7
00320E  6400 001E           4869:          bcc traperr      * n+1 befehl nicht da
003212  CEFC 0004           4870:          mulu #4,d7       * mal 4 fuer langwort
003216  4DFA D218           4871:          lea traptab(pc),a6
00321A  DDC7                4872:          adda.l d7,a6     * Pointer in Tabelle
00[redacted]  2C56          4873:          movea.l (a6),a6 * Relative Adresse holen
00[redacted]  2E0E          4874:          move.l a6,d7
003220  4DFA D20E           4875:          lea traptab(pc),a6       * Basisadr
003224  DDC7                4876:          adda.l d7,a6     * Da Eintrag: adr-traptab
003226  4E96                4877:          jsr (a6)         * und aufrufen damit
003228  40EF 0000           4878:          move sr,0(a7)    * status auf stack tauschen
                            4879:                           * Status des Uprgs bleibt somit
00322C  4E73                4880:          rte              * ohne Rueckholen des Status
                            4881:
                            4882:
00322E                      4883: traperr:
00322E  4E73                4884:          rte              * Fehlerausgang
                            4885:
003230                      4886: trap0a:
003230  4E73                4887:          rte
                            4888:
003232  4E75                4889: tnotimp:         rts      * nicht mehr vorhanden
                            4890:
                            4891: ** Tabelle am Anfang des Grundprogramms
```

```
003234                                  4892:
                                        4893: * rev 4.0 neue Codierung
003234                                  4894: gettrap:                  * a0 -> Symbol
                                        4895:                          * d0.l = wert adresse anschliessend
                                        4896:                          * d1.w behalten da interncode
                                        4897:                          * d2.l = index anschliessend
                                        4898:                          * carry wenn nicht gefunden
003234  48A7 4000                      4899:          movem d1.-(a7)  * wird zerstoert in setupname
003238  6100 E91C                      4900:          bsr setupname   *
00323C  4C9F 0002                      4901:          movem (a7)+.d1  * movem erhaelt flags
003240  6500 E87A                      4902:          bcs carset      * Fehler Namensconvention falsch
003244  43FA D3CA                      4903:          lea trapsvm(pc),a1      * d2=msb d3=lsb
003248  4284                          4904:          clr.l d4        * INDEX = 0. spaeter 1..n
00324A                                  4905: getltr:
00324A  0C91 00000000                  4906:          cmp.l #0.(a1)   *=0 dann ende
003250  6700 E86A                      4907:          beq carset      * Fehler nicht gefunden
003254  B4A9 0000                      4908:          cmp.l 0(a1),d2  *
003258  6600 0026                      4909:          bne get2tr
00325C  B6A9 0004                      4910:          cmp.l 4(a1),d3  * wenn beide gleich dann ok
003260  6600 001E                      4911:          bne get2tr
003264  2404                          4912:          move.l d4.d2    * ERGEBNIS INDEX
003266  5282                          4913:          addq.l #1.d2    * Bereich 1..n
003268  C8FC 0004                      4914:          mulu #4.d4      * fuer Adressberechnung Langwort
00326C  43FA D1C2                      4915:          lea traptab(pc),a1
003270  D3C4                          4916:          adda.l d4,a1
003272  2011                          4917:          move.l (a1),d0  * Adresse der Routine.relativ
003274  43FA D1BA                      4918:          lea traptab(pc),a1
003278  D3C0                          4919:          adda.l d0,a1
00327A  2009                          4920:          move.l a1.d0    * Ergebnis Abs. Adresse
00327C  6000 E844                      4921:          bra carres      * Fertig ohne Fehler
                                        4922:
                                        4923:
003280  5089                          4924: get2tr:  addq.l #8.a1    * naechster Eintrag
003282  5284                          4925:          addq.l #1.d4    * INDEX
003284  60C4                          4926:          bra getltr
                                        4927:
                                        4928:
                                        4929: *** Graphik-Zusatz
                                        4930:
                                        4931:
003286                                  4932: setfig:
003286  422D 0214                      4933:          clr.b oldsize(a5)
00328A  4E75                          4934:          rts
                                        4935:
00328C                                  4936: figur:   * d0=size. wenn=0 dann nur loeschen.
                                        4937:          * d1=x. d2=y a0->Daten
                                        4938:          *
00328C  C07C 00FF                      4939:          and.w #$ff.d0      * rev 4.0 nur Bytegroesse verw.
003290  48E7 E080                      4940:          movem.l d0-d2/a0.-(a7)
003294  6100 DAD8                      4941:          bsr erapen
003298  102D 0214                      4942:          move.b oldsize(a5).d0
00329C  6710                          4943:          beq.s figlur
00329E  322D 020C                      4944:          move oldx(a5).d1
0032A2  342D 020E                      4945:          move oldy(a5).d2
0032A6  206D 0210                      4946:          movea.l oldadr(a5).a0
0032AA  6100 002E                      4947:          bsr figset
0032AE                                  4948: figlur:
0032AE  4CDF 0107                      4949:          movem.l (a7)+.d0-d2/a0
0032B2  48E7 E080                      4950:          movem.l d0-d2/a0.-(a7)
0032B6  6100 DAC4                      4951:          bsr setpen
0032BA  4A00                          4952:          tst.b d0
0032BC  6700 0006                      4953:          beq fig2ur              *=0 dann nicht setzen
0032C0  6100 0018                      4954:          bsr figset
0032C4                                  4955: fig2ur:                  * aber sondern nur loeschen
0032C4  1B40 0214                      4956:          move.b d0.oldsize(a5)
0032C8  3B41 020C                      4957:          move d1.oldx(a5)
0032CC  3B42 020E                      4958:          move d2.oldy(a5)
0032D0  2B48 0210                      4959:          move.l a0.oldadr(a5)
0032D4  4CDF 0107                      4960:          movem.l (a7)+.d0-d2/a0
0032D8  4E75                          4961:          rts
                                        4962:
0032DA                                  4963: figset:                  * d0=size.d1=x.d2=y a0->Quelle
0032DA  48E7 FEC0                      4964:          movem.l d0-d6/a0/a1.-(a7)
0032DE  6100 DC02                      4965:          bsr moveto
0032E2  B03C 0004                      4966:          cmp.b #4.d0
0032E6  640E                          4967:          bcc.s setf1     * Wenn kleiner dann spezcode
0032E8  43FA 0047                      4968:          lea figtabl(pc).a1
0032EC  1631 0000                      4969:          move.b (a1.d0.w).d3     * code
0032F0  383C 0000                      4970:          move #1-1.d4    * Counter
0032F4  6008                          4971:          bra.s setf2
```

```
0032F6  363C 0028       4972: setf1:  move #%0101000.d3
0032FA  3800            4973:         move d0.d4
0032FC  5344            4974:         sub  #1.d4        * Counter
0032FE  43FA 0026       4975: setf2:  lea figtab(pc).a1       * Uebersetzungstabelle
003302  1018           4976: setf12: move.b (a0)+.d0
003304  4880            4977:         ext.w d0          * Rest 0-Setzen
003306  1031 0000       4978:         move.b (a1.d0.w).d0
00330A  6A0E            4979:         bpl.s setf14
00330C  8003            4980:         or.b d3.d0        * Code
00330E  3204            4981: setf13: move d4.d1        * Zaehler laden
003310  6100 DA54       4982: setf15: bsr cmd
003314  51C9 FFFA       4983:         dbra d1.setf15
003318  60E8            4984:         bra setf12
                        4985:
00331A                  4986: setf14:                   * Sondercodes
00331A  6700 0004       4987:         beq setffi        *0=Ende
00331E  60EE            4988:         bra.s setf13      * Weiter dort ohne or
                        4989:
003320                  4990: setffi:                   * Ende
003320  4CDF 037F       4991:         movem.l (a7)+.d0-d6/a0/a1
003324  4E75            4992:         rts
                        4993:
003326                  4994: figtab:
003326  80              4995:         dc.b %10000000    * Vektorrichtungen
[illegible]27  81       4996:         dc.b %10000001
[illegible]28  82       4997:         dc.b %10000010
[illegible]29  83       4998:         dc.b %10000011
00332A  86              4999:         dc.b %10000110
00332B  87              5000:         dc.b %10000111
00332C  84              5001:         dc.b %10000100
00332D  85              5002:         dc.b %10000101
00332E  03              5003:         dc.b $3           * Hebe
00332F  02              5004:         dc.b $2           * Senke
003330  00              5005:         dc.b $0           * Ende der Tabelle, Endekode
                        5006:
003331                  5007: figtab1:
003331  28              5008:         dc.b %0101000
003332  28              5009:         dc.b %0101000
003333  50              5010:         dc.b %1010000
003334  78              5011:         dc.b %1111000
                        5012:
003335  00              5013:         dc.b 0
003336                  5014:         ds 0              * Angleich
                        5015:
                        5016:
                        5017: *** Errors
                        5018: * A6 geht verloren beim Registerretten
                        5019:
003336                  5020: intlvx:
003336  4DFA 04F6       5021:         lea inttxt(pc).a6
00333A  6000 0050       5022:         bra error                 * Interrupt aufgetreten
                        5023:
00333E                  5024: buserr:
00333E  4DFA 04F8       5025:         lea bustxt(pc).a6
003342  6000 0042       5026:         bra spezerr
                        5027:
003346                  5028: adrerr:
003346  4DFA 04F9       5029:         lea adrtxt(pc).a6
00334A  6000 003A       5030:         bra spezerr      *spezial stack
                        5031:
00334E                  5032: illins:
[illegible]4E  4DFA 04FA 5033:        lea illtxt(pc).a6
[illegible]52  6000 0038 5034:        bra error
                        5035:
003356                  5036: zerdiv:
003356  4DFA 0500       5037:         lea zertxt(pc).a6
00335A  6000 0030       5038:         bra error
                        5039:
00335E                  5040: chkins:
00335E  4DFA 0504       5041:         lea chktxt(pc).a6
003362  6000 0028       5042:         bra error
                        5043:
003366                  5044: trapvins:
003366  4DFA 0502       5045:         lea traptxt(pc).a6
00336A  6000 0020       5046:         bra error
                        5047:
00336E                  5048: priv:
00336E  4DFA 0500       5049:         lea privtxt(pc).a6
003372  6000 0018       5050:         bra error
                        5051:
```

EXCEPT

```
                                5052: ** tracein in einzelschritt
                                5053:
003376                          5054: linea:
003376  4DFA 0508               5055:         lea linatxt(pc),a6
00337A  6000 0010               5056:         bra error
                                5057:
00337E                          5058: linef:
00337E  4DFA 0506               5059:         lea linftxt(pc),a6
003382  6000 0008               5060:         bra error
                                5061:
                                5062:
003386                          5063: spezerr:                          *einsprung bus.adrerror
003386  DFFC 00000008           5064:         adda.l #8,a7              *ignore access,cych,cycl.instr
                                5065:
00338C                          5066: error:                            * alle ausser bus.adr error
00338C  4BFA CC72               5067:         lea basis(pc).a5          * A5,A6 geht auch verloren
003390  DBFA D072               5068:         adda.l ramstart(pc),a5    * basisadresse
003394  48ED FFFF               5069:         movem.l d0-d7/a0-a7,regsave(a5)
        0178
00339A  2B4E 0064               5070:         move.l a6.txtmerk(a5)     * Textadresse merken
00339E  3B6F 0000               5071:         move 0(a7),srsave(a5)
        01C8
0033A4  2B6F 0002               5072:         move.l 2(a7).pcsave(a5)
        01CA
0033AA  4E68                    5073:         move.l usp,a0
0033AC  2B48 01CE               5074:         move.l a0,uspsave(a5)
0033B0  2B4F 01D2               5075:         move.l a7.sspsave(a5)             * ALLE REGISTER MERKEN
                                5076: *
0033B4  4FED 0A02               5077:         lea stack(a5).a7          * erst mal neu definieren
                                5078:
0033B8  422D 002E               5079:         clr.b flip(a5)
0033BC  422D 0030               5080:         clr.b flip1(a5)
0033C0  1B7C 0001               5081:         move.b #1.first(a5)
        021D
0033C6  6100 D9D8               5082:         bsr clrall        * loeschen
                                5083:
0033CA                          5084: errcom:                           *einsprung std
0033CA  6100 D9B0               5085:         bsr setpen                * schreibend
0033CE  206D 0064               5086:         movea.l txtmerk(a5).a0
0033D2  323C 0050               5087:         move #80,d1
0033D6  343C 00DC               5088:         move #220,d2
0033DA  103C 0033               5089:         move.b #$33,d0
0033DE  6100 DF9B               5090:         bsr print
0033E2  6100 EECC               5091:         bsr regdump
                                5092:
0033E6                          5093: errlp:
0033E6  6100 D67A               5094:         bsr ki            * warten
0033EA  B03C 004D               5095:         cmp.b #'M',d0     *bis M eingegeben
0033EE  66F6                    5096:         bne errlp
0033F0  6000 0002               5097:         bra start         *mit neuem pointer
                                5098:
                                5099:
                                5100:
                                5101: ************************************************
                                5102:
0033F4                          5103: start:                    * mit undef. Stack
0033F4  4BFA CC0A               5104:         lea basis(pc).a5
0033F8  DBFA D00A               5105:         adda.l ramstart(pc).a5
0033FC  4FED 0A02               5106:         lea stack(a5).a7          * jetzt Stackpointer definiert
003400  6100 FBE4               5107:         bsr init          * Variable
003404  6100 D8D2               5108:         bsr casinit       * Kassettenports
003408  6100 D90E               5109:         bsr loinit        * Centronix-Printer
00340C  6100 D65A               5110:         bsr ciinit2       * nur textpointer auf start legen
003410  6100 CBEE               5111:         bsr flinit        * nur Variable einstellen, kein Zugriff
                                5112: *
003414  6100 D9FE               5113:         bsr aktpage       * XOR,COLOR einstellen
003418  6100 D986               5114:         bsr clrall        * test sequenz
                                5115: *
00341C  41FA 046E               5116:         lea copyr(pc).a0
003420  323C 0000               5117:         move #0.d1
003424  343C 0078               5118:         move #120.d2
003428  303C 0033               5119:         move #$33.d0
00342C  6100 DF58               5120:         bsr textaus
003430  323C 0064               5121:         move #100.d1
003434  6100 DA02               5122: sta1:   bsr sync
003438  67FA                    5123:         beq sta1
00343A  51C9 FFF8               5124:         dbra d1.sta1
                                5125: *
00343E                          5126: rstart:                   * NEUSTART
00343E  304F                    5127:         movea a7.a0       * von DEFAULT an suchen nach abs. Ende
```

```
003440  6100 E8F2      5128:           bsr getram      * a0 -> rom nach ram   DAUERT einige sec
003444  2008            5129:           move.l a0,d0    * muss auf even gebracht werden
003446  5580            5130:           sub.l #2,d0
003448  C0BC FFFFFFFE   5131:           and.l #$fffffffe,d0     *
00344E  2E40            5132:           movea.l d0,a7   * NEUER Stackpointer am RAMENDE
                        5133: *
                        5134:
003450                  5135: mloop:
003450  6100 D94E      5136: men1:     bsr clrall      *
003454  41FA 0451      5137:           lea menue(pc),a0
003458  323C 0000      5138:           move #0,d1
00345C  343C 00DC      5139:           move #220,d2
003460  303C 0044      5140:           move #$44,d0
003464  6100 DF20      5141:           bsr textaus
                        5142: *
003468  41FA 0453      5143:           lea menue1(pc),a0
00346C  6100 DF84      5144:           bsr menueio     *spezial position fest d0=wert zeichen
                        5145:
003470  B03C 0031      5146:           cmp.b #'1',d0
003474  6600 0008      5147:           bne men11       * Aendern
003478  6100 F42A      5148:           bsr aendere
00347C  60D2            5149:           bra men1
                        5150:
003[illegible]  B03C 0032      5151: men11:   cmp.b #'2',d0
003[illegible]  6600 0008      5152:          bne men12       * STARTEN
003488  6100 F8F2      5153:          bsr startex
00348A  60C4            5154:          bra men1
                        5155:
                        5156:
00348C  B03C 0033      5157: men12:   cmp.b #'3',d0
003490  6600 0008      5158:          bne men13
003494  6100 FA8A      5159:          bsr speicheraus *Ausgabe des Speichers
003498  60B6            5160:          bra men1
                        5161:
00349A  B03C 0034      5162: men13:   cmp.b #'4',d0
00349E  6600 0008      5163:          bne men14
0034A2  6100 FAF2      5164:          bsr symbolaus
0034A6  60A8            5165:          bra men1
                        5166:
0034A8                  5167: men14:
0034A8  B03C 0057      5168:          cmp.b #'W',d0
0034AC  66A2            5169:          bne men1
                        5170:
0034AE  6100 D8F0      5171: men2:    bsr clrall
0034B2  41FA 043A      5172:          lea menue2(pc),a0
0034B6  6100 DF3A      5173:          bsr menueio
0034BA  B03C 0031      5174:          cmp.b #'1',d0
0034BE  6600 000C      5175:          bne men21
0034C2  6100 EA5A      5176:          bsr edittext    * EDITOR
0034C6  6100 01A6      5177:          bsr initdebug   * Info loeschen jedoch debug-flag erhalten
0034CA  60E2            5178:          bra men2        * Assembler Aufruf moeglich
                        5179:
0034CC                  5180: men21:
0034CC  B03C 0032      5181:          cmp.b #'2',d0
0034D0  6600 000E      5182:          bne men21a
0034D4  6100 0198      5183:          bsr initdebug   * Debugstart festlegen, debugflag nicht
0034D8  6100 F340      5184:          bsr asstext     * ASSEMBLER
0034DC  6000 FF72      5185:          bra men1        * Start moeglich
                        5186:
                        5187:
003[illegible]  B03C 0033      5188: men21a:  cmp.b #'3',d0   * Bibliothek
003[illegible]  6600 000A      5189:          bne men22
0034E8  6100 F2E0      5190:          bsr bibo
0034EC  6000 FF62      5191:          bra men1        * start
0034F0                  5192: men22:
0034F0  B03C 0034      5193:          cmp.b #'4',d0
0034F4  6600 01A6      5194:          bne men23       * OPTIONEN
0034F8  6100 D8A6      5195:          bsr clrall
0034FC  41FA 0426      5196:          lea menue21(pc),a0
003500  6100 DEF0      5197:          bsr menueio     * Untermenue
003504  B03C 0031      5198:          cmp.b #'1',d0   *ASSEMBLER-Opt
003508  6600 0054      5199:          bne men221
00350C  6100 D892      5200:          bsr clrall
003510  41FA 0455      5201:          lea menue211(pc),a0
003514  6100 DEDC      5202:          bsr menueio     * Untermenue-Assembler
003518  B03C 0031      5203:          cmp.b #'1',d0   * Fehler only
00351C  6600 000A      5204:          bne men22a
003520  1B7C 0001      5205:          move.b #1,iostat(a5)
        002A
003526  6086            5206:          bra men2
```

```
003528                          5207:
003528  B03C 0032               5208: men22a:
                                5209:           cmp.b #'2',d0   * CRT-Ausgabe
00352C  6600 000C               5210:           bne men22b
003530  1B7C 0002              5211:           move.b #2,iostat(a5)
        002A
003536  6000 FF76               5212:           bra men2
                                5213:
00353A                          5214: men22b:
00353A  B03C 0033               5215:           cmp.b #'3',d0   * LST-Ausgabe
00353E  6600 000C               5216:           bne men22c      * sonst zurueck in Hauptmenue
003542  1B7C 0003              5217:           move.b #3,iostat(a5)
        002A
003548  6000 FF64               5218:           bra men2
                                5219:
                                5220:
00354C                          5221: men22c:
00354C  B03C 0034               5222:           cmp.b #'4',d0
003550  6600 FF5C               5223:           bne men2
003554  1B7C 0004              5224:           move.b #4,iostat(a5)
        002A
00355A  6000 FF52               5225:           bra men2
                                5226:
00355E                          5227: men221:
00355E  B03C 0032               5228:           cmp.b #'2',d0   *Text
003562  6600 0098               5229:           bne men222
003566  6100 D838               5230:           bsr clrall
00356A  41FA 0640               5231:           lea textm1(pc),a0
00356E  6100 DE82               5232:           jsr menueio
003572  B03C 0031               5233:           cmp.b #'1',d0   * ALTER TEXT
003576  6632                    5234:           bne.s men221a
003578  41FA 0620               5235:           lea textdefa(pc),a0
00357C  6100 F974               5236:           bsr headaus
003580  6100 F896               5237:           bsr getadr
003584  4A41                    5238:           tst d1
003586  6700 001A               5239:           beq fmen2       * bei Fehler
00358A  2B40 0036               5240:           move.l d0,stxtxt(a5)
00358E  2B40 003A               5241:           move.l d0,akttxt(a5)    * auf Anfang legen
003592  2040                    5242:           movea.l d0,a0
003594                          5243: men2201p:
003594  4A18                    5244:           tst.b (a0)+     * =0 dann fertig
003596  66FC                    5245:           bne men2201p    * bis ende gefunden
003598  5388                    5246:           subq.l #1,a0    * auf ende zeigen
00359A  2B48 003E               5247:           move.l a0,etxtxt(a5)
00359E  2B48 0042               5248:           move.l a0,eottxt(a5)
0035A2                          5249: fmen2:
0035A2  6100 D7FC               5250:           bsr clrall
0035A6  6000 FF06               5251:           bra men2
                                5252:
0035AA                          5253: men221a:
0035AA  B03C 0032               5254:           cmp.b #'2',d0   * NEUER TEXT
0035AE  66F2                    5255:           bne fmen2       * Zurueck in Hauptmenue
0035B0  41FA 05F1               5256:           lea textdefn(pc),a0
0035B4  6100 F93C               5257:           bsr headaus
0035B8  6100 F85E               5258:           bsr getadr      * Adresse nach d0 holen
0035BC  4A41                    5259:           tst d1
0035BE  67E2                    5260:           beq fmen2       * Bei Fehler  d0 retten
0035C0  48E7 8000               5261:           movem.l d0,-(a7)
0035C4  41FA 0600               5262:           lea textm2(pc),a0
0035C8  103C 0022               5263:           move.b #$22,d0
0035CC  323C 000A               5264:           move #10,d1
0035D0  343C 0014               5265:           move #20,d2
0035D4  6100 DDB0               5266:           bsr textaus
0035D8  6100 D488               5267:           bsr ki          * =J
0035DC  B03C 004A               5268:           cmp.b #'J',d0
0035E0  4CDF 0001               5269:           movem.l (a7)+,d0        * Status bleibt
0035E4  66BC                    5270:           bne fmen2       * bei Fehler
0035E6  2B40 0036               5271:           move.l d0,stxtxt(a5)
0035EA  2B40 003A               5272:           move.l d0,akttxt(a5)
0035EE  2B40 003E               5273:           move.l d0,etxtxt(a5)
0035F2  2B40 0042               5274:           move.l d0,eottxt(a5)    * ++ hier dyn moeglich
0035F6  2040                    5275:           movea.l d0,a0
0035F8  4210                    5276:           clr.b (a0)
0035FA  60A6                    5277:           bra fmen2
                                5278:
0035FC                          5279: men222:
0035FC  B03C 0033               5280:           cmp.b #'3',d0   *Loeschen
003600  6600 001E               5281:           bne men223
003604  6100 D79A               5282:           bsr clrall
003608  41FA 03F0               5283:           lea menue23(pc),a0
```

```
003608C  6100 DDE4      5284:          bsr menueio
003610   B03C 0031      5285:          cmp.b #'1'.d0
003614   6600 FE98      5286:          bne men2
003618   6100 FAD0      5287:          bsr svmloesche
00361C   6000 FE90      5288:          bra men2
                        5289:
                        5290:
003620                  5291: men223:
003620   B03C 0034      5292:          cmp.b #'4'.d0   *Zeichen/Zeile & debug
003624   6600 0072      5293:          bne men224
003628   6100 D776      5294:          bsr clrall
00362C   41FA 0385      5295:          lea menue22(pc).a0
003630   6100 DDC0      5296:          bsr menueio
003634   B03C 0031      5297:          cmp.b #'1'.d0
003638   6600 000C      5298:          bne men223a
00363C   1B7C 0021      5299:          move.b #$21.groesse(a5)
         021B
003642   6000 FE6A      5300:          bra men2
                        5301:
003646                  5302: men223a:
003646   B03C 0032      5303:          cmp.b #'2'.d0
00364A   6600 000C      5304:          bne men223b
00364E   1B7C 0011      5305:          move.b #$11.groesse(a5)
         021B
003[redacted]  6000 FE58      5306:          bra men2
                        5307:
003658                  5308: men223b:                 * Debug an und INIT Startgebiet
003658   B03C 0033      5309:          cmp.b #'3'.d0
00365C   6600 002A      5310:          bne men223c
003660   1B7C 0001      5311:          move.b #1.debug(a5)
         0220
003666   6100 0006      5312:          bsr initdebug    * muss hier definiert werden
00366A   6000 FE42      5313:          bra men2         * sonst fehler wenn gleich Einzelschritt
                        5314:
00366E                  5315: initdebug:
00366E   202D 0042      5316:          move.l eottxt(a5).d0    * dahinter anlegen
003672   5680          5317:          add.l #3.d0       * Abstand savetv
003674   C0BC FFFFFFFE  5318:          and.l #$fffffffe.d0     * Wort-Angleich /-1
00367A   2B40 0204      5319:          move.l d0.debugst(a5)
00367E   2B40 0208      5320:          move.l d0.debugak(a5)   * Anfang definieren
003682   2040          5321:          movea.l d0.a0
003684   4290          5322:          clr.l (a0)              * loeschen - Endekennung
003686   4E75          5323:          rts
                        5324:
                        5325:
003688                  5326: men223c:
003688   B03C 0034      5327:          cmp.b #'4'.d0   * Debug aus
00368C   6600 FE20      5328:          bne men2
003690   422D 0220      5329:          clr.b debug(a5)
003694   6000 FE18      5330:          bra men2
                        5331:
                        5332:
003698   6000 FE14      5333: men224: bra men2         * In jedem Fall zurueck
                        5334:
                        5335:
00369C                  5336: men23:
00369C   B03C 0057      5337:          cmp.b #'W'.d0
0036A0   6600 FE0C      5338:          bne men2
                        5339:
0036A4   6100 D6FA      5340: men3:    bsr clrall
0036[redacted]   41FA 036C      5341:          lea menue3(pc).a0
003[redacted]   6100 DD44      5342:          bsr menueio
0036[redacted]   B03C 0031      5343:          cmp.b #'1'.d0   * speichern cas
0036B4   6628          5344:          bne.s men31
0036B6   6100 D6E8      5345:          bsr clrall
0036BA   41FA 039C      5346:          lea menue31(pc).a0
0036BE   6100 DD32      5347:          bsr menueio
0036C2   B03C 0031      5348:          cmp.b #'1'.d0   * text speichern
0036C6   6608          5349:          bne.s men31a
0036C8   6100 F3D8      5350:          bsr textcas
0036CC   6000 FDB2      5351:          bra men1
                        5352:
0036D0   B03C 0032      5353: men31a: cmp.b #'2'.d0   * daten speichern
0036D4   66CE          5354:          bne men3
0036D6   6100 F444      5355:          bsr datacas
0036DA   6000 FD74      5356:          bra men1
                        5357:
0036DE   B03C 0032      5358: men31:   cmp.b #'2'.d0   * laden cas
0036E2   662C          5359:          bne.s men32
0036E4   6100 D6BA      5360:          bsr clrall
```

```
0036E8  41FA 036E      5361:           lea menue31(pc),a0
0036EC  6100 DD04      5362:           bsr menueio
0036F0  B03C 0031      5363:           cmp.b #'1',d0    * text laden
0036F4  660A           5364:           bne.s men31b
0036F6  7C00           5365:           moveq #0,d6      * laden
0036F8  6100 F514      5366:           bsr castext
0036FC  6000 FD52      5367:           bra men1
                       5368:
003700  B03C 0032      5369: men31b:   cmp.b #'2',d0    * daten laden
003704  669E           5370:           bne men3
003706  7C00           5371:           moveq #0,d6
003708  6100 F5C8      5372:           bsr casdata
00370C  6000 FD42      5373:           bra men1
                       5374:
                       5375:
003710  B03C 0033      5376: men32:    cmp.b #'3',d0    * Pruefen Cas
003714  6600 0030      5377:           bne men33
003718  6100 D686      5378:           bsr clrall
00371C  41FA 033A      5379:           lea menue31(pc),a0
003720  6100 DCD0      5380:           bsr menueio
003724  B03C 0031      5381:           cmp.b #'1',d0    * Text vergleichen
003728  660A           5382:           bne.s men31c
00372A  7C01           5383:           moveq #1,d6
00372C  6100 F4E0      5384:           bsr castext
003730  6000 FD1E      5385:           bra men1
                       5386:
003734  B03C 0032      5387: men31c:   cmp.b #'2',d0    * Daten vergleichen
003738  6600 FF6A      5388:           bne men3
00373C  7C01           5389:           moveq #1,d6
00373E  6100 F592      5390:           bsr casdata
003742  6000 FD0C      5391:           bra men1
                       5392:
                       5393:
003746                 5394: men33:
003746  B03C 0034      5395:           cmp.b #'4',d0
00374A  6600 0040      5396:           bne men34        * Rev 4.1 FLOPPY-Boot
00374E  383C 0001      5397:           move #1,d4       * Laufwerk A, Seite 0
003752  6100 C8AC      5398:           bsr getflop      * Floppyformat nach D4 legen
003756  6400 0006      5399:           bcc men32a       * Fehler D1=Code
00375A  6000 FCF4      5400:           bra men1         * Rueckkehr bei Fehler, z.B.
                       5401:                            * Laufwerkstuer offen, nicht form. Diskett
00375E                 5402:
                       5403: men32a:                   * Kein Fehler, Boot ist moeglich
                       5404:                            * Das Programm wird 2k ueber den
                       5405:                            * Default-Textbereich gelegt.
                       5406:                            * Achtung, daher ggf. Texte verlagern
00375E  41ED 1800      5407:           lea $1800(a5),a0 * bevor Boot erfolgt.
003762  323C 0001      5408:           move #1,d1       * Lese-Operation
003766  343C 0001      5409:           move #1,d2       * Sektor 1
00376A  363C 0000      5410:           move #0,d3       * Spur 0
00376E  6100 C890      5411:           bsr floppy       * und Floppy laden
003772  6500 FCDC      5412:           bcs men1         * wenn kein Fehler, Programm dort starten
003776  41ED 1800      5413:           lea $1800(a5),a0 * Max 1k wird geladen (je nach Formatieru
00377A  3010           5414:           move.w (a0),d0   * Sicherheitspruefung
00377C  B07A 000C      5415:           cmp.w men32b(pc),d0
003780  6600 FCCE      5416:           bne men1         * Fehler, NOP muss an erster Stelle stehen
003784  4E90           5417:           jsr (a0)         * als Unterprogramm
003786  6000 FCC8      5418:           bra men1         * um Rueckkehr zu ermoeglichen
                       5419:
00378A  4E71           5420: men32b:   noo              * Fuer Sicherheitstest muss ein NOP-Befehl
                       5421:                            * am Anfang des BOOT-Sektors stehen
                       5422:
00378C                 5423: men34:
00378C  B03C 0057      5424:           cmp.b #'W',d0
003790  6600 FF12      5425:           bne men3
                       5426:
003794  6100 D60A      5427:
003798  41FA 02CD      5428: men4:     bsr clrall
00379C  6100 DC54      5429:           lea menue4(pc),a0
0037A0  B03C 0031      5430:           bsr menueio
0037A4  6600 000A      5431:           cmp.b #'1',d0    * EPROM prog
0037A8  6100 EDE8      5432:           bne men4a
0037AC  6000 FCA2      5433:           bsr promwrite
                       5434:           bra men1
                       5435:
0037B0                 5436: men4a:
0037B0  B03C 0032      5437:           cmp.b #'2',d0    * EPROM laden
0037B4  6600 000A      5438:           bne men4b
0037B8  6100 ED84      5439:           bsr promread
0037BC  6000 FC92      5440:           bra men1
```

```
0037C0
0037C0  B03C 0033        5441:
0037C4  6608             5442: men4b:
0037C6  6100 E618        5443:          cmp.b #'3',d0    * Speicherbereiche
0037CA  6000 FC84        5444:          bne.s men4c
                         5445:          bsr ausgaber     * Ausgabe Bereiche
0037CE                   5446:          bra men1         * Ganz zurueck
0037CE  B03C 0034        5447:
0037D2  6612             5448: men4c:
0037D4  6100 D292        5449:          cmp.b #'4',d0    * Text drucken  rev 4.2
0037D8  6100 D29C        5450:          bne.s men4d      * kein Seitenvorschub.
0037DC  4A00             5451:          bsr ci1init2     * Textzeiger auf Anfang
0037DE  67B4             5452: men4c1: bsr ci2          * Zeichen lesen
0037E0  6100 D546        5453:          tst.b d0         * =0 dann Ende. aber
0037E4  60F2             5454:          beq men4         * zurueck ins gleiche Menue
                         5455:          bsr io           * und ausgeben
0037E6                   5456:          bra.s men4c1
0037E6  B03C 0057        5457:
0037EA  66A8             5458: men4d:
                         5459:          cmp.b #'W',d0
                         5460:          bne men4
                         5461:
                         5462:
0037EC  6100 D5B2        5463: men5:    bsr clrall
0037F0  41FA 02BB        5464:          lea menue5(pc),a0
0037F4  6100 DBFC        5465:          bsr menueio
0037F8  B03C 0031        5466:          cmp.b #'1',d0    * IO lesen
0037FC  6600 0008        5467:          bne men51
003800  6100 EC0E        5468:          bsr ioread
003804  60E6             5469:          bra men5
                         5470:
003806  B03C 0032        5471: men51:   cmp.b #'2',d0    * IO setzen
00380A  6600 0008        5472:          bne men52
00380E  6100 EC84        5473:          bsr iowrite
003812  60D8             5474:          bra men5
                         5475:
003814  B03C 0033        5476: men52:   cmp.b #'3',d0    * Einzelschritt
003818  6600 000A        5477:          bne men53
00381C  6100 E706        5478:          bsr einzel
003820  6000 FC2E        5479:          bra men1         * OK
                         5480:
003824                   5481: men53:
003824  B03C 0057        5482:          cmp.b #'W',d0
003828  66C2             5483:          bne men5
                         5484:
                         5485:
00382A  6000 FC24        5486:          bra mloop
                         5487:
                         5488:
                         5489:
                         5490:
                         5491: ************************************************
                         5492:
                         5493: * textblock da bytealigned
                         5494:
00382E                   5495: inttxt:
00382E  49 4E 54 45      5496:          dc.b 'INTERRUPT'
        52 52 55 50 54
003837  00               5497:          dc.b 0
                         5498:
003838                   5499: bustxt:
003838  42 55 53 45      5500:          dc.b 'BUSERROR'
        52 52 4F 52
003840  00               5501:          dc.b 0
                         5502:
003841                   5503: adrtxt:
003841  41 44 52 45      5504:          dc.b 'ADRERROR'
        52 52 4F 52
003849  00               5505:          dc.b 0
                         5506:
00384A                   5507: illtxt:
00384A  49 4C 4C 45      5508:          dc.b 'ILLEGAL INSTR'
        47 41 4C 20 49
        4E 53 54 52
003857  00               5509:          dc.b 0
                         5510:
003858                   5511: zertxt:
003858  44 49 56 20      5512:          dc.b 'DIV BY ZERO'
        42 59 20 5A 45
        52 4F
003863  00               5513:          dc.b 0
```

```
003864
003864  43 48 45 43 4B                5514:
                                      5515: chktxt:
                                      5516:           dc.b 'CHECK'
003869  00                            5517:           dc.b 0
                                      5518:
00386A
00386A  54 52 41 50 56                5519: traptxt:
                                      5520:           dc.b 'TRAPV'
00386F  00                            5521:           dc.b 0
                                      5522:
003870
003870  50 52 49 56 49 4C 45 47 45    5523: privtxt:
                                      5524:           dc.b 'PRIVILEGE'
003879  00                            5525:           dc.b 0
                                      5526:
                                      5527:
00387A
00387A  54 52 41 43 45                5528: tracetxt:
                                      5529:           dc.b 'TRACE'
00387F  00                            5530:           dc.b 0
                                      5531:
003880
003880  4C 49 4E 45 41                5532: linatxt:
                                      5533:           dc.b 'LINEA'
003885  00                            5534:           dc.b 0
                                      5535:
003886
003886  4C 49 4E 45 46                5536: linftxt:
                                      5537:           dc.b 'LINEF'
00388B  00                            5538:           dc.b 0
                                      5539:
                                      5540:
00388C
00388C  28 43 29 20 31 39 38 34 20    5541: copyr:
        52 6F 6C 66 2D                5542:           dc.b '(C) 1984 Rolf-Dieter Klein'
        44 69 65 74 65
        72 20 4B 6C 65
        69 6E
0038A6  00                            5543:           dc.b 0
                                      5544:
0038A7
0038A7  52 44 4B 2D 47 72 75 6E 64    5545: menue:
        70 72 6F 67 72                5546:           dc.b 'RDK-Grundprogramm 68K'
        61 6D 6D 20 36
        38 4B
0038BC  00                            5547:           dc.b 0
                                      5548:
0038BD
0038BD  31 3D 61 65 6E 64 65 72 6E    5549: menue1:
                                      5550:           dc.b '1=aendern'
0038C6  0A                            5551:           dc.b $A
0038C7  32 3D 73 74 61 72 74 65 6E    5552:           dc.b '2=starten'
0038D0  0A                            5553:           dc.b $A
0038D1  33 3D 61 6E 73 65 68 65 6E    5554:           dc.b '3=ansehen'
0038DA  0A                            5555:           dc.b $A
0038DB  34 3D 53 79 6D 62 6F 6C 65    5556:           dc.b '4=Symbole'
0038E4  0A                            5557:           dc.b $A
0038E5  57 3D 77 65 69 74 65 72       5558:           dc.b 'W=weiter'
0038ED  00                            5559:           dc.b 0
                                      5560:
                                      5561:
0038EE
0038EE  31 3D 45 64 69 74 6F 72       5562: menue2:
                                      5563:           dc.b '1=Editor'
0038F6  0A                            5564:           dc.b $A
0038F7  32 3D 41 73 73 65 6D 62 6C    5565:           dc.b '2=Assembler'
        65 72
003902  0A                            5566:           dc.b $A
003903  33 3D 42 69 62 6C 69 6F 74    5567:           dc.b '3=Bibliothek'
        68 65 6B
00390F  0A                            5568:           dc.b $A
```

```
003910  34 3D 4F 70     5569:           dc.b '4=Optionen'
        74 69 6F 6E 65
        6E
00391A  0A              5570:           dc.b $A
00391B  57 3D 77 65     5571:           dc.b 'W=weiter'
        69 74 65 72
003923  00              5572:           dc.b 0
                        5573:
                        5574:
003924                  5575: menue21:
003924  31 3D 41 73     5576:           dc.b '1=Assembler-Opt'
        73 65 6D 62 6C
        65 72 2D 4F 70
        74
003933  0A              5577:           dc.b $a
003934  32 3D 54 65     5578:           dc.b '2=Textstart'
        78 74 73 74 61
        72 74
00393F  0A              5579:           dc.b $a
003940  33 3D 4C 6F     5580:           dc.b '3=Loeschen Symbole'
        65 73 63 68 65
        6E 20 53 79 6D
        62 6F 6C 65
003952  0A              5581:           dc.b $a
003953  34 3D 5A 65     5582:           dc.b '4=Zeichen/Zeile & Debug
        69 63 68 65 6E
        2F 5A 65 69 6C
        65 20 26 20 44
        65 62 75 67
00396A  00              5583:           dc.b 0
                        5584:
00396B                  5585: menue211:
00396B  31 3D 4E 75     5586:           dc.b '1=Nur Fehlerausgabe'
        72 20 46 65 68
        6C 65 72 61 75
        73 67 61 62 65
00397E  0A              5587:           dc.b $a
00397F  32 3D 41 75     5588:           dc.b '2=Ausgabe auf CRT'
        73 67 61 62 65
        20 61 75 66 20
        43 52 54
003990  0A              5589:           dc.b $a
003991  33 3D 41 75     5590:           dc.b '3=Ausgabe auf LST'
        73 67 61 62 65
        20 61 75 66 20
        4C 53 54
0039A2  0A              5591:           dc.b $a
0039A3  34 3D 77 69     5592:           dc.b '4=wie 3 ohne LF'
        65 20 33 20 6F
        68 6E 65 20 4C
        46
0039B2  00              5593:           dc.b 0
                        5594:
0039B3                  5595: menue22:
0039B3  31 3D 34 30     5596:           dc.b '1=40 Zeichen/Zeile'
        20 5A 65 69 63
        68 65 6E 2F 5A
        65 69 6C 65
0039C5  0A              5597:           dc.b $a
0039C6  32 3D 38 30     5598:           dc.b '2=80 Zeichen/Zeile'
        20 5A 65 69 63
        68 65 6E 2F 5A
        65 69 6C 65
0039D8  0A              5599:           dc.b $a
0039D9  33 3D 44 65     5600:           dc.b '3=Debug-Info AN'
        62 75 67 2D 49
        6E 66 6F 20 41
        4E
0039E8  0A              5601:           dc.b $a
0039E9  34 3D 44 65     5602:           dc.b '4=Debug-Info AUS'
        62 75 67 2D 49
        6E 66 6F 20 41
        55 53
0039F9  00              5603:           dc.b 0
                        5604:
0039FA                  5605: menue23:
0039FA  31 3D 4C 6F     5606:           dc.b '1=Loeschen '
        65 73 63 68 65
        6E 20
003A05  0A              5607:           dc.b $a
```

```
003A06  20 20 61 6C 6C 65 72 20 53 79 6D 62 6F 6C 65   5608:          dc.b '  aller Symbole'
003A15  00                                             5609:          dc.b 0
                                                       5610:
003A16                                                 5611: menue3:
003A16  31 3D 53 70 65 69 63 68 65 72 6E 20 43 41 53   5612:          dc.b '1=Speichern CAS'
003A25  0A                                             5613:          dc.b $A
003A26  32 3D 4C 61 64 65 6E 20 43 41 53               5614:          dc.b '2=Laden CAS'
003A31  0A                                             5615:          dc.b $A
003A32  33 3D 50 72 75 65 66 65 6E 20 43 41 53         5616:          dc.b '3=Pruefen CAS'
003A3F  0A                                             5617:          dc.b $A
003A40  34 3D 46 6C 6F 70 70 79 20 53 74 61 72 74      5618:          dc.b '4=Floppy Start'
003A4E  0A                                             5619:          dc.b $A
003A4F  57 3D 77 65 69 74 65 72                        5620:          dc.b 'W=weiter'
003A57  00                                             5621:          dc.b 0
                                                       5622:
003A58                                                 5623: menue31:
003A58  31 3D 54 65 78 74                              5624:          dc.b '1=Text'
003A5E  0A                                             5625:          dc.b $a
003A5F  32 3D 44 61 74 65 6E                           5626:          dc.b '2=Daten'
003A66  00                                             5627:          dc.b 0
                                                       5628:
                                                       5629:
003A67  31 3D 45 50 52 4F 4D 20 70 72 6F 67            5630: menue4:  dc.b '1=EPROM prog'
003A73  0A                                             5631:          dc.b $A
003A74  32 3D 45 50 52 4F 4D 20 6C 65 73 65 6E         5632:          dc.b '2=EPROM lesen'
003A81  0A                                             5633:          dc.b $A
003A82  33 3D 53 70 65 69 63 68 65 72 62 65 72 65 69   5634:          dc.b '3=Speicherbereiche'
003A94  0A                                             5635:          dc.b $A
003A95  34 3D 54 65 78 74 20 64 72 75 63 6B 65 6E      5636:          dc.b '4=Text drucken'
003AA3  0A                                             5637:          dc.b $A
003AA4  57 3D 77 65 69 74 65 72                        5638:          dc.b 'W=weiter'
003AAC  00                                             5639:          dc.b 0
                                                       5640:
                                                       5641:
003AAD  31 3D 49 4F 20 6C 65 73 65 6E                  5642: menue5:  dc.b '1=IO lesen'
003AB7  0A                                             5643:          dc.b $A
003AB8  32 3D 49 4F 20 73 65 74 7A 65 6E               5644:          dc.b '2=IO setzen'
003AC3  0A                                             5645:          dc.b $A
003AC4  33 3D 45 69 6E 7A 65 6C 73 63 68 72 69 74      5646:          dc.b '3=Einzelschritt'
003AD3  0A                                             5647:          dc.b $A
003AD4  57 3D 77 65 69 74 65 72                        5648:          dc.b 'W=weiter'
003ADC  00                                             5649:          dc.b 0
                                                       5650:
003ADD  53 70 65 69 63 68 65 72 20 61 6E 73 65 68 65   5651: spmsg1:  dc.b 'Speicher ansehen'
```

```
003AED  00                                5652:           dc.b 0
                                          5653:
003AEE  2B 3D 77 65                       5654: spmsg2: dc.b '+=weiter -=rueckw R=Adr M=Menue'
        69 74 65 72 20
        2D 3D 72 75 65
        63 6B 77 20 52
        3D 41 64 72 20
        4D 3D 4D 65 6E
        75 65
003B0D  00                                5655:           dc.b 0
                                          5656:
003B0E  2D 2D 2D 2D                       5657: spmsg3: dc.b '---- 0 1 2 3 4 5 6 7 8 9 A B C D E F'
        20 30 20 31 20
        32 20 33 20 34
        20 35 20 36 20
        37 20 38 20 39
        20 41 20 42 20
        43 20 44 20 45
        20 46
003B32  00                                5658:           dc.b 0
                                          5659:
                                          5660:
003B33  41 64 72 3A                       5661: adrmsg: dc.b 'Adr:
        20 20 20 20
003B3B  00                                5662:           dc.b 0
                                          5663:
003B3C  50 72 6F 67                       5664: stmsg1: dc.b 'Programm starten'
        72 61 6D 6D 20
        73 74 61 72 74
        65 6E
003B4C  00                                5665:           dc.b 0
                                          5666:
003B4D  46 3D 46 6C                       5667: fimsg:  dc.b 'F=Flip M=Menue'
        69 70 20 4D 3D
        4D 65 6E 75 65
003B5B  00                                5668:           dc.b 0
                                          5669:
                                          5670:
003B5C  52 41 4D 2D                       5671: sober1: dc.b 'RAM-Bereiche'
        42 65 72 65 69
        63 68 65
003B6B  00                                5672:           dc.b 0
                                          5673:
003B69  45 72 73 74                       5674: sober2: dc.b 'Erster Bereich: '
        65 72 20 42 65
        72 65 69 63 68
        3A 20
003B79  00                                5675:           dc.b 0
                                          5676:
003B7A  30 20 31 20                       5677: sober3: dc.b '0 1 2 3 4 5 6 7 8 9 A B C D E F'
        32 20 33 20 34
        20 35 20 36 20
        37 20 38 20 39
        20 41 20 42 20
        43 20 44 20 45
        20 46
003B99  00                                5678:           dc.b 0
                                          5679:
003B9A                                    5680: textdefa:
003B9A  54 65 78 74                       5681:           dc.b 'Text alt'
        20 61 6C 74
003BA2  00                                5682:           dc.b 0
                                          5683:
003BA3                                    5684: textdefn:
003BA3  54 65 78 74                       5685:           dc.b 'Text neu'
        20 6E 65 75
003BAB  00                                5686:           dc.b 0
                                          5687:
003BAC                                    5688: textm1:
003BAC  31 3D 61 6C                       5689:           dc.b '1=alter Text'
        74 65 72 20 54
        65 78 74
003BB8  0A                                5690:           dc.b $a
003BB9  32 3D 6E 65                       5691:           dc.b '2=neuer Text'
        75 65 72 20 54
        65 78 74
003BC5  00                                5692:           dc.b 0
                                          5693:
003BC6                                    5694: textm2:
003BC6  77 69 72 6B                       5695:           dc.b 'wirklich ?. Ja=J'
```

```
                6C 69 63 68 20
                3F 2C 20 4A 61
                3D 4A
003BD6  00                          5696:           dc.b 0
                                    5697:
                                    5698:
003BD7                              5699: textcas1:
003BD7  53 70 65 69                 5700:           dc.b 'Speichern Text'
        63 68 65 72 6E
        20 54 65 78 74
003BE5  00                          5701:           dc.b 0
                                    5702:
003BE6                              5703: textcas2:
003BE6  53 70 65 69                 5704:           dc.b 'Speichern Daten'
        63 68 65 72 6E
        20 44 61 74 65
        6E
003BF5  00                          5705:           dc.b 0
                                    5706:
003BF6                              5707: textcas3:
003BF6  4C 61 64 65                 5708:           dc.b 'Laden Text'
        6E 20 54 65 78
        74
003C00  00                          5709:           dc.b 0
                                    5710:
003C01                              5711: textcas4:
003C01  4C 61 64 65                 5712:           dc.b 'Laden Daten'
        6E 20 44 61 74
        65 6E
003C0C  00                          5713:           dc.b 0
                                    5714:
003C0D                              5715: textcas5:
003C0D  50 72 75 65                 5716:           dc.b 'Pruefen Text'
        66 65 6E 20 54
        65 78 74
003C19  00                          5717:           dc.b 0
                                    5718:
003C1A                              5719: textcas6:
003C1A  50 72 75 65                 5720:           dc.b 'Pruefen Daten'
        66 65 6E 20 44
        61 74 65 6E
003C27  00                          5721:           dc.b 0
                                    5722:
003C28                              5723: textcas7:
003C28  53 74 61 72                 5724:           dc.b 'Startadr. :'
        74 61 64 72 2E
        20 3A
003C33  0A                          5725:           dc.b $a
003C34  45 6E 64 61                 5726:           dc.b 'Endadr.    :'
        64 72 2E 20 20
        20 3A
003C3F  0A                          5727:           dc.b $a
003C40  4E 61 6D 65                 5728:           dc.b 'Name       :'
        20 20 20 20 20
        20 3A
003C4B  00                          5729:           dc.b 0
                                    5730:
003C4C                              5731: textcas8:
003C4C  53 74 61 72                 5732:           dc.b 'Startadr. :'
        74 61 64 72 2E
        20 3A
003C57  0A                          5733:           dc.b $a
003C58  4E 61 6D 65                 5734:           dc.b 'Name       :'
        20 20 20 20 20
        20 3A
003C63  00                          5735:           dc.b 0
                                    5736:
003C64                              5737: textcas9:
003C64  56 65 72 67                 5738:           dc.b 'Vergleichsfehler'
        6C 65 69 63 68
        73 66 65 68 6C
        65 72
003C74  00                          5739:           dc.b 0
                                    5740:
003C75                              5741: textcas10:
003C75  50 72 75 65                 5742:           dc.b 'Pruefsumme falsch'
        66 73 75 6D 6D
        65 20 66 61 6C
        73 63 68
003C86  00                          5743:           dc.b 0
```

```
                                5744:
003C87                          5745: textcas11:
003C87  4F 4B                   5746:           dc.b 'OK'
003C89  00                      5747:           dc.b 0
                                5748:
003C8A                          5749: textcas12:
003C8A  46 61 6C 73             5750:           dc.b 'Falsches Daten-Format'
        63 68 65 73 20
        44 61 74 65 6E
        2D 46 6F 72 6D
        61 74
003C9F  00                      5751:           dc.b 0
                                5752:
003CA0  53 70 65 69             5753: txta1:    dc.b 'Speicher aendern'
        63 68 65 72 20
        61 65 6E 64 65
        72 6E
003CB0  00                      5754:           dc.b 0
                                5755:
003CB1                          5756: txta2:
003CB1  63 72 20 3D             5757:           dc.b 'cr = weiter.  - = zurueck'
        20 77 65 69 74
        65 72 2C 20 20
        2D 20 3D 20 7A
        75 72 75 65 63
        6B
003CCA  0A                      5758:           dc.b $a
003CCB  20 52 20 3D             5759:           dc.b ' R = Adresse. M = Menue'
        20 41 64 72 65
        73 73 65 2C 20
        4D 20 3D 20 4D
        65 6E 75 65
003CE2  00                      5760:           dc.b $0
                                5761:
003CE3                          5762: txtbi1:
003CE3  42 69 62 6C             5763:           dc.b 'Bibliothek'.0
        69 6F 74 68 65
        6B 00
                                5764:
003CEE                          5765: txtbi2:
003CEE  4E 61 6D 65             5766:           dc.b 'Name    : '.0
        20 20 20 3A 20
        00
                                5767:
003CF8                          5768: txtbi3:
003CF8  53 74 61 72             5769:           dc.b 'Start   : '.0
        74 20 20 3A 20
        00
                                5770:
003D02                          5771: txtbi4:
003D02  4C 61 65 6E             5772:           dc.b 'Laenge : '.0
        67 65 20 3A 20
        00
                                5773:
003D0C                          5774: txtbi5:
003D0C  53 74 61 72             5775:           dc.b 'Starten J = JA'
        74 65 6E 20 4A
        20 3D 20 4A 41
003D1A  0A                      5776:           dc.b $a
003D1B  63 72 20 3D             5777:           dc.b 'cr = weiter. M = Menue'.0
        20 77 65 69 74
        65 72 2C 20 4D
        20 3D 20 4D 65
        6E 75 65 00
                                5778:
                                5779:
003D32                          5780: txto1:
003D32  45 50 52 4F             5781:           dc.b 'EPROM prog.'.0
        4D 20 70 72 6F
        67 2E 00
003D3E                          5782: txto2:
003D3E  45 50 52 4F             5783:           dc.b 'EPROM lesen'.0
        4D 20 6C 65 73
        65 6E 00
003D4A                          5784: txto3:
003D4A  76 6F 6E 20             5785:           dc.b 'von '.$a
        0A
003D4F  62 69 73 20             5786:           dc.b 'bis '.$a
        0A
003D54  6E 61 63 68             5787:           dc.b 'nach'.0
```

```
                00
003D57                                  5788: txto4:
003D59  42 65 72 65                     5789:         dc.b 'Bereit = B'.0
        69 74 20 3D 20
        42 00
003D64                                  5790: txto5:
003D64  45 50 52 4F                     5791:         dc.b 'EPROM OK. M=Menue'.0
        4D 20 4F 4B 2C
        20 4D 3D 4D 65
        6E 75 65 00
003D76                                  5792: txto6:
003D76  45 50 52 4F                     5793:         dc.b 'EPROM FEHLER M=Menue'.0
        4D 20 46 45 48
        4C 45 52 20 4D
        3D 4D 65 6E 75
        65 00
                                        5794:
003D8B                                  5795: iomsg:
003D8B  49 4F 20 6C                     5796:         dc.b 'IO lesen'.0
        65 73 65 6E 00
003D94                                  5797: io1msg:
003D94  52 3D 41 64                     5798:         dc.b 'R=Adr D=Dauer S=Stop M=Menue'.0
        72 20 44 3D 44
        61 75 65 72 20
        53 3D 53 74 6F
        70 20 4D 3D 4D
        65 6E 75 65 00
                                        5799:
003DB1                                  5800: iopmsg:
003DB1  49 4F 20 73                     5801:         dc.b 'IO setzen'.0
        65 74 7A 65 6E
        00
                                        5802:
003DBB                                  5803: datamsg:
003DBB  44 61 74 61                     5804:         dc.b 'Data:    '.0
        3A 20 20 20 00
                                        5805:
003DC4                                  5806: io01msg:
003DC4  4D 3D 4D 65                     5807:         dc.b 'M=Menue R=Adr'.0
        6E 75 65 20 52
        3D 41 64 72 00
                                        5808:
003DD2                                  5809: txtreg1:
003DD2  44 30 20 20                     5810:  dc.b 'D0       D1       D2       D3       D4       D5       D6
        20 20 20 20 20
        44 31 20 20 20
        20 20 20 20 44
        32 20 20 20 20
        20 20 20 44 33
        20 20 20 20 20
        20 20 44 34 20
        20 20 20 20 20
        20 44 35 20 20
        20 20 20 20 20
        44 36 20 20 20
        20 20 20 20 44
        37
003E13  00                              5811:  dc.b 0
                                        5812:
003E14                                  5813: txtreg2:
003E14  41 30 20 20                     5814:  dc.b 'A0       A1       A2       A3       A4       A5       A6
        20 20 20 20 20
        41 31 20 20 20
        20 20 20 20 41
        32 20 20 20 20
        20 20 20 41 33
        20 20 20 20 20
        20 20 41 34 20
        20 20 20 20 20
        20 41 35 20 20
        20 20 20 20 20
        41 36 20 20 20
        20 20 20 20 41
        37
003E55  00                              5815:  dc.b 0
                                        5816:
003E56                                  5817: txtreg3:
003E56  53 3D 53 65                     5818:  dc.b 'S=Seite N=n mal B=Bis L=Loeschen M=Menue'
        69 74 65 20 4E
        3D 6E 20 6D 61
```

```
                6C 20 42 3D 42
                69 73 20 4C 3D
                4C 6F 65 73 63
                68 65 6E 20 4D
                3D 4D 65 6E 75
                65
003E7E  00                      5819:   dc.b 0
                                5820:
003E7F                          5821: txtreg4:
003E7F  53 52 20 20             5822:   dc.b 'SR   USP       SSP       PC'
                20 55 53 50 20
                20 20 20 20 20
                53 53 50 20 20
                20 20 20 20 50
                43
003E98  00                      5823:   dc.b 0
                                5824:
003E99                          5825: txtreg5:
003E99  45 69 6E 7A             5826:   dc.b 'Einzelschritt'.0
                65 6C 73 63 68
                72 69 74 74 00
                                5827:
                                5828:
                00 -- inserted
003EA6                          5829:           ds 0            * Angleich
                                5830:
                                5831: ** Ende Insert
                                5832:
                                5833:
                                5834:
                                5835.
                                5836. ** ENDE TEIL B
003EA8                          5837.           end
    0 Errors detected
```

```
...A68K 4.0a 06/13/83   ...Run on 01/22/85
                          1. ******************************************************
                          2. * CRT Paket  mit Bildschirmroutinen fuer Grundprog. *
                          3. * (C) 1984 Rolf-Dieter Klein. Muenchen V 4.2        *
                          4. * rev 4.2 Cursor auf Seite 0 eingeblendet          *
                          5. ******************************************************
                          6.
000000                    7.          section 1
                          8.
                          9.          include cpu
                         10: ********************************
                         11: * CPU Include File   840930    *
                         12: * wird von GRUND.FLO2.UTIL.CRT *
                         13: * benoetigt um IO-Adressen     *
                         14: * festzulegen                 *
                         15: ********************************
                         16:
00000001                 17: cpu     equ     1
                         18:
                         19: * 1= CPU 68008       8-Bit-Datenbus
                         20: * 2= CPU 68000/10   16-Bit-Datenbus
                         21: * 4= CPU 68020      32-Bit-Datenbus (32er Mode)
                         22:
                         23:
                         24: *
                         25: * Ende CPU include
                         26: * Achtung, bei Aendern der Datei, alle vier
                         27: * betroffenen Programme neu uebersetzen
                         28: *
                         29:
                         30:
                         31:
                         32:
                         33:
                         34:
FFFFFF70                 35. gdp equ $ffffff70*cpu
                         36.
                         37.
                         38.         xref wrtpage.viewpage.curon
                         39.         xref groesse.curx.curv
                         40.         xref lineotr.linecnt.screen
                         41.         xref color.xormode.optflag
                         42.         xref neuzeich.escmerker.tempzeich
                         43.         xref flip1.flip1cnt.flip.flipcnt
                         44. *
                         45.         xref aktpage.setpen.erapen
                         46.         xref cmd.wait.moveto.clrall
                         47.         xref getxy
                         48.
                         49. *
                         50.
                         51.         xdef cursorein.cursoraus.co
                         52.         xdef clrscreen.aktcur.charhandler
                         53.         xdef progzge.getcode.putchar.getline
                         54.         xdef prtline
                         55.
                         56.         xdef inschar.delchar.insline.delline
                         57.         xdef calccur.eraeoln.charoscr
                         58.
                         59. *
                         60. *
                         61.
                         62.
                         63.
000000
000000  4A2D'0000        64. cursorein:              *alle Register behalten
000004  6736              65.         tst.b curon(a5) * rev 4.0
000006  48E7 F8E0        66.         beq.s curf
00000A  6100 0106        67.         movem.l d0-d4/a0-a2,-(a7)
                         68.         bsr calccur
                         69.
00000E  422D'0000        70.         clr.b wrtpage(a5)       * rev 4.2
000012  6100'FFEC        71.         bsr aktpage             * Cursor auf Seite 0
000016  6100 0070        72.         bsr aktcur              * dann auch ohne
00001A  6100'FFE4        73.         bsr setpen              * autoflip sichtbar
00001E  103C 000A        74.         move.b #10,d0
000022  6100'FFDC        75.         bsr cmd
000026  6100 0060        76.         bsr aktcur
00002A  6100'FFD4        77.         bsr erapen
00002E  1012              78.         move.b (a2),d0  * aktuelles zeichen
000030  6100 0046        79.         bsr cmdout      * auch sonderzeichen
```

```
000034  6100'FFCA          80.          bsr setoen      * Seite 0 bleibt eingestellt.
000038  4CDF 071F          81.          movem.l (a7)+.d0-d4/a0-a2
00003C  4E75               82. curf:    rts
                           83.
                           84.
                           85.
                           86.
00003E                     87. cursoraus:              * alle Register behalten
00003E  4A2D'0000          88.          tst.b curon(a5) * rev 4.2 Seite 0
000042  6732               89.          beq.s curf1
000044  48E7 F8E0          90.          movem.l d0-d4/a0-a2.-(a7)
000048  6100 00C8          91.          bsr calccur
00004C  422D'0000          92.          clr.b wrtoage(a5)       * rev 4.2
000050  6100'FFAE          93.          bsr aktoage
000054  6100 0032          94.          bsr aktcur
000058  6100'FFA6          95.          bsr eraoen
00005C  103C 000A          96.          move.b #10.d0
000060  6100'FF9E          97.          bsr cmd
000064  6100 0022          98.          bsr aktcur
000068  6100'FF96          99.          bsr setoen
00006C  1012              100.          move.b (a2).d0  * aktuelles Zeichen
00006E  6100 0008         101.          bsr cmdout      * auch Sonderzeichen. Seite 0
000072  4CDF 071F         102.          movem.l (a7)+.d0-d4/a0-a2
000076  4E75              103. curf1:   rts
                          104.
                          105.
000078  4A00              106. cmdout: tst.b d0
00007A  6B06              107.          bmi.s cmdo1
00007C  6100'FF82         108.          bsr cmd
000080  4E75              109.          rts
000082  6100 022E         110. cmdo1:   bsr outchar
000086  4E75              111.          rts
                          112.
000088                    113. aktcur:          * Cursor Position in x.v-Wert  umsetzen
000088  48E7 E000         114.          movem.l d0-d2.-(a7)
00008C  6100'FF72         115.          bsr wait          * warten bis GDP fertig
000090  102D'0000         116.          move.b groesse(a5).d0
000094  11C0 FF73         117.          move.b d0.gdo+3*cou * +++ neu setzen hier +++
000098  B03C 0011         118.          cmp.b #$11.d0     * SIZE=11  Sonderfall wegen Speed rev 4.2
00009C  6744              119.          beq.s aktlcur     * Schnellberechnung
                          120.                            *
00009E  102D'0000         121.          move.b curx(a5).d0
0000A2  C07C 00FF         122.          and #$FF.d0
0000A6  C0FC 0006         123.          mulu #6.d0        *curx * 6 *sizex = position x
0000AA  1238 FF73         124.          move.b gdo+3*cou.d1       * size x holen
0000AE  E859              125.          ror #4.d1
0000B0  C27C 000F         126.          and #$f.d1        * nur 1..15 erlaubt
0000B4  C2C0              127.          mulu d0.d1        * mit d0 multiplizieren
                          128.                            *x=parameter
0000B6  102D'0000         129.          move.b curv(a5).d0
0000BA  C07C 00FF         130.          and #$FF.d0
0000BE  C0FC 000A         131.          mulu #10.d0       * curv * 10
0000C2  1438 FF73         132.          move.b gdo+3*cou.d2
0000C6  C47C 000F         133.          and #$f.d2
0000CA  C0C2              134.          mulu d2.d0        * nach d0
0000CC  C4FC 000A         135.          mulu #10.d2       * sizev*10   rev 4.2 obere Seite ok.
0000D0  4442              136.          neg d2            * -sizev*10
0000D2  D47C 0100         137.          add #256.d2       * rev 4.2 oben auch berechnen
0000D6  9440              138.          sub d0.d2         * pos v := 256-10*sizev-curv*10*sizev
0000D8  6100'FF26         139.          bsr moveto        * dorthin stellen
0000DC  4CDF 0007         140.          movem.l (a7)+.d0-d2
0000E0  4E75              141.          rts
                          142.
0000E2                    143. aktlcur:         * rev 4.2 . Sonderfall fuer schnelle Ausgabe
0000E2  4241              144.          clr d1
0000E4  4242              145.          clr d2
0000E6  4240              146.          clr d0
0000E8  122D'0000         147.          move.b curx(a5).d1        * x=curx*6
0000EC  1001              148.          move.b d1.d0
0000EE  D241              149.          add.w d1.d1       * curx*2
0000F0  D240              150.          add.w d0.d1       * (curx*2+curx)
0000F2  D241              151.          add.w d1.d1       * (curx*2+curx)*2 =x
0000F4  102D'0000         152.          move.b curv(a5).d0        * v=256-10-curv*10
0000F8  1400              153.          move.b d0.d2
0000FA  D040              154.          add.w d0.d0       * curv*2
0000FC  D040              155.          add.w d0.d0       * curv*2
0000FE  D042              156.          add.w d2.d0       * (curv*4+curv)
000100  D040              157.          add.w d0.d0       * (curv*4+curv)*2
000102  343C 00F6         158.          move #256-10.d2
000106  9440              159.          sub.w d0.d2       * v=256-10-curv*10
```

```
000108  6100'FEF6          160.          bsr moveto       * und positionieren
00010C  4CDF 0007          161.          movem.l (a7)+.d0-d2
000110  4E75              162.          rts
                           163.
000112                     164. calccur:         *
000112  102D'0000          165.          move.b curv(a5).d0
000116                     166. getline:                  * d0 = zeile 0..23 die gewuenscht wird
                           167.                           * wie curv position auf dem schirm
                           168.                           * a0 = startadresse der zeileninfo
                           169.                           * a1 = startadresse der zeilenlaenge
                           170.                           * a2 = startadresse zeichen
000116  41ED'0000          171.          lea lineptr(a5).a0       *basis indirect
00011A  43ED'0000          172.          lea linecnt(a5).a1       *basis zeilenlaenge
00011E  C07C 00FF          173.          and #$FF.d0      *auf wort ergaenzen
000122  1030 0000          174.          move.b 0(a0.d0).d0       *0..23 indirekte adresse
000126  C07C 00FF          175.          and #$FF.d0
00012A  D2C0               176.          add d0.a1        *adresse zeilenlaenge
00012C  C0FC 0050          177.          mulu #80.d0      *laenge einer zeile
000130  41ED'0000          178.          lea screen(a5).a0
000134  D0C0               179.          add d0.a0        *adresse in bildschirm
000136  2448               180.          movea.l a0.a2    *auch dorthin
000138  102D'0000          181.          move.b curx(a5).d0       *0..79
00013C  C07C 00FF          182.          and #$FF.d0
000140  D4C0               183.          add d0.a2
000142  4E75               184.          rts
                           185.
                           186.
000144                     187.
                           188. clrscreen:       * init aller variable flip etc.
                           189.                  * Farbe=0 XOR=0
000144  48E7 C080          190.          movem.l d0/d1/a0.-(a7)
000148  422D'0000          191.          clr.b color(a5)
00014C  422D'0000          192.          clr.b xormode(a5)
000150  6100'FEAE          193.          bsr aktpage      * und setzen. rev 4.0
000154  1B7C 0001          194.          move.b #1.curon(a5)      *nun cursor an
        '0000
00015A  6100'FEA4          195.          bsr clrall               * loeschen bildschirm alle seiten
00015E  6100'FEA0          196.          bsr wait
000162  11ED'0000          197.          move.b groesse(a5).qdp+3*cou     * size = std
        FF73
000168  41ED'0000          198.          lea screen(a5).a0
00016C  303C 077F          199.          move #80*24-1.d0 *zaehler
000170  10FC 0020          200. clrlo:   move.b #' '.(a0)+
000174  51C8 FFFA          201.          dbra d0.clrlo    *screen leeren
000178  422D'0000          202.          clr.b curx(a5)
00017C  422D'0000          203.          clr.b curv(a5)
000180  41ED'0000          204.          lea lineptr(a5).a0
000184  303C 0017          205.          move #24-1.d0    *zeilenzahl
000188  4241              206.          clr d1
00018A  10C1               207. clrlo1:  move.b d1.(a0)+
00018C  5241               208.          addq #1.d1       *0.1.2.3..23
00018E  51C8 FFFA          209.          dbra d0.clrlo1
000192  41ED'0000          210.          lea linecnt(a5).a0
000196  303C 0017          211.          move #24-1.d0
00019A  4218               212. clrlo2:  clr.b (a0)+
00019C  51C8 FFFC          213.          dbra d0.clrlo2   *alle zeilen auch
0001A0  422D'0000          214.          clr.b flip(a5)
0001A4  1B7C 000A          215.          move.b #10.flip(a5)
        0000
0001AA  1B7C 000A          216.          move.b #10.flipcnt(a5)
        '0000
0001B0  422D'0000          217.          clr.b viewpage(a5)
0001B4  422D'0000          218.          clr.b wrtpage(a5) *seite 0 verwenden
0001B8  6100'FE46          219.          bsr aktpage
0001BC  4CDF 0103          220.          movem.l (a7)+.d0/d1/a0
0001C0  4E75               221.          rts
                           222.
                           223.
0001C2  48E7 F8E0          224. prt79:   movem.l d0-d4/a0-a2.-(a7)
0001C6  363C 004F          225.          move #79.d3      *Zeichenzaehler
0001CA  6008               226.          bra.s prtx
                           227.
0001CC                     228. prtline:                  *eine zeile von aktueller pos ausgeben
0001CC  48E7 F8E0          229.          movem.l d0-d4/a0-a2.-(a7)
0001D0  363C 0050          230.          move #80.d3
0001D4  6100 FF3C          231. prtx:    bsr calccur      *a2 ist pointer
0001D8  204A               232.          move.l a2.a0     *zeichenpos  a
0001DA  182D'0000          233.          move.b curx(a5).d4
0001DE  45F8 FF70          234.          lea qdp.a2
0001E2  0812 0002          235. prtxlp:  btst.b #2.(a2)
```

```
0001E6  67FA
0001E8  1018
0001EA  6B06
0001EC  1480
0001EE  6000 0006
0001F2
0001F2  6100 00BE
0001F6
0001F6  5244
0001F8  B803
0001FA  66E6
0001FC  4CDF 071F
000200  4E75



000202
000202  48E7 F8E0
000206  323C 0017
00020A  6008

00020C
00020C  48E7 F8E0
000210  323C 0018
000214  6100 FEFC
000218  204A
00021A  1611
00021C  142D'0000
000220  182D'0000
000224  6000 0026

000228
000228  48E7 F8E0
00022C  323C 0018
000230  343C 0001
000234  6000 000C


000238
000238  48E7 F8E0
00023C  323C 0018
000240  4242
000242
000242  300?
000244  6100 FED0
000248  1611
00024A  4244
00024C
00024C  4A03
00024E  6722
000250  45F8 FF70
000254
000254  0812 0002
000258  67FA
00025A  1018
00025C  6B0A
00025E  1480
000260  5244
000262  B604
000264  66EE
000266  600A
000268
000268  6100 0048

00026C  5244
00026E  B604
000270  66E2
000272  6100'FD8C
000276  4238 FF78
00027A  4236 FF79
00027E  1038 FF7A
000282  E158
000284  1038 FF7B
000288  E940
00028A  48C0
00028C  E840
00028E  1638 FF73
000292  C67C 000F
000296  C6FC 000A
00029A  9043
```

```
236.            beq ortx1lo
237.            move.b (a0)+,d0
238.            bmi.s ortx11lo
239.            move.b d0,(a2)  *ausgeben
240.            bra ortx2lo
241. ortx11lo:
242.            bsr putchar
243. ortx2lo:
244.            addq #1,d4
245.            cmp.b d3,d4     *alle zeichen
246.            bne ortx1lo
247.            movem.l (a7)+,d0-d4/a0-a2
248.            rts
249.
250.
251.
252. ortr23:                    *ausgabe nur bis zeile 23
253.            movem.l d0-d4/a0-a2,-(a7)
254.            move #23,d1
255.            bra.s ort2      *weiter in restausgabe
256.
257. ortrest:                   *von aktueller position an bis ende
258.            movem.l d0-d4/a0-a2,-(a7)
259.            move #24,d1     *anzahl der zeilen
260. ort2:     bsr calccur     *a0,a1,a2=zeichenposition
261.            movea.l a2,a0   *zeichen pos hier verwenden
262.            move.b (a1),d3  *Zeichenzaehler max
263.            move.b curv(a5),d2      *v start zeile
264.            move.b curx(a5),d4      *x - zeichenpositon
265.            bra ort1x               *einsprung schleifen
266.
267. orttop:                    *nur 1..23 ausgeben
268.            movem.l d0-d4/a0-a2,-(a7)
269.            move #24,d1     *letzte Zeile
270.            move #1,d2      *start bei zeile 1
271.            bra ort1o1      *in schleife
272.
273.                            *von aktuelle gdp-position an
274. ortall:                    *gesamten Schirm ausgeben
275.            movem.l d0-d4/a0-a2,-(a7)
276.            move #24,d1     *linemax in d1
277.            clr d2          *temp zaehler zeile
278. ort1o1:                    *schleife
279.            move d2,d0      *zeile als index
280.            bsr getline     *a0,a1,a2 berechnen start zeichen
281.            move.b (a1),d3  *zeilenlaenge holen
282.            clr d4          *zaehler 0..79
283. ort1x:                     *einsprung d2,d3,d4
284.            tst.b d3        *wenn zeilenlaenge <> 0
285.            beq.s ort1o2    *sonst ignore
286.            lea gdp,a2      *start zeichen in a2 frei
287. ort1o2:                    *wiederhole zeichenausgabe
288.            btst.b #2,(a2)  *port test
289.            beq ort1o2      *bis gdp ready
290.            move.b (a0)+,d0 *datenwert
291.            bmi.s ort1spez
292.            move.b d0,(a2)  *nach cmd Kanal ausgeben
293.            addq #1,d4
294.            cmp.b d4,d3     * doppelter code wegen geschw.
295.            bne ort1o2
296.            bra.s ort1o2    * dann weiter
297. ort1spez:                  *ausgabe sonderzeichen
298.            bsr putchar     *zeichen spez
299.                            *ende sonderzeichenausgabe
300. ort1o2:    addq #1,d4      *zeichenzaehler
301.            cmp.b d4,d3     *=max zeile
302.            bne ort1o2      *ende wiederhole
303. ort1o1:    bsr wait
304.            clr.b gdp+8*cou *loeschen msb x
305.            clr.b gdp+9*cou *loeschen lsb x
306.            move.b gdp+$A*cpu.d0    *msb v
307.            rol.w #8,d0
308.            move.b gdp+$B*cpu.d0    *lsb v
309.            asl #4,d0
310.            ext.l d0
311.            asr #4,d0       *4096 vorzeichenbehaftet
312.            move.b gdp+3*cou,d3
313.            and #$F,d3      *size 1.. 15  0 nicht ok
314.            mulu #10,d3
315.            sub d3,d0       *v := v - 10 * sizev  mit v mult
```

```
00029C  11C0 FF7B          316.          move.b d0,gdo+$B*cou
0002A0  E058               317.          ror #8,d0
0002A2  11C0 FF7A          318.          move.b d0,gdo+$A*cou
0002A6  5242               319.          addo #1,d2        *neue zeile
0002A8  B401               320.          cmp.b d1,d2       *=maxzeile
0002AA  6696               321.          bne prtlp1        *bis ausgegeben
0002AC  4CDF 071F          322.          movem.l (a7)+,d0-d4/a0-a2
0002B0  4E75               323.          rts
                           324.
                           325.
                           326. * Sonderzeichenausgabe
                           327. * Damit kann man Fremdsprachen-Zeichen ausgeben
                           328.
0002B2                     329. outchar:                  * d0 = zeichen
                           330.                           * Zeichen ausgeben
                           331.                           * Groesse beruecksichtigen
                           332.                           * Bit 7 gesetzt
0002B2  48E7 C0C0          333.          movem.l d0-d1/a0-a1,-(a7)
0002B6  41FA 015E          334.          lea ztab1(pc),a0        * Start Texttabelle
0002BA  43FA 0164          335.          lea ztab2(pc),a1
0002BE  0880 0007          336.          bclr #7,d0        * Ruecksetzen
0002C2  323C 0009          337.          move #anztab-1,d1
0002C6  B018               338. out1c:   cmp.b (a0)+,d0
0002C8  670C               339.          beq.s out2c
0002CA  D3FC 00000005      340.          adda.l #5,a1      * Pointer auf Zeichengen
0002D0  51C9 FFF4          341.          dbra d1,out1c
0002D4  6006               342.          bra.s out3c       * Zeichen nicht gefunden
0002D6                     343. out2c:
0002D6  2049               344.          movea.l a1,a0     * a0 ist Parameter
0002D8  6100 0008          345.          bsr progzge       * Zeichen ablegen
0002DC                     346. out3c:
0002DC  4CDF 0303          347.          movem.l (a7)+,d0-d1/a0-a1
0002E0  4E75               348.          rts
                           349.
                           350. * Programmierbarer Zeichengenerator
                           351. * Damit kann man selbst definierte Zeichen in
                           352. * unterschiedlicher Groesse auf dem Bildschirm
                           353. * ausgeben. Dabei werden die Zeichen in Matrixform
                           354. * gespeichert.
                           355.
0002E2                     356. progzge:                  * Globaler Eingang
                           357.                           * a0-> Zeichengenerator
                           358.                           * Wird an aktueller Stelle
                           359.                           * mit SIZE X,Y ausgegeben
                           360.                           *rev 3.1 speed
0002E2  48E7 FEC0          361.          movem.l d0-d6/a0/a1,-(a7)
0002E6  43F8 FF70          362.          lea gdp,a1        * gdpadresse in a1
0002EA  6100 FD14          363.          bsr wait
0002EE  1029 0003          364.          move.b cou*3(a1),d0
0002F2  B03C 0011          365.          cmp.b #$11,d0
0002F6  6700 007E          366.          beq prog11zge     * size 11 x,y=1
0002FA  B03C 0021          367.          cmp.b #$21,d0
0002FE  6700 00C2          368.          beq prog21zge     * size 21 x=2, y=1
000302  6100 FCFC          369.          bsr getxy         * koordinaten holen d1=x d2=y
000306  363C 0005          370.          move #6-1,d3      * Spalten, 5 ist dummy leer
00030A                     371. prog1zg:
00030A  6100 FCF4          372.          bsr wait
00030E  1C29 0003          373.          move.b cou*3(a1),d6
000312  E81E              374.          ror.b #4,d6
000314  CC3C 000F          375.          and.b #$f,d6      * SIZE x
000318                     376. prog11z:
000318  4A43               377.          tst d3
00031A  6700 003E          378.          beq progsk
00031E  3F02               379.          move d2,-(a7)     * y retten
000320  1010              380.          move.b (a0),d0    * Holen des Wertes
000322  383C 0007          381.          move #8-1,d4      * Zeilen
000326                     382. prog2zg:
000326  6100 FCD8          383.          bsr wait
00032A  1A29 0003          384.          move.b cou*3(a1),d5     * CSIZE %..y
00032E  CA3C 000F          385.          and.b #$f,d5      * 0 ist 16 mal
000332                     386. prog22z:
000332  0800 0007          387.          btst #7,d0        *  DOT  first out
000336  6708              388.          beq.s prog3zg
000338  6100 FCC6          389.          bsr wait
00033C  12BC 0080          390.          move.b #$80,(a1) * D0 unveraendert
000340                     391. prog3zg:
000340  5242              392.          add #1,d2         * Y:=Y+1
000342  6100 FCBC          393.          bsr wait          * nur lsb Y setzen, wrap bei Randueberlauf
000346  1342 000B          394.          move.b d2,cou*$b(a1) * da zeitkritisch besser so
00034A  5305               395.          sub.b #1,d5
```

```
00034C  CA3C 000F       396.          and.b #$f,d5     * 16 Mal max
000350  66E0           397.          bne prog22z
000352  E318           398.          rol.b #1,d0      * Next Bit
000354  51CC FFD0       399.          dbra d4,prog2zg
00035B  341F           400.          move (a7)+,d2    * y zurueck
00035A                 401. progsk:                   * letzte Zeile nur rechnen
00035A  5241           402.          add #1,d1        * x:=x+1
00035C  6100 FCA2       403.          bsr moveto       * Neue Position
000360  5306           404.          sub.b #1,d6      * size y
000362  CC3C 000F       405.          and.b #$f,d6     * 16 Mal max
000366  66B0           406.          bne prog11z
000368  D0FC 0001       407.          adda #1,a0       * nun erst neue Spalte
00036C  51CB FF9C       408.          dbra d3,prog1zg
000370  4CDF 037F       409.          movem.l (a7)+,d0-d6/a0/a1
000374  4E75           410.          rts
                       411.
                       412. * High speed Sonderfaelle  REV 3.1
                       413.
                       414.
000376                 415. prog11zge:                * Einsprung nach registerretten
000376  6100 FC88       416.          bsr getxy        * Koordinaten holen d1=x d2=y
00037A  363C 0005       417.          move #6-1,d3     * Spalten, 6 ist dummy leer
00037E                 418. pro1g1zg:
0003[illegible]  4A43           419.          tst d3
0003[illegible]  672C           420.          beq.s pro1gsk
0003[illegible]  3A02           421.          move d2,d5       * y retten, hier d5 frei
000384  1010           422.          move.b (a0),d0   * Holen des Wertes
000386  383C 0007       423.          move #8-1,d4     * Zeilen
00038A                 424. pro1g2zg:
00038A  0800 0007       425.          btst #7,d0       *  DOT  first out
00038E  670A           426.          beq.s pro1g3zg
000390                 427. pro1w1:
000390  0811 0002       428.          btst.b #2,(a1)   * ready
000394  67FA           429.          beq.s pro1w1
000396  12BC 0080       430.          move.b #$80,(a1) * D0 unveraendert
00039A                 431. pro1g3zg:
00039A  5242           432.          add #1,d2        * Y:=Y+1
00039C  E318           433.          rol.b #1,d0      * Next Bit
00039E                 434. pro1w2:
00039E  0811 0002       435.          btst.b #2,(a1)
0003A2  67FA           436.          beq.s pro1w2
                       437.                           * nur lsb Y setzen, wrap bei Randueberlau
0003A4  1342 000B       438.          move.b d2,cou*$b(a1) * da zeitkritisch besser so. rev 4.1b
0003A8  51CC FFE0       439.          dbra d4,pro1g2zg
0003AC  3405           440.          move d5,d2       * y zurueck, hier d5 frei
0003AE                 441. pro1gsk:                  * letzte Zeile nur rechnen
0003AE  5241           442.          add #1,d1        * x:=x+1
0003B0  6100 FC4E       443.          bsr moveto       * Neue Position
0003B4  D0FC 0001       444.          adda #1,a0       * nun erst neue Spalte
0003B8  51CB FFC4       445.          dbra d3,pro1g1zg
0003BC  4CDF 037F       446.          movem.l (a7)+,d0-d6/a0/a1
0003C0  4E75           447.          rts
                       448.
0003C2                 449. prog21zge:                * size=21
0003C2  6100 FC3C       450.          bsr getxy        * Koordinaten holen d1=x d2=y
0003C6  363C 0005       451.          move #6-1,d3     * Spalten, 6 ist dummy leer
0003CA                 452. pro2g1zg:
0003CA  3C3C 0001       453.          move #2-1,d6
0003CE                 454. pro2g11z:
0003CE  4A43           455.          tst d3
0003[illegible]  672C           456.          beq.s pro2gsk
0003[illegible]  3A02           457.          move d2,d5       * y retten hier d5 frei
0003D4  1010           458.          move.b (a0),d0   * Holen des Wertes
0003D6  383C 0007       459.          move #8-1,d4     * Zeilen
0003DA                 460. pro2g2zg:
0003DA  0800 0007       461.          btst #7,d0       *  DOT  first out
0003DE  670A           462.          beq.s pro2g3zg
0003E0                 463. pro2w1:
0003E0  0811 0002       464.          btst.b #2,(a1)
0003E4  67FA           465.          beq.s pro2w1
0003E6  12BC 0080       466.          move.b #$80,(a1) * D0 unveraendert
0003EA                 467. pro2g3zg:
0003EA  5242           468.          add #1,d2        * Y:=Y+1
0003EC  E318           469.          rol.b #1,d0      * Next Bit
0003EE                 470. pro2w2:
0003EE  0811 0002       471.          btst.b #2,(a1)
0003F2  67FA           472.          beq.s pro2w2
                       473.                           * nur lsb Y setzen, wrap bei Randueberla
0003F4  1342 000B       474.          move.b d2,cou*$b(a1) * da zeitkritisch besser so. rev 4.1
0003F8  51CC FFE0       475.          dbra d4,pro2g2zg
```

```
0003FC  3405                    move d5.d2      * v zurueck. hier d5 frei
0003FE                  pro2gsk:                        * letzte Zeile nur rechnen
0003FE  5241                    add #1.d1       * x:=x+1
000400  6100'FBFE              bsr moveto      * Neue Position
000404  51CE FFC8              dbra d6.pro2g11z * doppelt in x-Richtung
000408  D0FC 0001              adda #1.a0      * nun erst neue Spalte
00040C  51CB FFBC              dbra d3.pro2g1zg
000410  4CDF 037F              movem.l (a7)+.d0-d6/a0/a1
000414  4E75                    rts


0000000A                anztab equ 10

000416  5B 5C 5D 7B    ztab1:  dc.b $5b.$5c.$5d.$7b.$7c.$7d.$7e
        7C 7D 7E
00041D  01 02 03                dc.b 1.2.3


000420  7D              ztab2:  dc.b %01111101  * 5b
000421  0A                      dc.b %00001010
000422  09                      dc.b %00001001
000423  0A                      dc.b %00001010
000424  7D                      dc.b %01111101

000425  3D                      dc.b %00111101  * 5c
000426  42                      dc.b %01000010
000427  42                      dc.b %01000010
000428  42                      dc.b %01000010
000429  3D                      dc.b %00111101

00042A  7D                      dc.b %01111101  * 5d
00042B  40                      dc.b %01000000
00042C  40                      dc.b %01000000
00042D  40                      dc.b %01000000
00042E  7D                      dc.b %01111101

00042F  71                      dc.b %01110001  * 7b
000430  54                      dc.b %01010100
000431  54                      dc.b %01010100
000432  78                      dc.b %01111000
000433  41                      dc.b %01000001

000434  00                      dc.b %00000000  * 7c
000435  39                      dc.b %00111001
000436  44                      dc.b %01000100
000437  44                      dc.b %01000100
000438  39                      dc.b %00111001

000439  3D                      dc.b %00111101  * 7d
00043A  40                      dc.b %01000000
00043B  40                      dc.b %01000000
00043C  7D                      dc.b %01111101
00043D  40                      dc.b %01000000

00043E  00                      dc.b %00000000  * 7e
00043F  7F                      dc.b %01111111
000440  01                      dc.b %00000001
000441  4D                      dc.b %01001101
000442  32                      dc.b %00110010

000443  00                      dc.b %00000000  * 1  Start Block
000444  7F                      dc.b %01111111
000445  3E                      dc.b %00111110
000446  1C                      dc.b %00011100
000447  08                      dc.b %00001000

000448  08                      dc.b %00001000  * 2 Ende Block
000449  1C                      dc.b %00011100
00044A  3E                      dc.b %00111110
00044B  7F                      dc.b %01111111
00044C  00                      dc.b %00000000

00044D  40                      dc.b %01000000  * Einfuege Block
00044E  40                      dc.b %01000000
00044F  7F                      dc.b %01111111
000450  40                      dc.b %01000000
000451  40                      dc.b %01000000


                        ** Ende Zeichensatz
```

```
000452                                555.
                                      556.          ds 0              * adjust
                                      557.
                                      558.
                                      559.
                                      560.
000452                                561. getcode:                   * d0 = zeichen
                                      562.                            * Pruefen ob in Tabelle
                                      563.                            * wenn ja dann eintragen mit Bit 7=1
000452  48E7 4080                     564.          movem.l d1/a0.-(a7)
000456  41FA FFBE                     565.          lea ztab1(pc).a0
00045A  323C 0009                     566.          move #anztab-1.d1       * Zaehler max
00045E                                567. getc1:
00045E  B018                          568.          cmo.b (a0)+.d0
000460  670A                          569.          beq.s getc2
000462  51C9 FFFA                     570.          dbra d1.getc1           * nicht gefunden
000466  4CDF 0102                     571.          movem.l (a7)+.d1/a0
00046A  4E75                          572.          rts
00046C  08C0 0007                     573. getc2:   bset #7.d0      * Bit setzen
000470  4CDF 0102                     574.          movem.l (a7)+.d1/a0
000474  4E75                          575.          rts
                                      576.
                                      577.
                                      578. **** Screen-Bearbeitung
                                      579.
000476                                580. lindown:                   * Zeilen verschieben
000476  48E7 C080                     581.          movem.l d0/d1/a0.-(a7)
00047A  41ED 0000                     582.          lea lineptr(a5).a0
00047E  1010                          583.          move.b (a0).d0  *merken erste zeile
000480  323C 0016                     584.          move #23-1.d1   *zaehler
000484                                585. linlo1:
000484  10A8 0001                     586.          move.b 1(a0).(a0)
000488  5288                          587.          addq.l #1.a0
00048A  51C9 FFF8                     588.          dbra d1.linlo1
00048E  1080                          589.          move.b d0.(a0)  *last
000490  4CDF 0103                     590.          movem.l (a7)+.d0/d1/a0
000494  4E75                          591.          rts
                                      592.
                                      593.
000496                                594. scroll:                    *scrollen ausfuehren
000496  48E7 F0E0                     595.          movem.l d0-d3/a0-a2.-(a7)
00049A  1B7C 0000 0000                596.          move.b #0.viewpage(a5)
0004A0  1B7C 0001 0000                597.          move.b #1.wrtpage(a5)
0004A6  612C                          598.          bsr.s scrlupr
0004A8  1B7C 0001 0000                599.          move.b #1.viewpage(a5)
0004AE  1B7C 0000 0000                600.          move.b #0.wrtpage(a5)
0004B4  611E                          601.          bsr.s scrlupr
0004B6  61BE                          602.          bsr lindown
0004B8  303C 0017                     603.          move #23.d0       *zeile 23 holen
0004BC  6100 FC5B                     604.          bsr getline       *a0=start zeichen a1=zeichenzahl
0004C0  4211                          605.          clr.b (a1)        *keine gueltigen zeichen dort
0004C2  303C 004F                     606.          move #80-1.d0     *loeschen buffer
0004C6  10FC 0020                     607. scrlo:   move.b #' '.(a0)+
0004CA  51C8 FFFA                     608.          dbra d0.scrlo     *bis 80 zeichen geloescht
0004CE  4CDF 070F                     609.          movem.l (a7)+.d0-d3/a0-a2
0004D2  4E75                          610.          rts
                                      611.
                                      612.
                                      613.
0004D4                                614. scrlupr:                   *intern
0004D4  6100 FB2A                     615.          bsr aktpage       *unsichtbar schreiben
0004D8  6100 FB26                     616.          bsr eraoen        *altes bild loeschen
0004DC  4241                          617.          clr d1            *x:=0
0004DE  6100 001E                     618.          bsr setdvmax
0004E2  6100 FB1C                     619.          bsr moveto        *dorthin adostart
0004E6  6100 FD50                     620.          bsr prtall        *alles ausgeben
0004EA  6100 FB14                     621.          bsr setoen        *neues bild schreiben
0004EE  4241                          622.          clr d1
0004F0  6100 000C                     623.          bsr setdvmax
0004F4  6100 FB0A                     624.          bsr moveto
0004F8  6100 FD2E                     625.          bsr prttop        *nur 23 zeilen ausgeben von 1 an
0004FC  4E75                          626.          rts
                                      627.
0004FE                                628. setdvmax:                  * 256-10*sizev einstellen rev 4.2
0004FE  1438 FF73                     629.          move.b gdp+3*cou.d2
000502  C47C 000F                     630.          and #$f.d2
```

```
000506  B43C 0001      631.          cmo.b #1,d2
00050A  6606           632.          bne.s scr1uo
00050C  343C 00F6      633.          move #256-10,d2 *v:=256-10*sizev  rev 4.2
000510  4E75          634.          rts
000512                 635. scr1uo:
000512  C4FC 000A      636.          mulu #10,d2     * sizev*10   rev 4.2 obere Seite ok.
000516  4442           637.          neg d2          * -sizev*10
000518  D47C 0100      638.          add #256,d2     * rev 4.2 oben auch berechnen
00051C  4E75           639.          rts
                       640.
00051E                 641. charhandler:             *Textzeichen in D0
00051E  6100 0044      642.          bsr charlhan    * intern ohne scroll effekte fuer editor
000522  102D'0000      643.          move.b curx(a5),d0
000526  5200           644.          addq.b #1,d0    *neue x-position
000528  1B40'0000      645.          move.b d0,curx(a5)
00052C  B03C 004F      646.          cmp.b #79,d0    *wenn d0-79 <=0  dann weiter
000530  631E           647.          bls.s char9
000532  422D'0000      648.          clr.b curx(a5)  *cursor auf 0 zurueck
000536  102D'0000      649.          move.b curv(a5),d0
00053A  5200           650.          addq.b #1,d0
00053C  1B40'0000      651.          move.b d0,curv(a5)
000540  B03C 0017      652.          cmp.b #23,d0    *wenn d0-23 <=0 dann weiter
000544  630A           653.          bls.s char9
000546  1B7C 0017      654.          move.b #23,curv(a5)
        '0000
00054C  6100 FF48      655.          bsr scroll
000550  4E75          656. char9:   rts
                       657.
                       658.
000552                 659. charoscr:
000552  6100 0010      660.          bsr charlhan
000556  0C2D 004F      661.          cmp.b #79,curx(a5) * wenn gleich 79 dann bleibt wert
        '0000
00055C  6704           662.          beq.s charloscr
00055E  522D'0000      663.          addq.b #1,curx(a5)
000562                 664. charloscr:
000562  4E75          665.          rts
                       666.
000564                 667. charlhan:                * Unterprogramm gemeinsam, rev 4.1 0..ff
000564  4A2D'0000      668.          tst.b notflag(a5) *=0 dann amerikan zge
000568  6704           669.          beq.s char1
00056A  6100 FEE6      670.          bsr getcode     *ggf. umkodieren bit 7 setzen
00056E  1B40'0000      671. char1:   move.b d0,neuzeich(a5)
000572  422D'0000      672.          clr.b wrtpage(a5)
000576  422D'0000      673.          clr.b viewpage(a5)
00057A  6100'FAB4      674.          bsr aktpage
00057E  6100 FB08      675.          bsr aktcur      *an cursor position
000582  6100 FB8E      676.          bsr calccur     *a2 zeichenpointer a1=zeichenanzahl
000586  102D'0000      677.          move.b curx(a5),d0
00058A  5200           678.          addq.b #1,d0    *zeichen mehr
00058C  B011           679.          cmp.b (a1),d0   *alterwert <= curxneu dann rueckschreiben
00058E  6502           680.          bcs.s char2     *neu - alt >= 0 dann ausfuehren
000590  1280           681.          move.b d0,(a1)
000592  1B52'0000      682. char2:   move.b (a2),temozeich(a5)
000596  1012           683.          move.b (a2),d0
000598  B03C 0020      684.          cmp.b #' ',d0
00059C  6710           685.          beq.s char3     *bei leerzeichen ignorieren
00059E  6100'FA60      686.          bsr eraoen
0005A2  103C 000A      687.          move.b #10,d0   *loeschen
0005A6  6100'FA58      688.          bsr cmd
0005AA  6100 FADC      689.          bsr aktcur      *zurueck positionieren
0005AE  6100'FA50      690. char3:   bsr setoen
0005B2  102D'0000      691.          move.b neuzeich(a5),d0  *Neu im Wiederholspeicher
0005B6  1480           692.          move.b d0,(a2)
0005B8  6B06           693.          bmi.s char4     *
0005BA  6100'FA44      694.          bsr cmd
0005BE  6004           695.          bra.s char5
0005C0  6100 FCF0      696. char4:   bsr putchar     *bit 7 gesetzt
0005C4  1B7C 0001      697. char5:   move.b #1,wrtpage(a5)   *nun auf andere seite
        '0000
0005CA  6100'FA34      698.          bsr aktpage
0005CE  6100 FAB8      699.          bsr aktcur
0005D2  102D'0000      700.          move.b temozeich(a5),d0
0005D6  B03C 0020      701.          cmp.b #' ',d0   *blanks ignorieren
0005DA  6714           702.          beq.s char6
0005DC  6100'FA22      703.          bsr eraoen
0005E0  103C 000A      704.          move.b #10,d0   *loeschen
0005E4  6100'FA1A      705.          bsr cmd
0005E8  6100 FA9E      706.          bsr aktcur
0005EC  6100'FA12      707.          bsr setoen
```

```
0005F0  102D'0000       708. char6:  move.b neuzeich(a5).d0
0005F4  6B06            709.         bmi.s char7
0005F6  6100'FA0E       710.         bsr cmd           *normale zeichen
0005FA  4E75            711.         rts               * ehem. char8.
0005FC  6100 FCB4       712. char7:  bsr putchar
000600  4E75            713.         rts               * ohne cursor advance
                        714.
                        715.
                        716.
000602  422D'0000       717. seite0: clr.b wrtpage(a5)       *anwahl seite 0 als schreibseite
000606  1B7C 0001       718.         move.b #1.viewpage(a5)
        0000
00060C  6100'F9F2       719.         bsr aktpage
000610  6100 FA76       720.         bsr aktcur
000614  6100'F9EA       721.         bsr eraoen
000618  4E75            722.         rts
                        723.
                        724.
                        725.
00061A  422D'0000       726. seite1: clr.b viewpage(a5)      *anwahl seite 1 als schreibseite
00061E  1B7C 0001       727.         move.b #1.wrtpage(a5)
        0000
000624  6100'F9DA       728.         bsr aktpage
000628  6100 FA5E       729.         bsr aktcur
00062C  6100'F9D2       730.         bsr eraoen
000630  4E75            731.         rts
                        732.
                        733.
                        734.
                        735.
000632                  736. eraeoln:                     *von Cursor Position bis Zeilenende
                        737.                              *loeschen
000632  48E7 E0F8       738.         movem.l d0-d2/a0-a4.-(a7)
000636  61E2            739.         bsr seite1
000638  6100 FB92       740.         bsr prtline      *zeile ausgeben
00063C  61C4            741.         bsr seite0
00063E  6100 FB8C       742.         bsr prtline      *ausgeben
000642  6100 0008       743.         bsr eraln
000646  4CDF 1F07       744.         movem.l (a7)+.d0-d2/a0-a4
00064A  4E75            745.         rts
                        746. *
00064C                  747. eraln:
00064C  6100 FAC4       748.         bsr calccur      *speicheradressen a0.a1.a2
000650  102D'0000       749.         move.b curx(a5).d0      *cursorposition
000654  1280            750.         move.b d0.(a1)   *neue zeichenzahl
000656  103C 0050       751.         move.b #80.d0
00065A  9011            752.         sub.b (a1).d0    *80-zahl
00065C  14FC 0020       753. eral1:  move.b #' '.(a2)+
000660  5300            754.         subq.b #1.d0
000662  66F8            755.         bne eral1        *restzeile loeschen
000664  4E75            756.         rts
                        757.
000666                  758. eraeos:
000666  48E7 E0F8       759.         movem.l d0-d2/a0-a4.-(a7)
00066A  61AE            760.         bsr seite1
00066C  6100 FB9E       761.         bsr prtrest
000670  6190            762.         bsr seite0
000672  6100 FB98       763.         bsr prtrest
000676  61D4            764.         bsr eraln        *restzeile loeschen zuerst
                        765.                          *nun restliche zeilen cury+1 ...
000678  102D'0000       766.         move.b cury(a5).d0      *start position alte zeile
00067C                  767. eraxlo:                     *wiederholen bis letzte zeile erreicht
00067C  5240            768.         addq #1.d0       *naechste zeilen 0..23 erlaubt
00067E  3200            769.         move d0.d1
000680  B07C 0018       770.         cmp #24.d0       *wenn 24 dann fertig
000684  6716            771.         beq.s erafx1
000686  6100 FABE       772.         bsr getline      *a0.a1.a2 berechnen
00068A  4211            773.         clr.b (a1)       *quelltoe zeichen loeschen
00068C  103C 004F       774.         move.b #80-1.d0 *anzahl org zeile
000690                  775. eraxllo:
000690  10FC 0020       776.         move.b #' '.(a0)+
000694  51C8 FFFA       777.         dbra d0.eraxllo *loeschen der zeile
000698  3001            778.         move d1.d0
00069A  60E0            779.         bra eraxlo
00069C  4CDF 1F07       780. erafx1: movem.l (a7)+.d0-d2/a0-a4
0006A0  4E75            781.         rts
                        782.
                        783.
0006A2                  784. inschar:
0006A2  48E7 E0F8       785.         movem.l d0-d2/a0-a4.-(a7)
```

```
0006A6  6100 FF72      bsr seite1
0006AA  6100 FB20      bsr prtline
0006AE  6100 F9D8      bsr aktcur        *nun versetzt ausgeben
0006B2  6100'F94C      bsr setpen
0006B6  103C 0020      move.b #' '.d0
0006BA  6100'F944      bsr cmd           *leerzeichen
0006BE  102D'0000      move.b curx(a5).d0
0006C2  B03C 004F      cmp.b #79.d0      *<79 dann ausgeben
0006C6  6A04           bpl.s ins1
0006C8  6100 FAF8      bsr prt79         *n-1 zeichen
0006CC  6100 FF34      bsr seite0
0006D0  6100 FAFA      bsr prtline
0006D4  6100 F9B2      bsr aktcur
0006D8  6100'F926      bsr setpen
0006DC  103C 0020      move.b #' '.d0
0006E0  6100'F91E      bsr cmd
0006E4  102D'0000      move.b curx(a5).d0
0006E8  B03C 004F      cmp.b #79.d0
0006EC  6A04           bpl.s ins11
0006EE  6100 FAD2      bsr prt79
0006F2  6100 FA1E      bsr calccur       *a0.a1.a2
0006F6  0C11 0050      cmp.b #80.(a1)    *anzahl
0006FA  6A02           bpl.s ins2
0006FC  5211           addq.b #1.(a1)    *erhoehen
0006FE  102D'0000      move.b curx(a5).d0      *verschieben rest
000702  123C 0020      move.b #' '.d1    *start wert
000706  1412           move.b (a2).d2    *alter wert
000708  14C1           move.b d1.(a2)+   *neuer wert
00070A  3202           move d2.d1        *fuer naechsten
00070C  5240           addq #1.d0
00070E  B03C 0050      cmp.b #80.d0      *dann ende
000712  66F2           bne ins3
000714  4CDF 1F07      movem.l (a7)+.d0-d2/a0-a4
000718  4E75           rts


00071A
00071A  48E7 E0F8      movem.l d0-d2/a0-a4.-(a7)
00071E  6100 FEFA      bsr seite1
000722  6100 FAA8      bsr prtline
000726  6100 F960      bsr aktcur
00072A  6100'F8D4      bsr setpen
00072E  102D'0000      move.b curx(a5).d0
000732  B03C 004F      cmp.b #79.d0
000736  6A0C           bpl.s del1
000738  522D'0000      addq.b #1.curx(a5)
00073C  6100 FA8E      bsr prtline       *damit ein zeichen weniger aus
000740  1B40'0000      move.b d0.curx(a5)
000744  6100 FEBC      bsr seite0
000748  6100 FA82      bsr prtline
00074C  6100 F93A      bsr aktcur
000750  6100'F8AE      bsr setpen
000754  102D'0000      move.b curx(a5).d0
000758  B03C 004F      cmp.b #79.d0
00075C  6A0C           bpl.s del2
00075E  522D'0000      addq.b #1.curx(a5)
000762  6100 FA6B      bsr prtline       *damit ein zeichen weniger aus
000766  1B40'0000      move.b d0.curx(a5)
00076A  6100 F9A6      bsr calccur       *zeichen loeschen
00076E  102D'0000      move.b curx(a5).d0      *linecnt nicht updaten
000772  B03C 004F      cmp.b #79.d0      *79 dann fertig
000776  6A0C           bpl.s del3
000778  156A 0001      move.b 1(a2).0(a2)
        0000
00077E  528A           addq.l #1.a2
000780  5200           addq.b #1.d0
000782  60EE           bra del2o1
000784  14BC 0020      move.b #' '.(a2)
000788  4CDF 1F07      movem.l (a7)+.d0-d2/a0-a4
00078C  4E75           rts


00078E
00078E  48E7 E0F8      movem.l d0-d2/a0-a4.-(a7)
000792  422D'0000      clr.b curx(a5)    *stets auf den anfang setzen
000796  6100 FE82      bsr seite1
00079A  6100 FA70      bsr prtrest
00079E  6100'F860      bsr setpen
0007A2  102D'0000      move.b curv(a5).d0
0007A6  B03C 0017      cmp.b #23.d0
```

```
0007AA  6A10            865.            bol.s insl1
0007AC  522D'0000       866.            addq.b #1.curv(a5)      *erhoehen
0007B0  6100 F8D6       867.            bsr aktcur
0007B4  1B40'0000       868.            move.b d0.curv(a5)
0007B8  6100 FA48       869.            bsr ortr23      *ausgabe
0007BC  6100 FE44       870. insl1:     bsr seite0
0007C0  6100 FA4A       871.            bsr ortrest
0007C4  6100'F83A       872.            bsr setpen
0007C8  102D'0000       873.            move.b curv(a5).d0
0007CC  B03C 0017       874.            cmp.b #23.d0
0007D0  6A10            875.            bol.s insl2
0007D2  522D'0000       876.            addq.b #1.curv(a5)      *erhoehen
0007D6  6100 F9B0       877.            bsr aktcur
0007DA  1B40'0000       878.            move.b d0.curv(a5)
0007DE  6100 FA22       879.            bsr ortr23      *ausgabe
0007E2  41ED'0000       880. insl2:     lea lineotr(a5).a0
0007E6  102D'0000       881.            move.b curv(a5).d0
0007EA  C07C 00FF       882.            and #$FF.d0     *auf wort angleichen
0007EE  D0C0            883.            add d0.a0       *adresse zeile
0007F0  122D'0017       884.            move.b lineotr+23(a5).d1
0007F4  B03C 001B       885. insl3:     cmp.b #24.d0    *<24
0007F8  6A0A            886.            bol.s insl4
0007FA  1410            887.            move.b (a0).d2  *hole alte zeile
0007FC  10C1            888.            move.b d1.(a0)+ *neue zeile
0007FE  3202            889.            move d2.d1      *weiter so
000800  5200            890.            addq.b #1.d0
000802  60F0            891.            bra insl3
000804  6100 F90C       892. insl4:     bsr calccur
000808  4211            893.            clr.b (a1)      *zeichenzahl = 0 setzen
00080A  303C 004F       894.            move #80-1.d0   *anzahl loeschne
00080E  10FC 0020       895. insl5:     move.b #' '.(a0)+
000812  51C8 FFFA       896.            dbra d0.insl5
000816  4CDF 1F07       897.            movem.l (a7)+.d0-d2/a0-a4
00081A  4E75            898.            rts
                        899.
                        900.
                        901.
00081C                  902. delline:
00081C  48E7 E0F8       903.            movem.l d0-d2/a0-a4.-(a7)
000820  422D'0000       904.            clr.b curx(a5)
000824  6100 FDF4       905.            bsr seite1
000828  6100 F9E2       906.            bsr ortrest
00082C  6100'F7D2       907.            bsr setpen
000830  6100 F856       908.            bsr aktcur
000834  102D'0000       909.            move.b curv(a5).d0
000838  B03C 0017       910.            cmp.b #23.d0
00083C  6A0C            911.            bol.s dell1
00083E  522D'0000       912.            addq.b #1.curv(a5)
000842  6100 F9C8       913.            bsr ortrest
000846  1B40'0000       914.            move.b d0.curv(a5)
00084A  6100 FDB6       915. dell1:     bsr seite0
00084E  6100 F9BC       916.            bsr ortrest
000852  6100'F7AC       917.            bsr setpen
000856  6100 F830       918.            bsr aktcur
00085A  102D'0000       919.            move.b curv(a5).d0
00085E  B03C 0017       920.            cmp.b #23.d0
000862  6A0C            921.            bol.s dell2
000864  522D'0000       922.            addq.b #1.curv(a5)
000868  6100 F9A2       923.            bsr ortrest
00086C  1B40'0000       924.            move.b d0.curv(a5)
000870  41ED'0000       925. dell2:     lea lineotr(a5).a0
000874  102D'0000       926.            move.b curv(a5).d0
000878  C07C 00FF       927.            and #$FF.d0
00087C  D0C0            928.            add d0.a0
00087E  1210            929.            move.b (a0).d1  *zeilennummer
000880  B03C 0017       930. dell3:     cmp.b #23.d0
000884  6A0C            931.            bol.s dell4
000886  1168 0001 0000  932.            move.b 1(a0).0(a0)
00088C  5288            933.            addq.l #1.a0
00088E  5200            934.            addq.b #1.d0
000890  60EE            935.            bra dell3
000892  1141 0000       936. dell4:     move.b d1.0(a0)
000896  103C 0017       937.            move.b #23.d0
00089A  6100 F87A       938.            bsr getline
00089E  4211            939.            clr.b (a1)      *keine zeichen dort
0008A0  103C 004F       940.            move.b #80-1.d0
0008A4  10FC 0020       941. dell5:     move.b #' '.(a0)+
0008A8  51C8 FFFA       942.            dbra d0.dell5
0008AC  4CDF 1F07       943.            movem.l (a7)+.d0-d2/a0-a4
```

```
0008B0  4E75                    944.            rts
                                945.
                                946.
0008B2                          947.  eschandler:
0008B2  4A2D'0000               948.            tst.b escmerker(a5)
0008B6  6608                    949.            bne.s esc1
0008B8  1B7C 0001               950.            move.b #1.escmerker(a5) *einschalten sonst
        '0000
0008BE  4E75                    951.            rts
0008C0  0C2D 0002               952.  esc1:     cmp.b #2.escmerker(a5)
        '0000
0008C6  661C                    953.            bne.s esc11             *esc =  row
0008C8  903C 0020               954.            sub.b #32.d0            *y-adresse
0008CC  C0BC 000000FF           955.            and.l #$FF.d0           *div bezieht sich auf 32bit
0008D2  80FC 0018               956.            divu #24.d0             *rest ist modulo ergebnis
0008D6  4840                    957.            swap d0                 *
0008D8  1B40'0000               958.            move.b d0.curv(a5)              *0..23 erlaubt
0008DC  1B7C 0003               959.            move.b #3.escmerker(a5)
        '0000
0008E2  4E75                    960.            rts
                                961.
0008E4  0C2D 0003               962.  esc11:    cmp.b #3.escmerker(a5)
        '0000
0008EA  661C                    963.            bne.s esc12             *esc = row column
0008EC  903C 0020               964.            sub.b #32.d0            *x-position 0..79
0008F0  C0BC 000000FF           965.            and.l #$FF.d0           *div bezieht sich auf 32 [illegible]
0008F6  80FC 0050               966.            divu #80.d0             *0..79
0008FA  4840                    967.            swap d0                 *bereich
0008FC  1B40'0000               968.            move.b d0.curx(a5)
000900  6100 F786              969.            bsr aktcur
000904  6000 0082              970.            bra escfin              *ende esc-sequenz
                                971.
000908                          972.  esc12:                    *Zeichen in d0 nach esc sequenz
000908  B03C 0051              973.            cmp.b #'Q'.d0   *ins character
00090C  6608                    974.            bne.s esc2      *Ein Zeichen einfuegen
00090E  6100 FD92               975.            bsr inschar
000912  6000 0074              976.            bra escfin
                                977.
000916  B03C 0057              978.  esc2:     cmp.b #'W'.d0   *del character
00091A  6608                    979.            bne.s esc3      *Ein Zeichen loeschen
00091C  6100 FDFC               980.            bsr delchar
000920  6000 0066              981.            bra escfin
                                982.
000924  B03C 0045              983.  esc3:     cmp.b #'E'.d0   *ins line
000928  6608                    984.            bne.s esc4      *Eine Zeile einfuegen
00092A  6100 FE62               985.            bsr insline
00092E  6000 0058              986.            bra escfin
                                987.
000932  B03C 0052              988.  esc4:     cmp.b #'R'.d0   *del line
000936  6608                    989.            bne.s esc5      *Eine Zeile loeschen
000938  6100 FEE2               990.            bsr delline
00093C  6000 004A              991.            bra escfin
                                992.
000940  B03C 0054              993.  esc5:     cmp.b #'T'.d0   *erase to ende of line
000944  6608                    994.            bne.s esc6      *Zeile bis zum Ende loeschen
000946  6100 FCEA              995.            bsr eraeoln
00094A  6000 003C              996.            bra escfin
                                997.
00094E  B03C 0059              998.  esc6:     cmp.b #'Y'.d0   *erase to ende of page
000952  6608                    999.            bne.s esc7      *Bis zum Seitenende loeschen
000954  6100 FD10              1000.           bsr eraeos
000958  6000 002E              1001.           bra escfin
                               1002.
00095C  B03C 003D              1003. esc7:     cmp.b #'='.d0   *set cursor r c
000960  6608                    1004.           bne.s esc8      *Cursor positionieren
000962  1B7C 0002              1005.           move.b #2.escmerker(a5)
        '0000
000968  4E75                   1006.           rts
                               1007.
00096A  B03C 0024              1008. esc8:     cmp.b #'$'.d0   *Deutsch an
00096E  660A                    1009.           bne.s esc9
000970  1B7C 0001              1010.           move.b #1.optflag(a5)
        '0000
000976  6000 0010              1011.           bra escfin
                               1012.
00097A  B03C 0025              1013. esc9:     cmp.b #'%'.d0   *STD-Zeichensatz
00097E  6608                    1014.           bne.s escfin    *Amerika ASCII
000980  422D'0000              1015.           clr.b optflag(a5)
000984  6000 0002              1016.           bra escfin
                               1017.
```

```
                                1018.
                                1019. *
0009B8  422D'0000               1020. escfin: clr.b escmerker(a5)
0009BC  4E75                    1021.         rts
                                1022.
                                1023.
00098E                          1024. ctrlhandler:              *control zeichen auswertung
                                1025.
00098E  B03C 0008               1026.         cmp.b #$8,d0      *BS
000992  6620                    1027.         bne.s ctr1        *Eine Zeichen zurueck
000994  102D'0000               1028.         move.b curx(a5),d0
000998  6700 0008               1029.         beq ctr11
00099C  532D'0000               1030.         subq.b #1,curx(a5)
0009A0  4E75                    1031. ctr12:  rts
0009A2                          1032. ctr11:                    *linker Anschlag
0009A2  102D'0000               1033.         move.b curv(a5),d0
0009A6  67F8                    1034.         beq ctr12         *links oben
0009A8  1B7C 004F               1035.         move.b #79,curx(a5)
        0000
0009AE  532D'0000               1036.         subq.b #1,curv(a5)
0009B2  4E75                    1037.         rts               *ende BS
                                1038.
0009B4  B03C 001A               1039. ctr1:   cmp.b #$1a,d0     *Clear
0009B8  6606                    1040.         bne.s ctr2
0009BA  6100 F78B               1041.         bsr clrscreen     *loeschen alles
0009BE  4E75                    1042.         rts
                                1043.
0009C0  B03C 001E               1044. ctr2:   cmp.b #$1e,d0     *Home
0009C4  660A                    1045.         bne.s ctr3
0009C6  422D'0000               1046.         clr.b curx(a5)
0009CA  422D'0000               1047.         clr.b curv(a5)
0009CE  4E75                    1048.         rts
                                1049.
0009D0  B03C 000B               1050. ctr3:   cmp.b #$b,d0      *Cursor Up
0009D4  660C                    1051.         bne.s ctr4
0009D6  102D'0000               1052.         move.b curv(a5),d0
0009DA  6704                    1053.         beq.s ctr31
0009DC  532D'0000               1054.         subq.b #1,curv(a5)
0009E0  4E75                    1055. ctr31:  rts
                                1056.
0009E2  B03C 000A               1057. ctr4:   cmp.b #$a,d0      *Linefeed
0009E6  6616                    1058.         bne.s ctr5
0009E8  102D'0000               1059.         move.b curv(a5),d0
0009EC  B03C 0017               1060.         cmp.b #23,d0      *wenn >= 23 scroll
0009F0  6B06                    1061.         bmi.s ctr41       *d0-23 <0
0009F2  6100 FA42               1062.         bsr scroll        *d0 >= 23
0009F6  4E75                    1063.         rts
0009F8  522D'0000               1064. ctr41:  addq.b #1,curv(a5)      *sonst nach unten
0009FC  4E75                    1065.         rts
                                1066.
0009FE  B03C 0016               1067. ctr5:   cmp.b #$16,d0       * Cursor Down
000A02  6612                    1068.         bne.s ctr6          * rev 4.2 gueltig
000A04  102D'0000               1069.         move.b curv(a5),d0
000A08  B03C 0017               1070.         cmp.b #23,d0
000A0C  6A00 0006               1071.         bpl ctr51
000A10  522D'0000               1072.         addq.b #1,curv(a5) * ja d0<23, rev 4.2
000A14  4E75                    1073. ctr51:  rts
                                1074.
000A16  B03C 000C               1075. ctr6:   cmp.b #$C,d0
000A1A  6706                    1076.         beq.s ctr61
000A1C  B03C 0009               1077.         cmp.b #$9,d0
000A20  662A                    1078.         bne.s ctr7
000A22  102D'0000               1079. ctr61:  move.b curx(a5),d0
000A26  B03C 004F               1080.         cmp.b #79,d0
000A2A  661A                    1081.         bne.s ctr62
000A2C  422D'0000               1082.         clr.b curx(a5)
000A30  102D'0000               1083.         move.b curv(a5),d0
000A34  B03C 0017               1084.         cmp.b #23,d0
000A38  6606                    1085.         bne.s ctr63
000A3A  6100 FA5A               1086.         bsr scroll
000A3E  4E75                    1087.         rts
000A40  522D'0000               1088. ctr63:  addq.b #1,curv(a5)
000A44  4E75                    1089.         rts
                                1090. *
000A46  522D'0000               1091. ctr62:  addq.b #1,curx(a5)      *nach rechts
000A4A  4E75                    1092.         rts
                                1093.
000A4C  B03C 000D               1094. ctr7:   cmp.b #$d,d0
000A50  6606                    1095.         bne.s ctr8
000A52  422D'0000               1096.         clr.b curx(a5)
```

```
000A56  4E75            1097.           rts
000A58
000A58  4E75            1098. ctr8:
                        1099.           rts
                        1100.
                        1101.
000A5A                  1102. co:                       * Zeichen in d0 ausgeben
                        1103.                           * mit Steuerfunktionen
000A5A  6100'F5A4       1104.           bsr wait        * Immer Groesse neu setzen, rev 4.2
000A5E  11ED'0000       1105.           move.b groesse(a5),gdo+3*cpu * rev 4.2 SIZE-Befehl gueltig
        FF73
000A64  4A2D'0000       1106.           tst.b escmerker(a5)
000A68  6600 FE48       1107.           bne eschandler  *esc sequenz aktive alle zeichen
000A6C  B03C 001B       1108.           cmp.b #$1b,d0
000A70  6700 FE40       1109.           beq eschandler  *escape sequenz
000A74  B03C 0020       1110.           cmp.b #' ',d0
000A78  6500 FF14       1111.           bcs ctrlhandler * rev 4.1 bcs da voller Bereich 0..ff
000A7C  6000 FAA0       1112.           bra charhandler
                        1113.
000A80                  1114. prtco:                    *a0 -> start eines Textgebietes
                        1115.                           * 0 = textende
000A80  1018            1116.           move.b (a0)+,d0
000A82  670C            1117.           beq.s prtlco
000A84  48E7 0080       1118.           movem.l a0,-(a7)
000A88  61D0            1119.           bsr co
000A8A  4CDF 0100       1120.           movem.l (a7)+,a0
000A8E  60F0            1121.           bra prtco       *bis ende erreicht
                        1122.
000A90  4E75            1123. prtlco: rts
                        1124.
                        1125. * Ende CRT-Paket
                        1126.
000A92                  1127.           end
    0 Errors detected
```

FLOPPY

```
                    1. ****************************************
                    2. * Floopy-Disk Unterprogramme fuer       *
                    3. * 68000/68008/68020      840907        *
                    4. * (C) 1984 Rolf-Dieter Klein           *
                    5. * rev 1.1   840908        RDK           *
                    6. * rev 4.2   841030  RDK. SSO mit side  *
                    7. * nur Floopycontroller FD 1797         *
                    8. ****************************************
                    9.
000000             10.  section 1
                   11.
                   12.  xdef flinit.floopy.getflop     * Globale Routinen
                   13.  xref steprate.fehza.ccode.drvat.drvtab * Speicherzellen
                   14.  xref srsto
                   15.  include cpu
                   16: ********************************
                   17: * CPU Include File   840930    *
                   18: * wird von GRUND.FLO2.UTIL.CRT *
                   19: * benoetigt um IO-Adressen     *
                   20: * festzulegen                  *
                   21: ********************************
                   22:
00000001           23: cpu     equ     1
                   24:
                   25: * 1= CPU 68008       8-Bit-Datenbus
                   26: * 2= CPU 68000/10   16-Bit-Datenbus
                   27: * 4= CPU 68020      32-Bit-Datenbus (32er Mode)
                   28:
                   29:
                   30: *
                   31: * Ende CPU include
                   32: * Achtung, bei Aendern der Datei, alle vier
                   33: * betroffenen Programme neu uebersetzen
                   34: *
                   35:
                   36:
                   37:
                   38:
                   39:
                   40.
                   41.
                   42. * Floopycontroller FD 1797
                   43.
FFFFFFC0           44. flo0 equ $ffffffc0*cpu   * Kommandoregister
FFFFFFC1           45. flo1 equ $ffffffc1*cpu   * Track Register
FFFFFFC2           46. flo2 equ $ffffffc2*cpu   * Sektor Register
FFFFFFC3           47. flo3 equ $ffffffc3*cpu   * Daten Register
FFFFFFC4           48. flo4 equ $ffffffc4*cpu   * Status und Steuerung
                   49.
                   50. * flo4
                   51. * als Leseport
                   52. *
                   53. * 7     6       5       4       3       2       1       0
                   54. * DRQ   INTRQ   headld  x       x       x       x       x
                   55. * 1=Request liegt an
                   56. *       1=Interruot liegt an
                   57. *               1=Kopf liegt auf
                   58. *
                   59. *
                   60. *
                   61. * flo4
                   62. * als Schreibport
                   63. *
                   64. * 7     6       5       4       3       2       1       0
                   65. * sso   moton   mini    dense   d       c       b       a
                   66. *
                   67. * 1=Seite 1 mit Extraport, wenn Step-Bit 7=0, sonst mit SSO
                   68. *       0=Motor an
                   69. *               1=Minilaufwerk
                   70. *                       1=Single Dense
                   71. *                                 -------------------------
                   72. *                                   1=Laufwerke an
                   73. *
                   74.
                   75. * Floopy.Init-Aufruf
                   76. * a5 zeigt auf lokalen Speicherbereich ++++ WICHTIG ++++
                   77. * damit ist RAM-Bereich verschiebbar
                   78. * a0 zeigt auf Ziel- oder Quelladresse
                   79. * d0 ist frei
```

Floppy

```
                                * d1 Befehlscode 0=Steprate setzen.1=Lesen.2=Schreiben
                                * d2 Sektor
                                * d3 Track oder (steprate plus sso Bit 7)
                                * d4 =Laufwerkscode+dense+seite
                                *      entspricht dem Internformat von Port: flo4
                                *
                                * ACHTUNG FLO2 arbeitet beim 68000/8 ohne Interruots.
                                * Daher ist es besser die INT-Leitung zu trennen. um
                                * keine Stoerung von Anwenderprogrammen zu erleben.
                                * Man achte darauf moeglichst nicht die Bruecke
                                * fuer den NMI auf der CPU aktiviert zu haben. obwohl
                                * fuer diesen Fall vorgesorgt ist.
                                *
000000                          flinit:
000000  1B7C 0003 0000          move.b #3.steprate(a5) * Langsamste Rate ist Default.sso=0
000006  1B7C 00FF 0000          move.b #$ff.drvat(a5)  * Flag. Laufwerk unqueltig
00000C  43ED 0000               lea drvtab(a5).a1      * alle Laufwerke auf Spur 0
000010  303C 000F               move #16-1.d0
000014                          flinitlp:
000014  12FC 00FF               move.b #$ff.(a1)+      * Undefinierte Spuren
000018  51C8 FFFA               dbra d0.flinitlp       * dem einschalten
00001C  4E75                    rts
                                *
                                *                      * Ankommende NMI-Interruots werden
                                *                      * in dieser Zeit ignoriert.
                                *                      * Die Floppy-Routine darf nicht durch
00001E                          *                      * Interruots gestoert werden.
                                floppy:                * Aufruf des Hauptprogramms
                                *                      * Alle Register bleiben erhalten
00001E  40E7                    move sr.-(a7)          * Status retten
000020  003C 0007               or #7.ccr              * Level 7 setzen bei Interruot. Disable
000024  2F38 007C               move.l $7c.-(a7)       * alten Level 7 Autointerruot retten
000028  487A 002C               pea autoint(pc)        * Adresse merken
00002C  21DF 007C               move.l (a7)+.$7c       * und ablegen im Autointerruot
000030  11FC 00D0 FFC0          move.b #$d0.flo0       * Floppy-Controller ruecksetzen
000036  4A38 FFC0               tst.b flo0             * Interruots loeschen
00003A  48E7 78F0               movem.l d1-d4/a0-a3.-(a7)
00003E  6100 0144               bsr softex             * und Auffuehrung des Befehls
000042  4CDF 0F1E               movem.l (a7)+.d1-d4/a0-a3
000046  40ED 0000               move sr.srsto(a5)      * Status retten
00004A  21DF 007C               move.l (a7)+.$7c       * alter Autointerruot zurueck
00004E  46DF                    move (a7)+.sr          * Status zurueck. Int etc.
000050  44ED 0000               move srsto(a5).ccr     * aber nur Cond-Code.
000054  4E75                    rts                    * d0 ist Ergebnis 0=ok
                                *                      * ffff = Fehler. + carry bei Fehler
000056                          *
000056  4E73                    autoint:               * falls auftritt. ignorieren
                                rte
                                *
000058                          getfloo:               * Formatierung feststellen
000058  11FC 00D0 FFC0          move.b #$d0.flo0       * Floppy-Controller ruecksetzen
00005E  4A38 FFC0               tst.b flo0             * Interruots loeschen
                                *                      * d4=Laufwerkscode. ohne Dichtecode
                                *                      * anschliessend mit Dichtecode als
                                *                      * Ergebnis. sonst CARRY. wenn Format
                                *                      * nicht gefunden.
00062  C83C 008F               and.b #$8f.d4          * d1=Fehlercode Floppy Typ 1 (gaf. [illegible]te o
                                *                      * nur 0..15 erlaubt als Driveselec
                                *                      * und Sideselect
                                *                      * testet mini.sd  11
                                *                      *        mini.dd  10
                                *                      *        maxi.sd  01
                                *                      *        maxi.dd  00
                                *                      * Test durch Restore-Befehl mit min Steora
00066  6100 0070               bsr wawa               * Delay aus Sicherheitsgruenden
00066A 883C 0030               or.b #%00110000.d4     * mini sd
00066E 6100 002C               bsr getfloo
000672 6400 01C6               bcc noerr              * OK. kein Fehler MINI SD
000676 C83C 00AF               and.b #%10101111.d4    * mini dd
00067A 6100 0020               bsr getfloo
00067E 6400 01BA               bcc noerr              * MINI DD
000682 C83C 008F               and.b #%10001111.d4
000686 883C 0010               or.b  #%00010000.d4    * maxi sd
00068A 6100 0010               bsr getfloo
00068E 6400 01AA               bcc noerr              * MAXI SD
000692 C83C 008F               and.b #%10001111.d4    * maxi dd
```

```
000096  6100 0004       156.  bsr getlfloo            * nocarry (und 0=d0) kein Fehler
00009A  4E75            157.  rts                     * CARRY, Fehler z.B. NOT READY etc.
                        158.
                        159.
                        160.
00009C                  161. getlfloo:                * Unterprog. Restore mit
00009C  11C4 FFC4       162.  move.b d4.flo4          * quelligem Code
0000A0  11FC 000F       163.  move.b #$f.flo0         * RESTORE mit Verify und Headload. ohne SSO
        FFC0
0000A6  6100 005A       164.  bsr ready               * Warten bis Ready (INT gesetzt)
0000AA  1200            165.  move.b d0.d1            * Dort merken fuer Analyse
0000AC  C03C 0098       166.  and.b #%10011000.d0     * ready,seek,crc
0000B0  6600 017E       167.  one error
0000B4  6000 0184       168.  bra noerr
                        169.
                        170.
                        171. * Diverse Unterprogramme
                        172.
0000B8                  173. setup:           * Testet ob Floppykopf aufliegt
                        174.                  * und liefert SSO-Code, falls
                        175.                  * Bit 7 von STEPRATE=1
0000B8  4240            176.  clr d0          * 0=Ergebnis, wenn Kopf aufliegt
0000BA  082D 0007       177.  btst.b #7.steprate(a5) * =1, dann SSO setzen, freigegeben
        0000
0000[illegible]  6708   178.  beq.s setlup     * sonst auch Rueckseite = 0 beim Fruefen.
0000C2  0804 0007       179.  btst #7.d4       * Codierung Dichte,Laufwerk
0000C6  6702            180.  beq.s setlup     * Seite 0, dann ueberspringen
0000C8  7002            181.  moveq #2.d0      * Side Select Flag, beim 1797
                        182.                  * Damit wird auch die Formatierung auf Side 1
                        183.                  * geprueft, sonst Record not found err.
0000CA                  184. setlup:
0000CA  0838 0005       185.  btst.b #5.flo4 * =1 dann liegt Kopf auf
        FFC4
0000D0  6604            186.  bne.s noset
0000D2  807C 0004       187.  or #4.d0        * wenn nicht dann Maske 4 laden
0000D6                  188. noset:           * um Befehlscode fuer FDC vorzubereiten
0000D6  4E75            189.  rts
                        190.
0000D8                  191. wawa:
0000D8  303C 03E8       192.  move #1000.d0   * ca 3.25 ms
                        193.                  *
0000DC                  194. delay:           * fuer manche Befehle noetig
0000DC  4E71            195.  nop
0000DE  51C8 FFFC       196.  dbra d0.delay   * d0=Zaehlregister
0000E2  4E75            197.  rts             * Durchlauf =26 Zyklen beim 68008
                        198.                  * bei f=8MHZ 3.25 ys
                        199.
0000E4                  200. seek:            * Spur anfahren, D4=Laufwerk,D3=Spur,D2=Sektor
0000E4  61F2            201.  bsr wawa        * ein paar ms verzoegern
0000E6  11C4 FFC4       202.  move.b d4.flo4  * erst Laufwerkscode und Dichte einstellen
0000EA  11C3 FFC3       203.  move.b d3.flo3  * dann Spur, als Dateninformation
0000EE  11C2 FFC2       204.  move.b d2.flo2  * dann Sektor
0000F2  102D 0000       205.  move.b steprate(a5).d0
0000F6  C03C 0003       206.  and.b #3.d0     * ohne SSO bit, rev 4.2
0000FA  803C 001C       207.  or.b #$1c.d0    * dann Befehl  or Stepratencode
0000FE  11C0 FFC0       208.  move.b d0.flo0  * Als Befehl ausgeben
                        209.                  *
000102                  210. ready:           * Warten auf Befehlsende
000102  0838 0006      211.  btst.b #6.flo4 * Dann int-Bit gesetzt
        FFC4
000[illegible]  67F8    212.  beq.s ready
000[illegible]  1038 FFC0 213.  move.b flo0.d0 * Status einlesen und INT loeschen
00010E  4E75            214.  rts
                        215.
000110                  216. rstore:          * Kopf auf Spur 0 fahren
000110  61C6            217.  bsr wawa        * ein paar ms verzoegern
000112  11C4 FFC4       218.  move.b d4.flo4 * Dichte und Laufwerkscode einstellen
000116  102D 0000       219.  move.b steprate(a5).d0
00011A  C03C 0003       220.  and.b #3.d0     * ohne SSO Bit, rev 4.2
00011E  803C 000B       221.  or.b #8.d0      * Headload, aber kein Verify
000122  11C0 FFC0       222.  move.b d0.flo0 * und Befehl ausfuehren
000126  60DA            223.  bra ready       * und warten bis ausgefuehrt
                        224.
                        225. *
                        226. * Zeitkritische Routinen folgen nun
                        227. *
                        228.
000128                  229. rdflo:            * einen Sektor lesen, a0->Zieladresse
000128  11C4 FFC4       230.  move.b d4.flo4 * Dichte und Laufwerkscode ausgeben
00012C  11C2 FFC2       231.  move.b d2.flo2 * Sektor einstellen
```

```
00130   43F8 FFC0          232.   lea flo0.a1     * Adressen vorbelegen
00134   45F8 FFC4          233.   lea flo4.a2     * Status
00138   47F8 FFC3          234.   lea flo3.a3     * Daten
0013C   6100 FF7A          235.   bsr setup       * d0=Headload-Code + SSO-Code . rev 4.2
00140   D03C 0088          236.   add.b #$88.d0   * Lese-Befehl. sidesel=0
00144   1280               237.   move.b d0.(a1)  * Und Befehl ausgeben
00146                      238. rd1:              * Lese-Schleife
00146   1012               239.   move.b (a2).d0  * Status pruefen
00148   E318               240.   rol.b #1.d0     * und setzen der Flags
0014A   6B06               241.   bmi.s rdle      * Lesen beendet durch INT
0014C   64F8               242.   bcc.s rd1       * DRQ noch nicht da
0014E   10D3               243.   move.b (a3).(a0)+ * Daten ablegen wenn da
00150   60F4               244.   bra.s rd1       * bis alles gelesen
00152                      245. rdle:             * OK Lesen beendet
00152   1011               246.   move.b (a1).d0  * Status einlesen
00154   4E75               247.   rts             * und damit Ende
                           248.
00156                      249. wrflo:            * einen Sektor schreiben. a0->Quelladresse
00156   11C4 FFC4          250.   move.b d4.flo4  * Dichte und Laufwerkscode ausgeben
0015A   11C2 FFC2          251.   move.b d2.flo2  * Sektor einstellen
0015E   43F8 FFC0          252.   lea flo0.a1     * Adressen vorbelegen
00162   45F8 FFC4          253.   lea flo4.a2     * Status
00166   47F8 FFC3          254.   lea flo3.a3     * Daten
0016A   6100 FF4C          255.   bsr setup       * d0=Headload-Code und SSO-Code. rev 4.2
0016E   D03C 00A8          256.   add.b #$a8.d0   * Schreib-Befehl.
00172   1280               257.   move.b d0.(a1)  * Und Befehl ausgeben
00174                      258. wr1:
00174   1012               259.   move.b (a2).d0  * Status pruefen
00176   E318               260.   rol.b #1.d0     * und Flags setzen
00178   6B06               261.   bmi.s wrle      * INT. dann Schreiben beendet
0017A   64F8               262.   bcc.s wr1       * DRQ dann Anforderung neue Daten
0017C   1698               263.   move.b (a0)+.(a3) * Ja. dann Daten an Controller geben
0017E   60F4               264.   bra.s wr1       * und wieder warten
00180                      265. wrle:
00180   1011               266.   move.b (a1).d0  * Status einlesen
00182   4E75               267.   rts             * und damit Ende
                           268.
                           269. *
                           270. * Ende der zeitkritischen Routinen
                           271. *
00184                      272.
                           273. softex: * Kernroutine zur Befehlausfuehrung
                           274.         * a0->adresse
                           275.         * d1 Befehlscode 0=Steprate setzen.1=Lesen.2=Schreiben
                           276.         * d2 Sektor
                           277.         * d3 Track
                           278.         * d4 =Laufwerkscode+dense+seite
                           279.         *     entspricht dem Internformat von Port: flo4
                           280.         * 7     6       5       4       3       2       1       0
                           281.         * sso   mot     min     dense   d       c       b       a
                           282.         * 1=Seite 1
                           283.         *       0=Motor on
                           284.         *               1=Minilaufwerk
                           285.         *                       1=Single dense
                           286.         *                               1=Laufwerk an.
                           287.         *                               binaere Codierung moeglich
                           288.         *
                           289.         * Register D3 bestimmt die Steprate. wenn D1=0
                           290.         * Steprate setzen. 0 ist die schnellste Steprate
                           291.         * 3 die langsamste Steprate
                           292.         * wenn man zusaetzlich Bit 7 setzt. so wird bei all
                           293.         * Schreib-Lesezugriffen. das SSO-Bit im Controller
                           294.         * wenn man mit dem Drivecode (D4) auf die zweite Seite
                           295.         * zugreift. Dadurch kann man auch Formatierungen verwenden
                           296.         * die im Format das Side-Bit gesetzt haben. den der Contro
                           297.         * prueft dieses bei allen Zugriffen.
0184   4A01                298.   tst.b d1
0186   660A                299.   bne.s soft1     * Befehl=0. dann Steprate setzen
0188   C63C 0083           300.   and.b #$83.d3   * nur 0.1.2.3 als Code erlaubt. und Bit7=1 fuer SS
018C   1B43'0000           301.   move.b d3.steprate(a5) * in D3 ist die neue Steprate und SSO-Bit.
0190   4E75                302.   rts
0192                       303.
0192   422D'0000           304. soft1:
0196   1B44'0000           305.   clr.b fehza(a5)         * Keine Fehler bisher
019A   C87C 000F           306.   move.b d4.ccode(a5)     * Laufwerkscode merken
019E   102D'0000           307.   and.w #$f.d4            * Fuer Laufwerksvergleich 0..15
01A2   B03C 00FF           308.   move.b drvat(a5).d0     * Altes laufwerk
01A6   6712                309.   cmp.b #$ff.d0           * war es ungueltig
01A8   C07C 000F           310.   beq.s notdef            * dann notdef - Teil
                           311.   and.w #$f.d0            * 0..15
```

```
0001AC  B800            312.  cmp.b d0.d4               * wenn beide gleich
0001AE  6734            313.  beq.s nurseek             * dann nur Seek noetig ggf.
0001B0  43ED'0000       314.  lea drvtab(a5).a1         * Sonst Tabelle als Basisadresse
0001B4  13B8 FFC1       315.  move.b flo1.0(a1.d0.w)    * alte Sour dort ablegen. d0 ist wort
        0000
0001BA                  316. notdef:                    * altes Laufwerk war nicht da
0001BA  43ED'0000       317.  lea drvtab(a5).a1         * Tabelle als Basisadresse
0001BE  11F1 4000       318.  move.b 0(a1.d4.w).flo1    * aktuelle Spurnummer ablegen
        FFC1
0001C4  1B6D'0000       319.  move.b ccode(a5).drvat(a5) * ab sofort neues Laufwerk
        0000
0001CA  303C 039B       320.  move #923.d0              * ca 3ms warten
0001CE  6100 FF0C       321.  bsr delay                 * ca 3.25 ys
0001D2  11ED'0000       322.  move.b ccode(a5).flo4     * und dann neues Laufwerk ansprechen
        FFC4
0001D8  303C 2A12       323.  move #10770.d0            * ca 15ms..35ms warten
0001DC  6100 FEFE       324.  bsr delay                 * fuer Headload.Headsettle-Time
0001E0  6000 000A       325.  bra nur1                  * Immer auf Readv pruefen danach
0001E4                  326. nurseek:
0001E4  0838 0005       327.  btst.b #5.flo4            * Wenn Head down. kein dummy-Seek
        FFC4
0001EA  6610            328.  bne.s nodum
0001EC                  329. nur1:                      * sonst pruefen ob Laufwerk readv
0001EC  182D 0000       330.  move.b ccode(a5).d4       *
0001F0  6100 FEF2       331.  bsr seek                  * Selektiert als Dummy
0001F4  0838 0007       332.  btst.b #7.flo0            * Bis Readv erscheint
        FFC0
0001FA  66F0            333.  bne.s nur1                * Warten   ---- ----
0001FC                  334. nodum:                     * Wenn head down oder Readv
0001FC  0C38 00FF       335.  cmp.b #$ff.flo1           * Wenn Track undefiniert dann RSTORE
        FFC1
000202  6714            336.  beq.s trv1                * dort Restore
000204  B638 FFC1       337.  cmp.b flo1.d3             * wenn Trackflo=Trackakuell
000208  6738            338.  beq.s sk11                * dann kein Seek ausfuehren
00020A                  339. trv:                       * Seek ausfuehren
00020A  182D'0000       340.  move.b ccode(a5).d4       * Rest queltig
00020E  6100 FED4       341.  bsr seek                  * Ausfuehren von Seek
000212  C03C 0090       342.  and.b #%10010000.d0       * moegliche Fehler
000216  672A            343.  beq.s sk11                *=0 dann kein Fehler da
000218                  344. trv1:                      * sonst erst mal Restore ausfuehren
000218  182D'0000       345.  move.b ccode(a5).d4
00021C  6100 FEF2       346.  bsr rstore                * und Sour 0 anfahren
000220  102D'0000       347.  move.b fehza(a5).d0       * Fehlerzaehler veraendern
000224  5200            348.  addq.b #1.d0              * +1
000226  1B40'0000       349.  move.b d0.fehza(a5)
00022A  B03C 000A       350.  cmp.b #10.d0              * Max 10 Mal
00022E  65DA            351.  bcs.s trv                 * dann wieder SEEK ausfuehren
000230                  352. error:                     * FEHLER AUFGETRETEN
000230  303C FFFF       353.  move.w #$ffff.d0          * Error-Code
000234                  354. carset:
000234  003C 0001       355.  or.b #1.ccr
000238  4E75            356.  rts
                        357. *
00023A                  358. noerr:                     * AUSGANG KEIN FEHLER
00023A  4240            359.  clr d0
00023C                  360. carres:
00023C  023C 00FE       361.  and.b #$fe.ccr
000240  4E75            362.  rts
                        363. *
000242                  364. sk11:                      * R/W ausfuehren
000242  B23C 0002       365.  cmp.b #2.d1               * =2 dann schreiben
000246  6700 0032       366.  beq wrtex
00024A  182D'0000       367.  move.b ccode(a5).d4      * Code setzen
00024E  2F08            368.  move.l a0.-(a7)          * Ziel retten
000250  6100 FED6       369.  bsr rdflo                 * und lesen
000254  205F            370.  movea.l (a7)+.a0          *
000256  C03C 009C       371.  and.b #%10011100.d0       * Lost data.crc etc.
00025A  67DE            372.  beq noerr                 * =0 dann kein Fehler da
00025C  C03C 0010       373.  and.b #%00010000.d0       * Bei Record not found
000260  66B6            374.  bne trv1                  * Seek ausfuehren
000262  102D'0000       375.  move.b fehza(a5).d0       * Anzahl der Fehler
000266  5200            376.  addq.b #1.d0              * erhoehen
000268  1B40'0000       377.  move.b d0.fehza(a5)       * und merken
00026C  B03C 0005       378.  cmp.b #5.d0               * bis 5 Fehler
000270  65D0            379.  bcs.s sk11                * Befehl wiederholen
000272  B03C 000A       380.  cmp.b #10.d0              * dann Seek ausfuehren
000276  65A0            381.  bcs trv1                  * oder Rstore
000278  60B6            382.  bra error                 * dann FEHLER
                        383.
00027A                  384. wrtex:
```

```
00027A  182D'0000       385.  move.b ccode(a5).d4     * Code setzen
00027E  2F08            386.  move.l a0.-(a7)
000280  6100 FED4       387.  bsr wrflg
000284  205F            388.  movea.l (a7)+.a0       * Quelle retten
000286  C03C 00FC       389.  and.b #%11111100.d0    * Fehlercode
00028A  67AE            390.  beq noerr              *=0 dann fertig. kein Fehler da
00028C  C03C 0010       391.  and.b #%00010000.d0    * Record not found. dann P
000290  6686            392.  bne trv1               * Seek ausfuehren
000292  102D'0000       393.  move.b fehza(a5).d0    * Fehler Zaehler
000296  5200            394.  addq.b #1.d0
000298  1B40'0000       395.  move.b d0.fehza(a5)    * erhoehen
00029C  B03C 0004       396.  cmp.b #4.d0            * <4 dann SEEK versuchen
0002A0  6500 FF76       397.  bcs trv1
0002A4  608A            398.  bra error             * Sonst Fehlerabbruch
                        399.
0002A6                  400.  end
    0 Errors detected
```

```
...A68K 4.0a 06/13/83   ...Run on 01/22/85
                         1. ************************************
                         2. * Nuetzliche Unterprogramme. die  *
                         3. * fuer spezielle IO-Baugruppen    *
                         4. * vorhanden sind.                *
                         5. * Rev 4.1   840923  (C) 1984     *
                         6. * Muenchen Rolf-Dieter Klein     *
                         7. ************************************
                         8.
000000                   9.  section 1
                        10.
                        11.  include cou
                        12: ************************************
                        13: * CPU Include File   840930     *
                        14: * wird von GRUND.FLO2.UTIL.CRT  *
                        15: * benoetigt um IO-Adressen      *
                        16: * festzulegen                   *
                        17: ************************************
                        18:
00000001                19: cou     equ     1
                        20:
                        21: * 1= CPU 68008       8-Bit-Datenbus
                        22: * 2= CPU 68000/10   16-Bit-Datenbus
                        23: * 4= CPU 68020      32-Bit-Datenbus (32er Mode)
                        24:
                        25:
                        26: *
                        27: * Ende CPU include
                        28: * Achtung, bei Aendern der Datei, alle vier
                        29: * betroffenen Programme neu uebersetzen
                        30: *
                        31:
                        32:
                        33:
                        34:
                        35:
                        36.
                        37.
                        38.  xdef siinit.si.so.sists.sosts
                        39.  xdef getad8.getad10.setda
                        40.  xdef getuhr.setuhr
                        41.  xdef speak.speak1.sound
                        42.
                        43.
                        44.
                        45. *
                        46. * Serielle Schnittstelle mit Baugruppe SER
                        47. * der Baustein 6551 wird dabei verwendet
                        48. * somit kann jede Schnittstelle auf der
                        49. * angegebenen Adresse damit bedient werden.
                        50.
FFFFFFF0                51. ser0 equ cou*$fffffff0
FFFFFFF1                52. ser1 equ cou*$fffffff1
FFFFFFF2                53. ser2 equ cou*$fffffff2
FFFFFFF3                54. ser3 equ cou*$fffffff3
                        55.
                        56. * Initialisieren der SER-Baugruppe
                        57. * Control-Register:
                        58. * 7       6       5       4       3       2       1       0
                        59. * 0=1 Stop Bit
                        60. * 1=2 Stop Bits
                        61. * 1       Stop. wenn 8Bit Data+paritaet
                        62. * 1 1/2 Stop. 5Bits ohne Par
                        63. *         ---------
                        64. *         0       0 = 8 Datenbits
                        65. *         0       1 = 7 Datenbits
                        66. *         1       0 = 6 Datenbits
                        67. *         1       1 = 5 Datenbits
                        68. *                       0=externer Takt. 1=interner Takt
                        69. *                       -------------------------
                        70. *                               0000 16x externer Takt
                        71. *                               0001   50 Baud
                        72. *                               0010   75
                        73. *                               0011  109.92
                        74. *                               0100  134.58
                        75. *                               0101  150
                        76. *                               0110  300
                        77. *                               0111  600
                        78. *                               1000 1200
                        79. *                               1001 1800
```

```
                          *                                      1010 2400
                          *                                      1011 3600
                          *                                      1100 4800
                          *                                      1101 7200
                          *                                      1110 9600
                          *                                      1111 19200 Baud
                          *
                          * Command-Register:
                          * 7     6       5        4        3       2       1       0
                          * ---------------        !                        !       !
                          * --0 keine Paritaet    0=normal.1=Echo           !       0=-DTR auf
                          * 001 Odd Par Rec+Trans                           !       1=enable T
                          * 011 Even Par Rec+Trans                          !
                          * 101 Mark Par. Tran                              1=Int disable
                          * 111 Space Par. Tran
                          *                                 ---------
                          *                                 00 disable Int. -RTS High
                          *                                 01 enable  Int. -RTS Low
                          *                                 10 disable Int. -RTS Low
                          *                                 11 disable Int. -RTS Low. BREAK

                          * Beispiel: 9600 Baud. 8Bits. keine Paritaet
                          * D0 = %00011110 ($1E)  D1 = %00001011 ($B)
                          *
                          * D0=Control-Register
                          * D1=Command-Register
                          *

000000                    siinit:
000000  11C0 FFF3          move.b d0.ser3
000004  11C1 FFF2          move.b d1.ser2
000008  4E75               rts

                          *
                          * Daten werden in D0.B uebertragen
                          *

00000A                    si:
00000A  0838 0003          btst.b #3.ser1 * Pruefen ob ein Zeichen da ist
        FFF1
000010  67F8               beq.s si       * Nein. dann erneut versuchen
000012  4280               clr.l d0       * Auch Langwort sollte gueltig sein
000014  1038 FFF0          move.b ser0.d0 * und Datenbyte holen
000018  4E75               rts

                          *
                          * ACHTUNG Eingang C an SER muss auf +12V liegen.
                          * wenn er nicht benutzt wird.
                          * Man kann ihn z.B. mit RT verbinden
                          *

00001A                    so:             * D0.B ist das auszugebende Zeichen
00001A  0838 0006          btst.b #6.ser1 * Pruefen ob DSR bereit ist (+12V)
        FFF1
000020  66F8               bne.s so       * nein. dann warten
000022  0838 0004          btst.b #4.ser1 * Dann warten bis Zeichen ausgegeben
        FFF1
000028  67F0               beq.s so       * falls noch eines im Sendebuffer ist
00002A  11C0 FFF0          move.b d0.ser0 * sonst ausgeben
00002E  4E75               rts

                                          * ACHTUNG. Status nach unmittelbarer
                                          * SO-Ausgabe ungueltig
000030                    sists:          * D0.L =FFFFFFFF. wenn Zeichen da.
                                          * sonst =0
000030  4280               clr.l d0       * False annehmen
000032  0838 0003          btst.b #3.ser1
        FFF1
000038  6702               beq.s sisno
00003A  70FF               move.l #$ffffffff.d0
00003C                    sisno:
00003C  4A80               tst.l d0       * Flags setzen
00003E  4E75               rts

000040                    sosts:          * D0.L =FFFFFFFF. wenn Buffer leer.
                                          * und Status frei. sonst =0
000040  4280               clr.l d0       * False annehmen
000042  0838 0006          btst.b #6.ser1
        FFF1
```

```
000048  660A          155.  bne.s sosno     * DSR nicht auf +12V
00004A  0838 0004     156.  btst.b #4.ser1
        FFF1
000050  6702          157.  beq.s sosno     * Buffer noch voll
000052  70FF          158.  move.l #$ffffffff.d0
000054                159. sosno:
000054  4A80          160.  tst.l d0        * Flags setzen
000056  4E75          161.  rts
                      162.
                      163. *
                      164. * Analog/Digital-Umsetzer und Digital/Analog-Umsetzer
                      165. *
                      166. *
                      167.
      FFFFFFE0        168. adc0816 equ cpu*$ffffffe0
      FFFFFFFC        169. adc1001 equ cpu*$fffffffc
      FFFFFFF8        170. da0802  equ cpu*$fffffff8
                      171.
000058                172. getad8:          * D0.B =Kanalnummer 0..15
                      173.                  * D0.L = Ergebnis anschliessen
000058  48E7 4080     174.  movem.l d1/a0.-(a7) * keine Register veraendern
00005C  41F8 FFE0     175.  lea adc0816.a0  * Basisadresse laden
000060  323C 0001     176.  move #cpu.d1    * Multiplikator
000064  C07C 000F     177.  and.w #$F.d0
000068  B27C 0001     178.  cmp #1.d1       * wenn 68008
00006C  6702          179.  beq.s get0ad8   * ok. Data so lassen
00006E  C0C1          180.  mulu d1.d0      * Sonst Adresse neu berechnen
000070                181. get0ad8:
000070  1180 0000     182.  move.b d0.(a0.d0.w) * Start des Wandlers mit Dummy Wert
000074                183. get1ad8:
000074  1010          184.  move.b (a0).d0
000076  6AFC          185.  bpl.s get1ad8  * Warten bis gestartet
000078                186. get2ad8:
000078  1010          187.  move.b (a0).d0
00007A  68FC          188.  bmi.s get2ad8  * Warten bis gewandelt
00007C  4280          189.  clr.l d0        * Langwort quelltig
00007E  1028 0001     190.  move.b cpu*1(a0).d0  * 1.2.4 als Displacement
000082  4CDF 0101     191.  movem.l (a7)+.d1/a0
000086  4E75          192.  rts
                      193.
000088                194. getad10:         *
000088  11C0 FFFC     195.  move.b d0.adc1001  * Starten des Wandlers mit Dummy Wert
00008C  303C 0001     196.  move #cpu.d0    * 1=68008
000090  B07C 0001     197.  cmp #1.d0
000094  6718          198.  beq.s get0ad10
000096                199. get2ad10:
000096  0838 0007     200.  btst.b #7.adc1001
        FFFC
00009C  66F8          201.  bne.s get2ad10 * Warten bis gewandelt
00009E  4280          202.  clr.l d0
0000A0  1038 FFFC     203.  move.b adc1001.d0 * MSB
0000A4  E158          204.  rol.w #8.d0     * LSB
0000A6  1038 FFFD     205.  move.b adc1001+cpu.d0 * (1.)2.4
0000AA  4A40          206.  tst.w d0        * kompatibeler Status
0000AC  4E75          207.  rts             * und Ergebnis in D0.L
                      208.
0000AE                209. get0ad10:        * STD 68008
0000AE  4280          210.  clr.l d0
0000B0                211. get1ad10:
0000B0  3038 FFFC     212.  move adc1001.d0 * als Wort laden
0000B4  6BFA          213.  bmi.s get1ad10 * da Status zuerst ist Reihenfolge ok
0000B6  4E75          214.  rts             * Ergebnis fertig 10 Bit
                      215.
0000B8                216. setda:           * D1.B =DA Kanal 0. D2.B =DA Kana 1
0000B8  11C1 FFF8     217.  move.b d1.da0802   * kompat. zu x.v-Konvention
0000BC  11C2 FFF9     218.  move.b d2.da0802+cpu
0000C0  4E75          219.  rts
                      220.
                      221. *
                      222. * Bedienung der Baugruppe UHR
                      223. * Baustein E050-16
                      224. *
                      225.
      FFFFFFFE        226. uhr equ cpu*$fffffffe
                      227.
0000C2                228. wait:            * Verzoegerung
                      229.                  * fuer sicheren Betrieb
0000C2  2F00          230.  move.l d0.-(a7)
0000C4  303C 00C8     231.  move #200.d0    * Delay >>> 14 Mikrosekunden
0000C8                232. wi:
```

```
0000C8  4E71
0000CA  51C8 FFFC
0000CE  201F
0000D0  4E75

0000D2
0000D2  11FC 0004
        FFFE
0000D8  61E8
0000DA  11FC 0006
        FFFE
0000E0  1038 FFFE
0000E4  61DC
0000E6  4E75

0000E8
0000E8  11FC 0004
        FFFE
0000EE  C03C 0001
0000F2  803C 0004
0000F6  11C0 FFFE
0000FA  61C6
0000FC  803C 0002
000100  11C0 FFFE
000104  61BC
000106  C03C 00FE
00010A  11C0 FFFE
00010E  4E75

000110

000110  48E7 FC00
000114  363C 0003
000118  11FC 0004
        FFFE
00011E
00011E  4240
000120  61C6
000122  51CB FFFA
000126  363C 0006
00012A
00012A  383C 0007
00012E  4245
000130
000130  61A0
000132  C03C 0001
000136  8A00
000138  E21D
00013A  51CC FFF4
00013E  10C5
000140  51CB FFE8
000144  4238 FFFE
000148  4CDF 003F
00014C  4E75

00014E

00014E  48E7 FC00
000152  11FC 0004
        FFFE
000158  363C 0002
00015C
00015C  4240
00015E  6188
000160  51CB FFFA
000164  303C 0001
000168  6100 FF7E
00016C  363C 0006
000170
000170  383C 0007
```

```
233.  nop
234.  dbra d0.w1
235.  move.l (a7)+.d0
236.  rts
237.
238. pulsrd:              * Lesen ein Bit
239.  move.b #4.uhr      * CS setzen

240.  bsr wait
241.  move.b #6.uhr      * CS und CLK

242.  move.b uhr.d0      * in D0 ist Datenwert
243.  bsr wait
244.  rts
245.
246. pulswr:              * Schreiben LSB D0
247.  move.b #4.uhr      * CS setzen

248.  and.b #1.d0        * Savety
249.  or.b #4.d0         * CS und Data
250.  move.b d0.uhr      * ausgeben
251.  bsr wait
252.  or.b #2.d0         * CS und CLK
253.  move.b d0.uhr      * ausgeben
254.  bsr wait
255.  and.b #$fe.d0      * Data neutral
256.  move.b d0.uhr
257.  rts
258.
259. *
260. * Datenbuffer Beispiel (BCD - Notation beachten):
261. * Stunden Minuten Datum Monat Jahr Tag Sekunde
262. *  $18     $15     $23   $09   $84  $01  $00
263. *
264. *  dc.b $18.$15.$23.$09.$84.$01.$00
265. *
266.
267. getuhr:              * A0 zeigt auf den Datenbuffer
268.                      * Uhrzeit lesen nach Buffer
269.  movem.l d0-d5.-(a7)
270.  move #4-1.d3
271.  move.b #4.uhr      * CS setzen

272. rd1:                * Viermal 1 (0 inv ) ausgeben
273.  clr d0
274.  bsr pulswr         * als Lesebefehl
275.  dbra d3.rd1
276.  move #7-1.d3       * Dann Daten lesen
277. lom:
278.  move #8-1.d4       * Datenbits
279.  clr d5             * ergebnis
280. lo2:
281.  bsr pulsrd
282.  and.b #1.d0
283.  or.b d0.d5
284.  ror.b #1.d5        * Einzelbits einlesen
285.  dbra d4.lo2
286.  move.b d5.(a0)+          * Byte ablegen
287.  dbra d3.lom
288.  clr.b uhr          * CS ausschalten
289.  movem.l (a7)+.d0-d5
290.  rts
291.
292. *
293. *
294. setuhr:              * A0 zeigt auf den Datenbuffer
295.                      * Die Uhr wird gestellt
296.  movem.l d0-d5.-(a7)
297.  move.b #4.uhr      * CS setzen

298.  move #3-1.d3       * Start Info
299. lo11:
300.  clr d0
301.  bsr pulswr
302.  dbra d3.lo11       * 000 ausgeben
303.  move #1.d0         * 1 danach als Kennung
304.  bsr pulswr
305.  move #7-1.d3       * Anzahl bytes
306. lomm:
307.  move #8-1.d4       * Dann Rest ausgeben
```

```
000174  1A18                308.  move.b (a0)+,d5 * Data holen
000176                      309. lo22:
000176  E319                310.  rol.b #1,d5     * Einzelbits ausgeben
000178  1005                311.  move.b d5,d0
00017A  0A00 0001           312.  eor.b #1,d0     * Komplement bilden
00017E  C03C 0001           313.  and.b #1,d0     * bei Ausgabe
000182  6100 FFE4          314.  bsr pulswr
000186  51CC FFEE          315.  dbra d4,lo22    * Alle Bits
00018A  51CB FFE4          316.  dbra d3,lomm    * Alle Bytes
00018E  4238 FFFE          317.  clr.b uhr       * und CS neutral stellen
000192  4CDF 003F          318.  movem.l (a7)+,d0-d5
000196  4E75               319.  rts
                            320.
                            321. *
                            322. * Baugruppe SPRACHE mit dem Baustein SSI 263
                            323. * einem Phonemsynthesizer
                            324. *
                            325.
        FFFFFFD8            326. spra0 equ cpu*$ffffffd8
        FFFFFFD9            327. spra1 equ cpu*$ffffffd9
        FFFFFFDA            328. spra2 equ cpu*$ffffffda
        FFFFFFDB            329. spra3 equ cpu*$ffffffdb
        FFFFFFDC            330. spra4 equ cpu*$ffffffdc
                            331.
0001[illegible]8            332. speak:           * a0-> Datenblock
0001[illegible]B  48E7 8080 333.  movem.l d0/a0,-(a7)
                            334.                  * mit 5 Parametern, allen Optionen
                            335.                  * erster Datenwert ist Anzahl.W (1..n)
                            336.                  * dc.w anzahl
                            337.                  * dc.b 1,2,3,4,5
                            338.                  * dc.b 1,2,3,4,5
0001[illegible]C  11FC 0000 339.  move.b #$00,spra0      * Stop
          FFD8
0001A2  11FC 0050           340.  move.b #$50,spra1      * Defaultwerte fuer Sinnvolle Sprache
          FFD9
0001A8  11FC 00A8           341.  move.b #$a8,spra2
          FFDA
0001AE  11FC 005C           342.  move.b #$5c,spra3
          FFDB
0001B4  11FC 00E9           343.  move.b #$e9,spra4
          FFDC
0001BA  3018                344.  move.w (a0)+,d0 * Anzahl laden
0001BC  5340                345.  sub.w #1,d0     * wegen DBRA
0001BE                      346. spk0lo:
0001BE  0838 0007           347.  btst.b #7,spra0
          FFD8
0001C4  67F8                348.  beq.s spk0lo    * =1 dann fertig
0001C6  11D8 FFD8           349.  move.b (a0)+,spra0
0001CA  11D8 FFD9           350.  move.b (a0)+,spra1
0001CE  11D8 FFDA           351.  move.b (a0)+,spra2
0001D2  11D8 FFDB           352.  move.b (a0)+,spra3
0001D6  11D8 FFDC           353.  move.b (a0)+,spra4
0001DA  51C8 FFE2           354.  dbra d0,spk0lo
0001DE  4CDF 0101           355.  movem.l (a7)+,d0/a0
0001E2  4E75                356.  rts
                            357.
0001E4                      358. speak1:          * a0-> Datenblock mit 1 Parameter
0001E4  48E7 8080           359.  movem.l d0/a0,-(a7)
                            360.                  * dem Phonemcode
                            361.                  * erster Datenwert ist Anzahl.W (1..n)
                            362.                  * dann Codes
                            363.                  * dc.w anzahl
                            364.                  * dc.b 1,1,1,1,1,1,1,1,1,1,1,1
0001E8  11FC 0000           365.  move.b #$00,spra0      * Stop
          FFD8
0001EE  11FC 0050           366.  move.b #$50,spra1      * Defaultwerte fuer Sinnvolle Sprache
          FFD9
0001F4  11FC 00A8           367.  move.b #$a8,spra2
          FFDA
0001FA  11FC 005C           368.  move.b #$5c,spra3
          FFDB
000200  11FC 00E9           369.  move.b #$e9,spra4
          FFDC
000206  3018                370.  move.w (a0)+,d0 * Anzahl laden
000208  5340                371.  sub.w #1,d0     * wegen DBRA
00020A                      372. sok1lo:
00020A  0838 0007           373.  btst.b #7,spra0 * =1 dann fertig
          FFD8
000210  67F8                374.  beq.s sok1lo
000212  11D8 FFD8           375.  move.b (a0)+,spra0
```

```
000216  51C8 FFF2          376.  dbra d0.sqklp
00021A  4CDF 0101          377.  movem.l (a7)+.d0/a0
00021E  4E75               378.  rts
                           379.
                           380.
                           381. *
                           382. * Baugruppe SOUND
                           383. * Baustein AY-3-8912
                           384. *
                           385.
      FFFFFF40             386. snd0 equ cou*$ffffff40  * ADR-Merker  Basisadresse SOUND
      FFFFFF41             387. snd1 equ cou*$ffffff41  * DATA
                           388.
000220                     389. sound:           * a0 zeigt auf die Parametertabelle
                           390.                  * dc.b 0.1.2.3.4.5.6.7.8.9.10.11.12.13.14.15
000220  48E7 C080          391.  movem.l d0/d1/a0.-(a7)
000224  303C 000F          392.  move #16-1.d0   * Anzahl der Parameter
000228  4241               393.  clr d1          * Kanal Nummer
00022A                     394. sould:
00022A  11C1 FF40          395.  move.b d1.snd0     * ausgeben Nummer
00022E  11D8 FF41          396.  move.b (a0)+.snd1  * und Datenwert
000232  5241               397.  addq.w #1.d1       * naechster Kanal
000234  51C8 FFF4          398.  dbra d0.sould
000238  4CDF 0103          399.  movem.l (a7)+.d0/d1/a0
00023C  4E75               400.  rts
                           401.
                           402.
                           403.
                           404.
00023E                     405.  end
    0 Errors detected
```

SOUND

```
...A68K 4.0a 06/13/83   ...Run on 01/22/85
                         1. ****************************************
                         2. * Editor Modul  fuer Grundprogramm     *
                         3. * Rolf-Dieter Klein 830912     1.0     *
                         4. * Bildschirm - Texteditor             *
                         5. * wird mit Linker gebunden            *
                         6. * rev 2.0    RDK   831203             *
                         7. *            RDK   831211             *
                         8. * rev 3.0    RDK   840129 Erweiterungen *
                         9. * rev 3.1    RDK   840228   --        *
                        10. * rev 4.0    RDK   840903 Shift.Ctrl bei*
                        11. *                        zweit Buchstb.*
                        12. * rev 4.0b   RDK 840909 relokativ      *
                        13. * rev 4.1    RDK 840924 Blockbefehle   *
                        14. *            und Zweisprachenfaehig    *
                        15. * rev 4.2    841016 /4.3 850117        *
                        16. ****************************************
                        17.
                        18.
                        19. * INSL   mode-Flag
                        20. *  7      6      5      4      3      2      1      0
                        21. *                                                 insert=1
                        22. *                                          1=Blockanf markiert
                        23. *                                   1=Blockende markiert
                        24. *                            1=Block Einfuegen
                        25.
                        26.          xref cirscreen.ci.co.csts.calccur
                        27.          xref delline.insline.delchar.inschar.charoscr.eraeoln
                        28.          xref clrall.printbx.textein.textaus.moveto.drawto
                        29.          xref setpen.erapen.aktpage
                        30.          xref curon.flip.flipl.wrtpage.viewpage
                        31.          xref carset.carres
                        32.          xref einbuf.ausbuf
                        33.          xref stxtxt.akttxt.etxtxt.eottxt
                        34.          xref insl.curx.cury
                        35.          xref eotflag
                        36.          xref clrinvis
                        37.          xref prtline.aktcur.iqbn.exor
                        38.
                        39.          xdef edit
                        40.
000000                  41.          section 1
                        42.
                        43.
                        44.                              * Ausbuf. Ausbuf+40 hat Doppeltfunktion
                        45.                              * Einbuf fuer Suche
                        46.                              * Ausbuf fuer Ersetze und Statuszeile
                        47.
000000                  48. edit:                        * start Editieren
                        49.                              * STXTXT zeigt auf den Textanfang
                        50.                              * AKTTXT zeigt auf die Startzeile
                        51.                              * ETXTXT zeigt auf das Textende
                        52.                              * EOTTXT zeigt auf das Ende aller Texte
000000  422D 0000       53.          clr.b einbuf(a5)            * Flag fuer CTRL-L
000004  422D 0000       54.          clr.b ausbuf(a5)            * Flag fuer CTRL-L
000008  422D 0000       55.          clr.b curon(a5)
00000C  6100 FFF2       56.          bsr clrscreen               * Bildschirm loeschen und flip ei
000010  422D 0000       57.          clr.b insl(a5)                    * kein Insert-Mode = 0
000014  2B6D 0000       58.          move.l stxtxt(a5),akttxt(a5)      * auf Anfang bringen
        0000
0000    6100 10CA       59.          bsr getscreen    * hole schirmbild
                        60.                           * Buffer holen
00001E                  61. editlo:                   * Schleife EDITIEREN
00001E  6100 0DB2       62.          bsr crtstatus    * immer ausgeben
000022  6100 FFDC       63.          bsr ci           * zeichen holen dann Analyse
000026  B03C 007F       64.          cmp.b #$7f,d0
00002A  6700 0D32       65.          beq editdel      * loeschen mit DEL
00002E  B03C 0020       66.          cmp.b #' ',d0    * < leerzeichen dann Steuerbefehl
000032  6516            67.          bcs.s editl
000034  082D 0000       68.          btst.b #0,insl(a5)          * bei Insertmode zuerst einfuegen
        0000
00003A  6708            69.          beq.s editl2                * Bit 0
00003C  3F00            70.          move d0,-(a7)
00003E  6100 FFC0       71.          bsr inschar      * zeichen einfuegen davor
000042  301F            72.          move (a7)+,d0
000044  6100 FFBA       73. editl2:  bsr charoscr     * Zeichen ausgeben ohne scroll last line
000048  60D4            74.          bra editlo
                        75.
00004A                  76. editl:                    * Steuerzeichen
                        77.                           * ueber Tabelle suchen, die zwei Eintraeg
```

```
                                                78.                             * enthaelt   wert1 wert2 adresse
00004A  41FA 0018                               79.             lea edittab(pc).a0      * fuer Vergleich
00004E                                          80. edit21:
00004E  4A10                                    81.             tst.b (a0)
000050  67CC                                    82.             beq editlo      * Nicht gefunden Ignorieren
000052  B010                                    83.             cmp.b (a0).d0   * wenn gleich ok
000054  6704                                    84.             beq.s edit22    * weiter ggf. Zweiter Teil
000056  5888                                    85.             addq.l #4.a0    * Naechster eintrag rev 4.0b
000058  60F4                                    86.             bra edit21
                                                87.
00005A                                          88. edit22:
00005A  43FA 0008                               89.             lea edittab(pc).a1
00005E  D2E8 0002                               90.             adda.w 2(a0).a1 * plus displacement
000062  4ED1                                    91.             jmp (a1)        * und anspringen
                                                92.
                                                93.
000064                                          94. edittab:                    * 0.0 = ende
000064  01 00                                   95.             dc.b $1.0       * CTRL-A  Wort zurueck
000066  0300                                    96.             dc.w edita-edittab
000068  03 00                                   97.             dc.b $3.0
00006A  0C1A                                    98.             dc.w editctc-edittab
00006C  04 00                                   99.             dc.b $4.0
00006E  0C2E                                   100.             dc.w editctd-edittab
000070  05 00                                   101.             dc.b $5.0
000072  0C52                                   102.             dc.w editcte-edittab
000074  06 00                                   103.             dc.b $6.0
000076  0330                                   104.             dc.w editf-edittab      * CTRL-F Wort vor
000078  07 00                                   105.             dc.b $7.0
00007A  0C70                                   106.             dc.w editctg-edittab
00007C  08 00                                  107.             dc.b 8.0        * BS
00007E  0B94                                   108.             dc.w editbs-edittab     * Zeichen nach links
000080  09 00                                   109.             dc.b 9.0
000082  0B0E                                   110.             dc.w edittb-edittab     * Tabulator
000084  0A 00                                   111.             dc.b $a.0
000086  0060                                   112.             dc.w edithelp-edittab   * CTRL-J ist Help
000088  0B 00                                   113.             dc.b $b.0       * CTRL K
00008A  0618                                   114.             dc.w editk-edittab      * spez Umschaltmodee
00008C  0C 00                                  115.             dc.b $c.0       * CTRL-L
00008E  03BA                                   116.             dc.w editl-edittab
000090  0D 00                                  117.             dc.b $d.0
000092  0D1A                                   118.             dc.w editcr-edittab
                                               119.             * CR ausgeben
000094  0E 00                                  120.             dc.b $e.0
000096  0C78                                   121.             dc.w editctn-edittab
000098  10 00                                  122.             dc.b $10.0
00009A  03A8                                   123.             dc.w edito-edittab      * CRTL-P
00009C  11 00                                  124.             dc.b $11.0
00009E  0B34                                   125.             dc.w edito-edittab      * CTRL Q
0000A0  12 00                                  126.             dc.b $12.0
0000A2  0C84                                   127.             dc.w editctr-edittab
0000A4  13 00                                  128.             dc.b $13.0
0000A6  0B94                                   129.             dc.w editbs-edittab
0000A8  14 00                                  130.             dc.b $14.0
0000AA  0C98                                   131.             dc.w editctt-edittab
0000AC  16 00                                  132.             dc.b $16.0      * CTRL V
0000AE  0D10                                   133.             dc.w editinsl-edittab   * insl umschalten bit 0 Insert-Mod
0000B0  17 00                                  134.             dc.b $17.0
0000B2  0CA0                                   135.             dc.w editctw-edittab
0000B4  18 00                                  136.             dc.b $18.0
0000B6  0CB6                                   137.             dc.w editctx-edittab
0000B8  19 00                                  138.             dc.b $19.0
0000BA  0CD6                                   139.             dc.w editcty-edittab
0000BC  1A 00                                  140.             dc.b $1A.0
0000BE  0CE6                                   141.             dc.w editctz-edittab
0000C0  00 00                                  142.             dc.b 0.0
0000C2  0000                                   143.             dc.w 0
                                               144.
                                               145.
0000C4                                         146. edithelp:                  * Hilfefunktion
                                               147.                            * Auf Seite 3.3
0000C4  43FA 005E                              148.             lea txtind(pc).a1
0000C8                                         149. editmno:
0000C8  1B7C 0003                              150.             move.b #3.wrtpage(a5)
        0000
0000CE  1B7C 0003                              151.             move.b #3.viewpage(a5)
        0000
0000D4  6100 FF2A                              152.             bsr aktpage
0000D8  6100 FF26                              153.             bsr clrinvis    * Immer vorher loeschen
0000DC  343C 00EB                              154.             move #235.d2    * 20=delta. start v
0000E0                                         155. editlno:
```

```
0000E0  4280                    156.          clr.l d0          * MSB=0 setzen
0000E2  3019                    157.          move.w (a1)+,d0
0000E4  6700 0024               158.          beq edithl1
0000E8  41FA 003A               159.          lea txtind(pc).a0
0000EC  D1C0                    160.          adda.l d0,a0      * Adress Start absolut
0000EE  303C 0022               161.          move #$22,d0
0000F2  4241                    162.          clr d1            * x:=0
0000F4  6100 FF0A               163.          bsr textaus
0000F8  947C 0014               164.          sub #20,d2
0000FC  6AE2                    165.          bpl editlho       * Wenn <0 dann neue Seite noetig
0000FE                          166. edit2ho:
0000FE  6100 FF00               167.          bsr csts
000102  67FA                    168.          beq edit2ho       * nun Cursor einlesen dann neu
000104  6100 FEFA               169.          bsr ci            * Zeichen holen
000108  60BE                    170.          bra editmho       * neues al gueltig
                                171.
00010A                          172. edithl1:
00010A  6100 FEF4               173.          bsr csts
00010E  67FA                    174.          beq edithl1       * ok wenn Taste gedrueckt
000110  422D 0000               175.          clr.b wrtpage(a5)
000114  422D 0000               176.          clr.b viewpage(a5)
000118  6100 FEE6               177.          bsr aktpage
00011C  6100 FEE2               178.          bsr ci            * Zeichen wegwerfen
000120  6000 FEFC               179.          bra editlo
                                180.
                                181.
000124                          182. txtind:                    * Adresstabelle. 0=ende
000124  002E                    183.          dc.w txtl1-txtind
000126  0039                    184.          dc.w txtl2-txtind
000128  005B                    185.          dc.w txtl3-txtind
00012A  007F                    186.          dc.w txtl4-txtind
00012C  00A0                    187.          dc.w txtl5-txtind
00012E  00BE                    188.          dc.w txtl6-txtind
000130  00E3                    189.          dc.w txtl7-txtind
000132  00F6                    190.          dc.w txtl8-txtind
000134  0115                    191.          dc.w txtl9-txtind
000136  0136                    192.          dc.w txtl10-txtind
000138  015A                    193.          dc.w txtl11-txtind
00013A  0179                    194.          dc.w txtl12-txtind
00013C  0191                    195.          dc.w txtl13-txtind
00013E  019C                    196.          dc.w txtl14-txtind
000140  01B2                    197.          dc.w txtl15-txtind
000142  01B4                    198.          dc.w txtl16-txtind
000144  01BD                    199.          dc.w txtl17-txtind
000146  01BF                    200.          dc.w txtl18-txtind
000148  01DE                    201.          dc.w txtl19-txtind
00014A  01F1                    202.          dc.w txtl20-txtind
00014C  020E                    203.          dc.w txtl21-txtind
00014E  022C                    204.          dc.w txtl22-txtind
000150  0000                    205.          dc.w 0
                                206.
                                207.
                                208. *
                                209. * Textblock fuer Hilfefunktion
                                210. *
                                211.
000152  52 45 56 34             212. txtl1:   dc.b 'REV4.3 -1-'.0
        2E 33 20 2D 31
        2D 00
00    0 5E 45 3D 5A             213. txtl2:   dc.b '^E=Zeile hoch      ^X=Zeile runter'.0
        65 69 6C 65 20
        68 6F 63 68 20
        20 20 20 20 5E
        58 3D 5A 65 69
        6C 65 20 72 75
        6E 74 65 72 00
00017F  5E 53 3D 5A             214. txtl3:   dc.b '^S=Zeichen links  ^D=Zeichen rechts'.0
        65 69 63 68 65
        6E 20 6C 69 6E
        6B 73 20 20 5E
        44 3D 5A 65 69
        63 68 65 6E 20
        72 65 63 68 74
        73 00
0001A3  5E 41 3D 57             215. txtl4:   dc.b '^A=Wort links      ^F=Wort rechts'.0
        6F 72 74 20 6C
        69 6E 6B 73 20
        20 20 20 20 5E
        46 3D 57 6F 72
        74 20 72 65 63
```

```
        68 74 73 00
0001C4  5E 5A 3D 5A      216. txtl5:  dc.b '^Z=Zeile auf      ^W=Zeile ab'.0
        65 69 6C 65 20
        61 75 66 20 20
        20 20 20 20 5E
        57 3D 5A 65 69
        6C 65 20 61 62
        00
0001E2  5E 59 3D 5A      217. txtl6:  dc.b '^Y=Zeile loeschen ^N=Zeile einfuegen'.0
        65 69 6C 65 20
        6C 6F 65 73 63
        68 65 6E 20 5E
        4E 3D 5A 65 69
        6C 65 20 65 69
        6E 66 75 65 67
        65 6E 00
000207  5E 54 3D 4C      218. txtl7:  dc.b '^T=Loeschen rechts'.0
        6F 65 73 63 68
        65 6E 20 72 65
        63 68 74 73 00
00021A  5E 52 3D 53      219. txtl8:  dc.b '^R=Seite zurueck  ^C=Seite vor'.0
        65 69 74 65 20
        7A 75 72 75 65
        63 6B 20 20 5E
        43 3D 53 65 69
        74 65 20 76 6F
        72 00
000239  5E 56 3D 45      220. txtl9:  dc.b '^V=Einfuegemode   ^P=Zeichensatz'.0
        69 6E 66 75 65
        67 65 6D 6F 64
        65 20 20 20 5E
        50 3D 5A 65 69
        63 68 65 6E 73
        61 74 7A 00
00025A  5E 51 43 3D      221. txtl10: dc.b '^QC=an Textende   ^QR=an Textanfang'.0
        61 6E 20 54 65
        78 74 65 6E 64
        65 20 20 20 5E
        51 52 3D 61 6E
        20 54 65 78 74
        61 6E 66 61 6E
        67 00
00027E  5E 51 46 3D      222. txtl11: dc.b '^QF=suchen        ^QA=ersetzen'.0
        73 75 63 68 65
        6E 20 20 20 20
        20 20 20 20 5E
        51 41 3D 65 72
        73 65 74 7A 65
        6E 00
00029D  5E 47 2C 44      223. txtl12: dc.b '^G.DEL=Zeichen loeschen'.0
        45 4C 3D 5A 65
        69 63 68 65 6E
        20 6C 6F 65 73
        63 68 65 6E 00
0002B5  52 45 56 34      224. txtl13: dc.b 'REV4.3 -2-'.0
        2E 33 20 2D 32
        2D 00
0002C0  5E 4C 3D 77      225. txtl14: dc.b '^L=wiederhole ^QF.^QA'.0
        69 65 64 65 72
        68 6F 6C 65 20
        5E 51 46 2C 5E
        51 41 00
0002D6  20 00            226. txtl15: dc.b ' '.0
0002D8  5E 4B 58 3D      227. txtl16: dc.b '^KX=ENDE'.0
        45 4E 44 45 00
0002E1  20 00            228. txtl17: dc.b ' '.0
0002E3  5E 4B 42 3D      229. txtl18: dc.b '^KB=Block Anf.  ^KK=Block Ende'.0
        42 6C 6F 63 6B
        20 41 6E 66 2E
        20 20 5E 4B 4B
        3D 42 6C 6F 63
        6B 20 45 6E 64
        65 00
000302  5E 4B 59 3D      230. txtl19: dc.b '^KY=Block loeschen'.0
        42 6C 6F 63 6B
        20 6C 6F 65 73
        63 68 65 6E 00
000315  5E 4B 56 3D      231. txtl20: dc.b '^KV=verschieben ^KC=kopieren'.0
        76 65 72 73 63
        68 69 65 62 65
```

```
                6E 20 5E 4B 43
                3D 6B 6F 70 69
                65 72 65 6E 00
000332  5E 4B 52 3D          232. txt121: dc.b '^KR=lesen        ^KW=speichern',0
                6C 65 73 65 6E
                20 20 20 20 20
                20 20 5E 4B 57
                3D 73 70 65 69
                63 68 65 72 6E
                00
000350  5E 4B 48 3D          233. txt122: dc.b '^KH=Marken loeschen',0
                4D 61 72 6B 65
                6E 20 6C 6F 65
                73 63 68 65 6E
                00
000364                       234.         ds 0             * Textangleich
                             235.
                             236. ***
                             237.
                             238.
000364                       239. edita:                   * Wort zurueck
000364  4A2D'0000            240.         tst.b curx(a5)
000368  6700 089C            241.         beq editlbs      * wie BS behandeln wenn Zeilenstart
00036C  532D'0000            242.         subq.b #1,curx(a5)      * mind eines zurueck
000370  6100'FC8E            243.         bsr calccur
000374  0C12 0020            244.         cmp.b #' ',(a2)
000378  67EA                 245.         beq edita        * Suchen nach Leerzeichen
00037A                       246. edita1:
00037A  4A2D'0000            247.         tst.b curx(a5)
00037E  6700 FC9E            248.         beq editlo       * fertig
000382  532D'0000            249.         subq.b #1,curx(a5)      * mind eines zurueck
000386  6100'FC78            250.         bsr calccur
00038A  0C12 0020            251.         cmp.b #' ',(a2)
00038E  66EA                 252.         bne edita1       * Suchen nach Leerzeichen
000390  6000 0900            253.         bra editctd      * Dann wieder eins vor
                             254.
                             255.
000394                       256. editf:                   * Wort vor
000394  6100 FC6A            257.         bsr calccur
000398  522D'0000            258.         addq.b #1,curx(a5)
00039C  102D'0000            259.         move.b curx(a5),d0      * wenn am Zeilenende dann weitersc
0003A0  B011                 260.         cmp.b (a1),d0
0003A2  6200 090A            261.         bhi editctd      * Wie CTRL-D am Zeilenende
0003A6  B03C 0050            262.         cmp.b #80,d0
0003AA  6700 0902            263.         beq editctd      * Zeilenende dann naechste Zeile
0003AE  601A                 264.         bra.s editf01
0003B0                       265. editf0:
0003B0  6100'FC4E            266.         bsr calccur
0003B4  522D'0000            267.         addq.b #1,curx(a5)
0003B8  102D'0000            268.         move.b curx(a5),d0      * wenn am Zeilenende dann weitersc
0003BC  B011                 269.         cmp.b (a1),d0
0003BE  6200 0044            270.         bhi editf2       * Stoppen dort
0003C2  B03C 0050            271.         cmp.b #80,d0
0003C6  6700 003C            272.         beq editf2       * Zeilenende
                             273.
0003CA                       274. editf01:
0003CA  6100'FC34            275.         bsr calccur
0003CE  0C12 0020            276.         cmp.b #' ',(a2)
0003D2  67DC                 277.         beq editf0
0003D4  0C2A 0020            278.         cmp.b #' ',-1(a2)
        FFFF
0003[illegible] 6700 FC42    279.         beq editlo       * gefunden
                             280.
0003DE                       281. editf1:
0003DE  6100'FC20            282.         bsr calccur
0003E2  522D'0000            283.         addq.b #1,curx(a5)
0003E6  102D'0000            284.         move.b curx(a5),d0      * wenn am Zeilenende dann weiters
0003EA  B011                 285.         cmp.b (a1),d0
0003EC  6200 0016            286.         bhi editf2       * Zeilenende erreicht ok
0003F0  B03C 0050            287.         cmp.b #80,d0
0003F4  6700 000E            288.         beq editf2       * auch bei Zeilenende
0003F8  6100'FC06            289.         bsr calccur
0003FC  0C12 0020            290.         cmp.b #' ',(a2)
000400  66DC                 291.         bne editf1       * Bis max Zeilenende
000402  60AC                 292.         bra editf0       * wieder zurueck
                             293.
000404  6100 0D5E            294. editf2: bsr adjcurx      * AAAAA am Zeilenende
000408  6000 FC14            295.         bra editlo
                             296.
                             297.
```

```
                                298.
00040C                          299. edito:                     * hier nur flip fuer ootflag
00040C  102D'0000               300.            move.b ootflag(a5),d0
000410  5200                   301.            add.b #1.d0     * ggf Zaehler spaeter Mal
000412  C03C 0001               302.            and.b #$1.d0
000416  1B40'0000               303.            move.b d0,ootflag(a5)
00041A  6000 FC02               304.            bra editlp
                                305.
00041E                          306. editl:
00041E  4A2D'0000               307.            tst.b einbuf(a5)
000422  6700 FBFA               308.            beq editlp      * Undef
000426  4A2D'0000               309.            tst.b ausbuf(a5)
00042A  6700 0054               310.            beq editofsu    * suchen
00042E  6000 01BA               311.            bra editoasu    * ersetzen
                                312.
                                313.
                                314.
                                315.
000432                          316. editof:                    * Suchen eines Namens
000432  422D'0000               317.            clr.b einbuf(a5)
000436  422D'0000               318.            clr.b ausbuf(a5)
00043A  1B7C 0003               319.            move.b #3.viewpage(a5)
        '0000
000440  1B7C 0003               320.            move.b #3.wrtpage(a5)
        '0000
000446  6100'FBB8               321.            bsr aktpage
00044A  6100'FBB4               322.            bsr clrinvis
00044E  41FA 021C               323.            lea sutxt(pc).a0
000452  303C 0022               324.            move #$22.d0
000456  323C 0002               325.            move #2.d1
00045A  343C 0082               326.            move #130.d2
00045E  6100'FBA0               327.            bsr textaus
000462  41ED'0000               328.            lea einbuf(a5).a0       * Suche Text
000466  303C 0022               329.            move #$22.d0    *
00046A  323C 0002               330.            move #2.d1
00046E  343C 0064               331.            move #100.d2
000472  363C 0026               332.            move #38.d3     * Max Count
000476  6100'FB88               333.            bsr textein
00047A  4A44                    334.            tst d4
00047C  6700 000A               335.            beq editof1
                                336.
000480                          337. editofsu:                  * Suchen nun moeglich
                                338.                            * Eingang CTRL-L curx(a5),curv(a5) gueltig
                                339.                            * Suchen in Bildschirmspeicher zuerst
000480  6100 001C               340.            bsr suchedit    * UPRG Aufrufen
000484  6000 FB9B               341.            bra editlp      * ok fertig
000488                          342. editof1:
000488  422D'0000               343.            clr.b viewpage(a5)
00048C  422D'0000               344.            clr.b wrtpage(a5)
000490  6100'FB6E               345.            bsr aktpage
000494  1B7C 0001               346.            move.b #1.curon(a5)     * textein loescht
        '0000
00049A  6000 FB82               347.            bra editlp
                                348.
00049E                          349. suchedit:                  * carry wenn fehler. d0=curx(a5) Startposi
00049E  422D'0000               350.            clr.b viewpage(a5)      * Bildschirmorientiert suchen
0004A2  422D'0000               351.            clr.b wrtpage(a5)       * mit Cursoranzeige
0004A6  6100'FB58               352.            bsr aktpage
0004AA  1B7C 0001               353.            move.b #1.curon(a5)     * textein loescht
        '0000
                                354.
0004B0                          355. editf11:                   * Suchschleife. cursor hinter Wort
                                356.                            * Zeilenorientiert
0004B0  6100'FB4E               357.            bsr calccur     * a2-> Textstart  a1 -> Laenge max
0004B4  41ED'0000               358.            lea einbuf(a5).a0       * Start Suchtext
0004B8  122D'0000               359.            move.b curx(a5).d1      * merken Cursor
0004BC                          360. editf122:
0004BC  1010                    361.            move.b (a0),d0  * Vergleich
0004BE  671A                    362.            beq.s editfind  * STOP gefunden
0004C0  B012                    363.            cmp.b (a2),d0
0004C2  6626                    364.            bne.s editf12   * naechstes Wort
0004C4  5201                    365.            add.b #1.d1     * weiter vergleichen bis (einbuf)=0
0004C6  D1FC 00000001           366.            adda.l #1.a0    * Vergleichsspeicher
0004CC  D5FC 00000001           367.            adda.l #1.a2    * in Buffer
0004D2  B211                    368.            cmp.b (a1),d1
0004D4  65E6                    369.            bcs.s editf122  * bei Zeilenende.
0004D6  4A10                    370.            tst.b (a0)      * wenn letztes Zeichen dann ok
0004D8  6610                    371.            bne.s editf12   * sonst nicht gefunden. dann weiter suchen
                                372.
0004DA                          373. editfind:
```

```
0004DA  102D'0000          move.b curx(a5),d0      * alte Cursorposition merken +++
0004DE  1B41'0000          move.b d1,curx(a5)      * Gefundene Position in d1
0004E2  6100 0C80          bsr adicurx
0004E6  6000 FB18          bra cerres      * gefunden, auch bei Zeilenende

0004EA                                     * alte Position weiter
0004EA  522D'0000          add.b #1,curx(a5)       * Weiter schreiten
0004EE  102D'0000          move.b curx(a5),d0
0004F2  B011               cmp.b (a1),d0   * Zeilenlaenge
0004F4  65BA               bcs editf11     * weiter suchen ok
0004F6                     editf13:
0004F6  422D'0000          clr.b curx(a5)
0004FA  522D'0000          add.b #1,curv(a5)       * check ueberlauf
0004FE  102D'0000          move.b curv(a5),d0
000502  B03C 0018          cmp.b #24,d0    * 0..23 erlaubt
000506  66A8               bne editf11     *
000508  1B7C 0017          move.b #23,curv(a5)     * definieren
        0000

                                           * ganz in Hauptspeicher
00050E  6100 0C2A          bsr outscreen   * zurueck, akttxt hinter alten Text
                                           * dort suchen ab akttxt high speed
000512
000512  246D 0000          editf14:
000516  41ED 0000          movea.l akttxt(a5),a2   * Aktueller Textzeiger
00051A                     lea einbuf(a5),a0       * suchen nun besser
00051A  B5ED'0000          editf15:
00051E  6700 0032          cmpa.l etxtxt(a5),a2    * wenn ende
000522  1010               beq editnfnd    * nicht gefunden
000524  6700 001C          move.b (a0),d0
000528  B012               beq editf20     * gefunden
00052A  6600 0010          cmp.b (a2),d0
00052E  D1FC 00000001      bne editf16
000534  D5FC 00000001      adda.l #1,a0
00053A  60DE               adda.l #1,a2
                           bra editf15

00053C                                     * nicht gefunden
00053C  52AD'0000          editf16:        * weiter suchen naechstes Byte
000540  60D0               add.l #1,akttxt(a5)
                           bra editf14

000542                                     * gefunden
000542  284A 0000          editf20:
000546  6100 0992          move.l a2,akttxt(a5) * an Punkt setzen
00054A  6100 0B9A          bsr otzurueck   * Eine Zeile zurueck
00054E  6000 FF60          bsr getscreen   * Gefunden
                           bra editf11     * Text wieder holen

000552                                     * not found
000552  6100 0986          editnfnd:
000556  6100 0B8E          bsr otzurueck
00055A  422D'0000          bsr getscreen
00055E  1B7C 0001          clr.b curx(a5)
        0000               move.b #1,curv(a5)      * naechste leere Zeile nehmen

000564  4240               clr d0          * pos savety
000566  6000 FA98          bra cerset      * nicht gefunden


00056A                                     * Suchen und ersetzen eines Namens
00056A  422D'0000          editde:
00056E  422D 0000          clr.b einbuf(a5)
000572  1B7C 0003          clr.b ausbuf(a5)
        0000               move.b #3,viewpage(a5)
000578  1B7C 0003          move.b #3,wrtpage(a5)
        0000
00057E  6100'FA80          bsr aktpage
000582  6100'FA7C          bsr clrinvis
000586  41FA 00E4          lea sutxt(pc),a0
00058A  303C 0022          move #$22,d0
00058E  323C 0002          move #2,d1
000592  343C 00B4          move #180,d2
000596  6100'FA68          bsr textaus
00059A  41ED'0000          lea einbuf(a5),a0       * Suche Text
00059E  303C 0022          move #$22,d0    *
0005A2  323C 0002          move #2,d1
0005A6  343C 0096          move #150,d2
0005AA  363C 0026          move #38,d3     * Max Count
0005AE  6100'FA50          bsr textein
0005B2  4A44               tst d4
0005B4  6700 00A0          beq finsu
0005B8  41FA 00B8          lea ertxt(pc),a0
0005BC  303C 0022          move #$22,d0
0005C0  323C 0002          move #2,d1
```

```
0005C4  343C 0078          450.          move #120.d2
0005C8  6100'FA36          451.          bsr textaus
                           452.
0005CC  41ED'0000          453.          lea ausbuf(a5).a0       * Ersetze Text
0005D0  303C 0022          454.          move #$22.d0    *
0005D4  323C 0002          455.          move #2.d1
0005D8  343C 005A          456.          move #90.d2
0005DC  363C 0026          457.          move #38.d3     * Max Count < 40 wegen gemeinsamer
0005E0  6100'FA1E          458.          bsr textein     * Verwendung mit Statuszeile
0005E4  4A44               459.          tst d4
0005E6  6700 006E          460.          beq finsu
                           461.
0005EA                     462. editoasu:                * Einsprung CTRL-L
0005EA  6100 FEB2          463.          bsr suchedit    * Suchen der Position
                           464.                          * Cursor hinter Stelle
                           465.                          * zuerst loeschen n-Mal
                           466.                          * dann einfuegen k-Mal
0005EE  6500 FA2E          467.          bcs editlo      * nicht gefunden
0005F2  122D'0000          468.          move.b curx(a5).d1      * merken Position nach Wort
0005F6  1B40'0000          469.          move.b d0.curx(a5)      * neue Position Cursor
0005FA  6100'FA04          470.          bsr ci          * Zeichen holen J.N
0005FE  B03C 004A          471.          cmp.b #'J'.d0
000602  671A               472.          beq.s editoas1
000604  B03C 006A          473.          cmp.b #'j'.d0
000608  6714               474.          beq.s editoas1  * rev 4.0
00060A  B03C 006E          475.          cmp.b #'n .d0
00060E  6708               476.          beq.s editr4o   * rev 4.0
000610  B03C 004E          477.          cmp.b #'N'.d0   * dann weiterschalten
000614  6600 FA08          478.          bne editlo      * nicht ersetzen.abbruch
000618                     479. editr4o:
000618  1B41'0000          480.          move.b d1.curx(a5)      * wieder zuruecksetzen
00061C  60CC               481.          bra.s editoasu  * und weitersuchen
                           482.
00061E                     483. editoas1:
00061E  6100 0004          484.          bsr replace     * und ausfuehren
000622  60C6               485.          bra editoasu    * weitersuchen
                           486.
                           487.
000624                     488. replace:
000624  41ED'0000          489.          lea einbuf(a5).a0       * Loeschen zuerst n Zeichen
000628                     490. editoa1:
000628  4A10               491.          tst.b (a0)
00062A  6700 000E          492.          beq editoa2
00062E  6100'F9D0          493.          bsr delchar
000632  D1FC 00000001      494.          adda.l #1.a0
000638  60EE               495.          bra editoa1
00063A                     496. editoa2:
00063A  41ED'0000          497.          lea ausbuf(a5).a0
00063E                     498. editoa21:
00063E  4A10               499.          tst.b (a0)
000640  6700 0012          500.          beq editoa3
000644  6100'F9BA          501.          bsr inschar
000648  1018               502.          move.b (a0)+.d0
00064A  2F08               503.          move.l a0.-(a7)
00064C  6100'F9B2          504.          bsr charoscr
000650  205F               505.          movea.l (a7)+.a0
000652  60EA               506.          bra editoa21
000654                     507. editoa3:
000654  4E75               508.          rts
                           509.
                           510.
000656                     511.
000656  422D'0000          512. finsu:
00065A  422D'0000          513.          clr.b viewpage(a5)
00065E  6100'F9A0          514.          clr.b wrtpage(a5)
000662  1B7C 0001          515.          bsr aktpage
        0000               516.          move.b #1.curon(a5)     * textein loescht
000668  6000 F9B4          517.          bra editlo
                           518.
                           519.
00066C  53 75 63 68        520. sutxt:   dc.b 'Suche '.0     * <= laenge ertxt
        65 00
000672  45 72 73 65        521. ertxt:   dc.b 'Ersetze '.0         * laenger als sutxt. daher uebersc
        74 7A 65 00
00067A  00                 522.          dc.b 0
        00 -- inserted
00067C                     523.          ds 0
                           524.
                           525.
```

```
00067C                                526. editk:
00067C  6100 F982                     527.          bsr ci                  * weiter abfragen
000680  C03C 001F                     528.          and.b #%00011111.d0     * rev 4.1
000684  B03C 0018                     529.          cmp.b #'X'-$40.d0       * ENDE Editieren
000688  6700 004e                     530.          beq editend
00068C  B03C 0002                     531.          cmp.b #'B'-$40.d0       * Block Anfang
000690  6700 005C                     532.          beq blockan
000694  B03C 000B                     533.          cmp.b #'K'-$40.d0       * Block Ende
000698  6700 0080                     534.          beq blocken
00069C  B03C 0019                     535.          cmp.b #'Y'-$40.d0       * Block Loeschen
0006A0  6700 00B8                     536.          beq blockloe
0006A4  B03C 001e                     537.          cmp.b #'V'-$40.d0       * Block verschieben
0006A8  6700 012E                     538.          beq blockmve
0006AC  B03C 0003                     539.          cmp.b #'C'-$40.d0       * Block copieren
0006B0  6700 01EA                     540.          beq blockcop
0006B4  B03C 0012                     541.          cmp.b #'R'-$40.d0       * Block Lesen      (mem)
0006B8  6700 02B0                     542.          beq blockles
0006BC  B03C 0017                     543.          cmp.b #'W'-$40.d0       * Block Speichern (mem)
0006C0  6700 0314                     544.          beq blockspe
0006C4  B03C 0008                     545.          cmp.b #'H'-$40.d0       * Block Marken entfernen
0006C8  6700 0388                     546.          beq blockneutr
0006CC  6000 F950                     547.          bra editlo              * nicht gefunden
                                      548.
                                      549.
0006D0                                550. editend:
0006D0  6100 0A68                     551.          bsr cutscreen   * zurueck
0006D4  6100 F92A                     552.          bsr clrall        * rev4.1 ootflag wegen Zweisprach.
0006D8  422D 0000                     553.          clr.b ootflag(a5) * wieder ausschalten.
0006DC  422D 0000                     554.          clr.b curon(a5)
0006E0  422D 0000                     555.          clr.b flio(a5)
0006E4  422D 0000                     556.          clr.b flio1(a5) * kein Blinken mehr
0006E8  6100 03B4                     557.          bsr markeweg    * KB/KK-Marken entfernen
0006EC  4E75                          558.          rts
                                      559.
        00000081                      560. blkst equ $81   * Block - Startmarke
        00000082                      561. blken equ $82   * Block - Endemarke
        00000083                      562. blkins equ $83  * Block - Einfuegemarke
                                      563.
                                      564. * Bei Blocksetzen  - Suchen nach alter Marke
                                      565.
                                      566.
0006EE                                567. blockan:                 * Block Anfang markieren
0006EE  082D 0001                     568.          btst #1.insl(a5)
        0000
0006F4  6704                          569.          beq.s blocklan
0006F6  6100 0362                     570.          bsr blockck     * ACHTUNG. ALLE Marken entfernen
0006FA                                571. blocklan:                * Wenn schon Anfangsmarke da
0006FA  08ED 0001                     572.          bset #1.insl(a5)
        0000
000700  103C 0081                     573.          move.b #blkst.d0
000704  6100 0006                     574.          bsr blockhl
000708  6000 F914                     575.          bra editlo
                                      576.
00070C                                577. blockhl:
00070C  3F00                          578.          move d0.-(a7)
00070E  6100 F8F0                     579.          bsr inschar     * Einfuegen an aktueller Stelle
000712  301F                          580.          move (a7)+.d0
000714  6100 F8EA                     581.          bsr charoscr    * Und ausgeben auf dem Bildschirm
000718  4E75                          582.          rts
                                      583.
00071A                                584.
00071A  082D 0002                     585. blocken:                 * Blockende markieren
        0000                          586.          btst #2.insl(a5)
000720  6704                          587.          beq.s blocklen
000722  6100 0336                     588.          bsr blockck     * ACHTUNG. ALLE Marken entfernen
000726                                589. blocklen:                * im Fehlerfall. wenn schon Endmarke d
000726  08ED 0002                     590.          bset #2.insl(a5)
        0000
00072C  103C 0082                     591.          move.b #blken.d0
000730  61DA                          592.          bsr blockhl
000732  6000 F8EA                     593.          bra editlo
                                      594.
000736                                595. blkakt:                  * akttxt auf Blockanfang stellen
                                      596.                          * bzw. Auf Zeilenstart
000736  206D 0000                     597.          movea.l stxtxt(a5).a0
00073A                                598. blksakt:                 * Suche bei Textstart beginnen
00073A  1018                          599.          move.b (a0)+.d0 * suchen
00073C  6700 0012                     600.          beq blklakt
000740  B03C 0081                     601.          cmp.b #blkst.d0
```

```
000744  66F4                    602.            bne blksakt     * nein. dann weitersuchen
000746  5388                    603.            subq.l #1,a0    * Zeigt auf Blockanfang
000748  2B48 0000               604.            move.l a0,akttxt(a5)
00074C  6100 078C               605.            bsr otzurueck   * Muss Zeilenanfang sein
000750                          606. blklakt:                   * Textende aufgetreten. dann nicht aendern
000750  422D 0000               607.            clr.b curx(a5)  * jedoch immer
000754  422D 0000               608.            clr.b curv(a5)  * Home-Position anfahren
000758  4E75                    609.            rts             * Fertig
                                610.
                                611.
00075A                          612. blockloe:                  * Block loeschen
00075A  082D 0001               613.            btst.b #1,insl(a5)      * Anfang muss markiert sein
        0000
000760  6700 FBBC               614.            beq editlo      * sonst Fehler
000764  082D 0002               615.            btst.b #2,insl(a5)
        0000
00076A  6700 FBB2               616.            beq editlo      * und Blockende
00076E  2F2D 0000               617.            move.l akttxt(a5),-(a7)
000772  6100 09C6               618.            bsr outscreen   * zurueckschreiben
000776  2B5F 0000               619.            move.l (a7)+,akttxt(a5)
                                620.                            * Bearbeitung
00077A  6100 0390               621.            bsr findeae     * a0-> anf a1-> ende
00077E  6500 0012               622.            bcs blklloe     * Fehlerbedingung
000782  6100 001E               623.            bsr blklexe     * akttxt zeigt auf Anfang
000786  08AD 0001               624.            bclr.b #1,insl(a5)  * Marken wurden auch geloescht
        0000
00078C  08AD 0002               625.            bclr.b #2,insl(a5)
        0000
000792                          626. blklloe:
000792  6100 0952               627.            bsr getscreen
000796  422D 0000               628.            clr.b curx(a5)
00079A  422D 0000               629.            clr.b curv(a5)
00079E  6000 F87E               630.            bra editlo
                                631.
                                632.
007A2                           633. blklexe:                   * a0-> anf. a1-> ende
                                634.                            * loescht einen Block, allgemeine Routine
                                635.                            * Akttxt wird auf Anfang gestellt
007A2  48E7 00C0                636.            movem.l a0/a1,-(a7)
007A6  2209                     637.            move.l a1,d1
007A8  9288                     638.            sub.l a0,d1     * Distanz merken n-1
007AA  5281                     639.            addq.l #1,d1    * n = Anzahl der geloeschten Bytes
007AC  5289                     640.            addq.l #1,a1    * Quelle fuer Transport
007AE  202D 0000                641.            move.l etxtxt(a5),d0
007B2  9089                     642.            sub.l a1,d0     * n-1 Bytes fuer DBRA ok
007B4  10D9                     643. blklie: move.b (a1)+,(a0)+
007B6  51C8 FFFC                644.            dbra d0,blklie  * bis alles geloescht
007BA  4CDF 0300                645.            movem.l (a7)+,a0/a1
007BE  93AD 0000                646.            sub.l d1,etxtxt(a5)
007C2  93AD 0000                647.            sub.l d1,eotxt(a5) * neuer Stand
007C6  2B48 0000                648.            move.l a0,akttxt(a5) * auf Blockanfang stellen
007CA  48E7 00E0                649.            movem.l a0-a2,-(a7)
007CE  6100 070A                650.            bsr otzurueck   * Pointer auf Zeilenanfang
007D2  4CDF 0700                651.            movem.l (a7)+,a0-a2
007D6  4E75                     652.            rts
                                653.
                                654.
007D8                           655. blockmve:
007D8  082D 0001                656.            btst.b #1,insl(a5)      * Anfang muss markiert sein
        0000
007DE  6700 FB3E                657.            beq editlo      * sonst Fehler
007E2  082D 0002                658.            btst.b #2,insl(a5)
        0000
007E8  6700 FB34                659.            beq editlo      * und Blockende
007EC  08ED 0003                660.            bset #3,insl(a5)
        0000
007F2  103C 0083                661.            move.b #blkins,d0
007F6  6100 FF14                662.            bsr blockhi
007FA  2F2D 0000                663.            move.l akttxt(a5),-(a7)
007FE  6100 093A                664.            bsr outscreen   * zurueckschreiben
0802   2B5F 0000                665.            move.l (a7)+,akttxt(a5)
                                666.                            * Bearbeitung
0806   6100 032C                667.            bsr findeaei    * Alle drei Marken muessen ok sein
080A   6500 000C                668.            bcs blklmve     * Bei Fehler, nicht durchfuehren
080E   6100 0046                669.            bsr blkiexe     * Platz schaffen
0812   6100 0010                670.            bsr blkcexe     * Copieren
0816   618A                     671.            bsr blklexe     * und alten Block loeschen
0818                            672. blklmve:
0818   6100 FF1C                673.            bsr blkakt      * neuer Cursorstand, auf Blockanfang
081C   6100 08C8                674.            bsr getscreen
```

```
000820  6000 F7FC          675.          bra editlo
                           676.
000824                     677. blkcexe:                 * Kopieren von Quelle (a0->a1). nach Ziel
                           678.                          * Wenn dort Platz dafuer ist.
000824  48E7 00E0          679.          movem.l a0-a2,-(a7)
000828  2009               680.          move.l a1,d0
00082A  9088               681.          sub.l a0,d0     * Anzahl n-1 Zeichen
00082C  14D8               682. blkcle:  move.b (a0)+,(a2)+ * ins Zielgebiet transportieren
00082E  51C9 FFFC          683.          dbra d0,blkcle
000832  4CDF 0700          684.          movem.l (a7)+,a0-a2
000836  4E75               685.          rts
                           686.
                           687.
000838                     688. blkmoexe:                * Quellmarken entfernen
                           689.                          * a0-> erste Marke. a1-> zweite Marke
000838  48E7 00E0          690.          movem.l a0-a2,-(a7)
00083C  2049               691.          movea.l a1,a0   * Erst Ende loeschen
00083E  6100 FF62          692.          bsr blklexe
000842  4CDF 0700          693.          movem.l (a7)+,a0-a2
000846  48E7 00E0          694.          movem.l a0-a2,-(a7)
00084A  2248               695.          movea.l a0,a1   * Dann Quelle
00084C  6100 FF54          696.          bsr blklexe
000850  4CDF 0700          697.          movem.l (a7)+,a0-a2
000854  4E75               698.          rts
                           699.
                           700.
000856                     701. blklexe:                 * a0->anf a1->end a2->ziel
                           702.                          * a0,a1 bleiben korrekt auf
                           703.                          * die Marken eingestellt
                           704.                          * akttxt wird auf Ziel gesetzt
                           705.                          * Zielmarke wird freigestellt
                           706.                          * a2 zeigt auf die Zielmarke
000856  B1CA               707.          cmpa.l a2,a0    * ziel < anf
000858  6400 003A          708.          bcc blklie      * ja
                           709.                          * ziel > end ( durch findeaei geprueft )
00085C                     710. blk0ie:                  * Global Unterprog.. d0=Anzahl
00085C  2009               711.          move.l a1,d0
00085E  9088               712.          sub.l a0,d0     * n-1 Anzahl der Einzufuegenden Zeichen
000860  6700 F79E          713.          beq carres      * 1 Zeichen einfuegen. ok verwenden der I
000864  266D 0000          714.          movea.l etxtxt(a5),a3  * Quelle
000868  284B               715.          movea.l a3,a4   * Ziel
00086A  D9C0               716.          adda.l d0,a4    * Distanz n-1
00086C  220B               717.          move.l a3,d1    * Anzahl berechnen
00086E  928A               718.          sub.l a2,d1     * ende-insl  -> n-1 Zeichen incl. INSL
000870  528B               719.          addq.l #1,a3    * Wegen DECREMENT
000872  528C               720.          addq.l #1,a4
000874  1923               721. blk2ie:  move.b -(a3),-(a4)
000876  51C9 FFFC          722.          dbra d1,blk2ie  * und transportieren
00087A  D1AD 0000          723.          add.l d0,etxtxt(a5)  * Neues Ende
00087E  D1AD 0000          724.          add.l d0,eottxt(a5)
000882  2B4A 0000          725.          move.l a2,akttxt(a5) * Ziel ist neuer Pointerstand
000886  48E7 00E0          726.          movem.l a0-a2,-(a7)
00088A  6100 064E          727.          bsr otzurueck   * Zeile davor waehlen
00088E  4CDF 0700          728.          movem.l (a7)+,a0-a2
000892  4E75               729.          rts             * ok a0,a1,a2 quelltg
                           730.
000894                     731. blklie:                  * Ziel < Anf  Sonderfall
000894  61C6               732.          bsr blk0ie      * Wie >-Fall, jedoch nun Korrektur
000896  D1C0               733.          adda.l d0,a0    * Anzahl+(n-1)
000898  D3C0               734.          adda.l d0,a1    * jedoch a2. das Ziel bliebt
000[illegible]  4E75       735.          rts
                           736.
                           737.
00089C                     738. blockcoo:
00089C  082D 0001          739.          btst.b #1,insl(a5)      * Anfang muss markiert sein
        0000
0008A2  6700 F77A          740.          beq editlo      * sonst Fehler
0008A6  082D 0002          741.          btst.b #2,insl(a5)
        0000
0008AC  6700 F770          742.          beq editlo      * und Blockende
0008B0  08ED 0003          743.          bset #3,insl(a5)
        0000
0008B6  103C 0083          744.          move.b #blkins,d0
0008BA  6100 FE50          745.          bsr blockhl
0008BE  2F2D 0000          746.          move.l akttxt(a5),-(a7)
0008C2  6100 0876          747.          bsr outscreen   * zurueckschreiben
0008C6  2B5F 0000          748.          move.l (a7)+,akttxt(a5)
                           749.                          * Bearbeitung
0008CA  6100 0268          750.          bsr findeaei    * Alle drei Marken muessen ok sein
0008CE  6500 000C          751.          bcs blkicoo     * Bei Fehler, nicht durchfuehren
```

```
0008D2  6182                752.          bsr blkiexe     * Platz schaffen
0008D4  6100 FF4E           753.          bsr blkcexe     * Copieren
0008D8  6100 FF5E           754.          bsr blkmoexe    * alte Marken entfernen
0008DC                      755. blkicoo:                 * Neue Marken im Zielgebiet
0008DC  6100 FE58           756.          bsr blkakt      * Neuer Cursorstand bei Blockanf.
0008E0  6100 0804           757.          bsr getscreen
0008E4  6000 F738           758.          bra editlo
                            759.
                            760. *
                            761. * Block Lese- und Schreibbefehle
                            762. * Sie arbeiten im Speicher und dazu wird eine
                            763. * Ziel. bzw. Quelladresse angegeben.
                            764. *
                            765.
0008E8  51 75 65 6C         766. quelladr: dc.b 'Quelle',0
        6C 65 00
0008EF  5A 69 65 6C         767. zieladr:  dc.b 'Ziel',0
        00
0008F4                      768.          ds 0
                            769.
0008F4                      770. getvalx:
0008F4  2F08                771.          move.l a0,-(a7)
0008F6  422D 0000           772.          clr.b einbuf(a5)
0008FA  422D 0000           773.          clr.b ausbuf(a5)
0008FE  1B7C 0003 0000      774.          move.b #3,viewpage(a5)
000904  1B7C 0003 0000      775.          move.b #3,wrtpage(a5)
00090A  6100 F6F4           776.          bsr aktpage
00090E  6100 F6F0           777.          bsr clrinvis
000912  205F                778.          movea.l (a7)+,a0
000914  303C 0022           779.          move #$22,d0
000918  323C 0002           780.          move #2,d1
00091C  343C 00B4           781.          move #180,d2
000920  6100 F6DE           782.          bsr textaus
000924  41ED 0000           783.          lea ausbuf(a5),a0
000928  303C 0022           784.          move #$22,d0
00092C  323C 0002           785.          move #2,d1
000930  343C 005A           786.          move #90,d2
000934  363C 0026           787.          move #38,d3     * Achtung Statusvwdg.
000938  6100 F6C6           788.          bsr textein     * Zahl. Adresse holen
00093C  422D 0000           789.          clr.b viewpage(a5)
000940  422D 0000           790.          clr.b wrtpage(a5)
000944  1B7C 0001 0000      791.          move.b #1,curon(a5) * veraender durch textein
00094A  6100 F6B4           792.          bsr aktpage     * Zurueckstellen
00094E  41ED 0000           793.          lea ausbuf(a5),a0
000952  323C FFFF           794.          move #$ffff,d1
000956  6100 F6A8           795.          bsr igbn
00095A  6100 F6A4           796.          bsr exor        * QUELLADRESSE laden
00095E  B27C 0003           797.          cmp #3,d1       * nur Langwort zul.
000962  6700 F69C           798.          beq carres
000966  6000 F698           799.          bra carset      * Carry bei Fehler
                            800.
00096A                      801. blockles:
00096A  08ED 0003 0000      802.          bset #3,insl(a5)
000970  103C 0083           803.          move.b #blkins,d0
000974  6100 FD96           804.          bsr blockhl
000978  2F2D 0000           805.          move.l akttxt(a5),-(a7)
00097C  6100 07BC           806.          bsr outscreen   * zurueckschreiben
000980  2B5F 0000           807.          move.l (a7)+,akttxt(a5)
                            808.                          * Bearbeitung
000984  41FA FF62           809.          lea quelladr(pc),a0
000988  6100 FF6A           810.          bsr getvalx     * dc.l = Adr
00098C  6500 0038           811.          bcs blk2les
000990  2040               812.          move.l d0,a0    * Quelladresse
000992  2248               813.          movea.l a0,a1   * Endadresse suchen
000994  4A11               814.          tst.b (a1)      * =0 Dann Fehler. Leere Quelle
000996  6700 002E           815.          beq blk2les
00099A                      816. blk1les:
00099A  4A19               817.          tst.b (a1)+     * =0 dann Ende gefunden
00099C  66FC               818.          bne blk1les     * a1-> hinter Ende
00099E  5589               819.          subq.l #2,a1    * Zeigt auf Ende quelloses Zeichen
0009A0  48E7 00C0          820.          movem.l a0/a1,-(a7)
0009A4  6100 0180          821.          bsr findei      * Einfuege-Marke suchen
0009A8  4CDF 0300          822.          movem.l (a7)+,a0/a1
0009AC  6500 0018          823.          bcs blk2les
0009B0  48E7 00C0          824.          movem.l a0/a1,-(a7)     * Merken
0009B4  6100 FEA0          825.          bsr blkiexe     * Einfuegen eines freien Blockes
```

```
0009BB  4CDF 0300       826.            movem.l (a7)+.a0/a1     * Alter Wert stabil
0009BC  6100 FE66       827.            bsr blkcexe     * Und Kopieren in Hauptspeicher
0009C0  08AD 0003       828.            bclr.b #3.insl(a5) * ueberschrieben und geloescht
        0000
0009C6                  829. blk2les:
0009C6  6100 071E       830.            bsr getscreen
0009CA  422D 0000       831.            clr.b curx(a5)  * akttxt zeigt auf Textanfang
0009CE  422D 0000       832.            clr.b curv(a5)  * Blockanfang an Bildschirmanfang
0009D2  6000 F64A       833.            bra editlo
                        834.
0009D6                  835. blockspe:
0009D6  082D 0001       836.            btst.b #1.insl(a5)      * Anfang muss markiert sein
        0000
0009DC  6700 F640       837.            beq editlo      * sonst Fehler
0009E0  082D 0002       838.            btst.b #2.insl(a5)
        0000
0009E6  6700 F636       839.            beq editlo      * und Blockende
0009EA  1F2D 0000       840.            move.b curx(a5).-(a7)
0009EE  1F2D 0000       841.            move.b curv(a5).-(a7)
0009F2  2F2D 0000       842.            move.l akttxt(a5).-(a7)
0009F6  6100 0742       843.            bsr outscreen   * zurueckschreiben
0009FA  2B5F 0000       844.            move.l (a7)+.akttxt(a5)
                        845.                            * Bearbeitung
0009FE  41FA FEEF       846.            lea zieladr(pc).a0
000A02  6100 FEF0       847.            bsr getvalx     * d0.1 = Wert
000A06  6700 003A       848.            beq block1spe   * Fehler
                        849.                            * PRUEFUNG durchfuehren
000A0A  2440            850.            movea.l d0.a2   * Ziel ist A2
000A0C  206D 0000       851.            movea.l stxtxt(a5).a0
000A10  226D 0000       852.            movea.l etxtxt(a5).a1
000A14  B1CA            853.            cmpa.l a2.a0    * < ok
000A16  6400 0008       854.            bcc block2spe
000A1A  B5C9            855.            cmpa.l a1.a2    * > ok
000A1C  6500 0024       856.            bcs block1spe   * Fehler liegt dazwischen
000A20                  857. block2spe:
000A20  48E7 0020       858.            movem.l a2.-(a7)
000A24  6100 00E6       859.            bsr findeae     * Anfang und Ende suchen
000A28  4CDF 0400       860.            movem.l (a7)+.a2 * a2 zeigt auf Zielgebiet
000A2C  6500 0014       861.            bcs block1spe   * Fehler, nicht definiert
000A30  2009            862.            move.l a1.d0
000A32  9088            863.            sub.l a0.d0     * n-1 Zeichen incl. Marken
000A34  5380            864.            sub.l #1.d0     * n ohne Marken
000A36  5288            865.            addo.l #1.a0    * Start+1 ist Block ohne Marke
000A38  6002            866.            bra.s blk4spe   * =0 Dann nicht ausfuehren
000A3A                  867. blk3spe:
000A3A  14D8            868.            move.b (a0)+.(a2)+  * Ins Zielgebiet schaffen
000A3C                  869. blk4spe:
000A3C  51C8 FFFC       870.            dbra d0.blk3spe
000A40  4212            871.            clr.b (a2)      * Ende-Marke dort ablegen
000A42                  872. block1spe:
000A42  6100 06A2       873.            bsr getscreen
000A46  1B5F 0000       874.            move.b (a7)+.curv(a5)
000A4A  1B5F 0000       875.            move.b (a7)+.curx(a5)
000A4E  6000 F5CE       876.            bra editlo
                        877.
000A52                  878. blockneutr:
000A52  6100 0006       879.            bsr blockck
000A56  6000 F5C6       880.            bra editlo
                        881.
000A5A                  882. blockck:                   * gemeinsame Routine
000A5A  082D 0001       883.            btst.b #1.insl(a5)      * Anfang muss markiert sein
        0000
000A60  6600 0016       884.            bne blockn1     * sonst Fehler
000A64  082D 0002       885.            btst.b #2.insl(a5)
        0000
000A6A  6600 000C       886.            bne blockn1     * und Blockende
000A6E  082D 0003       887.            btst.b #3.insl(a5)
        0000
000A74  6700 0026       888.            beq block1ck    * Keine Marke gesetzt, dann ignorieren
000A78                  889. blockn1:                   * Eine Marke genuegt
000A78  1F2D 0000       890.            move.b curx(a5).-(a7)
000A7C  1F2D 0000       891.            move.b curv(a5).-(a7)
000A80  2F2D 0000       892.            move.l akttxt(a5).-(a7)
000A84  6100 06B4       893.            bsr outscreen   * zurueckschreiben
                        894.                            * Bearbeitung, Marken entfernen
000A88  6100 0014       895.            bsr markeweg
000A8C  2B5F 0000       896.            move.l (a7)+.akttxt(a5)
000A90  6100 0654       897.            bsr getscreen
000A94  1B5F 0000       898.            move.b (a7)+.curv(a5)
000A98  1B5F 0000       899.            move.b (a7)+.curx(a5)
```

```
000A9C
000A9C  4E75                    900. blockick:
                                901.         rts
                                902.
000A9E                          903. markeweg:          * Im Speicher Marken entfernen
                                904.                    * falls welche eingetragen sind
                                905.                    * nur jeweils eine Marke moeglich
                                906.                    * Text steht im Speicher.
000A9E  082D 0001               907.         btst.b #1.insl(a5)
        0000
000AA4  6612                    908.         bne.s malweg  * OK loeschen noetig
000AA6  082D 0002               909.         btst.b #2.insl(a5)
        0000
000AAC  660A                    910.         bne.s malweg
000AAE  082D 0003               911.         btst.b #3.insl(a5)
        0000
000AB4  6602                    912.         bne.s malweg
000AB6  4E75                    913.         rts      * sonst Ueberfluessig
                                914.
000AB8                          915. malweg:
000AB8  206D 0000               916.         movea.l stxtxt(a5).a0   * Start des Suchgebietes
000ABC  1018                    917. ma2weg: move.b (a0)+.d0
000ABE  6700 0038               918.         beq ma3weg              * =0 Dann Ende der Suche
000AC2  0800 0007               919.         btst #7.d0              * Bit 7 gesetzt, dann verdaechtig
000AC6  67F4                    920.         beq ma2weg              * nein, dann weitersuchen bis =0
000ACB  B03C 0081               921.         cmp.b #blkst.d0         * Erst wenn Bit 7, dann An[illegible]e
000ACC  670C                    922.         beq.s ma21weg
000ACE  B03C 0082               923.         cmp.b #blken.d0
000AD2  6706                    924.         beq.s ma21weg
000AD4  B03C 0083               925.         cmp.b #blkins.d0
000AD8  66E2                    926.         bne ma2weg              * Nicht gefunden
000ADA                          927. ma21weg:                        * Position gefunden, a0-> dahinter
000ADA  202D 0000               928.         move.l etxtxt(a5).d0    * BEREICH max 64K Transfer
000ADE  9088                    929.         sub.l a0.d0             * gibt n-1 Transfers fuer DBRA ok
000AE0  2248                    930.         movea.l a0.a1           * Temp Pointer Ziel
000AE2  2449                    931.         movea.l a1.a2           * Temp Pointer Quelle
000AE4  5389                    932.         subq.l #1.a1            * Ziel ist vorherioer Speicher
000AE6                          933. ma22weg:                        * incl. 0 als Ende bei ETXTXT
000AE6  12DA                    934.         move.b (a2)+.(a1)+
000AE8  51C8 FFFC               935.         dbra d0.ma22weg         * und ausfuehren
000AEC  53AD 0000               936.         subq.l #1.etxtxt(a5)
000AF0  53AD 0000               937.         subq.l #1.eottxt(a5)    * Zeichen geloescht
000AF4  5388                    938.         subq.l #1.a0            * Ein Zeichen vorher anfangen
000AF6  60C4                    939.         bra ma2weg              * und weitersuchen
                                940.
000AF8                          941. ma3weg:                         * Ende der Bearbeitung
000AF8  08AD 0001               942.         bclr.b #1.insl(a5)
        0000
000AFE  08AD 0002               943.         bclr.b #2.insl(a5)
        0000
000B04  08AD 0003               944.         bclr.b #3.insl(a5)
        0000
000B0A  4E75                    945.         rts
                                946.
000B0C                          947. findeae:                        * Anfangs und Endmarke
000B0C  6100 0042               948.         bsr finde
000B10  2008                    949.         move.l a0.d0
000B12  6700 F4EC               950.         beq carset
000B16  2009                    951.         move.l a1.d0            * Beide muessen definiert sein
000B18  6700 F4E6               952.         beq carset
000B1C  B1C9                    953.         cmpa.l a1.a0            * Ende > Anfang, dann ok
000B1E  6400 F4E0               954.         bcc carset              * nein, dann Fehler
000B22  6000 F4DC               955.         bra carres              * ja, dann ok
                                956.
000B26                          957. findei:                         * Einfuegemarke
000B26  6100 0028               958.         bsr finde
000B2A  200A                    959.         move.l a2.d0
000B2C  6700 F4D2               960.         beq carset              * Fehler, nicht definiert
000B30  6000 F4CE               961.         bra carres              * ok
                                962.
000B34                          963. findeaei:                       * Alle Marken, finde call
000B34  61D6                    964.         bsr findeae             * kein Carry, dann A < E
000B36  6500 F4C8               965.         bcs carset              * sonst Fehler
000B3A  200A                    966.         move.l a2.d0
000B3C  6700 F4C2               967.         beq carset              * Fehler, nicht da
000B40  B1CA                    968.         cmpa.l a2.a0            * Anfang <= insl <= Ende ist FEHLE
000B42  6400 F4BC               969.         bcc carres              * insl < Anf ist ok
000B46  B5C9                    970.         cmpa.l a1.a2            * Ende < insl ist ok
000B48  6400 F4B6               971.         bcc carres
000B4C  6000 F4B2               972.         bra carset              * sonst Fehler
                                973.
```

```
000B50                          974. finde:                             * Suchen nach Marken im Speicher
                                975.                                    * Carry bei Fehlerbedingung
                                976.                                    * a0->Blockst-Marke
                                977.                                    * a1->Blocken-Marke
                                978.                                    * a2->Blockins-Marke
000B50  266D'0000               979.            movea.l stxtxt(a5).a3   * Start der Suche
000B54  207C 00000000           980.            movea.l #0.a0
000B5A  227C 00000000           981.            movea.l #0.a1
000B60  247C 00000000           982.            movea.l #0.a2           * falls nicht gefunden=0
000B66  101B                    983. find1:     move.b (a3)+.d0
000B68  6700 002C               984.            beq find2               * Ende
000B6C  0800 0007               985.            btst #7.d0
000B70  67F4                    986.            beq find1               * weitersuchen bis Bit 7 gesetzt
000B72  B03C 0081               987.            cmp.b #blkst.d0
000B76  6712                    988.            beq.s find1a
000B78  B03C 0082               989.            cmp.b #blken.d0
000B7C  6712                    990.            beq.s find2a
000B7E  B03C 0083               991.            cmp.b #blkins.d0
000B82  66E2                    992.            bne find1               * Start der Marke ist A-1
000B84  244B                    993.            movea.l a3.a2
000B86  538A                    994.            subq.l #1.a2
000B88  60DC                    995.            bra find1
000B8A                          996. find1a:
000B  204B                      997.            movea.l a3.a0
000  5388                       998.            subq.l #1.a0
000B  60D6                      999.            bra find1
000B90  224B                   1000. find2a:    movea.l a3.a1
000B92  5389                   1001.            subq.l #1.a1
000B94  60D0                   1002.            bra find1               * Jeweils eintragen und weitersuch
000B96  4E75                   1003. find2:     rts
                               1004.
                               1005.
000B98                         1006. editc:                    * Umschaltmenue
                               1007.                           * REV 4.0 auch Kleinbuchstaben
                               1008.                           * und Controls
000B98  6100 F466              1009.            bsr ci         * neues Zeichen
000B9C  C03C 001F              1010.            and.b #%000111111.d0    * rev 4.1
000BA0  B03C 0012              1011.            cmp.b #'R'-$40.d0       * Q R
000BA4  6700 001E              1012.            beq editor
000BA8  B03C 0003              1013.            cmp.b #'C'-$40.d0
000BAC  6700 0028              1014.            beq editoc
000BB0  B03C 0006              1015.            cmp.b #'F'-$40.d0
000BB4  6700 F87C              1016.            beq editof
000BB8  B03C 0001              1017.            cmp.b #'A'-$40.d0
000BBC  6700 F9AC              1018.            beq editoa
000BC0  6000 F45C              1019.            bra editlo
                               1020.
000BC4                         1021. editor:                   * an den Anfang
000BC4  6100 0574              1022.            bsr outscreen
000BC8  2B6D'0000              1023.            move.l stxtxt(a5).akttxt(a5)
        '0000
000BCE  6100 0516              1024.            bsr getscreen
000BD2  6000 F44A              1025.            bra editlo
                               1026.
000BD6  6100 0562              1027. editoc:    bsr outscreen  * an das ende gehen
000BDA  2B6D'0000              1028.            move.l etxtxt(a5).akttxt(a5)
        '0000
000BE0  323C 0009              1029.            move #10-1.d1  * zehn zeilen sichbar lassen
000BE4                         1030. edit1oc:
000BE4  3F01                   1031.            move d1.-(a7)
000  6100 02F2                 1032.            bsr otzurueck  * rueckwaerts falls moeglich
000  321F                      1033.            move (a7)+.d1
000BEC  51C9 FFF6              1034.            dbra d1.edit1oc
000BF0  6100 04F4              1035.            bsr getscreen  * und wieder auffuellen
000BF4  6000 F428              1036.            bra editlo
                               1037.
                               1038.
000BF8  4A2D'0000              1039. editbs:    tst.b curx(a5)  * = 0 dann prev Zeilestop z.Z.
000BFC  6708                   1040.            beq.s edit1bs   * und CTRL - S
000BFE  532D'0000              1041.            subq.b #1.curx(a5)
000C02  6000 F41A              1042.            bra editlo
000C06                         1043. edit1bs:
000C06  4A2D'0000              1044.            tst.b curv(a5)
000C0A  6700 0016              1045.            beq edit2bs     * Sonderfall v=0
000C0E  532D'0000              1046.            subq.b #1.curv(a5)
000C12  6100 F3EC              1047.            bsr calccur     * a1-> Anzahl
000C16  1B51'0000              1048.            move.b (a1).curx(a5)    * Auf Zeilenende positionieren
000C1A  6100 0548              1049.            bsr adjcurx     * nur wenn 80 dann nicht durchf.
000C1E  6000 F3FE              1050.            bra editlo
000C22                         1051. edit2bs:                  * Scroll nach unten ausfuehren
```

```
000C22  202D'0000       1052.                           * letzte Zeile nach Speicher
000C26  B0AD'0000       1053.           move.l akttxt(a5).d0
000C2A  6700 F3F2       1054.           cmp.l stxtxt(a5).d0     * wenn gleich nicht mehr weiter mo
000C2E  6100 0494       1055.           beq editlp
000C32  6100'F3CC       1056.           bsr crtdown
000C36  1B51'0000       1057.           bsr calccur     * a1-> Anzahl
000C3A  6100 0528       1058.           move.b (a1).curx(a5)    * Auf Zeilenende positionieren
000C3E  6000 F3DE       1059.           bsr adjcurx
                        1060.           bra editlp
000C42                  1061.
000C42  4280            1062. edittb:                   * Tabulator, bei insmode mit leerzeichenfu
000C44  102D'0000       1063.           clr.l d0        * ganzes register loeschen
000C48  80FC 0008       1064.           move.b curx(a5).d0
000C4C  4840            1065.           divu #8.d0      * mod in msb
000C4E  4440            1066.           swap d0
000C50  5040            1067.           neg d0
000C52  082D 0000       1068.           add #8.d0       * 8-(curx(a5) mod 8)
        0000            1069.           btst.b #0.insl(a5)
000C58  661C            1070.           bne.s edit1tab
000C5A  D12D'0000       1071.           add.b d0.curx(a5)
000C5E  6100 0504       1072.           bsr adjcurx     * falls ueberlauf
000C62  6000 F3BA       1073.           bra editlp
                        1074.
000C66                  1075. edit12tab:
000C66  3F00            1076.           move d0.-(a7)
000C68  6100'F396       1077.           bsr inschar
000C6C  103C 0020       1078.           move.b #' '.d0
000C70  6100'F3BE       1079.           bsr charoscr    * bleibt in endpos
000C74  301F            1080.           move (a7)+.d0
000C76                  1081. edit1tab:                 * d0 = anzahl fuellzeichen 8..1
000C76  51C8 FFEE       1082.           dbra d0.edit12tab
000C7A  6000 F3A2       1083.           bra editlp
                        1084.
                        1085.
000C7E                  1086. editctc:
000C7E  323C 000E       1087.           move #15-1.d1   * 15 weiter
000C82                  1088. edit1ctc:
000C82  3F01            1089.           move d1.-(a7)
000C84  6100 041E       1090.           bsr crtuo
000C88  321F            1091.           move (a7)+.d1
000C8A  51C9 FFF6       1092.           dbra d1.edit1ctc
000C8E  6000 F38E       1093.           bra editlp
                        1094.
                        1095.
000C92                  1096. editctd:                  * Cursor nach rechts
000C92  6100'F36C       1097.           bsr calccur
000C96  522D'0000       1098.           addq.b #1.curx(a5)      * 1..80
000C9A  102D'0000       1099.           move.b curx(a5).d0      * checken ob ueberlauf nach rechts
000C9E  B011            1100.           cmp.b (a1).d0   * wenn groesser als max dann naechste zeil
000CA0  620C            1101.           bhi.s edit1ctd  * weiter wenn curx(a5) > maxzeile
000CA2  B03C 0050       1102.           cmp.b #80.d0
000CA6  6700 0006       1103.           beq edit1ctd    * naechste Zeile
000CAA  6000 F372       1104.           bra editlp
                        1105.
000CAE                  1106. edit1ctd:                 * naechste Zeile nehmen falls moeglich
000CAE  422D'0000       1107.           clr.b curx(a5)
000CB2  6000 0066       1108.           bra editctx     * wie Zeile nach unten jedoch cursorx=0
                        1109.
                        1110.
000CB6                  1111. editcte:                  * Cursor nach oben
000CB6  4A2D'0000       1112.           tst.b curv(a5)  * = 0 dann scrollen
000CBA  670C            1113.           beq.s edit1cte
000CBC  532D'0000       1114.           subq.b #1.curv(a5)      * subrahieren damit nach oben
000CC0  6100 0490       1115.           bsr curtoend
000CC4  6000 F358       1116.           bra editlp
                        1117.
000CC8                  1118. edit1cte:
000CC8  6100 03FA       1119.           bsr crtdown     * prev zeile holen
000CCC  6100 0484       1120.           bsr curtoend
000CD0  6000 F34C       1121.           bra editlp
                        1122.
000CD4                  1123. editctg:
000CD4  6100'F32A       1124.           bsr delchar
000CD8  6000 F344       1125.           bra editlp
                        1126.
000CDC                  1127. editctn:
000CDC  6100 0358       1128.           bsr bottoln
000CE0  6100'F31E       1129.           bsr insline
000CE4  6000 F338       1130.           bra editlp
```

```
                                1131.
000CE8                          1132. editctr:                  * CRTL R   seite zurueck
000CE8  323C 000E               1133.           move #15-1,d1   * 15 weiter
000CEC                          1134. editlctr:
000CEC  3F01                    1135.           move d1,-(a7)
000CEE  6100 03D4               1136.           bsr crtdown
000CF2  321F                    1137.           move (a7)+,d1
000CF4  51C9 FFF6               1138.           dbra d1,editlctr
000CF8  6000 F324               1139.           bra editlo
                                1140.
                                1141.
000CFC                          1142. editctt:                  * Loesche bis Zeilenende
000CFC  6100 F302               1143.           bsr eraeoln     * nicht wie wordstar nur ein Wort
000D00  6000 F31C               1144.           bra editlo
                                1145.
000D04                          1146. editctw:
000D04  6100 03BE               1147.           bsr crtdown
000D08  0C2D 0016               1148.           cmp.b #22,curv(a5)
        0000
000D0E  6400 F30E               1149.           bcc editlo
000D12  522D 0000               1150.           addq.b #1,curv(a5)              *
000D16  6000 F306               1151.           bra editlo              * cursor bleibt auf Text-Position
                                1152.
000D1A                          1153. editctx:                  * Cursor down
000D                            1154.           cmp.b #22,curv(a5)      * wenn in bereich 22..n dann spez
        0000
000D20  640C                    1155.           bcc.s editlctx
000D22  522D 0000               1156.           addq.b #1,curv(a5)
000D26  6100 042A               1157.           bsr curtoend
000D2A  6000 F2F2               1158.           bra editlo
                                1159.
000D2E                          1160. editlctx:                 * Sonderfall cursor schon unten
000D2E  6100 0374               1161.           bsr crtuo
000D32  6100 041E               1162.           bsr curtoend
000D36  6000 F2E6               1163.           bra editlo
                                1164.
                                1165.
000D3A                          1166. editctv:
000D3A  6100 F2C4               1167.           bsr delline             * aktuelle Zeile
000D3E  6100 032E               1168.           bsr lntobott
000D42  422D 0000               1169.           clr.b curx(a5)          * zurueck an alte Zeile
000D46  6000 F2D6               1170.           bra editlo
                                1171.
000D4A                          1172. editctz:
000D4A  6100 0358               1173.           bsr crtuo
000D4E  4A2D 0000               1174.           tst.b curv(a5)          * decr
000D52  6700 F2CA               1175.           beq editlo
000D56  532D 0000               1176.           subq.b #1,curv(a5)
000D5A  6000 F2C2               1177.           bra editlo              * cursor bleibt auf Text-Position
                                1178.
                                1179.
000D5E                          1180. editdel:
000D5E  4A2D 0000               1181.           tst.b curx(a5)          * =0 dann vorherige Zeile
000D62  670C                    1182.           beq.s editldel
000D64  532D 0000               1183.           subq.b #1,curx(a5)
000D68  6100 F296               1184.           bsr delchar     * erste verringern, dann loeschen
000D6C  6000 F2B0               1185.           bra editlo
                                1186.
000D70                          1187. editldel:                 * oof transport +++
000D70  6000 F2AC               1188.           bra editlo
                                1189.
000D                            1190. editinsl:
000D74  0A2D 0001               1191.           eor.b #1,insl(a5)       * 0->1  1->0
        0000
000D7A  6000 F2A2               1192.           bra editlo
                                1193.
000D7E                          1194. editcr:                   * bei INSL  crlf mit einfuegen
                                1195.                           * sonst nur cr lf ausser
                                1196.                           * wenn letzte Zeile
000D7E  082D 0000               1197.           btst.b #0,insl(a5)      * =  1 dann spezial
        0000
000D84  6600 0024               1198.           bne editlcr
000D88  422D 0000               1199.           clr.b curx(a5)  * auf 0 oos
000D8C  0C2D 0016               1200.           cmp.b #22,curv(a5)      * >= 22 dann scroll mode anwenden
        0000
000D92  6408                    1201.           bcc.s editlcr
000D94  522D 0000               1202.           addq.b #1,curv(a5)      * sonst nur weiter
000D98  6000 F284               1203.           bra editlo
000D9C                          1204. editlcr:
000D9C  6100 0306               1205.           bsr crtuo
```

```
000DA0  1B7C 0016 '0000      1206.          move.b #22.curv(a5)
000DA6  6000 F276            1207.          bra editlo
                             1208.
000DAA                       1209. edit1cr:                  * cr mit insert mode
000DAA  422D'0000            1210.          clr.b curx(a5)
000DAE  0C2D 0016 '0000      1211.          cmp.b #22.curv(a5)
000DB4  6406                 1212.          bcc.s edit3cr
000DB6  522D'0000            1213.          addq.b #1.curv(a5)
000DBA  600A                 1214.          bra.s edit4cr
                             1215.
000DBC                       1216. edit3cr:
000DBC  6100 02E6            1217.          bsr crtup
000DC0  1B7C 0016 '0000      1218.          move.b #22.curv(a5)
000DC6                       1219. edit4cr:
000DC6  6100 026E            1220.          bsr bottoln      * retten der unteren Zeile
000DCA  6100'F234            1221.          bsr insline
000DCE  6000 F24E            1222.          bra editlo
                             1223.
                             1224.
000DD2                       1225. crtstatus:                * Statuszeile
                             1226.                           * wird auf zwei Bildseiten geschrieben
                             1227.                           * Erneuern nur nach CLEAR
000DD2  1B7C 0001 '0000      1228.          move.b #1.wrtpage(a5)
000DD8  6100'F226            1229.          bsr aktpage
000DDC  6100 005C            1230.          bsr crtists
000DE0  422D'0000            1231.          clr.b wrtpage(a5)
000DE4  6100'F21A            1232.          bsr aktpage
000DE8  6100 0050            1233.          bsr crtists
000DEC  4E75                 1234.          rts
                             1235.
000DEE                       1236. cwaus:           * a1-> Text bis 0
                             1237.                  * Ausgabe nach a0
000DEE  1019                 1238.          move.b (a1)+.d0
000DF0  6700'F20E            1239.          beq carres
000DF4  10C0                 1240.          move.b d0.(a0)+
000DF6  60F6                 1241.          bra cwaus
                             1242.
000DF8  54 65 78 74          1243. txtstx: dc.b 'Textstart='.0
        73 74 61 72 74
        3D 00
000E03  20 46 65 6E          1244. txtakt: dc.b ' Fenster=' .0
        73 74 65 72 3D
        00
000E0D  20 54 6F 72          1245. txtetx: dc.b ' Tor='.0
        3D 00
000E13  20 65 69 6E          1246. txteinf:  dc.b ' einf'.0
        66 00
000E19  20 20 20 20          1247. txtneinf: dc.b '      '.0
        20 00
000E1F  20 64 65 75          1248. txtdeut:  dc.b ' deut'.0
        74 00
000E25  20 61 6D 65          1249. txtamer: dc.b ' amer'.0
        72 00
                             1250.
                             1251.
000E2B  20 43 54 52          1252. txthelp: dc.b ' CTRL-J=Hilfe'.0
        4C 2D 4A 3D 48
        69 6C 66 65 00
000E39  00                   1253.          dc.b 0
000E3A                       1254.          ds 0             * ALIGN
                             1255.
000E3A                       1256. crtists:
000E3A  6100'F1C4            1257.          bsr setpen       * pen down
000E3E  4241                 1258.          clr d1           * x=0
000E40  4242                 1259.          clr d2           * v=0
000E42  6100'F1BC            1260.          bsr moveto
000E46  323C 01FF            1261.          move #511.d1     * x=511
000E4A  6100'F1B4            1262.          bsr drawto
000E4E  343C 000A            1263.          move #10.d2      * v= 10
000E52  6100'F1AC            1264.          bsr drawto
000E56  4241                 1265.          clr d1           * x=0
000E58  6100'F1A6            1266.          bsr drawto
000E5C  4242                 1267.          clr d2           * v=0
000E5E  6100'F1A0            1268.          bsr drawto
                             1269.                           * Ausgabe der Statusinfos HIER
000E62  41ED'0028            1270.          lea ausbuf+40(a5).a0    * Anfang fuer Suche reserviert
```

```
000E66  43FA FF90      1271.          lea txtstx(pc).a1
000E6A  6182           1272.          bsr cwaus
000E6C  202D'0000      1273.          move.l stxtxt(a5).d0
000E70  6100'F18E      1274.          bsr print6x
000E74  43FA FF8D      1275.          lea txtakt(pc).a1
000E78  6100 FF74      1276.          bsr cwaus
000E7C  202D'0000      1277.          move.l akttxt(a5).d0
000E80  6100'F17E      1278.          bsr print6x
000E84  43FA FF87      1279.          lea txtetx(pc).a1
000E88  6100 FF64      1280.          bsr cwaus
000E8C  202D'0000      1281.          move.l etxtxt(a5).d0
000E90  6100'F16E      1282.          bsr print6x
000E94  43FA FF7D      1283.          lea txteinf(pc).a1
000E98  082D 0000      1284.          btst.b #0.insl(a5)
        0000
000E9E  6604           1285.          bne.s crt3sts
000EA0  43FA FF77      1286.          lea txtneinf(pc).a1
000EA4                 1287. crt3sts:
000EA4  6100 FF48      1288.          bsr cwaus
000EA8  43FA FF7B      1289.          lea txtamer(pc).a1
000EAC  4A2D'0000      1290.          tst.b ootflag(a5)
000EB0  6704           1291.          beq.s crt4sts
000EB2  43FA FF6B      1292.          lea txtdeut(pc).a1
000E[illegible]        1293. crt4sts:
000[illegible]  6100 FF36      1294.          bsr cwaus
000E[illegible]  43FA FF6F     1295.          lea txthelo(pc).a1
000EBE  6100 FF2E      1296.          bsr cwaus
000EC2  4210           1297.          clr.b (a0)        * ENDE Zeichen
000EC4  41ED'0028      1298.          lea ausbuf+40(a5).a0 * Anfang fuer Suche reserviert
000EC8  303C 0011      1299.          move #$11.d0
000ECC  323C 0003      1300.          move #3.d1        * x=3
000ED0  343C 0001      1301.          move #1.d2        * y=1
000ED4  6100'F12A      1302.          bsr textaus
000ED8  4E75           1303.          rts
                       1304.
                       1305.
000EDA                 1306. otzurueck:                 * AKTTXT eine Zeile zurueck
                       1307.                            * Carry wenn nicht moeglich
000EDA  206D'0000      1308.          movea.l akttxt(a5).a0   * auf aktuellen Text
000EDE  B1ED'0000      1309.          cmpa.l stxtxt(a5).a0    * wenn gleich dann fertig
000EE2  6700'F11C      1310.          beq carset        * CARRY da keine Verschiebung moeglich
000EE6                 1311. ot1zur:
000EE6  5388           1312.          subq.l #1.a0      * Zeiger zurueck
000EE8  B1ED'0000      1313.          cmpa.l stxtxt(a5).a0    * wenn gleich dann fertig
000EEC  6708           1314.          beq.s ot2zur
000EEE  0C28 000A      1315.          cmp.b #$A.-1(a0)        * wenn Vorgaenger = LF dann fertig
        FFFF
000EF4  66F0           1316.          bne ot1zur
000EF6                 1317. ot2zur:
000EF6  2B48'0000      1318.          move.l a0.akttxt(a5)    * Neuer Zeiger
000EFA  6000'F104      1319.          bra carres              * kein CARRY
                       1320.
000EFE                 1321. otvor:                     * AKTTXT eine Zeile vor
                       1322.                            * carry wenn nicht moeglich
000EFE  206D'0000      1323.          movea.l akttxt(a5).a0   * start text
000F02  B1ED'0000      1324.          cmpa.l etxtxt(a5).a0
000F06  6700'F0F8      1325.          beq carset        * ist schon am Ende
000F0A                 1326. ot1vor:
000F0A  5288           1327.          addq.l #1.a0
000F0C  B1ED'0000      1328.          cmpa.l etxtxt(a5).a0
000F[illegible]  6700 000A     1329.          beq ot2vor
000F[illegible]  0C28 000A     1330.          cmp.b #$a.-1(a0) * wenn Vorgaenger LF dann ok
        FFFF
000F1A  66EE           1331.          bne ot1vor
000F1C                 1332. ot2vor:
000F1C  2B48'0000      1333.          move.l a0.akttxt(a5)
000F20  6000'F0DE      1334.          bra carres
                       1335.
                       1336.                            * REV 4.1 High Speed Transfer direkter au
000F24                 1337. lntocrt:                   * Zeile von Speicher auf Bildschirm holen
                       1338.                            * Zeile wird im Speicher geloescht
                       1339.                            * d0 = curv(a5) Zeile der Einfuegung
000F24  1F2D'0000      1340.          move.b curx(a5).-(a7)
000F28  1F2D'0000      1341.          move.b curv(a5).-(a7)   * ootflag braucht nicht gerettet
000F2C  422D'0000      1342.          clr.b curx(a5)          * auf Zeilenstart
000F30  1B40'0000      1343.          move.b d0.curv(a5)      * neuer Cursor
000F34  6100'F0CA      1344.          bsr calccur             * a1->Zeichenzahl a2->Position Ze
000F38  206D'0000      1345.          movea.l akttxt(a5).a0   * Zeiger auf Start
000F3C  323C 004F      1346.          move #80-1.d1           * Zaehler
000F40  4242           1347.          clr d2                  * Zeichenzaehler
```

```
000F42
000F42  1018            1348. ln1tocrt:
000F44  B03C 0020       1349.         move.b (a0)+,d0
000F48  6508            1350.         cmp.b #' '.d0
                        1351.         bcs.s ln2tocrt          * Abbruch bei cr lf o.ae.
                        1352.                                 * REV 4.1 direkte Ausgabe
000F4A  14C0            1353.         move.b d0.(a2)+         * in Zeichenbuffer. dann Ausgabe
000F4C  5202            1354.         addq.b #1.d2            * Und Zeichenzaehler erhoehen
000F4E  51C9 FFF2       1355.         dbra d1.ln1tocrt
000F52                  1356. ln2tocrt:
000F52  1282            1357.         move.b d2.(a1)  * Neuer Zeichenzaehlerstand
000F54  1B7C 0001       1358.         move.b #1.wrtpage(a5)
        0000
000F5A  422D'0000       1359.         clr.b viewpage(a5)
000F5E  6100'F0A0       1360.         bsr aktpage     * Zeilen waren vorher auf dem Bildschirm 1
000F62  6100'F09C       1361.         bsr aktcur
000F66  6100'F098       1362.         bsr ortline     * REV 4.1 High-Speed-Ausgabe
000F6A  422D'0000       1363.         clr.b wrtpage(a5) * Ausgabe auf beide Bildseiten
000F6E  6100'F090       1364.         bsr aktpage     *
000F72  6100'F08C       1365.         bsr aktcur
000F76  6100'F088       1366.         bsr ortline     * Zeilen auf beide Seiten ausgeben
000F7A  4242            1367.         clr d2          * Zeichenzaehler fuers loeschen
000F7C  206D'0000       1368.         movea.l akttxt(a5).a0   * alter Zeiger bleibt
000F80                  1369. ln3tocrt:
000F80  4A10            1370.         tst.b (a0)      * = 0 dann stop zaehlen
000F82  6708            1371.         beq.s ln4tocrt
000F84  5242            1372.         addq #1.d2      * Zaehler + 1
000F86  0C18 000A       1373.         cmp.b #$a.(a0)+ * wenn =lf dann abbruch
000F8A  66F4            1374.         bne ln3tocrt    * A0 zeigt auf freien platz = quelle
000F8C                  1375. ln4tocrt:               * d2=0 dann keine Zeichen
000F8C  226D'0000       1376.         movea.l akttxt(a5).a1   * ziel = a1
000F90  202D'0000       1377.         move.l eottxt(a5).d0
000F94  2208            1378.         move.l a0.d1    * 8 - 5 = 3 zeichen. aber 4 uebertrg
000F96  9081            1379.         sub.l d1.d0     * eot - akttxt = anzahl-1 in D0
000F98                  1380. ln5tocrt:
000F98  12D8            1381.         move.b (a0)+.(a1)+      * nach ziel
000F9A  51C8 FFFC       1382.         dbra d0.ln5tocrt
000F9E  222D'0000       1383.         move.l eottxt(a5).d1
000FA2  48C2            1384.         ext.l d2
000FA4  9282            1385.         sub.l d2.d1
000FA6  2B41'0000       1386.         move.l d1.eottxt(a5)
000FAA  222D'0000       1387.         move.l etxtxt(a5).d1
000FAE  9282            1388.         sub.l d2.d1
000FB0  2B41'0000       1389.         move.l d1.etxtxt(a5)
000FB4  1B5F'0000       1390.         move.b (a7)+.curv(a5)
000FB8  1B5F'0000       1391.         move.b (a7)+.curx(a5)
000FBC  4E75            1392.         rts
                        1393.
000FBE                  1394.
                        1395. crttoln:                  * Zeile von Bildschirm nach Speicher
                        1396.                           * Zeile wird im Speicher eingefuegt
                        1397.                           * d0 = curv(a5) Zeile die geloscht wird
000FBE                  1398.                           * d2 = anzahl der eingebauten Zeichen
000FBE  1F2D'0000       1399.         move.b curx(a5).-(a7)
000FC2  1F2D'0000       1400.         move.b curv(a5).-(a7)
000FC6  422D'0000       1401.         clr.b curx(a5)  * auf Zeilenstart
000FCA  1B40'0000       1402.         move.b d0.curv(a5)      * neuer Cursor
000FCE  6100'F030       1403.         bsr calccur     * (a1).b = anzahl.a0 = zeilenstart
000FD2  4282            1404.         clr.l d2        * ergebnis auf 0 setzen
000FD4  0C10 007F       1405.         cmp.b #$7f.(a0) * MARKE FUER NIL
000FD8  6748            1406.         beq.s crtfln    * sonst auch leer Zeile als CRLF ausgeben
000FDA  246D'0000       1407.         movea.l eottxt(a5).a2   * quelle
000FDE  220A            1408.         move.l a2.d1
000FE0  4280            1409.         clr.l d0        * alles loeschen
000FE2  1011            1410.         move.b (a1).d0  * zeichen auch wenn keine da. ausgeben
000FE4  5480            1411.         addq.l #2.d0    * + CR + LF
000FE6  2400            1412.         move.l d0.d2    * D2 ist temo Anzahl  GLOBAL ALS ERGEBNIS
000FE8  D081            1413.         add.l d1.d0
000FEA  2640            1414.         movea.l d0.a3   * Ziel
000FEC  92AD'0000       1415.         sub.l akttxt(a5).d1     * ende-anf -> n-1 anzahl transfers
000FF0  528A            1416.         addq.l #1.a2
000FF2  528B            1417.         addq.l #1.a3    *
000FF4                  1418. crtlln:
000FF4  1722            1419.         move.b -(a2).-(a3)
000FF6  51C9 FFFC       1420.         dbra d1.crtlln * bis fertig
000FFA  4241            1421.         clr d1          * neuer Zaehler
000FFC  1211            1422.         move.b (a1).d1  * fuer Uebertragung
000FFE  6002            1423.         bra.s crll21    * 0= keinmal
01000                   1424. crtl2ln:
01000  14D8             1425.         move.b (a0)+.(a2)+      * ziel ist leerer Platz
01002                   1426. crll21:
```

```
00:202  51C9 FFFC      1427.          dbra d1.crtl2ln
001006  14FC 000D      1428.          move.b #$d.(a2)+
00100A  14FC 000A      1429.          move.b #$a.(a2)+          * CR LF als Abschluss
00100E  222D 0000      1430.          move.l eottxt(a5).d1
001012  D282          1431.          add.l d2.d1
001014  2B41 0000      1432.          move.l d1.eottxt(a5)
001018  222D 0000      1433.          move.l etxtxt(a5).d1
00101C  D282          1434.          add.l d2.d1
00101E  2B41 0000      1435.          move.l d1.etxtxt(a5)
001022                 1436. crtfln:                            * D2 ist Anzahl Zeichen
001022  1B5F 0000      1437.          move.b (a7)+.curv(a5)
001026  1B5F 0000      1438.          move.b (a7)+.curx(a5)
00102A  4E75           1439.          rts
                       1440.
                       1441.
00102C                 1442. toptoln:                           * zeile 0 an speicher geben
                       1443.                                    * AKTTXT zeigt dahinter
00102C  4240           1444.          clr d0
00102E  618E           1445.          bsr crttoln
001030  D5AD 0000      1446.          add.l d2.akttxt(a5)       * drauf addieren zeit auf Next
001034  4E75           1447.          rts
                       1448.
001036                 1449. bottoln:                           * zeile 23 an speicher geben
                       1450.                                    * AKTTXT zeigt darauf
[illegible]  7017      1451.          moveq #23.d0              *
[illegible]  6184      1452.          bsr crttoln
00103A  4E75           1453.          rts                       * AKTTXT nicht veraendern
                       1454.
                       1455.
00103C                 1456. lntotop:                           * Text vor AKTTXT an CRT Zeile 0
00103C  6100 FE9C      1457.          bsr otzurueck             * erst mal zurueck
001040  6500 000A      1458.          bcs lntotop
001044  4240           1459.          clr d0
001046  6100 FEDC      1460.          bsr lntocrt               * sonst ausgabe moeglich
00104A  4E75           1461.          rts
00104C                 1462. lntotop:                           * NIL einbauen als Kennung
00104C  1F2D 0000      1463.          move.b curx(a5).-(a7)
001050  1F2D 0000      1464.          move.b curv(a5).-(a7)
001054  422D 0000      1465.          clr.b curx(a5)
001058  422D 0000      1466.          clr.b curv(a5)
00105C  103C 007F      1467.          move.b #$7f.d0
001060  6100 EF9E      1468.          bsr charoscr              * dort ausgeben
001064  1B5F 0000      1469.          move.b (a7)+.curv(a5)
001068  1B5F 0000      1470.          move.b (a7)+.curx(a5)
00106C  4E75           1471.          rts
                       1472.
                       1473.
                       1474.
00106E                 1475. lntobott:                          * text bei AKTTXT an CRT Zeile 23
00106E  202D 0000      1476.          move.l akttxt(a5).d0
001072  B0AD 0000      1477.          cmp.l etxtxt(a5).d0       * wenn gleich dann del ausgeben
001076  6708           1478.          beq.s lntobott
001078  7017           1479.          moveq #23.d0
00107A  6100 FEAB      1480.          bsr lntocrt               * auf letzte Zeile bringen
00107E  4E75           1481.          rts                       * falls was da ist
001080                 1482. lntobott:
001080  1F2D 0000      1483.          move.b curx(a5).-(a7)
001084  1F2D 0000      1484.          move.b curv(a5).-(a7)
001088  422D 0000      1485.          clr.b curx(a5)
00108C  1B7C 0017      1486.          move.b #23.curv(a5)
        0000
[illegible]  103C 007F 1487.          move.b #$7f.d0
001096  6100 EF6B      1488.          bsr charoscr              * dort ausgeben
00109A  1B5F 0000      1489.          move.b (a7)+.curv(a5)
00109E  1B5F 0000      1490.          move.b (a7)+.curx(a5)
0010A2  4E75           1491.          rts
                       1492.
                       1493.
0010A4                 1494. crtuo:
0010A4  1F2D 0000      1495.          move.b curv(a5).-(a7)
0010A8  1F2D 0000      1496.          move.b curx(a5).-(a7)
0010AC  6100 FF7E      1497.          bsr toptoln
0010B0  422D 0000      1498.          clr.b curv(a5)  * Zeile 0
0010B4  6100 EF4A      1499.          bsr delline     * loeschen
0010B8  61B4           1500.          bsr lntobott    * letzte Zeile holen
0010BA  1B5F 0000      1501.          move.b (a7)+.curx(a5)
0010BE  1B5F 0000      1502.          move.b (a7)+.curv(a5)
0010C2  4E75           1503.          rts
                       1504.
0010C4                 1505. crtdown:
```

```
0010C4  1F2D'0000       1506.           move.b curv(a5).-(a7)
0010C8  1F2D'0000       1507.           move.b curx(a5).-(a7)
0010CC  6100 FF68       1508.           bsr bottoln
0010D0  422D'0000       1509.           clr.b curv(a5)  * Zeile 0 einfuegen
0010D4  6100'EF2A       1510.           bsr insline
0010D8  6100 FF62       1511.           bsr lntotoo
0010DC  1B5F'0000       1512.           move.b (a7)+.curx(a5)
0010E0  1B5F'0000       1513.           move.b (a7)+.curv(a5)
0010E4  4E75            1514.           rts
                        1515.
                        1516.
0010E6                  1517. getscreen:                * Bildschirm laden von
                        1518.                           * AKTTXT an
                        1519.                           * Rev 3.0 High Speed
0010E6  6100'EF18       1520.           bsr clrscreen   * zuerst loeschen
0010EA  4240            1521.           clr d0          * Start bei Zeile 0
0010EC                  1522. getlscr:
0010EC  3F00            1523.           move d0.-(a7)
0010EE  202D'0000       1524.           move.l akttxt(a5).d0
0010F2  B0AD'0000       1525.           cmo.l etxtxt(a5).d0     * Wenn Ende erreicht dann DEL ausg
0010F6  670A            1526.           beq.s get2scr
0010F8  301F            1527.           move (a7)+.d0
0010FA  3F00            1528.           move d0.-(a7)
0010FC  6100 FE26       1529.           bsr lntocrt     * Ausgabe auf dem Bildschirm
010100  6024            1530.           bra.s get3scr
                        1531.
010102                  1532. get2scr:
010102  301F            1533.           move (a7)+.d0
010104  3F00            1534.           move d0.-(a7)
010106  1F2D'0000       1535.           move.b curx(a5).-(a7)
01010A  1F2D'0000       1536.           move.b curv(a5).-(a7)
01010E  422D'0000       1537.           clr.b curx(a5)
010112  1B40'0000       1538.           move.b d0.curv(a5)      * aktuelle Position
010116  103C 007F       1539.           move.b #$7f.d0
01011A  6100'EEE4       1540.           bsr charoscr    * dort ausgeben
01011E  1B5F'0000       1541.           move.b (a7)+.curv(a5)
010122  1B5F'0000       1542.           move.b (a7)+.curx(a5)
010126                  1543. get3scr:
010126  301F            1544.           move (a7)+.d0
010128  5240            1545.           add #1.d0
01012A  B07C 0018       1546.           cmo #24.d0      * 0..23
01012E  66BC            1547.           bne getlscr
010130  422D'0000       1548.           clr.b curv(a5)
010134  422D'0000       1549.           clr.b curx(a5)  * auf Homeposition
010138  4E75            1550.           rts
                        1551.
0113A                   1552. outscreen:                * Bildschirm zurueck
                        1553.                           * rev 3.0  high speed
                        1554.                           * an aktuelle Stelle
0113A  4240             1555.           clr d0          * Start Zeile 0
0113C                   1556. outlscr:
0113C  3F00             1557.           move d0.-(a7)
0113E  6100 FE7E        1558.           bsr crttoln
010142  D5AD'0000       1559.           add.l d2.akttxt(a5)     * Weiter schalten
010146  301F            1560.           move (a7)+.d0
010148  5240            1561.           add #1.d0       * 0 bis 23
01014A  B07C 0018       1562.           cmo #24.d0      * AM SCHLUSS
01014E  66EC            1563.           bne outlscr     * WIRD NICHT GELOESCHT
010150  4E75            1564.           rts             *
                        1565.
                        1566. * aufruf bei allen curx(a5) setzbefehlen mit Spezbedeutung
                        1567.
010152                  1568. curtoend:                  * cursor auf ende
010152  6100'EEAC       1569.           bsr calccur     * a1-> anzahlpointer
010156  102D'0000       1570.           move.b curx(a5).d0      * alte position
01015A  B011            1571.           cmo.b (a1).d0   *
01015C  6500'EEA2       1572.           bcs carres      * wenn curx(a5) groesser dann ok
010160  1B51'0000       1573.           move.b (a1).curx(a5)    * sonst neu setzen
                        1574.           *
010164                  1575. adjcurx:                  * wenn > 79 dann zurueecksetzen
010164  0C2D 0050       1576.           cmo.b #80.curx(a5)
        '0000
01016A  6500'EE94       1577.           bcs carres      * 0..79 ok
01016E  1B7C 004F       1578.           move.b #79.curx(a5)
        '0000
010174  6000'EE8A       1579.           bra carset
                        1580.
                        1581.
                        1582.
010178                  1583.           end
```

```
...A68K 4.0a 06/13/83  ...Run on 01/22/85
                        1.
                        2. ****************************************
                        3. * Assemblermodul  fuer Grundprogramm    *
                        4. * (C) 1984  Rolf-Dieter Klein          *
                        5. * Version 1.0 830816                  *
                        6. * Verwendet Unterprogramme fuer Symbole*
                        7. * aus dem Grundprogramm               *
                        8. * rev 2.0   3.12.83                   *
                        9. * rev 3.1  28.2.84  RDK               *
                       10. * rev 4.0 -- RDK    840903           *
                       11. *              ds.l war bisher falsch *
                       12. * rev 4.0b  relokative Version        *
                       13. * rev 4.1                            *
                       14. * rev 4.2 ADDQ #8. und Seitennr./4.3  *
                       15. ****************************************
                       16. * fuer Linker
                       17.
                       18.          xdef assemble
                       19.          xref stxtxt,akttxt,etxtxt,eottxt
                       20.          xref ci2,co2,ciinit2,prtco2
                       21.          xref carres,carset,expr
                       22.          xref errart,errpoi,errcnt,errflag,pcstand,anfstand
                       23.          xref einbuf,ausbuf,passflag,befcode
                       24.          xref attcode,wordbyte,iostat,zeilen
                       25.          xref curon,flip,flip1
                       26.          xref clrscreen,setupname
                       27.          xref bucheck,setval,getval,newval
                       28.          xref ci,csts,igbn
                       29.          xref symtab,symnext,pcorg
                       30.          xref auszahl,auspoi
                       31.          xref debug,debugst,debugak
                       32.          xref offset
                       33.          xref pagecnt,pagebuf,print4d
                       34.
000000                 35.          section 1
                       36.
                       37. * Definitionen
                       38.
                       39. * Format fuer Ausgabe
                       40. * aaaaaa  bbbbbbbbbbbbbb
                       41. * 012345678901234567890123
0000001B               42. insst    equ 24          * Start Befehlsteil
                       43.
                       44. * rev 3.1   expr ohne igbn
000000                 45. expr1:
000000  6100'FFFE     46.          bsr igbn        * rev 3.1
000004  323C FFFF     47.          move #$ffff,d1
000008  6100'FFF6     48.          bsr expr        * d1 vorbelegen wegen symbolmodi
00000C  4A41           49.          tst d1
00000E  6100 1E9E      50.          bsr erresyntax  * wenn = 0 dann syntaxfehler
000012  B27C 0005      51.          cmp #5.d1
000016  6100 1ED8      52.          bsr erreundef   * undefiniert
00001A  4E75           53.          rts
                       54.
                       55. * textstart ist STXTXT
                       56.
00001C                 57.
                       58. geteinbuf:               * Eingabebuffer fuellen
                       59.                          * von Quelle
00[illegible]  41ED'0000     60.          lea einbuf(a5),a0
[illegible]  323C 0083      61.          move #132-1,d1  * Zeichenzaehler max.
000024  6100'FFDA     62. getllp: bsr ci2          * Zeichen holen ---
000028  B03C 000A      63.          cmp.b #$a,d0
00002C  67F6           64.          beq.s getllp    * Ignore LFs
00002E  B03C 0027      65.          cmp.b #$27,d0   * Anfuehrungszeichen
000032  6720           66.          beq.s get2ff
000034                 67. getlff:
000034  6100'FFCA     68.          bsr bucheck     * wandelt in Grossbuchstaben um
000038  10C0           69.          move.b d0,(a0)+ * und dort ablegen
00003A  B03C 000D      70.          cmp.b #$d,d0    * CR ist Endezeichen
00003E  57C9 FFE4      71.          dbeq d1,getllp
000042  4E75           72.          rts             *
                       73.
000044                 74. get2lp:                  * Bei Text keine Uebersetzung klein-gross
000044  6100'FFBA     75.          bsr ci2
000048  B03C 000A      76.          cmp.b #$a,d0
00004C  67F6           77.          beq get2lp
00004E  B03C 0027      78.          cmp.b #$27,d0
000052  67E0           79.          beq.s getlff
```

```
000054                          80. get2ff:
000054  10C0                    81.         move.b d0,(a0)+
000056  B03C 000D               82.         cmp.b #$d,d0
00005A  57C9 FFE8               83.         dbeq d1,get2lp
00005E  4E75                    84.         rts
                                85.
000060                          86. outausbuf:              * Ausgabebuffer ausgeben
000060  41ED'0000               87.         lea ausbuf(a5),a0
000064  323C 0083               88.         move #132-1,d1
000068  101B                    89. put1lp: move.b (a0)+,d0 * holen zeichen
00006A  6100'FF94               90.         bsr co2         * ausgeben   - d0 bleibt
00006E  B03C 000D               91.         cmp.b #$d,d0
000072  57C9 FFF4               92.         dbeq d1,put1lp
000076  103C 000A               93.         move.b #$A,d0
00007A  6100'FF84               94.         bsr co2         * CR LF
                                95.                         * REV 3.1  hier Abfrage wichtig
                                96.                         * da Zeilen-Ueberlauf moeglich.bei dc.x
00007E  0C2D 0042               97.         cmp.b #66,zeilen(a5)
        0000
000084  650C                    98.         bcs.s pu1ee
000086  103C 000C               99.         move.b #$c,d0
00008A  6100'FF74              100.         bsr co2
00008E  6100 0642              101.         bsr kopfaus     * Neue Ueberschrift setzen dannach
000092                         102. pu1ee:
000092  4E75                   103.         rts
                               104.
000094                         105. nachausbuf:            * Transfer von Befehl nach Ausgabe
                               106.                        * Buffer und vorloeschen mit Leerz.
000094  41ED'0000              107.         lea ausbuf(a5),a0
000098  323C 0017              108.         move #insst-1,d1        *zaehler
00009C                         109. nach1lp:
00009C  10FC 0020              110.         move.b #$20,(a0)+
0000A0  51C9 FFFA              111.         dbra d1,nach1lp * ok geloescht
0000A4  323C 006B              112.         move #132-insst-1,d1
0000A8  43ED'0000              113.         lea einbuf(a5),a1       *Quelle
0000AC                         114. nach2lp:
0000AC  10D9                   115.         move.b (a1)+,(a0)+
0000AE  51C9 FFFC              116.         dbra d1,nach2lp
0000B2  4E75                   117.         rts
                               118.
                               119.
0000B4                         120. setigname:             * wie setup jedoch mit leerzeichen
0000B4  6100'FF4A              121.         bsr igbn
0000B8  6000'FF46              122.         bra setupname
                               123.
                               124.
                               125.
                               126. ** Routinen fuer Codeausgabe
                               127.
0000BC                         128. out8adr:
0000BC  4840                   129.         swap d0
0000BE  6100 000E              130.         bsr out4adr
0000C2  4840                   131.         swap d0
0000C4  6008                   132.         bra.s out4adr
                               133.
0000C6                         134. out6adr:               * 3 Bytes (d0) ausgeben
                               135.                        * ohne Terminatorzeichen
0000C6  4840                   136.         swap d0
0000C8  6100 000C              137.         bsr out2adr
0000CC  4840                   138.         swap d0
0000CE                         139. out4adr:               * 2 Bytes ausgeben
0000CE  E098                   140.         ror.l #8,d0
0000D0  6100 0004              141.         bsr out2adr
0000D4  E198                   142.         rol.l #8,d0
0000D6                         143. out2adr:               * 1 Bytes ausgeben
0000D6  E898                   144.         ror.l #4,d0
0000D8  6100 0004              145.         bsr out1adr
0000DC  E998                   146.         rol.l #4,d0
0000DE                         147. out1adr:
0000DE  48E7 8000              148.         movem.l d0,-(a7)
0000E2  C03C 000F              149.         and.b #$f,d0
0000E6  B03C 000A              150.         cmp.b #10,d0
0000EA  6B0A                   151.         bmi.s out11adr
0000EC  903C 000A              152.         sub.b #10,d0
0000F0  D03C 0041              153.         add.b #'A',d0
0000F4  6004                   154.         bra.s out12adr
0000F6                         155. out11adr:
0000F6  D03C 0030              156.         add.b #'0',d0
0000FA                         157. out12adr:
0000FA  10C0                   158.         move.b d0,(a0)+
```

```
0000FC  4CDF 0001       159.            movem.l (a7)+.d0
000100  4E75           160.            rts
                        161.
                        162.
                        163. ** Codegenerator mit automatischer Neuausgabe des Buffers
                        164. ** falls ein Ueberlauf droht
                        165. ** uebernimmt auch die Ausgabeverwaltung
                        166. ** Fester Speicherplatz enthaelt die Adressen etc.
                        167.
                        168. * ANFSTAND haelt den Anfang fest und ist nach GETQUELLE def.
                        169. * PCSTAND wird als Adresse uebernommen dorthin Zielcode
                        170. * AUSPOI ist eigener Ausgabepointer
                        171. * AUSZAHL temp Zaehler fuer Bytezahl pro Zeile
                        172. * a0 bleibt erhalten  d0 = datenwert
                        173.
000102                  174. initcode:                   * Internes Init  auch einmal bei Start
000102  41ED'0008       175.            lea ausbuf+8(a5).a0
000106  2B48'0000       176.            move.l a0.auspoi(a5)
00010A  3B7C 0008       177.            move #8.auszahl(a5)     * Loeschen Ausgabezahl
        '0000
000110  323C 0017       178.            move #insst-1.d1
000114  41ED'0000       179.            lea ausbuf(a5).a0
000118                  180. init1c:
000118  10FC 0020       181.            move.b #' '.(a0)+
00011C  51C9 FFFA       182.            dbra d1.init1c
000120  10FC 000D       183.            move.b #$d.(a0)+        * Endezeichen
000124  4E75           184.            rts
                        185.
000126                  186. endecode:                  * Am Ende aufrufen
000126  6100 FF38       187.            bsr outausbuf   * Ausgeben falls noch was da
00012A  4E75           188.            rts
                        189.
                        190.                            * Datenwert immer in D0
00012C                  191. outlong:                   * 4 Werte
00012C  48E7 FF80       192.            movem.l d0-d7/a0.-(a7)
000130  206D'0000       193.            movea.l auspoi(a5).a0
000134  0C6D 000F       194.            cmp #insst-9.auszahl(a5)
        '0000
00013A  6504           195.            bcs.s put11g
00013C  6100 0106       196.            bsr newput
000140  6100 FF7A       197. put11g: bsr put8adr
000144  10FC 0020       198.            move.b #' '.(a0)+
000148  2B48'0000       199.            move.l a0.auspoi(a5)
00014C  066D 0009       200.            add #9.auszahl(a5)
        '0000
000152  206D'0000       201.            movea.l pcstand(a5).a0
000156  2208           202.            move.l a0.d1
000158  0801 0000       203.            btst.l #0.d1
00015C  671E           204.            beq.s put12lg
00015E  5248           205.            addq #1.a0      * neues a0
000160  2F08           206.            move.l a0.-(a7) *rev 4.0
000162  41ED'0000       207.            lea einbuf(a5).a0
000166  2B48'0000       208.            move.l a0.errpoi(a5)
00016A  205F           209.            movea.l (a7)+.a0
00016C  3B7C 0001       210.            move #1.errflag(a5)     * Fehler ist dominant
        '0000
000172  3B7C 0005       211.            move #5.errart(a5)      *
        '0000
000178  526D'0000       212.            add #1.errcnt(a5)
00017C                  213. put12lg:
00017C  58AD'0000       214.            addq.l #4.pcstand(a5)
000180  D1ED'0000       215.            adda.l offset(a5).a0
000184  20C0           216.            move.l d0.(a0)+         * Muss auf Wortgrenze sein
000186  4CDF 01FF       217.            movem.l (a7)+.a0/d0-d7
00018A  4E75           218.            rts
                        219.
00018C                  220. outword:
00018C  48E7 FF80       221.            movem.l d0-d7/a0.-(a7)
000190  206D'0000       222.            movea.l auspoi(a5).a0
000194  0C6D 0013       223.            cmp #insst-5.auszahl(a5)
        '0000
00019A  6504           224.            bcs.s put21g
00019C  6100 00A6       225.            bsr newput
0001A0  6100 FF2C       226. put21g: bsr put4adr
0001A4  10FC 0020       227.            move.b #' '.(a0)+
0001A8  2B48'0000       228.            move.l a0.auspoi(a5)
0001AC  5A6D'0000       229.            add #5.auszahl(a5)
0001B0  206D'0000       230.            movea.l pcstand(a5).a0
0001B4  2208           231.            move.l a0.d1
0001B6  0801 0000       232.            btst.l #0.d1
```

```
0001BA  671E            233.            beq.s put13lg
0001BC  5248            234.            addq #1,a0      * neues a0
0001BE  2F08            235.            move.l a0,-(a7)
0001C0  41ED'0000       236.            lea einbuf(a5),a0
0001C4  2B48'0000       237.            move.l a0,errpoi(a5)
0001C8  205F            238.            movea.l (a7)+,a0
0001CA  3B7C 0001       239.            move #1,errflag(a5)     * Fehler ist dominant
        '0000
0001D0  3B7C 0005       240.            move #5,errart(a5)      *
        '0000
0001D6  526D'0000       241.            add #1,errcnt(a5)
0001DA                  242. put13lg:
0001DA  54AD'0000       243.            addq.l #2,pcstand(a5)
0001DE  D1ED'0000       244.            adda.l offset(a5),a0
0001E2  30C0            245.            move.w d0,(a0)+
0001E4  4CDF 01FF       246.            movem.l (a7)+.a0/d0-d7
0001E8  4E75            247.            rts
                        248.
                        249.
0001EA                  250. putbyte:
0001EA  48E7 FF80       251.            movem.l d0-d7/a0.-(a7)
0001EE  206D'0000       252.            movea.l auspoi(a5).a0
0001F2  0C6D 0015       253.            cmp #insst-3,auszahl(a5)
        '0000
0001F8  6504            254.            bcs.s put31g
0001FA  6100 0048       255.            bsr newput
0001FE  6100 FED6       256. put31g: bsr put2adr
000202  10FC 0020       257.            move.b #' ',(a0)+
000206  2B48'0000       258.            move.l a0,auspoi(a5)
00020A  566D'0000       259.            add #3,auszahl(a5)
00020E                  260. put51g:                            * weiter von putobyte
00020E  206D'0000       261.            movea.l pcstand(a5),a0
000212  52AD'0000       262.            addq.l #1,pcstand(a5)
000216  D1ED'0000       263.            adda.l offset(a5),a0
00021A  10C0            264.            move.b d0,(a0)+
00021C  4CDF 01FF       265.            movem.l (a7)+.d0-d7/a0
000220  4E75            266.            rts
                        267.
000222                  268. putobyte:                          * Ohne Leerzeichen
000222  48E7 FF80       269.            movem.l d0-d7/a0.-(a7)
000226  206D'0000       270.            movea.l auspoi(a5),a0
00022A  0C6D 0015       271.            cmp #insst-3,auszahl(a5)
        '0000
000230  6504            272.            bcs.s put41g
000232  6100 0010       273.            bsr newput
000236  6100 FE9E       274. put41g: bsr put2adr
00023A  2B48'0000       275.            move.l a0,auspoi(a5)    * pointer weiter
00023E  546D'0000       276.            add #2,auszahl(a5)
000242  60CA            277.            bra put51g
                        278.
                        279.
000244                  280. newput:                            * neuer Aufbau einer Zeile
000244  48E7 8000       281.            movem.l d0.-(a7)
000248  6100 FE16       282.            bsr putausbuf   * Ausgeben
00024C  6100 FEB4       283.            bsr initcode
000250  6100 0022       284.            bsr getq1
000254  4CDF 0001       285.            movem.l (a7)+.d0
000258  206D'0000       286.            movea.l auspoi(a5),a0   * Pointer aktuell
00025C  4E75            287.            rts
                        288.
00025E                  289. getquelle:                         * holen der Quelldaten
                        290.                                    * Ausgabe des alten Buffers
00025E  2B6D'0000       291.            move.l pcstand(a5).anfstand(a5)  * festhalten den Anfstand
        '0000
000264  6100 FDFA       292.            bsr putausbuf   * alten Buffer ausgeben
000268  6100 FDB2       293.            bsr geteinbuf
00026C  6100 FE94       294.            bsr initcode    * Loescht ausgabebuffer
000270  6100 FE22       295.            bsr nachausbuf  * Ausgabe aktueller PCSTAND
000274                  296. getq1:
000274  202D'0000       297.            move.l pcstand(a5).d0
000278  41ED'0000       298.            lea ausbuf(a5),a0
00027C  6100 FE48       299.            bsr put6adr     * 6 bytes PC ausgeben
000280  4E75            300.            rts             * Ausbuf vorbereitet
                        301.
                        302.
                        303. * GETEA   hole effektive Adresse
                        304. * A0 ist zeiger auf den Text-Buffer
                        305. * Ergebnis
                        306. * D0.L ist der Datenwert, falls einer da
                        307. * D1.W  = 0 bei Fehler
```

```
                    308. *       = 1 bei Byte. =2 bei Wort. =3 bei Langwort, =4 wenn ohne d
                    309. *       = 5 bei Forwaertsreferenz  .. schon so eintragen
                    310. *         bei equ. label dann als langwort codieren
                    311. * D2.W  = Modecode nach A68k
                    312. * D3.W  = Extensioncode fuer Index-Mode
                    313. * D4    = Bitcode fuer Assemblercheck SET OF ADRMODE
                    314. *       15  14  13  12  11  10   9   8   7   6   5   4   3   2
                    315. *             abL abW ind decr incr dat adr displ pca indx
                    316. *
                    317. *
                    318.
 00002000           319. absl    equ     $2000    * Absolute Longword
 00001000           320. absw    equ     $1000    * Absolute Word
 00000800           321. indir   equ     $0800    * Indirekt (An)
 00000400           322. decre   equ     $0400    * Decrmode -(An)
 00000200           323. incre   equ     $0200    * Incrmode (An)+
 00000100           324. dreg    equ     $0100    * Datareg Dn
 00000080           325. areg    equ     $0080    * Adressreg An
 00000040           326. disol   equ     $0040    * Displacement  wert( )
 00000020           327. pcadr   equ     $0020    * PC ADR wert(PC )
 00000010           328. indx    equ     $0010    * Indexadr wert( .Xi)
                    329.
 000282             330. getea:                   * Hole effektive Adresse  a0 -> buffer
 000282  6100 0016  331.         bsr getmain     * aufruf
 000[illegible]  6100 1C1E  332.         bsr errcsvntax  * wenn carry dann fehler
 000[illegible]  4A41       333.         tst d1
 00028C  6100 1C20  334.         bsr erresvntax
 000290  B27C 0005  335.         cmp #5.d1
 000294  6100 1C5A  336.         bsr erreundef
 000298  4E75      337.         rts             * ok
                    338.
 00029A             339. getmain:
 00029A  4284       340.         clr.l d4        * Menge = leer
 00029C  4283       341.         clr.l d3        * kein mode aktiv
 00029E  4282       342.         clr.l d2        * Modecode = 0
 0002A0  4281       343.         clr.l d1        * 0= Fehler defaultausgang
 0002A2  4280       344.         clr.l d0        * Wert = 0 annehmen
 0002A4  6100 FD5A  345.         bsr igbn        * Vorblanks ignorieren
 0002A8  0C10 002D  346.         cmp.b #'-',(a0) * PRE DECREMENT
 0002AC  6600 0038  347.         bne getea2
 0002B0  0C28 0028  348.         cmp.b #'(',1(a0) * Muss danach Klammer kommen
         0001
 0002B6  6600 002E  349.         bne getea2
 0002BA  5448       350.         addq #2,a0
 0002BC  6100 020A  351.         bsr checkan     * dann Adresse -(An
 0002C0  6500 0022  352.         bcs getea12
 0002C4  6100 FD3A  353.         bsr igbn        * An  )
 0002C8  0C10 0029  354.         cmp.b #$29,(a0)
 0002CC  6600 FD32  355.         bne carset      * FEHLER
 0002D0  5248       356.         addq #1,a0
 0002D2  3405       357.         move d5,d2      * MODE
 0002D4  847C 0020  358.         or #%100000,d2  * 100 rrr
 0002D8  383C 0400  359.         move #decre,d4  * Decrement mode
 0002DC  323C 0004  360.         move #4,d1      * Ohne displ
 0002E0  6000 FD1E  361.         bra carres
                    362.
 0002E4             363. getea12:
 0002E4  5548       364.         subq #2,a0      * wieder auf Anfang dann weiter
 0002E6  0C10 0028  365. getea2: cmp.b #'(',(a0) * indirekt oder postincr
 0002EA  6600 004A  366.         bne getea3
 0002[illegible]  5248       367.         addq #1,a0      * next
 000[illegible]  6100 01D6  368.         bsr checkan
 000[illegible]  6500 003E  369.         bcs getea21
 0002F8  6100 FD06  370.         bsr igbn
 0002FC  0C10 0029  371.         cmp.b #')',(a0) *  ( An )
 000300  6600 FCFE  372.         bne carset      * sonst Fehler
 000304  5248       373.         addq #1,a0
 000306  0C10 002B  374.         cmp.b #'+',(a0) *  ( An ) +
 00030A  6600 0016  375.         bne getea22     * nein dann ( An )
 00030E  5248       376.         addq #1,a0
 000310  3405       377.         move d5,d2
 000312  847C 001B  378.         or #%011000,d2  * MODE 011 rrr
 000316  323C 0004  379.         move #4,d1      * Kein displacement
 00031A  383C 0200  380.         move #incre,d4
 00031E  6000 FCE0  381.         bra carres
                    382.
 000322             383. getea22:                 * ( An )
 000322  3405       384.         move d5,d2
 000324  847C 0010  385.         or #%010000,d2  * MODE 010 rrr
 000328  323C 0004  386.         move #4,d1      * kein displacement
```

```
00032C  383C 0800          move #indir,d4
00330   6000'FCCE          bra carres

00334
00334   5348
00336   6100 0160          subq #1,a0       * wieder zuruecksetzen
0033A   6500 0010  getea3: bsr checkdn      * REG Dn
0033E   3405               bcs getea4
00340   383C 0100          move d5,d2       * MODE 000 rrr
00344   323C 0004          move #dreg,d4    * MODE DATAREG
00348   6000'FCB6          move #4,d1       * ohne disolteil = 4
                           bra carres       * No carry ok fertig

0034C   6100 017A  getea4: bsr checkan      * REG An
00350   6500 0014          bcs getea5
00354   3405               move d5,d2
00356   847C 0008          or #%001000,d2   * MODE 001 rrr
0035A   383C 0080          move #areg,d4
0035E   323C 0004          move #4,d1       * ohne disolteil
00362   6000'FC9C          bra carres

00366

00366   323C FFFF  getea5:                 * expr ...
0036A   48E7 3800          movem.l d2-d4,-(a7)  * Bei Pass 1 Achtung undef Symbol ignore
0036E   6100 FC90          bsr expr1        * Fehlerfrei annehmen
                           movem.l (a7)+,d2-d4  * EXPR aufrufen d0 = WERT
                                            * d1 = 5 dann undefiniert d1 = 0 dann SYNT
00372   4CDF 001C          movem.l (a7)+,d2-d4  ****** Fehlercheck hier ggf
00376   4A01               tst.b d1
00378   6700'FC86          beq carset       * auch wenn Syntaxfehler
0037C   6100'FC82          bsr isbn
00380   0C10 0028          cmp.b #'(',(a0)  * wenn Klammer folgt dann INDEX MODE
00384   675A               beq.s getea6
                                            * ok absoluter wert d1 gueltig
                                            * = 5 wenn ungueltig aber wie langw
00386   0C10 002E          cmp.b #'.',(a0)  * .W .S .L
0038A   662A               bne.s getea51    * default ist .L
0038C   5248               addq #1,a0
0038E   0C10 004C          cmp.b #'L',(a0)
00392   6720               beq.s getea511
00394   0C10 0057          cmp.b #'W',(a0)
00398   6708               beq.s getea512
0039A   0C10 0053          cmp.b #'S',(a0)
0039E   6600'FC60          bne carset       * FEHLER
003A2
003A2   5248       getea512:
003A4   383C 1000          addq #1,a0
003A8   343C 0038          move #absw,d4    * WORD
003AC   323C 0002          move #%111000,d2       * MODE 111 000
003B0   6000'FC4E          move #2,d1       * WORD FLAG
                           bra carres

003B4
003B4   5248       getea511:
003B6                      addq #1,a0
003B6   383C 2000  getea51:                * ABS.LONG
003BA   343C 0039          move #absl,d4
003BE   B23C 0005          move #%111001,d2       * MODE 111 001
003C2   6700'FC3C          cmp.b #5,d1
003C6   323C 0003          beq carres       * LONG UNDEFINED
003CA   6000'FC34          move #3,d1       * LONG FLAG ELSE
                           bra carres

003CE
003CE   6100'FC30  kommack:                * testet ob komma da
003D2   0C10 002C          bsr isbn
003D6   6600'FC28          cmp.b #',',(a0)  * wenn fehler a0 bleibt
                           bne carset
003DA   5248               addq #1,a0       * a0 hinter wert
003DC   6000'FC22          bra carres       * CARRY WENN nicht komma

003E0
003E0   5248       getea6:                 * wert (
003E2   6100'FC1C          addq #1,a0       * d0 definiert d1 definiert
003E6   0C10 0050          bsr isbn
003EA   6634               cmp.b #'P',(a0)  * ( PC
003EC   0C28 0043          bne.s getea61
        0001               cmp.b #'C',1(a0)
003F2   662C
003F4   544B               bne.s getea61
003F6   61D6               addq #2,a0       * OK (PC
                           bsr kommack
```

```
0003F8  641A            466.          bcc.s geteab11
0003FA  6100'FC04       467.          bsr igbn
0003FE  0C10 0029       468.          cmp.b #')'.(a0) *
000402  6600'FBFC       469.          bne carset      * FEHLER
000406  5248            470.          addq #1.a0
000408  383C 0860       471.          move #displ!pcadr!indir.d4
00040C  343C 003A       472.          move #%111010.d2        * MODE 111 010
000410  6000'FBEE       473.          bra carres      * d0 = displ d1 = attr
                        474.
000414                  475. geteab11:                  * (PC,
000414  383C 0870       476.          move #displ!pcadr!indir!indx.d4
000418  343C 003B       477.          move #%111011.d2
00041C  6000 002E       478.          bra get600
                        479.
                        480.
000420                  481. geteab1:                   * (An
000420  6100 00A6       482.          bsr checkan       * Adresse
000424  6500'FBDA       483.          bcs carset        * FEHLER
000428  3405            484.          move d5.d2        * Mode 1
00042A  61A2            485.          bsr kommack
00042C  6416            486.          bcc.s geteab12    * ja dann weiter
00042E  0C10 0029       487.          cmp.b #')'.(a0)
000432  6600'FBCC       488.          bne carset
000436  5248            489.          addq #1.a0        * ok (An)
000438  383C 0840       490.          move #displ!indir.d4
00043C  847C 0028       491.          or #%101000.d2    * MODE 101 rrr
000440  6000'FBBE       492.          bra carres        * d0.d1 gueltig
                        493.
                        494.
000444                  495. geteab12:                  * (An,
000444  383C 0850       496.          move #displ!indir!indx.d4
000448  847C 0030       497.          or #%110000.d2    * MODE 110 rrr
                        498.                            *
00044C                  499. get600:                    * gemeinsamer code .An.d
00044C  6100 007A       500.          bsr checkan       * Ziel d5
000450  6500 000C       501.          bcs get601        * sonst Areg
000454  3605            502.          move d5.d3        * MODE ERGAENZUNG
000456  E85B            503.          ror #4.d3         * 0rrr000000000000
000458  867C 8000       504.          or #$8000.d3      * 1rrr000000000000
00045C  600C            505.          bra.s get602
00045E  6100 0038       506. get601: bsr checkdn
000462  6500'FB9C       507.          bcs carset
000466  3605            508.          move d5.d3
000468  E85B            509.          ror #4.d3         * 0rrr000000000000
00046A  0C10 002E       510. get602: cmp.b #'.'.(a0)
00046E  6616            511.          bne.s get603      * DEFAULT .W
000470  5248            512.          addq #1.a0
000472  0C10 0057       513.          cmp.b #'W'.(a0) * wenn .W
000476  670C            514.          beq.s get6031
000478  0C10 004C       515.          cmp.b #'L'.(a0)
00047C  6600'FB82       516.          bne carset        * Fehler
000480  867C 0800       517.          or #$0800.d3      * Long Index Mode
000484                  518. get6031:
000484  5248            519.          addq #1.a0
000486                  520. get603:
000486  6100'FB78       521.          bsr igbn
00048A  0C10 0029       522.          cmp.b #')'.(a0)
00048E  6600'FB70       523.          bne carset
000492  5248            524.          addq #1.a0
000494  6000'FB6A       525.          bra carres
                        526.
                        527.
                        528.
000498                  529. checkdn:                   * Test ob Dn. wenn ja regcode nach
                        530.                            * d5
000498  6100'FB66       531.          bsr igbn          * Leerzeichen ignore
00049C  0C10 0044       532.          cmp.b #'D'.(a0)
0004A0  6600'FB5E       533.          bne carset        * nein dann carry Dn ?
0004A4                  534. checkda1:                  * Gemeinsamer Teil
0004A4  0C28 0030       535.          cmp.b #'0'.1(a0)
        0001
0004AA  6500'FB54       536.          bcs carset        * < 0 dann falsch
0004AE  0C28 0038       537.          cmp.b #'7'+1.1(a0)  *REV 4.1 cc.cs
        0001
0004B4  6400'FB4A       538.          bcc carset        * > 7 Dann Fehler
0004B8  1A28 0001       539.          move.b 1(a0).d5 * nach ergebnis
0004BC  CA7C 0007       540.          and.w #7.d5       * 0..7 ist der Bereich
0004C0  D0FC 0002       541.          adda #2.a0        * dann weiter
0004C4  6000'FB3A       542.          bra carres
                        543.
```

```
0004C8                  checkan:                    * Test ob An. wenn ja regcode nach
                                                    * d5
0004C8  6100'FB36               bsr igbn
0004CC  0C10 0041               cmp.b #'A'.(a0)
0004D0  6600'FB2E               bne carset
0004D4  60CE                    bra checkda1


0004D6                  putorea:                    * aus befcode und d2 mode bilden nach d0
                                                    * und ausgeben
0004D6  48E7 F800               movem.l d0-d4.-(a7)
0004DA  302D'0000               move befcode(a5).d0
0004DE  8042                    or d2.d0            * ea aufgelegt
0004E0  6100 FCAA               bsr putword         * Befehl ablegen
0004E4  4CDF 001F               movem.l (a7)+.d0-d4
0004E8  6000'FB16               bra carres          * OK

0004EC                  putea:                      * Adressteile ablegen falls da
0004EC  3A02                    move d2.d5          * MOde merken
0004EE  CA7C 0038               and #%111000.d5     * nur Modebits dort
0004F2  BA7C 0028               cmp #%101000.d5
0004F6  6608                    bne.s put1ea
0004F8  6100 FC92               bsr putword         * d0 ausgeben INDIRECT WITH DISPLAC[illegible]T
0004FC  6000'FB02               bra carres          * hier long check moeglich

000500  BA7C 0030       put1ea: cmp #%110000.d5
000504  660E                    bne.s put2ea
000506  C07C 00FF               and #$00FF.d0       *
00050A  8043                    or d3.d0            * EXTENSION + DISPLACEMENT
00050C  6100 FC7E               bsr putword
000510  6000'FAEE               bra carres

000514  B47C 0038       put2ea: cmp #%111000.d2     * Alle Bits nun
000518  6608                    bne.s put3ea
00051A  6100 FC70               bsr putword         * D0 ablegen
00051E  6000'FAE0               bra carres

000522  B47C 0039       put3ea: cmp #%111001.d2
000526  6608                    bne.s put4ea
000528  6100 FC02               bsr putlong
00052C  6000'FAD2               bra carres

000530  B47C 003A       put4ea: cmp #%111010.d2
000534  6614                    bne.s put5ea        * disp1(PC)
000536  90AD'0000               sub.l pcstand(a5).d0  * RELATIVE ANGABE BESSER
00053A  6100 18E4               bsr rangewlck       * achtung pcstand(a5) verwendet da +-
00053E  6100 1942               bsr errcber         * bezug immer auf diesen
000542  6100 FC48               bsr putword
000546  6000'FAB8               bra carres

00054A  B47C 003B       put5ea: cmp #%111011.d2
00054E  661A                    bne.s put6ea        * disp1(PC.Xi)
000550  90AD'0000               sub.l pcstand(a5).d0
000554  6100 1882               bsr rangeblck       * eingeschraenker Bereich +-
000558  6100 1928               bsr errcber
00055C  C07C 00FF               and #$FF.d0         * RELATIVE ANGABE
000560  8043                    or d3.d0
000562  6100 FC28               bsr putword
000566  6000'FA98               bra carres

00056A  6000'FA94       put6ea: bra carres          * Ohne Zusatzwort

                        ************************************************************
                        * Assemblermodul  Hauptprogramm                            *
                        ************************************************************

00056E                  assemble:                   * uebersetze von stxtxt nach etxtxt
00056E  422D'0000               clr.b curon(a5)
000572  6100'FA8C               bsr clrscreen       * loeschen Bildschirm
000576  42AD'0000               clr.l offset(a5)          * Ohne Offset ist STD
00057A  3B7C 0001               move #1.passflag(a5)      * Pass 1 zuerst
        '0000
000580  2B6D'0000               move.l pcorg(a5).pcstand(a5)     * default
        '0000
000586  6100'FA78               bsr ciinit2         * init pointer auf start
00058A  6100 FB76               bsr initcode
00058E                  ass1:
00058E  426D'0000               clr errflag(a5)
```

```
000592  6100 FCCA       622.          bsr getquelle
000596  0C6D 0002       623.          cmp #2,errflag(a5)
        0000
00059C  6700 0012       624.          beq ass12       * FEHLER ENDE ZU FRUEH
0005A0  41ED 0000       625.          lea einbuf(a5),a0
0005A4  6100 02E6       626.          bsr assline     * uebersetzen
0005A8  0C6D 0002       627.          cmp #2,errflag(a5)
        0000
0005AE  66DE            628.          bne ass1        * bis ende gefunden
0005B0                  629.  ass12:
0005B0  6100 FB74       630.          bsr endecode    * letzte zeile
                        631.                          *
                        632.                          **********************
                        633.                          * PASS 2 folgt       *
                        634.                          **********************
                        635.                          *
0005B4  426D 0000       636.          clr.w pagecnt(a5) * Seite 0
0005B8  426D 0000       637.          clr errcnt(a5)  * keine Fehler annehmen
0005BC  42AD 0000       638.          clr.l offset(a5)
0005C0  3B7C 0002       639.          move #2,passflag(a5)    * dann Pass 2
        0000
0005C6  2B6D 0000       640.          move.l pcorg(a5),pcstand(a5)    * default ++++
        0000
0005CC  6100 FA32       641.          bsr ciinit2     * init pointer auf start
000[illegible]  6100 FB30       642.          bsr initcode
000[illegible]  6100 00FC       643.          bsr kopfaus     * OHNE FF erste Ueberschrift
0005D8                  644.  ass2:
0005D8  0C2D 0042       645.          cmp.b #66,zeilen(a5)
        0000
0005DE  650C            646.          bcs.s ass2a
0005E0  103C 000C       647.          move.b #$c,d0
0005E4  6100 FA1A       648.          bsr co2
0005E8  6100 00E8       649.          bsr kopfaus     * Header ausgeben
                        650.
0005EC                  651.  ass2a:
0005EC  426D 0000       652.          clr errflag(a5) *
                        653.
                        654.  ************            * DEBUG-INFO
                        655.                          * pcstand(a5),akttxt festhalten wenn debug
0005F0  4A2D 0000       656.          tst.b debug(a5)
0005F4  6700 0010       657.          beq ass21ff
0005F8  206D 0000       658.          movea.l debugak(a5),a0  * zielpointer
0005FC  20ED 0000       659.          move.l pcstand(a5),(a0)+
000600  20ED 0000       660.          move.l akttxt(a5),(a0)+
000604  4290            661.          clr.l (a0)
                        662.
000606                  663.  ass21ff:
                        664.  ************
000606  6100 FC56       665.          bsr getquelle
00060A  0C6D 0002       666.          cmp #2,errflag(a5)
        0000
000610  6700 002C       667.          beq ass22       * FEHLER ENDE ZU FRUEH
000614  41ED 0000       668.          lea einbuf(a5),a0
000618  6100 0272       669.          bsr assline     * uebersetzen
                        670.  ************
00061C  4A2D 0000       671.          tst.b debug(a5) * DEBUG-INFO
000620  6700 0014       672.          beq ass22ff
000624  206D 0000       673.          movea.l debugak(a5),a0  * start debug info
000628  2010            674.          move.l (a0),d0  * pcstand(a5) vorher
00062A  B0AD 0000       675.          cmp.l pcstand(a5),d0
000[illegible]  6700 0006       676.          beq ass22ff     * debugak nicht weiterschalten
000[illegible]  50AD 0000       677.          addq.l #8,debugak(a5)   * sonst ja
                        678.                          * damit nur code-zeile abgelegt
000636                  679.  ass22ff:
                        680.  ************
000636  0C6D 0002       681.          cmp #2,errflag(a5)
        0000
00063C  669A            682.          bne ass2        * bis ende gefunden
00063E                  683.  ass22:
00063E  6100 FAE6       684.          bsr endecode    * letzte zeile wird immer ausgegeben
                        685.
000642                  686.  assabbr:                * ABBRUCH bei CTRL-C
                        687.
000642  3B7C 0002       688.          move #2,errflag(a5)     * auf ende setzen
        0000
                        689.
000648  103C 000D       690.          move.b #$d,d0   * Abstand bilden
00064C  6100 F9B2       691.          bsr co2
000650  103C 000A       692.          move.b #$a,d0
000654  6100 F9AA       693.          bsr co2
```

```
000658  302D'0000          694.          move errcnt(a5).d0      * rev 4.0
00065C  671C               695.          beq.s assnoerr  * ohne FEHLERMELDUNG
00065E  41ED'0000          696.          lea ausbuf(a5).a0
000662  43FA 00D6          697.          lea fintxt(pc).a1
000666                     698.  ass23:
000666  10D9               699.          move.b (a1)+.(a0)+
000668  66FC               700.          bne ass23
00066A  41ED'0000          701.          lea ausbuf(a5).a0
00066E  302D'0000          702.          move errcnt(a5).d0
000672  6100 FA5A          703.          bsr put4adr     * HEX ausgeben
000676  6100 F9E8          704.          bsr putausbuf
00067A                     705.  assnoerr:                * Keine Fehler da
00067A  41ED'0000          706.          lea ausbuf(a5).a0
00067E  43FA 00D3          707.          lea fin1txt(pc).a1
000682                     708.  ass24:
000682  10D9               709.          move.b (a1)+.(a0)+
000684  66FC               710.          bne ass24
000686  41ED'0000          711.          lea ausbuf(a5).a0
00068A  4280               712.          clr.l d0
00068C  302D'0000          713.          move svmnext(a5).d0
000690  2F08               714.          move.l a0.-(a7) * rev 4.0
000692  41ED'0000          715.          lea svmtab(a5).a0
000696  D088               716.          add.l a0.d0
000698  205F               717.          movea.l (a7)+.a0        * Startadresse
00069A  6100 FA2A          718.          bsr put6adr     * HEX ausgeben
00069E  6100 F9C0          719.          bsr putausbuf
0006A2  4A2D'0000          720.          tst.b debug(a5) * DEBUG
0006A6  6700 0020          721.          beq ass26       * ENDE WENN KEIN DEBUG-MODE
0006AA  41ED'0000          722.          lea ausbuf(a5).a0
0006AE  43FA 00BF          723.          lea fin2txt(pc).a1
0006B2                     724.  ass25:
0006B2  10D9               725.          move.b (a1)+.(a0)+
0006B4  66FC               726.          bne ass25
0006B6  41ED'0000          727.          lea ausbuf(a5).a0
0006BA  4280               728.          clr.l d0
0006BC  202D'0000          729.          move.l debugak(a5).d0
0006C0  6100 FA04          730.          bsr put6adr     * HEX ausgeben
0006C4  6100 F99A          731.          bsr putausbuf
0006C8  103C 000C          732.  ass26:  move.b #$c.d0
0006CC  6100'F932          733.          bsr co2         * Vorschub am Zeilenende
0006D0  4E75               734.          rts             * Ende der Uebersetzung
                           735.
0006D2                     736.  kopfaus:                * Ueberschrift ausgeben
0006D2  422D'0000          737.          clr.b zeilen(a5)        * Zeile 0 nun wieder
0006D6  41FA 0028          738.          lea kopftxt(pc).a0
0006DA  6100'F924          739.          bsr prtco2
0006DE  526D'0000          740.          addq.w #1.pagecnt(a5) * naechste Seite
0006E2  41ED'0000          741.          lea pagebuf(a5).a0    * Buffer fuer Ausgabe der Nr.
0006E6  302D'0000          742.          move pagecnt(a5).d0
0006EA  6100'F914          743.          bsr print4d
0006EE  41ED'0000          744.          lea pagebuf(a5).a0
0006F2  6100'F90C          745.          bsr prtco2
0006F6  41FA 003D          746.          lea crlflf(pc).a0
0006FA  6100'F904          747.          bsr prtco2              * Vorschub
0006FE  4E75               748.          rts
                           749.
00700                      750.  kopftxt:
00700   52 6F 6C 66        751.          dc.b 'Rolf-D.Klein 68000/08 Assembler 4.3 (C) 1984. Seite
        2D 44 2E 4B 6C
        65 69 6E 20 36
        38 30 30 30 2F
        30 38 20 41 73
        73 65 6D 62 6C
        65 72 20 34 2E
        33 20 28 43 29
        20 31 39 38 34
        2C 20 53 65 69
        74 65 20
00734   00                 752.          dc.b 0
00735                      753.  crlflf:
00735   0D 0A 0D 0A        754.          dc.b $d.$a.$d.$a.0
        00
                           755.
0073A                      756.  fintxt:
0073A   20 20 20 20        757.          dc.b '          Fehler entdeckt'.$d.0
        20 20 20 20 46
        65 68 6C 65 72
        20 65 6E 74 64
        65 63 6B 74 0D
        00
```

```
                                                  758.
000753                                            759. finitxt:
000753  20 20 20 20                               760.          dc.b '          Ende-Symboltabelle'.$d.0
        20 20 20 20 45
        6E 64 65 2D 53
        79 6D 62 6F 6C
        74 61 62 65 6C
        6C 65 0D 00
                                                  761.
00076F                                            762. fin2txt:
00076F  20 20 20 20                               763.          dc.b '          Ende-Debug-Tabelle'.$d.0
        20 20 20 20 45
        6E 64 65 2D 44
        65 62 75 67 2D
        54 61 62 65 6C
        6C 65 0D 00

        00 -- inserted
                                                  764.
00078C                                            765.          ds 0
                                                  766.
                                                  767. ** Symbol - Teil auf Wort-Grenze
                                                  768.
                                                  769. ** Format   TEXT-8Zeichen Vorcode-16Bit Type-16Bit rel-adresse-16B
                                                  770.
      00000000                                    771. svmtext equ      0        * Displacements
      00000008                                    772. svmcode equ      8        * fuer einfaches ansprechen
      0000000A                                    773. svmtype equ      8+2
      0000000C                                    774. svmadr  equ      8+2+2
                                                  775.
      0000000E                                    776. svmlg   equ      8+2+2+2 * rev 4.0 Laenge eines Symboleintrages
                                                  777.
                                                  778. * type  0=ein Wort  1=mit immediate
                                                  779.
                                                  780.
                                                  781.
                                                  782. *** Suchalgorithmus fuer Befehle
                                                  783.
                                                  784. * a0 -> schluesselwort
                                                  785. * dann carry wenn nicht gefunden oder fehler
                                                  786. * sonst a3 -> record
                                                  787.
                                                  788.
                                                  789. * begin
                                                  790. *  id := keywort;
                                                  791. *  i := 1; j := symzahl
                                                  792. *  repeat
                                                  793. *    k := (i + j) div 2;
                                                  794. *    if id <= key[k] then j := k - 1;
                                                  795. *    if id >= key[k] then i := k + 1;
                                                  796. *  until i > j
                                                  797. *  if i - 1 > j then found else notfound;
                                                  798. * end;
                                                  799. *  bei gleichheit (id = key[k]) Schnellausgang in Programm
                                                  800. * rev 4.0b relative adressen
                                                  801.
00078C                                            802. suchkey:
00078C  6100 F926                                 803.          bsr setigname   * a0 hinter name d0.d1 temp
000790  6500 F86E                                 804.          bcs carset      * Fehler ungueltiger Name
                                                  805.                          * d2 = namemsb d3 = namelsb
000794  7001                                      806.          move.l #1.d0            * start i
0007[illegible]  223C 0000004D                    807.          move.l #symzahl.d1      * j auf ende
0007[illegible]                                   808. suchrep1:
0007[illegible]  2800                             809.          move.l d0.d4
00079E  D881                                      810.          add.l d1.d4     * (I + J)
0007A0  88FC 0002                                 811.          divu #2.d4      * DIV 2
0007A4  48C4                                      812.          ext.l d4
0007A6  2F04                                      813.          move.l d4.-(a7) * Merken
0007A8  5344                                      814.          subq.w #1.d4    * Bereich 0..n-1 als Index nehmen
0007AA  C8FC 000E                                 815.          mulu #svmlg.d4  * * Laenge eines Eintrages
0007AE  2F08                                      816.          move.l a0.-(a7) * rev 4.0
0007B0  41FA 02A8                                 817.          lea befehle(pc).a0
0007B4  D888                                      818.          add.l a0.d4
0007B6  205F                                      819.          movea.l (a7)+.a0
0007B8  2644                                      820.          movea.l d4.a3   * A3 ist Ziel-Pointer
0007BA  281F                                      821.          move.l (a7)+.d4 * K
0007BC  B4AB 0000                                 822.          cmp.l 0(a3).d2  * MSB CHECK  wenn groesser springen
0007C0  6B00 0016                                 823.          bmi suchkl1
0007C4  6200 0018                                 824.          bhi suchgr1
0007C8  B6AB 0004                                 825.          cmp.l 4(a3).d3  * LSB CHECK
0007CC  6B00 000A                                 826.          bmi suchkl1
```

```
0007D0  6200 000C                bhi suchgr1
                                                  * Gefunden  A3 ist gueltig
0007D4  6000'F82A                bra carres       * ENDE DES SUCHENS
0007D8                  suchkl1:
0007D8  5384                     subq.l #1,d4     * j := k - 1
0007DA  2204                     move.l d4,d1
0007DC  6004                     bra.s suchchk

0007DE                  suchgr1:
0007DE  5244                     addq #1,d4
0007E0  2004                     move.l d4,d0     * i := k + 1
0007E2                  suchchk:                  * ENDETEST
0007E2  B280                     cmp.l d0,d1      * until i > j
0007E4  6B00'F81A                bmi carset       * FERTIG, NICHT GEFUNDEN
0007E8  60B2                     bra suchrep1     * Weiter suchen


                        ***

0007EA                  checkcc:                  * conditioncode pruefen a0-> auf condition
                                                  * Carry wenn nicht
                                                  * sonst wert nach d0
                                                  * wert schon in conditionpposition
0007EA  1010                     move.b (a0),d0   * ohne IGBN hier
0007EC  6100'F812                bsr bucheck      * muss buchstabe sein
0007F0  6500'F80E                bcs carset       * Fehler
0007F4  E158                     rol #8,d0        * 8 nach links
0007F6  C07C FF00                and #$FF00,d0
0007FA  3200                     move d0,d1       * merken
0007FC  1028 0001                move.b 1(a0),d0
00800   6100'F7FE                bsr bucheck
00804   6500 0014                bcs ch11cc
00808   8240                     or d0,d1         * d1 ist fertiger code
0080A   1028 0002                move.b 2(a0),d0
0080E   6100'F7F0                bsr bucheck      * darf kein bu mehr sein
00812   6400'F7EC                bcc carset       * sonst Fehler
00816   5488                     addq.l #2,a0
00818   6006                     bra.s ch12cc
0081A                   ch11cc:
0081A   5288                     addq.l #1,a0
0081C   827C 0020                or #$0020,d1     * Leerzeichen fuellen sonst
00820   43FA 0022        ch12cc: lea cctab(pc),a1          * Pointer auf vgl
00824   303C 0011                move #18-1,d0    * Zaehler
00828                   ch13cc:
00828   B251                     cmp.w (a1),d1
0082A   670A                     beq.s ch14cc     * gefunden
0082C   5889                     addq.l #4,a1     * naechste Stelle
0082E   51C8 FFF8                dbra d0,ch13cc
00832   6000'F7CC                bra carset       * nicht gefunden Fehler
00836                   ch14cc:
00836   3029 0002                move.w 2(a1),d0  * Zielergebnis   CODE
0083A   E058                     ror.w #8,d0      * 0000cccc00000000
0083C   C07C 0F00                and.w #$0f00,d0
00840   6000'F7BE                bra carres



00844                   cctab:
00844   43 43 00 04              dc.b 'CC',0,%0100
00848   48 53 00 04              dc.b 'HS',0,%0100
0084C   43 53 00 05              dc.b 'CS',0,%0101
00850   4C 4F 00 05              dc.b 'LO',0,%0101
00854   45 51 00 07              dc.b 'EQ',0,%0111
00858   46 20 00 01              dc.b 'F ',0,%0001
0085C   47 45 00 0C              dc.b 'GE',0,%1100
00860   47 54 00 0E              dc.b 'GT',0,%1110
00864   48 49 00 02              dc.b 'HI',0,%0010
00868   4C 45 00 0F              dc.b 'LE',0,%1111
0086C   4C 53 00 03              dc.b 'LS',0,%0011
00870   4C 54 00 0D              dc.b 'LT',0,%1101
00874   4D 49 00 0B              dc.b 'MI',0,%1011
00878   4E 45 00 06              dc.b 'NE',0,%0110
0087C   50 4C 00 0A              dc.b 'PL',0,%1010
00880   54 20 00 00              dc.b 'T ',0,%0000
00884   56 43 00 08              dc.b 'VC',0,%1000
00888   56 53 00 09              dc.b 'VS',0,%1001


0088C                   assline:                  * a0 -> start einer Zeile
                                                  * uebersetzen und Fehlerbehandlung
```

```
000888C  426D'0000          907.          clr errflag(a5) * kein Fehler annehmen
0008890  6100'F76E          908.          bsr igbn
0008894  1010               909.          move.b (a0).d0
0008896  4A40               910.          tst d0
0008898  6700 14EE          911.          beq codend       * = binaere Null dann Ende
000889C  B07C 2A00          912.          cmp #'*'.d0      * Kommentar Teil
00088A0  6700'F75E          913.          beq carres       * ok weiter
00088A4  B07C 3B00          914.          cmp #':'.d0      * Auch Kommentar
00088A8  6700'F756          915.          beq carres
00088AC  2F08               916.          move.l a0.-(a7) * Start merken
00088AE  6100 F804          917.          bsr setigname    * d2.d3 = name
00088B2  6500 008E          918.          bcs ass12li      * nein kein Name dann weiter auch kein EQU
00088B6  0C10 003A          919.          cmp.b #':'.(a0) * danach Doppelpunkt ?
00088BA  6644               920.          bne.s ass1line  * Nein dann kein Label da
00088BC  205F               921.          move.l (a7)+.a0 * zurueck an Anfang
00088BE  202D'0000          922.          move.l pcstand(a5).d0
00088C2  323C 0003          923.          move #3.d1       * Attribute = Langwort
00088C6  2F08               924.          move.l a0.-(a7) * retten Zeiger
00088C8  6100'F736          925.          bsr newval       * ggf neu anlegen
00088CC  642C               926.          bcc.s ass2line  * ja ok
00088CE  205F               927.          move.l (a7)+.a0 * nein nur aendern
00088D0  2F08               928.          move.l a0.-(a7)
00088D2  2F00               929.          move.l d0.-(a7)
00088D4  6100'F72A          930.          bsr getval       * holen des Wertes
00088[illegible] 4CDF 0004  931.          movem.l (a7)+.d2        * nach D2 holen
00088[illegible] 205F       932.          move.l (a7)+.a0
00088DE  B2BC 00000005      933.          cmp.l #5.d1      * gilt nur dann wenn nicht Udef
00088E4  6706               934.          beq.s ass2211
00088E6  B480               935.          cmp.l d0.d2      * muss gleich sein
00088E8  6100 1624          936.          bsr errnmult     * nicht gleich dann mehrfach def
00088EC                     937. ass2211:
00088EC  323C 0003          938.          move #3.d1       * nun gueltig
00088F0  C142               939.          exg d0.d2        * neues d0 setzen
00088F2  6100'F70C          940.          bsr setval       * nochmals setzen d1 ist neu
00088F6  5248               941.          addq #1.a0       * hinter Doppelpunkt
00088F8  604A               942.          bra.s ass3line
00088FA                     943. ass2line:
00088FA  225F               944.          move.l (a7)+.a1
00088FC  5248               945.          addq #1.a0
00088FE  6044               946.          bra.s ass3line
                            947.
0008900                     948. ass1line:
0008900  6100 F7B2          949.          bsr setigname    * d2 = zweiter name
0008904  6500 003C          950.          bcs ass12li      * nein kein name dann weiter
0008908  B4BC 45515520      951.          cmp.l #'EQU '.d2
000890E  6600 0032          952.          bne ass12li
0008912  6100 F6EC          953.          bsr expr1        * Wert holen. d1 gueltig und d0
0008916  41ED'0000          954.          lea ausbuf(a5).a0
000891A  10FC 003D          955.          move.b #'='.(a0)+       * EQU
000891E  10FC 0020          956.          move.b #' '.(a0)+
0008922  6100 F798          957.          bsr out8adr      *= 00000000
0008926  205F               958.          move.l (a7)+.a0 * auf Anfang Label
0008928  2F08               959.          move.l a0.-(a7) * retten Zeiger fuer setval
000892A  6100'F6D4          960.          bsr newval       * ggf neu anlegen
000892E  6400 000C          961.          bcc ass15li      * ja ok
0008932  205F               962.          move.l (a7)+.a0 * nein nur aendern
0008934  6100'F6CA          963.          bsr setval       * ok nun
0008938  6000'F6C6          964.          bra carres       * neu definieren ok
                            965.                           * da der Wert sich aendern koennte
                            966.                           * wenn labels definiert werden
0008[illegible]             967. ass15li:
0008[illegible] 205F        968.          move.l (a7)+.a0
000893E  6000'F6C0          969.          bra carres       * fertig
                            970.
0008942                     971. ass12li:
0008942  205F               972.          move.l (a7)+.a0 * von Start an
0008944                     973. ass3line:
0008944  6100'F6BA          974.          bsr igbn
0008948  1010               975.          move.b (a0).d0
000894A  B03C 000D          976.          cmp.b #$d.d0
000894E  6710               977.          beq.s assfline  * fertig da ende der Zeile
0008950  B03C 002A          978.          cmp.b #'*'.d0
0008954  670A               979.          beq.s assfline  * Kommentar
0008956  B03C 003B          980.          cmp.b #':'.d0
000895A  6704               981.          beq.s assfline
000895C  6100 001C          982.          bsr verteile     * ausfuehren Assembler
0008960                     983. assfline:                 * Ausgang
0008960  4A6D'0000          984.          tst errflag(a5)
0008964  6700'F69A          985.          beq carres       * OK Fertig
                            986.                           * Fehleranalyse und Ausgabe
```

```
000968  0C6D 0002          987.          cmp #2.errflag(a5)
        '0000
00096E  6700'F690          988.          beq carres       * ok ende des programms kein Fehler
000972  6100 15B0          989.          bsr erranalyse
000976  6000'F688          990.          bra carset
                           991.
                           992.
00097A                     993. verteile:                  * hole symbol. rev 4.0b relok
                           994.                            * carry ausgang wenn fehler
                           995.                            * sonst flag je nach Unterprogramm
                           996.                            * fuer gemeinsame Fehlerbeh.
                           997.                            * fehlercode = d0 bei CARRY
                           998.                            * sonst unterprogrammaufruf
                           999.                            * der erkannten Routine
                          1000.                            * a0 - > hinter das Symbol
                          1001.                            * Speicher BEFCODE.W enthaelt Teilcode
                          1002.                            * Speicher attcode(a5).W enthaelt Attribut
                          1003.                            * Speicher WORDBYTE.W enthaelt byte.word.l
00097A  2F08               1004.          move.l a0.-(a7)  * merken falls undef
00097C  6100 FE0E          1005.          bsr suchkey      * nach Befehl suchen erste Artx
000980  6400 0076          1006.          bcc verteil1     * ja dann weiter
000984  205F               1007.          move.l (a7)+.a0  * wieder zurueck alter Pointer
                          1008.                            * falls Makros dann hier suchen vor cc
                          1009.                            * suchen nach Bcc DBcc Scc - Codes
                          1010.
000986  6100'F678          1011.          bsr igbn         * Vorblanks ignorieren
00098A  0C10 0053          1012.          cmp.b #'S'.(a0)
00098E  6600 001A          1013.          bne vert11
000992  5248               1014.          addq #1.a0
000994  6100 FE54          1015.          bsr checkcc
000998  6100 150C          1016.          bsr errcsyntax
00099C  807C 50C0          1017.          or #%0101000011000000.d0        * Befcode + cc
0009A0  3B40'0000          1018.          move d0.befcode(a5)
0009A4  6100 127C          1019.          bsr codscc       * spez check dort
0009A8  4E75               1020.          rts
                          1021.
0009AA                     1022. vert11:
0009AA  0C10 0042          1023.          cmp.b #'B'.(a0)
0009AE  6600 001E          1024.          bne vert12
0009B2  5248               1025.          addq #1.a0
0009B4  6100 FE34          1026.          bsr checkcc
0009B8  6100 14EC          1027.          bsr errcsyntax
0009BC  807C 6000          1028.          or #%0110000000000000.d0
0009C0  3B40'0000          1029.          move d0.befcode(a5)      * SHORT ETC MERKEN
0009C4  6100 0052          1030.          bsr setsbw       * BYTEWORD definieren
0009C8  6100 095C          1031.          bsr codbra
0009CC  4E75               1032.          rts
                          1033.
0009CE                     1034. vert12:
0009CE  0C10 0044          1035.          cmp.b #'D'.(a0)
0009D2  6100 14E0          1036.          bsr errnsyntax   * FEHLER ALLGEMEIN SYNTAX
0009D6  0C28 0042          1037.          cmp.b #'B'.1(a0)
        0001
0009DC  6100 14D6          1038.          bsr errnsyntax
0009E0  5448               1039.          addq #2.a0
0009E2  6100 FE06          1040.          bsr checkcc
0009E6  6100 14BE          1041.          bsr errcsyntax
0009EA  807C 50C8          1042.          or #%0101000011001000.d0
0009EE  3B40'0000          1043.          move d0.befcode(a5)
0009F2  6100 0A66          1044.          bsr coddbcc
0009F6  4E75               1045.          rts
                          1046.
0009F8                     1047. verteil1:                 * ok gefunden
0009F8  225F               1048.          move.l (a7)+.a1 * DUMMY
0009FA  3B6B 0008          1049.          move symcode(a3).befcode(a5)     * Befehlscode ablegen
        '0000
00A00   3B6B 000A          1050.          move symtype(a3).attcode(a5)     * Attributecode
        '0000
00A06   6100 0010          1051.          bsr setsbw                       * SET Byte Word Long
00A0A   302B 000C          1052.          move.w symadr(a3).d0             * Startadr
00A0E   47FA 004A          1053.          lea befehle(pc).a3
00A12   D6C0               1054.          adda.w d0.a3                     * Damit +- Bereich
                          1055.                                           * bildet Abs Adresse
00A14   4E93               1056.          jsr (a3)                         * nach Adresse springen
00A16   4E75               1057.          rts                      * Carry wie ende Unterprogramm
                          1058.
                          1059.
00A18                     1060. setsbw:
00A18   3B7C 0001          1061.          move #1.wordbyte(a5)     * WORD IST DEFAULT
        '0000
```

```
000A1E  0C10 002E      1062.          cmp.b #'.'.(a0)          * .B etc checken
000A22  6634           1063.          bne.s vere
000A24  0C28 0053      1064.          cmp.b #'S'.1(a0)                 * .S
        0001
000A2A  6608           1065.          bne.s ver2
000A2C                 1066. ver12:
000A2C  5448           1067.          addq #2.a0
000A2E  426D 0000      1068.          clr.w wordbyte(a5)               * .B code
000A32  6024           1069.          bra.s vere
000A34  0C28 0042      1070. ver2:    cmp.b #'B'.1(a0)
        0001
000A3A  67F0           1071.          beq.s ver12              * auch .B
000A3C  0C28 004C      1072.          cmp.b #'L'.1(a0)
        0001
000A42  660A           1073.          bne.s ver3
000A44  5448           1074.          addq #2.a0
000A46  3B7C 0002      1075.          move #2.wordbyte(a5)     *.L
        0000
000A4C  600A           1076.          bra.s vere
000A4E  0C28 0057      1077. ver3:    cmp.b #'W'.1(a0)
        0001
000A54  6602           1078.          bne.s vere
000A56  5448           1079.          addq #2.a0               *.W schon eingestellt
000A58                 1080. vere:
000A58  4E75           1081.          rts
                       1082.
                       1083.
                       1084. ***
                       1085.
                       1086.
000A5A                 1087. befehle:                  * Befehlstabelle
                       1088.                           * rev 4.0b
000A5A  41 42 43 44    1089.          dc.l 'ABCD'.'    '
        20 20 20 20
000A62  C100           1090.          dc.w %1100000100000000  * Basiscode
000A64  0000           1091.          dc.w 0                  * Type = nicht immediate
000A66  0436           1092.          dc.w codabcd-befehle
                       1093.
000A68  41 44 44 20    1094.          dc.l 'ADD '.'    '
        20 20 20 20
000A70  D000           1095.          dc.w %1101000000000000
000A72  0600           1096.          dc.w %0000011000000000
000A74  04FE           1097.          dc.w codadd-befehle
                       1098.
000A76  41 44 44 41    1099.          dc.l 'ADDA'.'    '
        20 20 20 20
000A7E  D000           1100.          dc.w %1101000000000000
000A80  0000           1101.          dc.w 0
000A82  05A4           1102.          dc.w codadda-befehle     * neu da an erlaubt ist
                       1103.
000A84  41 44 44 49    1104.          dc.l 'ADDI'.'    '
        20 20 20 20
000A8C  0600           1105.          dc.w %0000011000000000
000A8E  0001           1106.          dc.w 1                   * Imm
000A90  063E           1107.          dc.w codaddi-befehle
                       1108.
000A92  41 44 44 51    1109.          dc.l 'ADDQ'.'    '
        20 20 20 20
000A9A  5000           1110.          dc.w %0101000000000000
000A9C  0000           1111.          dc.w 0                   *
000A9E  06CA           1112.          dc.w codaddq-befehle     *
                       1113.
000AA0  41 44 44 58    1114.          dc.l 'ADDX'.'    '
        20 20 20 20
000AA8  D100           1115.          dc.w %1101000100000000
000AAA  0000           1116.          dc.w 0
000AAC  0716           1117.          dc.w codaddx-befehle
                       1118.
000AAE  41 4E 44 20    1119.          dc.l 'AND '.'    '
        20 20 20 20
000AB6  C000           1120.          dc.w %1100000000000000
000AB8  0200           1121.          dc.w %0000001000000000
000ABA  04AE           1122.          dc.w codand-befehle      *
                       1123.
000ABC  41 4E 44 49    1124.          dc.l 'ANDI'.'    '
        20 20 20 20
000AC4  0200           1125.          dc.w %0000001000000000
000AC6  0000           1126.          dc.w 0
000AC8  0A7E           1127.          dc.w codeori-befehle
                       1128.
```

```
000ACA  41 53 4C 20     1129.           dc.l 'ASL '.'    '
        20 20 20 20
000AD2  E100            1130.           dc.w %1110000100000000
000AD4  E1C0            1131.           dc.w %1110000111000000
000AD6  072C            1132.           dc.w codshift-befehle
                        1133.
000AD8  41 53 52 20     1134.           dc.l 'ASR '.'    '
        20 20 20 20
000AE0  E000            1135.           dc.w %1110000000000000
000AE2  E0C0            1136.           dc.w %1110000011000000
000AE4  072C            1137.           dc.w codshift-befehle
                        1138.
000AE6  42 43 48 47     1139.           dc.l 'BCHG'.'    '
        20 20 20 20
000AEE  0140            1140.           dc.w %0000000101000000
000AF0  0840            1141.           dc.w %0000100001000000  * fuer bitmode
000AF2  084E            1142.           dc.w codbchg-befehle
                        1143.
                        1144.
000AF4  42 43 4C 52     1145.           dc.l 'BCLR'.'    '
        20 20 20 20
000AFC  0180            1146.           dc.w %0000000110000000
000AFE  0880            1147.           dc.w %0000100010000000          * fuer bitmode
000B00  084E            1148.           dc.w codbchg-befehle
                        1149.
000B02  42 52 41 20     1150.           dc.l 'BRA '.'    '
        20 20 20 20
000B0A  6000            1151.           dc.w %0110000000000000
000B0C  0001            1152.           dc.w 1
000B0E  08CC            1153.           dc.w codbra-befehle
                        1154.
000B10  42 53 45 54     1155.           dc.l 'BSET'.'    '
        20 20 20 20
000B18  01C0            1156.           dc.w %0000000111000000
000B1A  08C0            1157.           dc.w %0000100011000000          * fuer bitmode
000B1C  084E            1158.           dc.w codbchg-befehle
                        1159.
000B1E  42 53 52 20     1160.           dc.l 'BSR '.'    '
        20 20 20 20
000B26  6100            1161.           dc.w %0110000100000000
000B28  0000            1162.           dc.w 0
000B2A  08CC            1163.           dc.w codbra-befehle             * wie bei BRA
                        1164.
000B2C  42 54 53 54     1165.           dc.l 'BTST'.'    '
        20 20 20 20
000B34  0100            1166.           dc.w %0000000100000000
000B36  0800            1167.           dc.w %0000100000000000  * fuer bitmode
000B38  07D0            1168.           dc.w codbtst-befehle
                        1169.
000B3A  43 48 4B 20     1170.           dc.l 'CHK '.'    '
        20 20 20 20
000B42  4180            1171.           dc.w %0100000110000000
000B44  0000            1172.           dc.w 0
000B46  0916            1173.           dc.w codchk-befehle
                        1174.
000B48  43 4C 52 20     1175.           dc.l 'CLR '.'    '
        20 20 20 20
000B50  4200            1176.           dc.w %0100001000000000
000B52  0000            1177.           dc.w 0
000B54  11B2            1178.           dc.w codneg-befehle
                        1179.
000B56  43 4D 50 20     1180.           dc.l 'CMP '.'    '
        20 20 20 20
000B5E  B000            1181.           dc.w %1011000000000000
000B60  0C00            1182.           dc.w %0000110000000000
000B62  097C            1183.           dc.w codcmp-befehle     * einschraenkung
                        1184.
000B64  43 4D 50 41     1185.           dc.l 'CMPA'.'    '
        20 20 20 20
000B6C  B000            1186.           dc.w %1011000000000000
000B6E  0000            1187.           dc.w 0
000B70  05A4            1188.           dc.w codadda-befehle    * genauso
                        1189.
000B72  43 4D 50 49     1190.           dc.l 'CMPI'.'    '
        20 20 20 20
000B7A  0C00            1191.           dc.w %0000110000000000
000B7C  0001            1192.           dc.w 1
000B7E  063E            1193.           dc.w codaddi-befehle    * wie addi
                        1194.
000B80  43 4D 50 4D     1195.           dc.l 'CMPM'.'    '
```

```
                20 20 20 20
000B88  B108            1196.           dc.w %1011000100001000
000B8A  0000            1197.           dc.w 0
000B8C  09B8            1198.           dc.w codcmom-befehle
                        1199.
000B8E  44 42 52 41     1200.           dc.l 'DBRA'.'    '
        20 20 20 20
000B96  51C8            1201.           dc.w %0101000111001000
000B98  0000            1202.           dc.w 0
000B9A  0A00            1203.           dc.w coddbcc-befehle
                        1204.
000B9C  44 43 20 20     1205.           dc.l 'DC  '.'    '      * DEFINE CONSTANT
        20 20 20 20
000BA4  0000            1206.           dc.w 0
000BA6  0002            1207.           dc.w 2
000BA8  125E            1208.           dc.w coddc-befehle
                        1209.
000BAA  44 49 56 53     1210.           dc.l 'DIVS'.'    '
        20 20 20 20
000BB2  81C0            1211.           dc.w %1000000111000000
000BB4  0000            1212.           dc.w 0
000BB6  0916            1213.           dc.w codchk-befehle     * wie chk
                        1214.
000BB8  44 49 56 55     1215.           dc.l 'DIVU'.'    '
        20 20 20 20
000BC0  80C0            1216.           dc.w %1000000011000000
000BC2  0000            1217.           dc.w 0
000BC4  0916            1218.           dc.w codchk-befehle
                        1219.
000BC6  44 53 20 20     1220.           dc.l 'DS  '.'    '
        20 20 20 20
000BCE  0000            1221.           dc.w 0
000BD0  0002            1222.           dc.w 2
000BD2  12E8            1223.           dc.w codds-befehle
                        1224.
000BD4  45 4E 44 20     1225.           dc.l 'END '.'    '
        20 20 20 20
000BDC  0000            1226.           dc.w 0
000BDE  0002            1227.           dc.w 2
000BE0  132E            1228.           dc.w codend-befehle
                        1229.
000BE2  45 4F 52 20     1230.           dc.l 'EOR '.'    '
        20 20 20 20
000BEA  B000            1231.           dc.w %1011000000000000
000BEC  0A00            1232.           dc.w %0000101000000000
000BEE  0A32            1233.           dc.w codeor-befehle     * source dn
                        1234.
000BF0  45 4F 52 49     1235.           dc.l 'EORI'.'    '
        20 20 20 20
000BF8  0A00            1236.           dc.w %0000101000000000
000BFA  0000            1237.           dc.w 0
000BFC  0A7E            1238.           dc.w codeori-befehle
                        1239.
000BFE  45 58 47 20     1240.           dc.l 'EXG '.'    '
        20 20 20 20
000C06  C100            1241.           dc.w %1100000100000000
000C08  0000            1242.           dc.w 0
000C0A  0B06            1243.           dc.w codexg-befehle
                        1244.
000C0C  45 58 54 20     1245.           dc.l 'EXT '.'    '
        20 20 20 20
000C14  4800            1246.           dc.w %0100100000000000
000C16  0000            1247.           dc.w 0
000C18  0B9A            1248.           dc.w codext-befehle
                        1249.
000C1A  49 4C 4C 45     1250.           dc.l 'ILLE'.'GAL '
        47 41 4C 20
000C22  4AFC            1251.           dc.w %0100101011111100
000C24  0000            1252.           dc.w 0
000C26  0BD8            1253.           dc.w codall-befehle
                        1254.
000C28  4A 4D 50 20     1255.           dc.l 'JMP '.'    '
        20 20 20 20
000C30  4EC0            1256.           dc.w %0100111011000000
000C32  0000            1257.           dc.w 0
000C34  0BF4            1258.           dc.w codjmp-befehle     *nur ea
                        1259.
000C36  4A 53 52 20     1260.           dc.l 'JSR '.'    '
        20 20 20 20
000C3E  4E80            1261.           dc.w %0100111010000000
```

```
00C40  0000              1262.          dc.w 0
00C42  0BF4              1263.          dc.w codjmp-befehle      * wie jmp
                         1264.
00C44  4C 45 41 20       1265.          dc.l 'LEA '.'    '
       20 20 20 20
00C4C  41C0              1266.          dc.w %0100000111000000
00C4E  0000              1267.          dc.w 0
00C50  0C0C              1268.          dc.w codlea-befehle      *adr teil
                         1269.
                         1270.
00C52  4C 49 4E 4B       1271.          dc.l 'LINK'.'    '
       20 20 20 20
00C5A  4E50              1272.          dc.w %0100111001010000
00C5C  0001              1273.          dc.w 1
00C5E  0C32              1274.          dc.w codlink-befehle
                         1275.
00C60  4C 53 4C 20       1276.          dc.l 'LSL '.'    '
       20 20 20 20
00C68  E108              1277.          dc.w %1110000100001000
00C6A  E3C0              1278.          dc.w %1110001111000000
00C6C  072C              1279.          dc.w codshift-befehle
                         1280.
00C6E  4C 53 52 20       1281.          dc.l 'LSR '.'    '
       20 20 20 20
00C76  E008              1282.          dc.w %1110000000001000
00C78  E2C0              1283.          dc.w %1110001011000000
00C7A  072C              1284.          dc.w codshift-befehle
                         1285.
00C7C  4D 4F 56 45       1286.          dc.l 'MOVE'.'    '
       20 20 20 20
00C84  0000              1287.          dc.w %0000000000000000
00C86  0000              1288.          dc.w 0
00C88  0D06              1289.          dc.w codmove-befehle
                         1290.
00C8A  4D 4F 56 45       1291.          dc.l 'MOVE'.'A   '
       41 20 20 20
00C92  0000              1292.          dc.w %0000000000000000
00C94  0000              1293.          dc.w 0
00C96  0C68              1294.          dc.w codmovea-befehle
                         1295.
00C98  4D 4F 56 45       1296.          dc.l 'MOVE'.'M   '
       4D 20 20 20
00CA0  4880              1297.          dc.w %0100100010000000
00CA2  0001              1298.          dc.w 1
00CA4  0F54              1299.          dc.w codmovem-befehle
                         1300.
00CA6  4D 4F 56 45       1301.          dc.l 'MOVE'.'P   '
       50 20 20 20
00CAE  0008              1302.          dc.w %0000000000001000
00CB0  0001              1303.          dc.w 1
00CB2  10B0              1304.          dc.w codmoveq-befehle
                         1305.
00CB4  4D 4F 56 45       1306.          dc.l 'MOVE'.'Q   '
       51 20 20 20
00CBC  7000              1307.          dc.w %0111000000000000
00CBE  0000              1308.          dc.w 0
00CC0  117A              1309.          dc.w codmoveq-befehle
                         1310.
00CC2  4D 55 4C 53       1311.          dc.l 'MULS'.'    '
       20 20 20 20
00CCA  C1C0              1312.          dc.w %1100000111000000
00CCC  0000              1313.          dc.w 0
00CCE  0916              1314.          dc.w codchk-befehle
                         1315.
00CD0  4D 55 4C 55       1316.          dc.l 'MULU'.'    '
       20 20 20 20
00CD8  C0C0              1317.          dc.w %1100000011000000
00CDA  0000              1318.          dc.w 0
00CDC  0916              1319.          dc.w codchk-befehle
                         1320.
00CDE  4E 42 43 44       1321.          dc.l 'NBCD'.'    '
       20 20 20 20
00CE6  4800              1322.          dc.w %0100100000000000
00CE8  0000              1323.          dc.w 0
00CEA  0BE4              1324.          dc.w codtas-befehle      * nur ea aber An,PC nicht erlaubt
                         1325.
00CEC  4E 45 47 20       1326.          dc.l 'NEG '.'    '
       20 20 20 20
00CF4  4400              1327.          dc.w %0100010000000000
00CF6  0000              1328.          dc.w 0
```

```
000CF8  11B2            1329.          dc.w codneg-befehle
                        1330.
000CFA  4E 45 47 58     1331.          dc.l 'NEGX'.'    '
        20 20 20 20
000D02  4000            1332.          dc.w %0100000000000000
000D04  0000            1333.          dc.w 0
000D06  11B2            1334.          dc.w codneg-befehle
                        1335.
000D08  4E 4F 50 20     1336.          dc.l 'NOP '.'    '
        20 20 20 20
000D10  4E71            1337.          dc.w %0100111001110001
000D12  0000            1338.          dc.w 0
000D14  0BD8            1339.          dc.w codall-befehle
                        1340.
000D16  4E 4F 54 20     1341.          dc.l 'NOT '.'    '
        20 20 20 20
000D1E  4600            1342.          dc.w %0100011000000000
000D20  0000            1343.          dc.w 0
000D22  11B2            1344.          dc.w codneg-befehle
                        1345.
000D24  4F 46 46 53     1346.          dc.l 'OFFS'.'ET  '
        45 54 20 20
000D2C  0000            1347.          dc.w 0
000D2E  0000            1348.          dc.w 0
000D30  12CE            1349.          dc.w codoff-befehle
                        1350.
000D32  4F 52 20 20     1351.          dc.l 'OR  '.'    '
        20 20 20 20
000D3A  8000            1352.          dc.w %1000000000000000
000D3C  0000            1353.          dc.w %0000000000000000
000D3E  04AE            1354.          dc.w codand-befehle      * wie and
                        1355.
000D40  4F 52 47 20     1356.          dc.l 'ORG '.'    '
        20 20 20 20
000D48  0000            1357.          dc.w 0
000D4A  0002            1358.          dc.w 2
000D4C  1350            1359.          dc.w codorg-befehle
                        1360.
000D4E  4F 52 49 20     1361.          dc.l 'ORI '.'    '
        20 20 20 20
000D56  0000            1362.          dc.w %0000000000000000
000D58  0000            1363.          dc.w 0
000D5A  0A7E            1364.          dc.w codeori-befehle
                        1365.
000D5C  50 45 41 20     1366.          dc.l 'PEA '.'    '
        20 20 20 20
000D64  4840            1367.          dc.w %0100100001000000
000D66  0000            1368.          dc.w 0
000D68  0BF4            1369.          dc.w codjmp-befehle      * nur ein adrmode
                        1370.
000D6A  52 45 53 45     1371.          dc.l 'RESE'.'T   '
        54 20 20 20
000D72  4E70            1372.          dc.w %0100111001110000
000D74  0000            1373.          dc.w 0
000D76  0BD8            1374.          dc.w codall-befehle
                        1375.
000D78  52 4F 4C 20     1376.          dc.l 'ROL '.'    '
        20 20 20 20
000D80  E118            1377.          dc.w %1110000100011000
000D82  E7C0            1378.          dc.w %1110011111000000
000D84  072C            1379.          dc.w codshift-befehle
                        1380.
000D86  52 4F 52 20     1381.          dc.l 'ROR '.'    '
        20 20 20 20
000D8E  E018            1382.          dc.w %1110000000011000
000D90  E6C0            1383.          dc.w %1110011011000000
000D92  072C            1384.          dc.w codshift-befehle
                        1385.
000D94  52 4F 58 4C     1386.          dc.l 'ROXL'.'    '
        20 20 20 20
000D9C  E110            1387.          dc.w %1110000100010000
000D9E  E5C0            1388.          dc.w %1110010111000000
000DA0  072C            1389.          dc.w codshift-befehle
                        1390.
000DA2  52 4F 58 52     1391.          dc.l 'ROXR'.'    '
        20 20 20 20
000DAA  E010            1392.          dc.w %1110000000010000
000DAC  E4C0            1393.          dc.w %1110010011000000
000DAE  072C            1394.          dc.w codshift-befehle
                        1395.
```

```
000DB0  52 54 45 20   1396.   dc.l 'RTE '.'    '
        20 20 20 20
000DB8  4E73          1397.   dc.w %0100111001110011
000DBA  0000          1398.   dc.w 0
000DBC  0BD8          1399.   dc.w codall-befehle
                      1400.
000DBE  52 54 52 20   1401.   dc.l 'RTR '.'    '
        20 20 20 20
000DC6  4E77          1402.   dc.w %0100111001110111
000DC8  0000          1403.   dc.w 0
000DCA  0BD8          1404.   dc.w codall-befehle
                      1405.
000DCC  52 54 53 20   1406.   dc.l 'RTS '.'    '
        20 20 20 20
000DD4  4E75          1407.   dc.w %0100111001110101
000DD6  0000          1408.   dc.w 0
000DD8  0BD8          1409.   dc.w codall-befehle
                      1410.
000DDA  53 42 43 44   1411.   dc.l 'SBCD'.'    '
        20 20 20 20
000DE2  8100          1412.   dc.w %1000000100000000
000DE4  0000          1413.   dc.w 0
000DE6  0436          1414.   dc.w codabcd-befehle
                      1415.
000DE8  53 54 4F 50   1416.   dc.l 'STOP'.'    '
        20 20 20 20
000DF0  4E72          1417.   dc.w %0100111001110010
000DF2  0000          1418.   dc.w 0
000DF4  11E0          1419.   dc.w codstop-befehle
                      1420.
000DF6  53 55 42 20   1421.   dc.l 'SUB '.'    '
        20 20 20 20
000DFE  9000          1422.   dc.w %1001000000000000
000E00  0400          1423.   dc.w %0000010000000000
000E02  04FE          1424.   dc.w codadd-befehle
                      1425.
000E04  53 55 42 41   1426.   dc.l 'SUBA'.'    '
        20 20 20 20
000E0C  9000          1427.   dc.w %1001000000000000
000E0E  0000          1428.   dc.w 0
000E10  05A4          1429.   dc.w codadda-befehle
                      1430.
000E12  53 55 42 49   1431.   dc.l 'SUBI'.'    '
        20 20 20 20
000E1A  0400          1432.   dc.w %0000010000000000
000E1C  0001          1433.   dc.w 1
000E1E  063E          1434.   dc.w codaddi-befehle
                      1435.
000E20  53 55 42 51   1436.   dc.l 'SUBQ'.'    '
        20 20 20 20
000E28  5100          1437.   dc.w %0101000100000000
000E2A  0000          1438.   dc.w 0
000E2C  06CA          1439.   dc.w codaddq-befehle
                      1440.
000E2E  53 55 42 58   1441.   dc.l 'SUBX'.'    '
        20 20 20 20
000E36  9100          1442.   dc.w %1001000100000000
000E38  0000          1443.   dc.w 0
000E3A  0716          1444.   dc.w codaddx-befehle
                      1445.
000E3C  53 57 41 50   1446.   dc.l 'SWAP'.'    '
        20 20 20 20
000E44  4840          1447.   dc.w %0100100001000000
000E46  0000          1448.   dc.w 0
000E48  1204          1449.   dc.w codswap-befehle
                      1450.
000E4A  54 41 53 20   1451.   dc.l 'TAS '.'    '
        20 20 20 20
000E52  4AC0          1452.   dc.w %0100101011000000
000E54  0000          1453.   dc.w 0
000E56  0BE4          1454.   dc.w codtas-befehle      *
                      1455.
000E58  54 52 41 50   1456.   dc.l 'TRAP'.'    '
        20 20 20 20
000E60  4E40          1457.   dc.w %0100111001000000
000E62  0000          1458.   dc.w 0
000E64  121A          1459.   dc.w codtrap-befehle
                      1460.
000E66  54 52 41 50   1461.   dc.l 'TRAP'.'V    '
        56 20 20 20
```

```
000E6E  4E76            1462.           dc.w %0100111001110110
000E70  0000            1463.           dc.w 0
000E72  0BD8            1464.           dc.w codall-befehle
                        1465.
000E74  54 53 54 20     1466.           dc.l 'TST '
        20 20 20 20
000E7C  4A00            1467.           dc.w %0100101000000000
000E7E  0000            1468.           dc.w 0
000E80  11B2            1469.           dc.w codneg-befehle
                        1470.
000E82  55 4E 4C 4B     1471.           dc.l 'UNLK'
        20 20 20 20
000E8A  4E58            1472.           dc.w %0100111001011000
000E8C  0000            1473.           dc.w 0
000E8E  1236            1474.           dc.w codunlk-befehle    * anderer code wie link
                        1475.
000E90                  1476. lastsvmn:
                        1477.
                        1478.                           * Anzahl der Svmbole
0000004D                1479. svmzahl equ (lastsvmn-befehle)/svmlg
                        1480.
                        1481.
                        1482.
000E90                  1483. codabcd:                   * abcd Familie
000E94  6100 F606       1484.           bsr checkdn
000E98  6500 002A       1485.           bcs codlabcd    * vielleicht -(An)
000E9C  8A6D'0000       1486.           or befcode(a5).d5
000EA0  3B45'0000       1487.           move d5.befcode(a5)     * dv Teil
000EA4  6100 F52C       1488.           bsr kommack
000EA8  6100 1000       1489.           bsr errcsvntax
000EAC  6100 0FF8       1490.           bsr checkdn     * ABCD Dv.Dx
000EB0  EE5D            1491.           bsr errcsvntax
000EB2  8A6D'0000       1492.           ror #7.d5       * 0000xxx000000vvv
000EB6  3005            1493.           or befcode(a5).d5
000EB8  6100 F2D2       1494.           move d5.d0
000EBC  6000'F142       1495.           bsr outword     * fertiger code r/m = 0
000EC0                  1496.           bra carres
000EC0  6100 F3C0       1497. codlabcd:
000EC4  B8BC 00000400   1498.           bsr getea
000ECA  6100 0FE8       1499.           cmp.l #decre.d4 * muss decrementmode sein
000ECE  C47C 0007       1500.           bsr errnsvntax
000ED2  847C 0008       1501.           and #$7.d2
000ED6  846D'0000       1502.           or #$8.d2
000EDA  3B42'0000       1503.           or befcode(a5).d2
000EDE  6100 F4EE       1504.           move d2.befcode(a5)
000EE2  6100 0FC2       1505.           bsr kommack
000EE6  6100 F39A       1506.           bsr errcsvntax
000EEA  B8BC 00000400   1507.           bsr getea       * zweiter Adressteil
000EF0  6100 0FC2       1508.           cmp.l #decre.d4
000EF4  C47C 0007       1509.           bsr errnsvntax
000EF8  EE5A            1510.           and #$7.d2
000EFA  846D'0000       1511.           ror #7.d2
000EFE  3002            1512.           or befcode(a5).d2
000F00  6100 F28A       1513.           move d2.d0
000F04  6000'F0FA       1514.           bsr outword
                        1515.           bra carres
                        1516.
000F08                  1517.
000F08  6100'F0F6       1518. codand:                   * aehnlich ADD jedoch An nicht erlaubt
000F0C  0C10 0023       1519.           bsr iobn
000F10  6600 000C       1520.           cmp.b #'#'.(a0) * immediate
000F14  3B6D'0000       1521.           bne cod12and
        '0000           1522.           move attcode(a5).befcode(a5)
000F1A  6000 05BC       1523.           bra codeori     * CCR und SR auch erlaubt
000F1E                  1524. cod12and:
000F1E  6100 F578       1525.           bsr checkdn     * add dn.ea  dn in d5
000F22  6400 0050       1526.           bcc codanaadd   * gemeinsamer Bereich
                        1527.
                        1528.
000F26                  1529. codland:                  * add ea.dn
000F26  6100 F35A       1530.           bsr getea       * hier zuerst EA angegeben
000F2A  C8BC 00000080   1531.           and.l #areg.d4  * An nicht als Quelle erlaubt
000F30  6100 0FA2       1532.           bsr errnadress
000F34  6100 F498       1533.           bsr kommack
000F38  6100 0F6C       1534.           bsr errcsvntax
000F3C  6100 F55A       1535.           bsr checkdn     * muss datenregister sein nach D5
000F40  6100 0F64       1536.           bsr errcsvntax  * rest erhalten
000F44  E75D            1537.           rol #3.d5
000F46  8A6D'0000       1538.           or wordbyte(a5).d5      * rrrmmm
```

```
000F4A  ED5D              1539.          rol #6.d5
000F4C  8A6D'0000         1540.          or befcode(a5).d5
000F50  3B45'0000         1541.          move d5.befcode(a5)
000F54  6000 0706         1542.          bra codea        * wie ea - mode weiter
                          1543.
                          1544.
000F58                    1545. codadd:                    * add Familie
000F58  6100'F0A6         1546.          bsr igbn
000F5C  0C10 0023         1547.          cmp.b #'#'.(a0)
000F60  660A              1548.          bne.s cod9add
000F62  3B6D'0000         1549.          move attcode(a5).befcode(a5)    * immediate code
        '0000
000F68  6000 012E         1550.          bra codaddi              * hier auch imm moeglich
                          1551.
000F6C                    1552. cod9add:
000F6C  6100 F52A         1553.          bsr checkdn      * add dn.ea
000F70  6500 0064         1554.          bcs cod1add      * dn in d5
000F74                    1555. codanaadd:
000F74  EE5D              1556.          ror #7.d5
000F76  302D'0000         1557.          move wordbyte(a5).d0
000F7A  807C 0004         1558.          or #%100.d0      * Ziel ist EA
000F7E  ED58              1559.          rol #6.d0
000F80  8A40              1560.          or d0.d5         * mode = 100..1nn
000F82  8A6D'0000         1561.          or befcode(a5).d5        * ZIel ist ea
000F86  3B45'0000         1562.          move d5.befcode(a5)      * ok vorbereitet
000F8A  6100 F442         1563.          bsr kommack
000F8E  6100 0F16         1564.          bsr errcsyntax
000F92  6100 F2EE         1565.          bsr getea        * effektive adresse
000F96  B8BC 00000100     1566.          cmp.l #dreg.d4
000F9C  670E              1567.          beq.s cod121a
000F9E  C8BC 000000A0     1568.          and.l #areg!pcadr.d4
000FA4  6100 0F2E         1569.          bsr errnadress   * AREG und PC nicht als Ziel erlaubt
000FA8  6000 06B2         1570.          bra codea
                          1571.
                          1572.
000FAC                    1573. cod121a:                   * Sonderfall Dn.Dn
000FAC  302D'0000         1574.          move befcode(a5).d0
000FB0  C07C FEFF         1575.          and #%1111111101111111.d0
000FB4  3B40'0000         1576.          move d0.befcode(a5)      * nicht EA als Ziel
000FB8  C07C 0E00         1577.          and #%0000111000000000.d0        * reg holen
000FBC  EF58              1578.          rol #7.d0        * nach 000rrr
000FBE  806D'0000         1579.          or befcode(a5).d0        * xxxxrrrooo000rrr
000FC2  C07C F1FF         1580.          and #%1111000111111111.d0
000FC6  C47C 0007         1581.          and #$7.d2       * Mode ea
000FCA  EE5A              1582.          ror #7.d2
000FCC  8042              1583.          or d2.d0
000FCE  6100 F1BC         1584.          bsr outword      *ok nun fertig
000FD2  6000'F02C         1585.          bra carres
                          1586.
                          1587.
000FD6                    1588. cod1add:                   * add ea.dn
000FD6  6100 F2AA         1589.          bsr getea        * hier zuerst EA angegeben
000FDA  6100 F3F2         1590.          bsr kommack
000FDE  6100 0EC6         1591.          bsr errcsyntax
000FE2  6100 F4B4         1592.          bsr checkdn      * muss datenregister sein nach D5
000FE6  6100 0EBE         1593.          bsr errcsyntax   * rest erhalten
000FEA  E75D              1594.          rol #3.d5
000FEC  8A6D'0000         1595.          or wordbyte(a5).d5       * rrrmmm
000FF0  ED5D              1596.          rol #6.d5
000FF2  8A6D'0000         1597.          or befcode(a5).d5
000FF6  3B45'0000         1598.          move d5.befcode(a5)
000FFA  6000 0660         1599.          bra codea        * wie ea - mode weiter
                          1600.
                          1601.
000FFE                    1602. codadda:                   * ea.An hier
000FFE  6100 0CA6         1603.          bsr immeck
001002  6500 005E         1604.          bcs cod1adda     * #data.An
001006  6100 EFF8         1605.          bsr exor1        * D0
00100A  6100 F3C2         1606.          bsr kommack
00100E  6100 0E96         1607.          bsr errcsyntax
001012  6100 F4B4         1608.          bsr checkan      * An
001016  6100 0EBE         1609.          bsr errcsyntax
00101A  4A6D'0000         1610.          tst wordbyte(a5)
00101E  6100 0EBE         1611.          bsr erresyntax   * nicht mit .B
001022  C145              1612.          exg d0.d5        * wechsel
001024  E558              1613.          rol #2.d0
001026  806D'0000         1614.          or wordbyte(a5).d0       * 0rrrmm
00102A  E358              1615.          rol #1.d0
00102C  807C 0003         1616.          or #3.d0         * rrrm11
001030  ED58              1617.          rol #6.d0
```

```
001032  806D 0000       1618.          or befcode(a5).d0
001036  807C 003C       1619.          or #%111100.d0  * MODE 111 100 IMMEDIATE
00103A  6100 F150       1620.          bsr outword
00103E  0C6D 0002       1621.          cmp #2.wordbyte(a5)      * long
        0000
001044  6712            1622.          beq.s cod2adda
001046  2005            1623.          move.l d5.d0
001048  6100 0DFA       1624.          bsr rangewck
00104C  6100 0E34       1625.          bsr errcber      * zu lang
001050  6100 F13A       1626.          bsr outword
001054  6000 EFAA       1627.          bra carres
                        1628.
001058                  1629. cod2adda:
001058  2005            1630.          move.l d5.d0
00105A  6100 F0D0       1631.          bsr outlong
00105E  6000 EFA0       1632.          bra carres       *
                        1633.
                        1634.
001062                  1635. cod1adda:
001062  6100 F21E       1636.          bsr getea        * hier zuerst EA angegeben
001066  6100 F366       1637.          bsr kommack
00106A  6100 0E3A       1638.          bsr errcsyntax
00106E  6100 F458       1639.          bsr checkan      * muss Adressregister sein nach D5
001[illegible]  6100 0E32       1640.          bsr errcsyntax   * rest erhalten
001[illegible]  4A6D 0000       1641.          tst wordbyte(a5)
00107A  6100 0E32       1642.          bsr errcsyntax   * nicht mit .B
00107E  E55D            1643.          rol #2.d5
001080  8A6D 0000       1644.          or wordbyte(a5).d5       * 0rrrmm
001084  E35D            1645.          rol #1.d5
001086  8A7C 0003       1646.          or #3.d5         * rrrm11
00108A  ED5D            1647.          rol #6.d5
00108C  8A6D 0000       1648.          or befcode(a5).d5
001090  3B45 0000       1649.          move d5.befcode(a5)
001094  6000 05C6       1650.          bra codea        * wie ea - mode weiter
                        1651.
00109B                  1652. codaddi:                  * addi Familie
00109B  6100 0076       1653.          bsr sizeein      * Einbau SIZE
00109C  6100 EF62       1654.          bsr iobn
0010A0  0C10 0023       1655.          cmp.b #'#'.(a0)
0010A4  6100 0E0E       1656.          bsr errnsyntax
0010A8  5248            1657.          addq #1.a0
0010AA  6100 EF54       1658.          bsr exori        * di queltig
                        1659.                           * d0 ist definiert
0010AE  6100 F31E       1660.          bsr kommack
0010B2  6100 0DF2       1661.          bsr errcsyntax
0010B6  2F00            1662.          move.l d0.-(a7)
0010B8  6100 F1C8       1663.          bsr getea        * nun Effektive Adresse holen
0010BC                  1664. codeoaddi:                * EINSPRUNG von codeori
0010BC  C8BC 000000A0   1665.          and.l #areg!ocadr.d4
0010C2  6100 0E10       1666.          bsr errnadress   * nicht zulaessige Adresse
0010C6  6100 F40E       1667.          bsr outorea      * befcode(a5) ablegen
0010CA  2A1F            1668.          move.l (a7)+.d5 * d5 ist noch frei
0010CC  0C6D 0001       1669.          cmp #1.wordbyte(a5)
        0000
0010D2  671C            1670.          beq.s cod1addi
0010D4  0C6D 0002       1671.          cmp #2.wordbyte(a5)
        0000
0010DA  6724            1672.          beq.s cod2addi
0010DC  CA7C 00FF       1673.          and #$ff.d5      * ohne rangecheck hier
001[illegible]  2F00            1674.          move.l d0.-(a7)
00[illegible]  3005            1675.          move d5.d0
001[illegible]  6100 F0A6       1676.          bsr outword
0010E8  201F            1677.          move.l (a7)+.d0
0010EA  6100 F400       1678.          bsr outea
0010EE  4E75            1679.          rts
                        1680.
0010F0                  1681. cod1addi:                 * word
0010F0  2F00            1682.          move.l d0.-(a7)
0010F2  3005            1683.          move d5.d0
0010F4  6100 F096       1684.          bsr outword
0010F8  201F            1685.          move.l (a7)+.d0
0010FA  6100 F3F0       1686.          bsr outea
0010FE  4E75            1687.          rts
                        1688.
001100                  1689. cod2addi:
001100  2F00            1690.          move.l d0.-(a7)
001102  2005            1691.          move.l d5.d0
001104  6100 F026       1692.          bsr outlong
001108  201F            1693.          move.l (a7)+.d0
00110A  6100 F3E0       1694.          bsr outea
```

```
00110E  4E75                1695.           rts
                            1696.
001110                      1697.  sizeein:                 * fuer Size an pos 7.6
001110  2F00                1698.           move.l d0,-(a7)
001112  302D'0000           1699.           move wordbyte(a5).d0
001116  ED58                1700.           rol #6.d0
001118  806D'0000           1701.           or befcode(a5).d0
00111C  3B40'0000           1702.           move d0.befcode(a5)
001120  201F                1703.           move.l (a7)+.d0
001122  4E75                1704.           rts
                            1705.
                            1706.
001124                      1707.  codaddo:                  * addq Familie
001124  61EA                1708.           bsr sizeein      * Einbau SIZE
001126  6100'EED8           1709.           bsr igbn
00112A  0C10 0023           1710.           cmp.b #'#',(a0)
00112E  6100 0D84           1711.           bsr errnsyntax
001132  5248                1712.           addq #1.a0
001134  6100 EECA           1713.           bsr exor1        * d1 gueltig
                            1714.                            * d0 ist definiert
001138  6100 F294           1715.           bsr kommack
00113C  6100 0D68           1716.           bsr errcsyntax
001140  2F00                1717.           move.l d0,-(a7)
001142  6100 F13E           1718.           bsr getea        * nun Effektive Adresse holen
001146  2A1F                1719.           move.l (a7)+.d5 * d5 ist noch frei
001148  2C05                1720.           move.l d5.d6
00114A  0C6D 0002           1721.           cmp #2.passflag(a5)
        '0000
001150  660C                1722.           bne.s codiaddo
001152  5386                1723.           subq.l #1.d6     * 1..8 ist erlaubt rev 4.2
001154  CCBC FFFFFFF8       1724.           and.l #$FFFFFFF8.d6      * 0..7  pruefen
00115A  6100 0D38           1725.           bsr errnber      * Bereichsfehler
00115E                      1726.  codiaddo:
00115E  EE5D                1727.           ror #7.d5        * 8 wird als 0 codiert rev 4.2
001160  CA7C 0E00           1728.           and #$0E00.d5    * fuer Codierung ok.
001164  8A6D'0000           1729.           or befcode(a5).d5        *
001168  3B45'0000           1730.           move d5.befcode(a5)
00116C  6000 04EE           1731.           bra codeal       * ohne check ausfuehren
                            1732.
                            1733.
001170                      1734.  codaddx:                  * addx Familie
001170  302D'0000           1735.           move wordbyte(a5).d0
001174  ED58                1736.           rol #6.d0
001176  C07C 00C0           1737.           and #%0000000011000000.d0
00117A  806D'0000           1738.           or befcode(a5).d0
00117E  3B40'0000           1739.           move d0.befcode(a5)
001182  6000 FD0C           1740.           bra codabcd      * Weiter wie bei ABCD
                            1741.
001186                      1742.  codshift:                 * Schiebe-Familie
001186  6100'EE78           1743.           bsr igbn
00118A  0C10 0023           1744.           cmp.b #'#',(a0) * anzahl schieben
00118E  6760                1745.           beq.s codish
001190  6100 F0F0           1746.           bsr getea        * wenn datenreg dann ggf komma
001194  B8BC 00000100       1747.           cmp.l #dreg.d4
00119A  662E                1748.           bne.s cod2sh     * nein dann EA Mode
00119C  6100 F230           1749.           bsr kommack
0011A0  6100 0D04           1750.           bsr errcsyntax
0011A4  6100 FF6A           1751.           bsr sizeein      * Size in Befcode
                            1752.                            * Dx.Dy
0011A8  C47C 0007           1753.           and #$7.d2       * aus mode heraus
0011AC  EE5A                1754.           ror #7.d2
0011AE  846D'0000           1755.           or befcode(a5).d2
0011B2  847C 0020           1756.           or #%100000.d2   * i/r = 1
0011B6  3002                1757.           move d2.d0
0011B8  6100 F2DE           1758.           bsr checkdn      * Dy holen
0011BC  6100 0CE8           1759.           bsr errcsyntax
0011C0  8045                1760.           or d5.d0         * ok eingeblendet
0011C2  6100 EFC8           1761.           bsr putword      * Fertig
0011C6  6000'EE38           1762.           bra carres
                            1763.
                            1764.
0011CA                      1765.  cod2sh:                   * ea mode shift
0011CA  C8BC 000001A0       1766.           and.l #areg!dreg!pcadr.d4
0011D0  6100 0D02           1767.           bsr errnadress
0011D4  4A6D'0000           1768.           tst wordbyte(a5)
0011D8  6100 0CF4           1769.           bsr erreadress
0011DC  0C6D 0002           1770.           cmp #2.wordbyte(a5)
        '0000
0011E2  6100 0CEA           1771.           bsr erreadress  * nicht zul. sind .B .L
0011E6  3B6D'0000           1772.           move attcode(a5).befcode(a5)     * rev da neuer code
```

```
          0000
0011EC    6000 046E      1773.          bra codea        * mit check ea Ausgabe
                         1774.
0011F0                   1775. cod1sh:                   * #data.Dn
0011F0    6100 FF1E      1776.          bsr sizeein      * size in befcode(a5)
0011F4    5248           1777.          addq #1.a0
0011F6    6100 EE08      1778.          bsr expr1        *
0011FA    5380           1779.          subq.l #1.d0     *1..8 -> 0..7
0011FC    6100 0AC0      1780.          bsr range07
001200    6100 0C80      1781.          bsr errcber
001204    5280           1782.          addq.l #1.d0     * 1..  .. 8
001206    C07C 0007      1783.          and #7.d0        * 1..7 .. 0
00120A    EE58           1784.          ror #7.d0
00120C    806D'0000      1785.          or befcode(a5).d0
001210    6100 F1BC      1786.          bsr kommack
001214    6100 0C90      1787.          bsr errcsyntax
001218    6100 F27E      1788.          bsr checkdn
00121C    6100 0C88      1789.          bsr errcsyntax
001220    8045           1790.          or d5.d0
001222    6100 EF68      1791.          bsr outword
001226    6000'EDD8      1792.          bra carres
                         1793.
                         1794.
0012[illegible]                        1795. codbtst:                  * wie bchg jedoch *(PC) erlaubt
                         1796.
00122A    6100'EDD4      1797.          bsr iqbn
00122E    0C10 0023      1798.          cmp.b #'#'.(a0)
001232    6600 0048      1799.          bne codibtst
001236    5248           1800.          addq #1.a0
001238    3B6D'0000      1801.          move attcode(a5).befcode(a5)    * neuer Befehlscode ist at
          0000
00123E    6100 EDC0      1802.          bsr expr1        * data holen 0..7 erlaubt
001242    6100 0BB8      1803.          bsr rangebck
001246    6100 0C3A      1804.          bsr errcber      * Bereichsfehler ---
00124A    6100 F182      1805.          bsr kommack
00124E    6100 0C56      1806.          bsr errcsyntax
001252    2F00           1807.          move.l d0.-(a7)
001254    6100 F02C      1808.          bsr getea
001258    2A1F           1809.          move.l (a7)+.d5
00125A    C8BC 00000080  1810.          and.l #areg.d4   * nicht erlaubt
001260    6100 0C72      1811.          bsr errnadress
001264    6100 F270      1812.          bsr putorea
001268    2F00           1813.          move.l d0.-(a7)
00126A    3005           1814.          move d5.d0
00126C    C07C 00FF      1815.          and #$FF.d0
001270    6100 EF1A      1816.          bsr outword
001274    201F           1817.          move.l (a7)+.d0
001276    6100 F274      1818.          bsr putea        * rest ea
00127A    4E75           1819.          rts
                         1820.
                         1821.
00127C                   1822. cod1btst:                 * Dn.ea
00127C    6100 F21A      1823.          bsr checkdn
001280    6100 0C24      1824.          bsr errcsyntax
001284    EE5D           1825.          ror #7.d5
001286    8A6D'0000      1826.          or befcode(a5).d5
00128A    3B45'0000      1827.          move d5.befcode(a5)
00128E    6100 F13E      1828.          bsr kommack
001292    6100 0C12      1829.          bsr errcsyntax
0012[illegible]    6100 EFEA      1830.          bsr getea        * ea holen
0012[illegible]    C8BC 00000080  1831.          and.l #areg.d4   * nicht erlaubt
0012[illegible]    6100 0C32      1832.          bsr errnadress
0012A4    6000 03B6      1833.          bra codea
                         1834.
                         1835.
0012A8                   1836. codbchg:                  * bchg spez  da immediate moegl
0012A8    6100'ED56      1837.          bsr iqbn
0012AC    0C10 0023      1838.          cmp.b #'#'.(a0)
0012B0    6600 0048      1839.          bne codibchg
0012B4    5248           1840.          addq #1.a0
0012B6    3B6D'0000      1841.          move attcode(a5).befcode(a5)    * neuer Befehlscode ist at
          0000
0012BC    6100 ED42      1842.          bsr expr1        * data holen 0..7 erlaubt
0012C0    6100 0B3A      1843.          bsr rangebck
0012C4    6100 0BBC      1844.          bsr errcber      * Bereichsfehler ---
0012C8    6100 F104      1845.          bsr kommack
0012CC    6100 0BDB      1846.          bsr errcsyntax
0012D0    2F00           1847.          move.l d0.-(a7)
0012D2    6100 EFAE      1848.          bsr getea
0012D6    2A1F           1849.          move.l (a7)+.d5
```

```
0012D8  C8BC 000000A0  1850.          and.l #areg!pcadr.d4    * nicht erlaubt
0012DE  6100 0BF4      1851.          bsr errnadress
0012E2  6100 F1F2      1852.          bsr putorea
0012E6  2F00           1853.          move.l d0,-(a7)
0012E8  3005           1854.          move d5.d0
0012EA  C07C 00FF      1855.          and #$FF.d0
0012EE  6100 EE9C      1856.          bsr putword
0012F2  201F           1857.          move.l (a7)+.d0 * altes d0
0012F4  6100 F1F6      1858.          bsr putea       * rest ea
0012F8  4E75           1859.          rts
                       1860.
                       1861.
0012FA                 1862. codibchg:                * Dn.ea
0012FA  6100 F19C      1863.          bsr checkdn
0012FE  6100 0BA6      1864.          bsr errcsyntax
001302  EE5D           1865.          ror #7.d5
001304  8A6D'0000      1866.          or befcode(a5).d5
001308  3B45'0000      1867.          move d5.befcode(a5)
00130C  6100 F0C0      1868.          bsr kommack
001310  6100 0B94      1869.          bsr errcsyntax
001314  6100 EF6C      1870.          bsr getea       * ea holen
001318  C8BC 000000A0  1871.          and.l #areg!pcadr.d4    * nicht erlaubt
00131E  6100 0BB4      1872.          bsr errnadress
001322  6000 0338      1873.          bra codea
                       1874.
                       1875.
001326                 1876. codbra:                  * branch
001326  0C6D 0001      1877.          cmp #1.wordbyte(a5)
        '0000
00132C  6722           1878.          beq.s codbra1   * long branch
00132E  6100 ECD0      1879.          bsr exori
001332  90AD'0000      1880.          sub.l pcstand(a5).d0
001336  5580           1881.          subq.l #2.d0    * - 2  da pcstand(a5) vor bef zeigt
001338  6100 0A9E      1882.          bsr rangebick
00133C  6100 0B44      1883.          bsr errcber     * Mit WORDAUSGABE gegen Phasenfehler
001340  C07C 00FF      1884.          and.w #$00ff.d0
001344  806D'0000      1885.          or befcode(a5).d0
001348  6100 EE42      1886.          bsr putword     * ok fertig
00134C  6000'ECB2      1887.          bra carres
                       1888.
001350                 1889. codbra1:
001350  302D'0000      1890.          move befcode(a5).d0
001354  6100 EE36      1891.          bsr putword     * erstes Wort ablegenm mit 0
001358  6100 ECA6      1892.          bsr exori
00135C  90AD'0000      1893.          sub.l pcstand(a5).d0
001360  6100 0ABE      1894.          bsr rangewick
001364  6100 0B1C      1895.          bsr errcber     * mit WORDAUSGABE
001368  6100 EE22      1896.          bsr putword     * ok fertig
00136C  6000'EC92      1897.          bra carres
                       1898.
001370                 1899. codchk:                  * CHK <ea>.Dn
001370  6100 0934      1900.          bsr immeck
001374  6500 0034      1901.          bcs codi1ck     * CHK #data.Dn
001378  6100 EC86      1902.          bsr exori
00137C  6100 0AC6      1903.          bsr rangewck
001380  6100 0B00      1904.          bsr errcber
001384  6100 F048      1905.          bsr kommack
001388  6100 0B1C      1906.          bsr errcsyntax
00138C  6100 F10A      1907.          bsr checkdn     * Register holen
001390  EE5D           1908.          ror #7.d5
001392  8A6D'0000      1909.          or befcode(a5).d5
001396  8A7C 003C      1910.          or #%111100.d5  * MODE IMME 111 100
00139A  C145           1911.          exg d0.d5       * nach d0
00139C  6100 EDEE      1912.          bsr putword
0013A0  3005           1913.          move d5.d0      * wort ablegen
0013A2  6100 EDE8      1914.          bsr putword
0013A6  6000'EC58      1915.          bra carres      * ok fertig
                       1916.
0013AA                 1917. codi1ck:
0013AA  6100 EED6      1918.          bsr getea       * eff adresse
0013AE  C8BC 00000080  1919.          and.l #areg.d4  * nicht erlaubt Adressregister
0013B4  6100 0B1E      1920.          bsr errnadress
0013B8  6100 F014      1921.          bsr kommack
0013BC  6100 0AE8      1922.          bsr errcsyntax  * ea.
0013C0  6100 F0D6      1923.          bsr checkdn     * ea.dn
0013C4  6100 0AE0      1924.          bsr errcsyntax
0013C8  EE5D           1925.          ror #7.d5       * maskieren
0013CA  8A6D'0000      1926.          or befcode(a5).d5
0013CE  3B45'0000      1927.          move d5.befcode(a5)
0013D2  6000 0288      1928.          bra codea       * code ablegen
```

```
                        1925.
0013D6                  1930. codcmo:                   * CMP.s <ea>.Dn
0013D6  6100'EC28       1931.           bsr igbn
0013DA  0C10 0023       1932.           cmp.b #'#',(a0)
0013DE  660A            1933.           bne.s cod9cmp
0013E0  3B6D'0000       1934.           move attcode(a5),befcode(a5)    * immediate code
        '0000
0013E6  6000 FCB0       1935.           bra codaddi
                        1936.
0013EA                  1937. cod9cmp:
0013EA  6100 EE96       1938.           bsr getea       * hier zuerst EA angegeben
0013EE  6100 EFDE       1939.           bsr kommack
0013F2  6100 0AB2       1940.           bsr errcsyntax
0013F6  6100 F0A0       1941.           bsr checkdn     * muss datenregister sein nach D5
0013FA  6100 0AAA       1942.           bsr errcsyntax  * rest erhalten
0013FE  E75D            1943.           rol #3,d5
001400  8A6D'0000       1944.           or wordbyte(a5),d5      * rrrmmm
001404  ED5D            1945.           rol #6,d5
001406  8A6D'0000       1946.           or befcode(a5),d5
00140A  3B45'0000       1947.           move d5,befcode(a5)
00140E  6000 024C       1948.           bra codea       * wie ea - mode weiter
                        1949.
                        1950.
0[illegible]2           1951. codcmom:                  * CMPM autoincr
0[illegible]12  6100 FCFC 1952.         bsr sizeein     * Size eintragen
001416  6100 EE6A       1953.           bsr getea       * muss incre sein  register in d2
00141A  B8BC 00000200   1954.           cmp.l #incre,d4
001420  6100 0A92       1955.           bsr errnsyntax
001424  C47C 0007       1956.           and #$7,d2
001428  846D'0000       1957.           or befcode(a5),d2
00142C  3B42'0000       1958.           move d2,befcode(a5)
001430  6100 EF9C       1959.           bsr kommack
001434  6100 0A70       1960.           bsr errcsyntax
001438  6100 EE48       1961.           bsr getea
00143C  B8BC 00000200   1962.           cmp.l #incre,d4
001442  6100 0A70       1963.           bsr errnsyntax
001446  C47C 0007       1964.           and #$7,d2
00144A  EE5A            1965.           ror #7,d2
00144C  846D'0000       1966.           or befcode(a5),d2
001450  3002            1967.           move d2,d0
001452  6100 ED38       1968.           bsr outword
001456  6000'EBA8       1969.           bra carres
                        1970.
                        1971.
00145A                  1972. coddbcc:                  * DBcc
00145A  6100 F03C       1973.           bsr checkdn     * Dataregister
00145E  6100 0A46       1974.           bsr errcsyntax  * D5 = registercode
001462  6100 EF6A       1975.           bsr kommack
001466  6100 0A3E       1976.           bsr errcsyntax  * Dn.value
00146A  302D'0000       1977.           move befcode(a5),d0
00146E  8045            1978.           or d5,d0
001470  6100 ED1A       1979.           bsr outword     * Befcode ablegen
001474  6100 EB8A       1980.           bsr expr1       * displacement
001478  90AD'0000       1981.           sub.l pcstand(a5),d0
00147C  6100 09A2       1982.           bsr rangewick
001480  6100 0A00       1983.           bsr errcber     * Bereichsfehler falls pass2 und > wort
001484  6100 ED06       1984.           bsr outword
001488  6000'EB76       1985.           bra carres
                        1986.
                        1987.
0[illegible]C           1988. codeor:                   * EOR OR AND
0014[illegible]8C  6100'EB72 1989.      bsr igbn
001490  0C10 0023       1990.           cmp.b #'#',(a0)
001494  6600 000C       1991.           bne codleor     * wenn ja bleibt a0
001498  3B6D'0000       1992.           move attcode(a5),befcode(a5)    * weiter wie eori
        '0000
00149E  6000 0038       1993.           bra codeori
                        1994.
0014A2                  1995. codleor:                  * Dn,ea
0014A2  6100 EFF4       1996.           bsr checkdn
0014A6  6100 09FE       1997.           bsr errcsyntax
0014AA  E75D            1998.           rol #3,d5
0014AC  8A6D'0000       1999.           or wordbyte(a5),d5
0014B0  8A7C 0004       2000.           or #%100,d5     * rrr1mm
0014B4  ED5D            2001.           rol #6,d5
0014B6  8A6D'0000       2002.           or befcode(a5),d5
0014BA  3B45'0000       2003.           move d5,befcode(a5)
0014BE  6100 EF0E       2004.           bsr kommack
0014C2  6100 09E2       2005.           bsr errcsyntax
0014C6  6100 EDBA       2006.           bsr getea
```

```
0014CA  C8BC 000000A0    2007.          and.l #areg!pcadr.d4
0014D0  6100 0A02        2008.          bsr errnadress  * nicht zulaessig
0014D4  6000 0186        2009.          bra codea       * und ablegen
                         2010.
                         2011.
0014D8                   2012. codeori:                  * EORI  ORI  ANDI   und CCR SR checken
0014D8  6100 07CC        2013.          bsr immeck
0014DC  6100 09CB        2014.          bsr errcsvntax  * muss # sein
0014E0  6100 EB1E        2015.          bsr expr1       * expression holen
0014E4  6100 EEE8        2016.          bsr kommack
0014E8  6100 09BC        2017.          bsr errcsvntax
0014EC  2F00             2018.          move.l d0,-(a7) * merken
0014EE  2F08             2019.          move.l a0,-(a7) * auch adresse start
0014F0  6100 EBC2        2020.          bsr setioname   *
0014F4  6500 0052        2021.          bcs codieori
0014F8  B4BC 43435220    2022.          cmp.l #'CCR ',d2
0014FE  6728             2023.          beq.s cod2eori
001500  B4BC 53522020    2024.          cmp.l #'SR  ',d2
001506  6640            2025.          bne.s codieori * .SR
001508  302D'0000        2026.          move befcode(a5).d0
00150C  807C 007C        2027.          or #%01111100.d0        * word
001510  6100 EC7A        2028.          bsr outword
001514  225F             2029.          move.l (a7)+.a1 * dummy
001516  201F             2030.          move.l (a7)+.d0
001518  6100 092A        2031.          bsr rangewck
00151C  6100 0964        2032.          bsr errcber
001520  6100 EC6A        2033.          bsr outword
001524  6000'EADA        2034.          bra carres
                         2035.
001528                   2036. cod2eori:                 * .CCR
001528  302D'0000        2037.          move befcode(a5).d0             * Byte
00152C  807C 003C        2038.          or #%00111100.d0
001530  6100 EC5A        2039.          bsr outword
001534  225F             2040.          move.l (a7)+.a1 * dummy
001536  201F             2041.          move.l (a7)+.d0
001538  6100 08C2        2042.          bsr rangebck
00153C  C07C 00FF        2043.          and #$FF.d0
001540  6100 EC4A        2044.          bsr outword
001544  6000'EABA        2045.          bra carres
                         2046.
001548                   2047. codieori:                 * muss ea sein
001548  6100 FBC6        2048.          bsr sizeein     * size einbauen
00154C  205F             2049.          move.l (a7)+.a0 * alter pointer zurueck
00154E  6100 ED32        2050.          bsr getea       * check fehler
001552  C8BC 000000A0    2051.          and.l #areg!pcadr.d4
001558  6100 097A        2052.          bsr errnadress  * auf Stack ist Datenwert
00155C  6000 FB5E        2053.          bra codeoaddi   * weiter dort
                         2054.
                         2055.
                         2056.
001560                   2057. codexg:                   * EXG
001560  6100 EF36        2058.          bsr checkdn
001564  6500 0044        2059.          bcs codiexg     * d5 = Dn.
001568  3005             2060.          move d5.d0      * merken
00156A  6100 EE62        2061.          bsr kommack
00156E  6100 0936        2062.          bsr errcsvntax
001572  6100 EF24        2063.          bsr checkdn     * Dn.Dn
001576  6500 0016        2064.          bcs cod12exg
00157A  EE58             2065.          ror #7.d0
00157C  8045             2066.          or d5.d0
00157E  806D'0000        2067.          or befcode(a5).d0
001582  807C 0040        2068.          or #%01000000.d0
001586  6100 EC04        2069.          bsr putword
00158A  6000'EA74        2070.          bra carres
                         2071.
00158E                   2072. cod12exg:                 * Dn.An
00158E  6100 EF38        2073.          bsr checkan
001592  6100 0912        2074.          bsr errcsvntax
001596  EE58             2075.          ror #7.d0
001598  8045             2076.          or d5.d0
00159A  806D'0000        2077.          or befcode(a5).d0
00159E  807C 0088        2078.          or #%10001000.d0
0015A2  6100 EBE8        2079.          bsr putword
0015A6  6000'EA58        2080.          bra carres
                         2081.
                         2082.
0015AA                   2083. codiexg:
0015AA  6100 EF1C        2084.          bsr checkan
0015AE  6100 08F6        2085.          bsr errcsvntax
0015B2  3005             2086.          move d5.d0      * d0 = An
```

```
0015B4  6100 EE1A       2087.           bsr kommack
0015B8  6100 08EC       2088.           bsr errcsyntax
0015BC  6100 EEDA       2089.           bsr checkdn     * An.Dn
0015C0  6500 0016       2090.           bcs cod2lexg
0015C4  EE5D            2091.           ror #7.d5
0015C6  8A6D 0000       2092.           or befcode(a5).d5
0015CA  8045            2093.           or d5.d0
0015CC  807C 0088       2094.           or #%10001000.d0
0015D0  6100 EBBA       2095.           bsr outword
0015D4  6000 EA2A       2096.           bra carres
                        2097.
0015D8                  2098. cod2lexg:                 * An.An
0015D8  6100 EEEE       2099.           bsr checkan
0015DC  6100 08C8       2100.           bsr errcsyntax
0015E0  EE58            2101.           ror #7.d0
0015E2  806D 0000       2102.           or befcode(a5).d0
0015E6  8045            2103.           or d5.d0
0015E8  807C 0048       2104.           or #%01001000.d0
0015EC  6100 EB9E       2105.           bsr outword
0015F0  6000 EA0E       2106.           bra carres
                        2107.
0015F4                  2108. codext:                   * EXT
0015F4  6100 EEA2       2109            bsr checkdn
00[illegible]  6100 08AC       2110.           bsr errcsyntax  * EXT Dn
00[illegible]  0C6D 0002       2111.           cmp #2.wordbyte(a5)
        0000
001602  671C            2112.           beq.s cod2ext
001604  0C6D 0001       2113.           cmp #1.wordbyte(a5)
        0000
00160A  6100 08A8       2114.           bsr errnsyntax
00160E                  2115. cod1ext:                  * EXT.W
00160E  8A6D 0000       2116.           or befcode(a5).d5
001612  8A7C 0080       2117.           or #%10000000.d5
001616  3005            2118.           move d5.d0
001618  6100 EB72       2119.           bsr outword
00161C  6000 E9E2       2120.           bra carres
                        2121.
001620                  2122. cod2ext:                  * EXT.L
001620  8A6D 0000       2123.           or befcode(a5).d5
001624  8A7C 00C0       2124.           or #%11000000.d5
001628  3005            2125.           move d5.d0
00162A  6100 EB60       2126.           bsr outword
00162E  6000 E9D0       2127.           bra carres
                        2128.
001632                  2129. codall:                   * Direkt op ohne adr
001632  302D 0000       2130.           move befcode(a5).d0
001636  6100 EB54       2131.           bsr outword     * einfach ablegen
00163A  6000 E9C4       2132.           bra carres
                        2133.
00163E                  2134. codtas:
00163E  6100 EC42       2135.           bsr getea
001642  C8BC 000000A0   2136.           and.l #areg!pcadr.d4
001648  6100 088A       2137.           bsr errnadress
00164C  600E            2138.           bra.s codea
                        2139.
00164E                  2140. codjmp:                   * JMP    <ea>
00164E  6100 EC32       2141.           bsr getea       * effektive Adresse holen
                        2142.                           * di = 0 dann fehler
001652  C8BC 00000780   2143.           and.l #decre!incre!areg!dreg.d4
00[illegible]  6100 087A       2144.           bsr errnadress  * FEHLER nicht erlaubter adressmode
00[illegible]                  2145. codea:                    * Einsprung effektive adresse als ergebni
00[illegible]                  2146. codeai:                   * ohne check
00165C  6100 EE78       2147.           bsr outorea     * ea odern und bef ausgeben
001660  6100 EE8A       2148.           bsr outea       * Adressteil falls da ablegen
001664  4E75            2149.           rts
                        2150.
                        2151.
                        2152.
001666                  2153. codlea:                   * LEA <ea>.An
001666  6100 EC1A       2154.           bsr getea       *
00166A  C8BC 00000780   2155.           and.l #areg!dreg!incre!decre.d4
001670  6100 0862       2156.           bsr errnadress
001674  6100 ED58       2157.           bsr kommack
001678  6100 082C       2158.           bsr errcsyntax
00167C  6100 EE4A       2159.           bsr checkan     * nach D5
001680  EE5D            2160.           ror #7.d5
001682  8A6D 0000       2161.           or befcode(a5).d5
001686  3B45 0000       2162.           move d5.befcode(a5)
00168A  60D0            2163.           bra codea       * wie ea weiter
                        2164.
```

```
00168C                          2166. codlink:                  * link An.#disp1
00168C  6100 EE3A               2167.           bsr checkan
001690  6100 0814               2168.           bsr errcsyntax
001694  8A6D'0000               2169.           or befcode(a5).d5
001698  3005                    2170.           move d5.d0
00169A  6100 EAF0               2171.           bsr outword
00169E  6100 ED2E               2172.           bsr kommack
0016A2  6100 0802               2173.           bsr errcsyntax
0016A6  6100 05FE               2174.           bsr immeck
0016AA  6100 07FA               2175.           bsr errcsyntax
0016AE  6100 E950               2176.           bsr expr1
0016B2  6100 0790               2177.           bsr rangewck
0016B6  6100 07CA               2178.           bsr errcber
0016BA  6100 EAD0               2179.           bsr outword     * Ausgabe
0016BE  6000'E940               2180.           bra carres
                                2181.
                                2182.
0016C2                          2183. codmovea:
0016C2  6100 05E2               2184.           bsr immeck      * immediate zul.
0016C6  6400 003A               2185.           bcc codme1
0016CA  6100 EBB6               2186.           bsr getea
0016CE  6100 ECFE               2187.           bsr kommack
0016D2  6100 07D2               2188.           bsr errcsyntax
0016D6  6100 EDF0               2189.           bsr checkan     * MOVEA ea.An
0016DA  6100 07CA               2190.           bsr errcsyntax
0016DE  4A6D'0000               2191.           tst wordbyte(a5)
0016E2  6100 07CA               2192.           bsr errcsyntax
0016E6  382D'0000               2193.           move wordbyte(a5).d4
0016EA  887C 0002               2194.           or #2.d4        * 11 oder 10
0016EE  E75C                    2195.           rol #3.d4
0016F0  8845                    2196.           or d5.d4
0016F2  E75C                    2197.           rol #3.d4
0016F4  887C 0001               2198.           or #%001.d4     * ssrrr001
0016F8  ED5C                    2199.           rol #6.d4
0016FA  3B44'0000               2200.           move d4.befcode(a5)
0016FE  6000 FF5C               2201.           bra codea1      * eff adr kein check mehr
                                2202.
001702                          2203. codme1:                   * MOVEA #data.An
001702  6100 E8FC               2204.           bsr expr1       * data holen
001706  6100 ECC6               2205.           bsr kommack
00170A  6100 079A               2206.           bsr errcsyntax  * ggf Fehler
00170E  6100 EDB8               2207.           bsr checkan
001712  6100 0792               2208.           bsr errcsyntax
001716  4A6D'0000               2209.           tst wordbyte(a5)
00171A  6100 0792               2210.           bsr errcsyntax
00171E  382D'0000               2211.           move wordbyte(a5).d4
001722  887C 0002               2212.           or #2.d4        * 11 oder 10
001726  E75C                    2213.           rol #3.d4
001728  8845                    2214.           or d5.d4
00172A  E75C                    2215.           rol #3.d4
00172C  887C 0001               2216.           or #%001.d4     * ssrrr001
001730  ED5C                    2217.           rol #6.d4
001732  887C 003C               2218.           or #%111100.d4  * wert
001736  C144                    2219.           exg d0.d4
001738  6100 EA52               2220.           bsr outword
00173C  C144                    2221.           exg d0.d4       *= d0.l = data
00173E  0C6D 0001               2222.           cmp #1.wordbyte(a5)     *= word ?
        '0000
001744  6700 000A               2223.           beq codme2
001748  6100 E9E2               2224.           bsr outlong
00174C  6000'E8B2               2225.           bra carres      *ok ablegen
                                2226.
001750  6100 06F2               2227. codme2: bsr rangewck
001754  6100 072C               2228.           bsr errcber
001758  6100 EA32               2229.           bsr outword
00175C  6000'E8A2               2230.           bra carres      * word ablegen
                                2231.
                                2232.
001760                          2233. codmove:                  * MOVE Gruppe
001760  2F08                    2234.           move.l a0.-(a7) * merken Anfang
001762  6100 E950               2235.           bsr setioname
001766  6500 0016               2236.           bcs codnv
00176A  B4BC 53522020           2237.           cmp.l #'SR  '.d2        *
001770  6700 0114               2238.           beq cod1move
001774  B4BC 55535020           2239.           cmp.l #'USP '.d2
00177A  6700 012C               2240.           beq cod2move
00177E                          2241. codnv:
00177E  205F                    2242.           move.l (a7)+.a0
001780  6100 0524               2243.           bsr immeck      * mit immediate teil
001784  6400 0142               2244.           bcc cod3move
```

```
                                2245.                           * MOVE ea.
001788  6100 EAF8               2246.           bsr getea       * alle modi erlaubt
00178C  6100 EC40               2247.           bsr kommack
001790  6100 0714               2248.           bsr errcsyntax
001794  48E7 F800               2249.           movem.l d0-d4,-(a7)     * d4 auch noetig
001798  2F08                    2250.           move.l a0,-(a7)
00179A  6100 E918               2251.           bsr setioname   * d2 = name
00179E  6500 0020               2252.           bcs codinv
0017A2  B4BC 53522020           2253.           cmp.l #'SR  '.d2
0017A6  6700 0086               2254.           beq codm10
0017AC  B4BC 43435220           2255.           cmp.l #'CCR '.d2
0017B2  6700 0096               2256.           beq codm11
0017B6  B4BC 55534020           2257.           cmp.l #'USP '.d2
0017EC  6700 00A6               2258.           beq codm12
0017C0                          2259.  codinv:
0017C0  205F                    2260.           move.l (a7)+,a0 * zurueck
                                2261.                           * MOVE ea,ea
0017C2  4CDF 001F               2262.           movem.l (a7)+,d0-d4
0017C6  48E7 F000               2263.           movem.l d0-d3,-(a7)     * !!! nur 4 regs dann
0017CA  6100 EAB6               2264.           bsr getea       * zweiteres EA holen
0017CE  C8BC 000000A0           2265.           and.l #areg!pcadr,d4
0017D4  6100 06FE               2266.           bsr errnadress
0017D8  48E7 F000               2267.           movem.l d0-d3,-(a7)     * destination
0017DC  302D 0000               2268.           move wordbyte(a5),d0    *0.1.2
0017E0  5240                    2269.           addq #1,d0      *1.2.3  -> 1.3.2
0017E2  B07C 0001               2270.           cmp #1,d0
0017E6  6704                    2271.           beq.s cod01mv
0017E8  0A40 0001               2272.           eori #1,d0      * 2 <-> 3
0017EC                          2273.  cod01mv:
0017EC  E758                    2274.           rol #3,d0
0017EE  3202                    2275.           move d2,d1      * mode umbauen
0017F0  C27C 0007               2276.           and #$7,d1
0017F4  8041                    2277.           or d1,d0
0017F6  E758                    2278.           rol #3,d0       *
0017F8  3202                    2279.           move d2,d1
0017FA  E659                    2280.           ror #3,d1       * Mode bits
0017FC  C27C 0007               2281.           and #$7,d1
001800  8041                    2282.           or d1,d0
001802  ED58                    2283.           rol #6,d0
001804  3B40 0000               2284.           move d0,befcode(a5)     * und ablegen als basiscode
001808  4CDF 000F               2285.           movem.l (a7)+,d0-d3     * destination
00180C  C144                    2286.           exg d0,d4
00180E  C345                    2287.           exg d1,d5
001810  C546                    2288.           exg d2,d6
001812  C747                    2289.           exg d3,d7
001814  4CDF 000F               2290.           movem.l (a7)+,d0-d3     * source
001818  6100 ECBC               2291.           bsr putorea     * source ablegen zuerst und befcode(a5)
00181C  6100 ECCE               2292.           bsr putea       * adressen source
001820  C144                    2293.           exg d0,d4
001822  C345                    2294.           exg d1,d5
001824  C546                    2295.           exg d2,d6
001826  C747                    2296.           exg d3,d7
001828  6100 ECC2               2297.           bsr putea       * adressen destination
00182C  6000 E7D2               2298.           bra carres
                                2299.
                                2300.
001830                          2301.  codm10:                  * MOVE ea,SR
001830  225F                    2302.           move.l (a7)+,a1
001832  3B7C 46C0               2303.           move #$46c0,befcode(a5)
        0000
001838  4CDF 001F               2304.           movem.l (a7)+,d0-d4
00183C  C8BC 00000080           2305.           and.l #areg,d4
001842  6100 0670               2306.           bsr errnsyntax
001846  6000 FE14               2307.           bra codea
                                2308.
00184A                          2309.  codm11:                  * MOVE ea,CCR
00184A  225F                    2310.           move.l (a7)+,a1
00184C  3B7C 44C0               2311.           move #$44c0,befcode(a5)
        0000
001852  4CDF 001F               2312.           movem.l (a7)+,d0-d4
001856  C8BC 00000080           2313.           and.l #areg,d4
00185C  6100 0656               2314.           bsr errnsyntax
001860  6000 FDFA               2315.           bra codea
                                2316.
001864                          2317.  codm12:                  * MOVE ea,USP  ea=An nur
001864  225F                    2318.           move.l (a7)+,a1
001866  4CDF 001F               2319.           movem.l (a7)+,d0-d4
00186A  B8BC 00000080           2320.           cmp.l #areg,d4
001870  6100 0642               2321.           bsr errnsyntax  * nur An erlaubt
001874  3002                    2322.           move d2,d0
```

```
001876  C07C 0007          2323.          and #$7.d0
00187A  807C 4E60          2324.          or #$4e60.d0
00187E  6100 E90C          2325.          bsr outword
001882  6000'E77C          2326.          bra carres
                           2327.
                           2328.
001886                     2329. cod1move:                * MOVE SR.ea
001886  225F               2330.          move.l (a7)+.a1 * a0 lassen
001888  6100 EB44          2331.          bsr kommack
00188C  6100 0618          2332.          bsr errcsyntax
001890  3B7C 40C0          2333.          move #$40C0.befcode(a5)
        '0000
001896  6100 E9EA          2334.          bsr getea       * eff adresse holen
00189A  C8BC 000000A0      2335.          and.l #areglocadr.d4
0018A0  6100 0632          2336.          bsr errnadress
0018A4  6000 FDB6          2337.          bra codea       * ablegen std.
                           2338.
                           2339.
0018A8                     2340. cod2move:                * MOVE USP.An
0018A8  225F               2341.          move.l (a7)+.a1
0018AA  6100 EB22          2342.          bsr kommack
0018AE  6100 05F6          2343.          bsr errcsyntax
0018B2  6100 EC14          2344.          bsr checkan
0018B6  6100 05EE          2345.          bsr errcsyntax
0018BA  303C 4E68          2346.          move #$4e68.d0  *
0018BE  8045               2347.          or d5.d0        * irrr
0018C0  6100 E8CA          2348.          bsr outword
0018C4  6000'E73A          2349.          bra carres
                           2350.
0018C8                     2351. cod3move:                * MOVE #data.
0018C8  6100 E736          2352.          bsr exori       * wert holen nach D0
0018CC  6100 EB00          2353.          bsr kommack
0018D0  6100 05D4          2354.          bsr errcsyntax  * komma muss da sein
0018D4  2F00               2355.          move.l d0.-(a7)
0018D6  2F08               2356.          move.l a0.-(a7)
0018D8  6100 E7DA          2357.          bsr setioname   * fuer vergleich
0018DC  6500 0016          2358.          bcs cod2nv
0018E0  B4BC 43435220      2359.          cmp.l #'CCR '.d2
0018E6  6700 008E          2360.          beq cod4move
0018EA  B4BC 53522020      2361.          cmp.l #'SR  '.d2
0018F0  6700 00A0          2362.          beq cod5move
0018F4                     2363. cod2nv:
0018F4  205F               2364.          move.l (a7)+.a0 * MOVE #data.ea
0018F6  2A1F               2365.          move.l (a7)+.d5
0018F8  2F05               2366.          move.l d5.-(a7)
0018FA  6100 E986          2367.          bsr getea
0018FE  2A1F               2368.          move.l (a7)+.d5
001900  C8BC 000000A0      2369.          and.l #areglocadr.d4
001906  6100 058C          2370.          bsr errnber
00190A  48E7 F800          2371.          movem.l d0-d4.-(a7)     * ACHTUNG STACK BELEGT
00190E  302D'0000          2372.          move wordbyte(a5).d0    *0.1.2
001912  5240               2373.          addq #1.d0      *1.2.3  -> 1.3.2
001914  B07C 0001          2374.          cmp #1.d0
001918  6704               2375.          beq.s cod12mv
00191A  0A40 0001          2376.          eori #1.d0      * 2 <-> 3
00191E                     2377. cod12mv:
00191E  E758               2378.          rol #3.d0
001920  3202               2379.          move d2.d1      * mode umbauen
001922  C27C 0007          2380.          and #$7.d1
001926  8041               2381.          or d1.d0
001928  E758               2382.          rol #3.d0       *
00192A  3202               2383.          move d2.d1
00192C  E659               2384.          ror #3.d1       * Mode bits
00192E  C27C 0007          2385.          and #$7.d1
001932  8041               2386.          or d1.d0
001934  ED58               2387.          rol #6.d0
001936  807C 003C          2388.          or #%111100.d0  * IMM Bits
00193A  6100 E850          2389.          bsr outword     * Befehl ausgegeben
00193E  302D'0000          2390.          move wordbyte(a5).d0
001942  4A40               2391.          tst d0
001944  6700 001A          2392.          beq cod13mv
001948  B07C 0001          2393.          cmp #1.d0
00194C  6700 000A          2394.          beq cod14mv
                           2395.                          * Long
001950  2005               2396.          move.l d5.d0
001952  6100 E7D8          2397.          bsr outlong
001956  6012               2398.          bra.s cod15mv
                           2399.
001958                     2400. cod14mv:                 * Word
001958  2005               2401.          move.l d5.d0
```

```
00195A  6100 E830       2402.          bsr outword
00195E  600A            2403.          bra.s cod15mv
                        2404.
001960                  2405. cod13mv:                 * Byte
001960  2005            2406.          move.l d5.d0
001962  C07C 00FF       2407.          and #$FF.d0
001966  6100 E824       2408.          bsr outword
00196A                  2409. cod15mv:                 * weiter hier
00196A  4CDF 001F       2410.          movem.l (a7)+.d0-d4
00196E  6100 EB7C       2411.          bsr outea
001972  6000 E68C       2412.          bra carres
                        2413.
                        2414.
                        2415.
001976                  2416. cod4move:                * MOVE #data.CCR
001976  225F            2417.          move.l (a7)+.a1
001978  303C 44FC       2418.          move #$44fc.d0
00197C  6100 E80E       2419.          bsr outword
001980  201F            2420.          move.l (a7)+.d0
001982  6100 04C0       2421.          bsr rangewck
001986  6100 04FA       2422.          bsr errcber
00198A  6100 E800       2423.          bsr outword
00198E  6000 E670       2424.          bra carres
                        2425.
                        2426.
001992                  2427. cod5move:                * MOVE #data.SR
001992  225F            2428.          move.l (a7)+.a1
001994  303C 46FC       2429.          move #$46fc.d0
001998  6100 E7F2       2430.          bsr outword
00199C  201F            2431.          move.l (a7)+.d0
00199E  6100 04A4       2432.          bsr rangewck
0019A2  6100 04DE       2433.          bsr errcber
0019A6  6100 E7E4       2434.          bsr outword
0019AA  6000 E654       2435.          bra carres
                        2436.
                        2437.
0019AE                  2438. codmovem:                * MOVEM reglist.ea
                        2439.                          * MOVEM ea.reglist
0019AE  2F08            2440.          move.l a0.-(a7)
0019B0  6100 EAE6       2441.          bsr checkdn     * start bei Dn ?
0019B4  6406            2442.          bcc.s cod1mm
0019B6  6100 EB10       2443.          bsr checkan
0019BA  6560            2444.          bcs.s cod2mm    *
0019BC                  2445. cod1mm:                  * reglist.ea   D5 = erster Wert
0019BC  205F            2446.          move.l (a7)+.a0
0019BE  6100 00AC       2447.          bsr getregli    * registerliste holen nach D4
0019C2  6100 04E2       2448.          bsr errcsyntax
0019C6  6100 EA06       2449.          bsr kommack
0019CA  6100 04DA       2450.          bsr errcsyntax
0019CE  2F04            2451.          move.l d4.-(a7)
0019D0  6100 E8B0       2452.          bsr getea
0019D4  2A1F            2453.          move.l (a7)+.d5 * wert merken
0019D6  B8BC 00000400   2454.          cmp.l #decre.d4 * dann revlist
0019DC  6604            2455.          bne.s cod12mm
0019DE  6100 0112       2456.          bsr revlist     * bei predecrement D0..D7 A0..A7
0019E2                  2457. cod12mm:
0019E2  C8BC 000003A0   2458.          and.l #dreg!areg!incre!ocadr.d4
0019E8  6100 04EA       2459.          bsr errnadress  * fehler adresse
                        2460.                          * dr = 0
0019EC  4A6D 0000       2461.          tst wordbyte(a5)
0019F0  6100 04BC       2462.          bsr erresyntax  * .B nicht erlaubt
0019F4  0C6D 0001       2463.          cmp #1.wordbyte(a5)
        0000
0019FA  670C            2464.          beq.s cod13mm
0019FC  382D 0000       2465.          move befcode(a5).d4
001A00  887C 0040       2466.          or #%1000000.d4
001A04  3B44 0000       2467.          move d4.befcode(a5)     *
001A08                  2468. cod13mm:
001A08  6100 EACC       2469.          bsr outorea     * befcode(a5) zuerst
001A0C  C145            2470.          exg d0.d5
001A0E  6100 E77C       2471.          bsr outword
001A12  C145            2472.          exg d0.d5       * displacement
001A14  6100 EAD6       2473.          bsr outea
001A18  6000 E5E6       2474.          bra carres
                        2475.
                        2476.
001A1C                  2477. cod2mm:                  * ea.reglist
001A1C  205F            2478.          move.l (a7)+.a0
001A1E  6100 E862       2479.          bsr getea       * ea holen
001A22  C8BC 00000580   2480.          and.l #dreg!areg!decre.d4
```

```
001A28  6100 04AA       2481.           bsr errnadress
001A2C  6100 E9A0       2482.           bsr kommack
001A30  6100 0474       2483.           bsr errcsvntax
001A34  6100 0036       2484.           bsr getreqli    * registerliste nach d4
001A38  3A2D 0000       2485.           move befcode(a5).d5
001A3C  8A7C 0400       2486.           or #$400.d5     * dr = 1
001A40  4A6D 0000       2487.           tst wordbyte(a5)
001A44  6100 0468       2488.           bsr erresvntax
001A48  0C6D 0001       2489.           cmp #1.wordbyte(a5)
        0000
001A4E  6704            2490.           beq.s cod21mm
001A50  8A7C 0040       2491.           or #%1000000.d5
001A54                  2492. cod21mm:
001A54  3B45 0000       2493.           move d5.befcode(a5)
001A58  6100 EA7C       2494.           bsr outorea
001A5C  C940            2495.           exg d4.d0
001A5E  6100 E72C       2496.           bsr outword
001A62  C940            2497.           exg d4.d0
001A64  6100 EA86       2498.           bsr outea
001A68  6000 E596       2499.           bra carres
                        2500.
                        2501.
001A6C                  2502. getreqli:
001A6C  4284            2503.           clr.l d4        * Ziel
001A6E  48E7 F400       2504.           movem.l d0-d3/d5.-(a7)
001A72  6100 000A       2505.           bsr get1reqli
001A76  4CDF 002F       2506.           movem.l (a7)+.d0-d3/d5  * rev 4.0 . aber unkrit
001A7A  4E75            2507.           rts
                        2508.
001A7C                  2509. getreqli:
001A7C  5248            2510.           addq #1.a0
001A7E                  2511. get1reqli:                * Dn  - Dn / An - An ///
                        2512.                           * Ergebnis A7..A0 D7..D0
001A7E  6100 EA18       2513.           bsr checkdn     * Nach D4
001A82  6500 0030       2514.           bcs get1re
001A86  08C4            2515.           bset d5.d4      * setzen dynamisch
001A88  6100 E576       2516.           bsr iobn
001A8C                  2517. get2reqli:
001A8C  0C10 002F       2518.           cmp.b # / .(a0)
001A90  67EA            2519.           beq getreqli
001A92  0C10 002D       2520.           cmp.b # - .(a0)
001A96  6600 E568       2521.           bne carres      * ENDE
001A9A  5248            2522.           addq #1.a0
001A9C  3005            2523.           move d5.d0      * D0 destroyed
001A9E  6100 E9F8       2524.           bsr checkdn
001AA2  6500 E55C       2525.           bcs carset      * Fehler
001AA6                  2526. get1re:
001AA6  BA40            2527.           cmp d0.d5
001AA8  67E2            2528.           beq get2reqli   * fertig
001AAA  5240            2529.           add #1.d0
001AAC  C07C 0007       2530.           and #$7.d0      * max 0..7
001AB0  01C4            2531.           bset d0.d4      * Ziel
001AB2  60F2            2532.           bra get1re
                        2533.
                        2534.
001AB4                  2535. getire:                   * An
001AB4  6100 EA12       2536.           bsr checkan     *
001AB8  6500 E546       2537.           bcs carset      * Abbruch mit FEHLER
001ABC  5045            2538.           add #8.d5       * in MSB-Bereich
001ABE  08C4            2539.           bset d5.d4
001AC0  6100 E53E       2540.           bsr iobn
001AC4  0C10 002F       2541.           cmp.b #'/'.(a0)
001AC8  67B2            2542.           beq getreqli
001ACA  0C10 002D       2543.           cmp.b # -'.(a0)
001ACE  6600 E530       2544.           bne carres      * ENDE
001AD2  5248            2545.           addq #1.a0
001AD4  3005            2546.           move d5.d0      * Merken dort
001AD6  6100 E9F0       2547.           bsr checkan
001ADA  5045            2548.           add #8.d5       * Bereich 8..15
001ADC  6500 E522       2549.           bcs carset
001AE0                  2550. get12re:
001AE0  BA40            2551.           cmp d0.d5
001AE2  67A8            2552.           beq get2reqli   * fertig
001AE4  5240            2553.           add #1.d0
001AE6  C07C 0007       2554.           and #$7.d0      * max 0..7
001AEA  807C 0008       2555.           or #$8.d0       * Bereich 8..15
001AEE  01C4            2556.           bset d0.d4      * Ziel
001AF0  60EE            2557.           bra get12re
                        2558.
                        2559.
```

```
                                2560.
                                2561.
001AF2                          2562. revlist:                  * umdrehen d15 .. d0
                                2563.                           * reg D5
001AF2  48E7 6000               2564.           movem.l d1-d2,-(a7)
001AF6  343C 000F               2565.           move #16-1,d2
001AFA                          2566. revllist:
001AFA  E355                    2567.           roxl #1,d5      * ziel d1
001AFC  E251                    2568.           roxr #1,d1
001AFE  51CA FFFA               2569.           dbra d2,revllist
001B02  3A01                    2570.           move d1,d5
001B04  4CDF 0006               2571.           movem.l (a7)+,d1-d2
001B08  4E75                    2572.           rts
                                2573.
                                2574.
                                2575.
001B0A                          2576. codmoveo:                 * MOVEP Dn,d(Av)
                                2577.                           * MOVEP d(Av),Dn
001B0A  6100 E98C               2578.           bsr checkdn     * a0 retten
001B0E  6500 0060               2579.           bcs codlmoveo   * Dn,d(Av)
001B12  EE5D                    2580.           ror #7,d5
001B14  8A6D 0000               2581.           or befcode(a5),d5
001B18  4A6D 0000               2582.           tst wordbyte(a5)
001B1C  6100 0390               2583.           bsr erresyntax
001B20  0C6D 0001               2584.           cmp #1,wordbyte(a5)
        0000
001B26  6700 000A               2585.           beq cod12mo
001B2A  8A7C 01C0               2586.           or #%111000000,d5       * long
001B2E  6000 0006               2587.           bra cod13mo
001B32                          2588. cod12mo:
001B32  8A7C 0180               2589.           or #%110000000,d5
001B36                          2590. cod13mo:
001B36  3B45 0000               2591.           move d5,befcode(a5)     * neuer Befehlscode
001B3A  6100 E892               2592.           bsr kommack
001B3E  6100 E742               2593.           bsr getea
001B42  3202                    2594.           move d2,d1      * register merken
001B44  C47C 0038               2595.           and #%111000,d2
001B48  B47C 0028               2596.           cmp #%101000,d2
001B4C  6100 0386               2597.           bsr errnadress
001B50  C27C 0007               2598.           and #$7,d1
001B54  C141                    2599.           exg d0,d1
001B56  806D 0000               2600.           or befcode(a5),d0
001B5A  6100 E630               2601.           bsr putword
001B5E  C141                    2602.           exg d0,d1       * Datenwert
001B60  6100 02E2               2603.           bsr rangewck
001B64  6100 031C               2604.           bsr errcber
001B68  6100 E622               2605.           bsr putword
001B6C  6000 E492               2606.           bra carres      * ok fertig
                                2607.
                                2608.
001B70                          2609. codlmoveo:                * d(Av),Dn
001B70  6100 E710               2610.           bsr getea       * check zuerst
001B74  3202                    2611.           move d2,d1      * d0 = displacement
001B76  C47C 0038               2612.           and #%111000,d2
001B7A  B47C 0028               2613.           cmp #%101000,d2
001B7E  6100 0354               2614.           bsr errnadress
001B82  6100 E84A               2615.           bsr kommack
001B86  6100 031E               2616.           bsr errcsyntax
001B8A  6100 E90C               2617.           bsr checkdn
001B8E  6100 0316               2618.           bsr errcsyntax
001B92  EE5D                    2619.           ror #7,d5
001B94  8A6D 0000               2620.           or befcode(a5),d5
001B98  C27C 0007               2621.           and #$7,d1      * displ
001B9C  8245                    2622.           or d5,d1        * ziel
001B9E  4A6D 0000               2623.           tst wordbyte(a5)
001BA2  6100 030A               2624.           bsr erresyntax
001BA6  0C6D 0001               2625.           cmp #1,wordbyte(a5)
        0000
001BAC  6700 000A               2626.           beq cod2mo
001BB0  827C 0140               2627.           or #%101000000,d1       * LONG
001BB4  6000 0006               2628.           bra cod3mo
                                2629.
001BB8                          2630. cod2mo:
001BB8  827C 0100               2631.           or #%100000000,d1
001BBC  C141                    2632. cod3mo:   exg d0,d1
001BBE  6100 E5CC               2633.           bsr putword     * Befehle
001BC2  C141                    2634.           exg d0,d1
001BC4  6100 027E               2635.           bsr rangewck
001BC8  6100 02DC               2636.           bsr errcsyntax
001BCC  6100 E5BE               2637.           bsr putword
```

```
001BD0  6000'E42E          2638.          bra carres
                           2639.
                           2640.
001BD4                     2641. codmoveq:                * MOVEQ #data.Dn
001BD4  6100 00D0          2642.          bsr immeck
001BD8  6500'E426          2643.          bcs carset
001BDC  6100 E422          2644.          bsr exor1
001BE0  6100 E7EC          2645.          bsr kommack
001BE4  6100 02C0          2646.          bsr errcsyntax
001BE8  6100 E8AE          2647.          bsr checkdn
001BEC  6100 02B8          2648.          bsr errcsyntax
001BF0  EE5D               2649.          ror #7.d5
001BF2  8A6D'0000          2650.          or befcode(a5).d5
001BF6  6100 0204          2651.          bsr rangebck
001BFA  6100 02B6          2652.          bsr errcber
001BFE  C07C 00FF          2653.          and #$FF.d0      * 0..255 Bereich zulaessig
001C02  8045               2654.          or d5.d0
001C04  6100 E586          2655.          bsr outword      * 0111rrr0ddddddddd
001C08  6000'E3F6          2656.          bra carres
                           2657.
001C0C                     2658. codneg:                  * NEG.s <ea>
001C0C  6100 F502          2659.          bsr sizeein
001C10  6100 E670          2660.          bsr getea
001C14  C8BC 000000A0      2661.          and.l #areg!pcadr.d4
001C1A  6100 02B8          2662.          bsr errnadress
001C1E  6000 FA3C          2663.          bra codea        * dann wie eff mode
                           2664.
                           2665.
                           2666.
001C22                     2667. codscc:                  * Scc
001C22  6100 E65E          2668.          bsr getea        * effektive Adresse holen
                           2669.                           * d1 = 0 dann fehler
001C26  C8BC 00000080      2670.          and.l #areg.d4   * nur AREG nicht erlaubt
001C2C  6100 02A6          2671.          bsr errnadress   * FEHLER nicht erlaubter adressmode
                           2672.
001C30  6100 E8A4          2673.          bsr putorea      * ea odern und bef ausgeben
001C34  6100 E8B6          2674.          bsr putea        * Adressteil falls da ablegen
001C38  4E75               2675.          rts
                           2676.
                           2677.
                           2678.
001C3A                     2679. codstop:                 * STOP
001C3A  6100 006A          2680.          bsr immeck
001C3E  6100 0266          2681.          bsr errcsyntax   * STOP #data
001C42  302D'0000          2682.          move befcode(a5).d0
001C46  6100 E544          2683.          bsr outword
001C4A  6100 E3B4          2684.          bsr exor1        * STOP data
001C4E  6100 01F4          2685.          bsr rangewck
001C52  6100 022E          2686.          bsr errcber
001C56  6100 E534          2687.          bsr outword      * Ausgabe
001C5A  6000'E3A4          2688.          bra carres
                           2689.
001C5E                     2690. codswap:                 * SWAP Dn
001C5E  6100 E838          2691.          bsr checkdn
001C62  6100 0242          2692.          bsr errcsyntax
001C66  8A6D'0000          2693.          or befcode(a5).d5
001C6A  3005               2694.          move d5.d0
001C6C  6100 E51E          2695.          bsr outword
001C70  6000'E38E          2696.          bra carres
                           2697.
                           2698.
                           2699.
001C74                     2700. codtrap:                 * TRAP
001C74  6100 0030          2701.          bsr immeck       * TRAP #data
001C78  6100 022C          2702.          bsr errcsyntax
001C7C  6100 E382          2703.          bsr exor1
001C80  C07C 000F          2704.          and #$f.d0
001C84  806D'0000          2705.          or befcode(a5).d0
001C88  6100 E502          2706.          bsr outword
001C8C  6000'E372          2707.          bra carres
                           2708.
001C90                     2709. codunlk:                 * UNLK  An
001C90  6100 E836          2710.          bsr checkan
001C94  6100 0210          2711.          bsr errcsyntax
001C98  8A6D'0000          2712.          or befcode(a5).d5
001C9C  3005               2713.          move d5.d0
001C9E  6100 E4EC          2714.          bsr outword
001CA2  6000'E35C          2715.          bra carres
                           2716.
                           2717.
```

```
001CA6                  2718. immeck:
001CA6  6100 E358       2719.           bsr iobn
001CAA  0C10 0023       2720.           cmp.b #'#'.(a0)
001CAE  6600 E350       2721.           bne carset
001CB2  5248            2722.           addq #1.a0
001CB4  6000 E34A       2723.           bra carres
                        2724.
                        2725. ** Pseudobefehle
                        2726.
001CB8                  2727. coddc:                    * CONSTANT
001CB8  302D 0000       2728.           move wordbyte(a5).d0
001CBC  4A40            2729.           tst d0
001CBE  672C            2730.           beq.s coddcb
001CC0  0C6D 0001       2731.           cmp #1.wordbyte(a5)
        0000
001CC6  6712            2732.           beq.s coddcw
                        2733.                           * long
001CC8                  2734. coddcl:                   * in Schleife ausfuehren
001CC8  6100 E336       2735.           bsr exor1       * bis error oder undef
001CCC  6100 E45E       2736.           bsr outlong
001CD0  6100 E6FC       2737.           bsr kommack     * Ende
001CD4  6500 E32A       2738.           bcs carres
001CD8  60EE            2739.           bra coddcl
                        2740.
001CDA                  2741. coddcw:                   * word
001CDA  6100 E324       2742.           bsr exor1
001CDE  6100 E4AC       2743.           bsr outword
001CE2  6100 E6EA       2744.           bsr kommack
001CE6  6500 E318       2745.           bcs carres
001CEA  60EE            2746.           bra coddcw
                        2747.
001CEC                  2748. coddcb:                   * byte
001CEC  6100 E312       2749.           bsr iobn
001CF0  0C10 0027       2750.           cmp.b #$27.(a0) * Text dann spezial
001CF4  6716            2751.           beq.s cod2dcb
001CF6  6100 E308       2752.           bsr exor1
001CFA  C07C 00FF       2753.           and #$ff.d0
001CFE  6100 E4EA       2754.           bsr outbyte
001D02                  2755. cod1dcb:
001D02  6100 E6CA       2756.           bsr kommack
001D06  6500 E2F8       2757.           bcs carres
001D0A  60E0            2758.           bra coddcb
                        2759.
001D0C                  2760. cod2dcb:
001D0C  5248            2761.           addq #1.a0
001D0E  0C10 0027       2762.           cmp.b #$27.(a0) * ''
001D12  6710            2763.           beq.s cod3dcb   * weiter dort
001D14  0C10 0020       2764.           cmp.b #' '.(a0)
001D18  6100 018C       2765.           bsr errcsyntax  * < blank nicht erlaubt
001D1C  1010            2766.           move.b (a0).d0
001D1E  6100 E502       2767.           bsr outobyte    * ok code ablegen ohne blank
001D22  60E8            2768.           bra cod2dcb
001D24                  2769. cod3dcb:
001D24  5248            2770.           addq #1.a0      * ok erkannt
001D26  60DA            2771.           bra cod1dcb
                        2772.
001D28                  2773. codoff:
001D28  6100 E2D6       2774.           bsr exor1       * Wert holen
001D2C  0800 0000       2775.           btst.l #0.d0
001D30  6100 0182       2776.           bsr errnsyntax  * Adjust Fehler Syntax
001D34  C0BC FFFFFFFE   2777.           and.l #$fffffffe.d0
001D3A  2B40 0000       2778.           move.l d0.offset(a5)    * Neuer Wert
001D3E  6000 E2C0       2779.           bra carres
                        2780.
001D42                  2781. codds:                    * STORAGE
                        2782.                           * rev 4.0. Korrektur
                        2783.                           * DS.L war vorher falsch
001D42  6100 E2BC       2784.           bsr exor1
001D46  322D 0000       2785.           move wordbyte(a5).d1    * 0.1.2
001D4A  4A80            2786.           tst.l d0
001D4C  6700 0044       2787.           beq codadjust
001D50  4282            2788.           clr.l d2
001D52  B27C 0000       2789.           cmp #0.d1
001D56  6604            2790.           bne.s codds2a
001D58  D480            2791.           add.l d0.d2
001D5A  6020            2792.           bra.s codds2f
001D5C                  2793. codds2a:
001D5C  B27C 0001       2794.           cmp #1.d1
001D60  6606            2795.           bne.s codds2b
001D62  D480            2796.           add.l d0.d2
```

```
001D64  D480            2797.           add.l d0.d2
001D66  6014            2798.           bra.s codds2f
001D68                  2799.  codds2b:
001D68  B27C 0002       2800.           cmp #2.d1
001D6C  660A            2801.           bne.s codds2c
001D6E  D480            2802.           add.l d0.d2
001D70  D480            2803.           add.l d0.d2
001D72  D480            2804.           add.l d0.d2
001D74  D480            2805.           add.l d0.d2
001D76  6004            2806.           bra.s codds2f
001D78                  2807.  codds2c:
001D78  6000'E286       2808.           bra carset      * Fehlerhafter Code
                        2809.
001D7C                  2810.  codds2f:                 * Ende Bearbeitung
001D7C  D4AD'0000       2811.           add.l pcstand(a5).d2
001D80  2B42'0000       2812.           move.l d2.pcstand(a5)   * reservierte Symbole
001D84  6000'E27A       2813.           bra carres
                        2814.
                        2815.
                        2816.
001D88                  2817.  codend:                  * END
001D88  3B7C 0002       2818.           move #2.errflag(a5)     * ENDE KENNUNG
        '0000
001D8E  6000'E270       2819.           bra carres      * flag Setzen vorher
                        2820.
                        2821.
001D92                  2822.  codadjust:               * wenn ungleicher PC dann
                        2823.                           * angleichen
001D92  202D'0000       2824.           move.l pcstand(a5).d0
001D96  C0BC 00000001   2825.           and.l #$1.d0
001D9C  6700'E262       2826.           beq carres      * ja fertig
001DA0  4240            2827.           clr d0
001DA2  6100'E446       2828.           bsr putbyte
001DA6  6000'E258       2829.           bra carres      * sonst Fuell-Byte ausgeben
                        2830.
                        2831.
001DAA                  2832.  codorg:                  * ORG
001DAA  6100'E254       2833.           bsr exori       * von a0 an holen
001DAE  2B40'0000       2834.           move.l d0.pcstand(a5)   * neuer PC - Stand
001DB2  41ED'0000       2835.           lea ausbuf(a5).a0       * a0 destroy
001DB6  6100'E30E       2836.           bsr putadr      * neue Adresse gleich eintragen
001DBA  6000'E244       2837.           bra carres      * ok kein Fehler
                        2838.
                        2839.
                        2840.
                        2841.  ** Fehlerbehandlung verschiedene Einsprunge
                        2842.
                        2843.
001DBE                  2844.  range07:                 * nur 0..7 erlaubt
001DBE  0C6D 0002       2845.           cmp #2.passflag(a5)
        '0000
001DC4  6600'E23A       2846.           bne carres      * Nur wenn PASS2
001DC8  2F00            2847.           move.l d0.-(a7)
001DCA  C0BC FFFFFFF8   2848.           and.l #$FFFFFFF8.d0
001DD0  6700 0098       2849.           beq rangok      * keine negativen Zahlen hier
001DD4  6000 008E       2850.           bra rangfal
                        2851.
001DD8                  2852.  rangebick:               * d0 auf Bereich testen in BYTE Grenze
001DD8  0C6D 0002       2853.           cmp #2.passflag(a5)     * nur -n..0..+n
        '0000
001DDE  6600'E220       2854.           bne carres      * Nur wenn PASS2
001DE2  2F00            2855.           move.l d0.-(a7)
001DE4  C0BC FFFFFF80   2856.           and.l #$FFFFFF80.d0
001DEA  6700 007E       2857.           beq rangok      * 0..7f ist OK
001DEE  B0BC FFFFFF80   2858.           cmp.l #$FFFFFF80.d0     * 80h .. ffh auch ok wenn negativ
001DF4  6700 0074       2859.           beq rangok      * auch negative Zahlen ok
001DF8  6000 006A       2860.           bra rangfal
                        2861.
001DFC                  2862.  rangebck:               * d0 auf Bereich testen in BYTE Grenze
001DFC  0C6D 0002       2863.           cmp #2.passflag(a5)
        '0000
001E02  6600'E1FC       2864.           bne carres      * Nur wenn PASS2
001E06  2F00            2865.           move.l d0.-(a7)
001E08  C0BC FFFFFF00   2866.           and.l #$FFFFFF00.d0
001E0E  6700 005A       2867.           beq rangok
001E12  B0BC FFFFFF00   2868.           cmp.l #$FFFFFF00.d0     * Auch ok
001E18  6700 0050       2869.           beq rangok      * auch negative Zahlen ok
001E1C  6000 0046       2870.           bra rangfal
                        2871.
001E20                  2872.  rangewick:               * d0 auf Bereich testen in WORT Grenze
```

```
001E20  0C6D 0002       2873.           cmp #2.passflag(a5)
        0000
001E26  6600 E1DB       2874.           bne carres      * erst bei PASS2 als Fehler ok
001E2A  2F00            2875.           move.l d0.-(a7)
001E2C  C0BC FFFF8000   2876.           and.l #$FFFF8000.d0     *0..7fff
001E32  6700 0036       2877.           beq rangok
001E36  B0BC FFFF8000   2878.           cmp.l #$FFFF8000.d0     * Auch ok 8000..ffff wenn negative
001E3C  6700 002C       2879.           beq rangok      * auch negative Zahlen ok
001E40  6000 0022       2880.           bra rangfal
                        2881.
001E44                  2882. rangewck:                 * d0 auf Bereich testen in WORT Grenze
001E44  0C6D 0002       2883.           cmp #2.passflag(a5)
        0000
001E4A  6600 E184       2884.           bne carres      * erst bei PASS2 als Fehler ok
001E4E  2F00            2885.           move.l d0.-(a7)
001E50  C0BC FFFF0000   2886.           and.l #$FFFF0000.d0
001E56  6700 0012       2887.           beq rangok
001E5A  B0BC FFFF0000   2888.           cmp.l #$FFFF0000.d0     * Auch ok
001E60  6700 0008       2889.           beq rangok      * auch negative Zahlen ok
001E64                  2890. rangfal:                  * CARRY bei Fehler
001E64  201F            2891.           move.l (a7)+.d0
001E66  6000 E198       2892.           bra carset
                        2893.
001E6A  201F            2894. rangok: move.l (a7)+.d0
001E6C  6000 E192       2895.           bra carres
                        2896.
                        2897.
                        2898.
                        2899. * Fehler werden alle auch in PASS1 abgelegt. jedoch
                        2900. * erfolgt Meldung erst in PASS2
                        2901. * nachfolgende Programme werden als Unterprogramm
                        2902. * aufgerufen. wenn carry oder not equal erfolgt
                        2903. * eintrag. wenn errflag ungleich 0 wird fehler
                        2904. * der zuerst da war. beibehalten
                        2905.
001E70                  2906. errende:
001E70  3B7C 0001       2907.           move #1.errflag(a5)
        0000
001E76  2B48 0000       2908.           move.l a0.errpoi(a5)
001E7A  526D 0000       2909.           add #1.errcnt(a5)
001E7E  6000 E180       2910.           bra carset
                        2911.
001E82                  2912. errcber:
001E82  6400 E17C       2913.           bcc carres      * kein fehler
001E86  4A6D 0000       2914.           tst errflag(a5)
001E8A  6600 E174       2915.           bne carres
001E8E  426D 0000       2916.           clr errart(a5)
001E92  60DC            2917.           bra errende
                        2918.
001E94                  2919. errnber:
001E94  6700 E16A       2920.           beq carres
001E98  4A6D 0000       2921.           tst errflag(a5)
001E9C  6600 E162       2922.           bne carres
001EA0  426D 0000       2923.           clr errart(a5)
001EA4  60CA            2924.           bra errende
                        2925.
001EA6                  2926. errcsyntax:
001EA6  6400 E158       2927.           bcc carres
001EAA  6000 000C       2928.           bra errn1
                        2929.
001E                    2930. erresyntax:
001E    6600 E150       2931.           bne carres
001EB2  6004            2932.           bra.s errn1
                        2933.
001EB4                  2934. errnsyntax:
001EB4  6700 E14A       2935.           beq carres
001EB8                  2936. errn1:
001EB8  4A6D 0000       2937.           tst errflag(a5)
001EBC  6600 E142       2938.           bne carres
001EC0  3B7C 0001       2939.           move #1.errart(a5)
        0000
001EC6  60A8            2940.           bra errende
                        2941.
001EC8                  2942. errcadress:
001EC8  6400 E136       2943.           bcc carres
001ECC  600A            2944.           bra.s erradr1
                        2945.
001ECE                  2946. erreadress:
001ECE  6600 E130       2947.           bne carres
001ED2  6004            2948.           bra.s erradr1
```

```
                                2949.
001ED4                          2950. errnadress:
001ED4  6700'E12A               2951.           beq carres
001ED8                          2952. erradr1:
001ED8  4A6D'0000               2953.           tst errflag(a5)
001EDC  6600'E122               2954.           bne carres
001EE0  3B7C 0002               2955.           move #2.errart(a5)
        '0000
001EE6  6088                    2956.           bra errende
                                2957.
001EE8                          2958. errcundef:
001EE8  6400'E116               2959.           bcc carres
001EEC  6000 000E               2960.           bra errniu
                                2961.
001EF0                          2962. erreundef:
001EF0  6600'E10E               2963.           bne carres
001EF4  6000 0006               2964.           bra errniu
001EF8                          2965. errnundef:
001EF8  6700'E106               2966.           beq carres
001EFC                          2967. errniu:
001EFC  4A6D'0000               2968.           tst errflag(a5)
001F00  6600'E0FE               2969.           bne carres
001F04  3B7C 0003               2970.           move #3.errart(a5)
        '0000
001F0A  6000 FF64               2971.           bra errende
                                2972.
001F0E                          2973. errnmult:
001F0E  6700'E0F0               2974.           beq carres
001F12  4A6D'0000               2975.           tst errflag(a5)
001F16  6600'E0E8               2976.           bne carres
001F1A  3B7C 0004               2977.           move #4.errart(a5)
        '0000
001F20  6000 FF4E               2978.           bra errende
                                2979.
                                2980. * 5 ist adjust
                                2981.
                                2982.
001F24                          2983. erranalyse:               * Fehler analyse mit Ausgabe
001F24  4A6D'0000               2984.           tst errflag(a5)
001F28  6700'E0D6               2985.           beq carset      * Fehler, kein Fehler da
001F2C  6100 E316               2986.           bsr newout      * alte Zeile ausgeben
001F30  206D'0000               2987.           movea.l errpoi(a5).a0   * Pointer auf Fehlerquelle
001F34  43FA 0020               2988.           lea errtab1(pc).a1
001F38  302D'0000               2989.           move errart(a5).d0
001F3C  C07C 0007               2990.           and #$7.d0      *0.1.2.3 als Fehlerart ok
001F40  C0FC 0002               2991.           mulu #2.d0
001F44  D2C0                    2992.           adda d0.a1      * Wortgross rev 4.0b
001F46  3011                    2993.           move.w (a1).d0  * indirekte Adresse
001F48  43FA 000C               2994.           lea errtab1(pc).a1
001F4C  D2C0                    2995.           adda.w d0.a1    * relativ rev 4.0b
001F4E  6100 0016               2996.           bsr errprt
001F52  6000'E0AC               2997.           bra carres      * ok Fehler draussen
                                2998.
                                2999.
001F56                          3000. errtab1:
001F56  0056                    3001.           dc.w txtber-errtab1     * 0
001F58  006C                    3002.           dc.w txtsyn-errtab1     * 1
001F5A  007C                    3003.           dc.w txtadr-errtab1     * 2
001F5C  0097                    3004.           dc.w txtundef-errtab1   * 3
001F5E  00AE                    3005.           dc.w txtmult-errtab1    * 4
001F60  00C3                    3006.           dc.w txtadj-errtab1
001F62  006C                    3007.           dc.w txtsyn-errtab1
001F64  006C                    3008.           dc.w txtsyn-errtab1
                                3009.
001F66                          3010. errprt:                   * a0 ist fehlerposition
001F66  2008                    3011.           move.l a0.d0
001F68  41ED'0001               3012.           lea einbuf+1(a5).a0 *rev 4.0b
001F6C  9088                    3013.           sub.l a0.d0     * eins subtr. mehr wegen dbra
001F6E  D0BC 00000018           3014.           add.l #insst.d0 * Laenge der Zeile addieren
001F74  C07C 00FF               3015.           and #$ff.d0     *MAX 255
001F78  3200                    3016.           move d0.d1
001F7A                          3017. errprt1:
001F7A  103C 005F               3018.           move.b #'_'.d0
001F7E  6100'E080               3019.           bsr co2         * ** Ausgabe soez
001F82  51C9 FFF6               3020.           dbra d1.errprt1
001F86  103C 005E               3021.           move.b #'^'.d0
001F8A  6100'E074               3022.           bsr co2         * dort markieren
001F8E  103C 000D               3023.           move.b #$d.d0
001F92  6100'E06C               3024.           bsr co2
001F96  103C 000A               3025.           move.b #$a.d0
```

```
001F9A  6100'E064       3026.           bsr co2
                        3027.
001F9E                  3028.  txtort:
001F9E  1019            3029.           move.b (a1)+,d0
001FA0  4A00            3030.           tst.b d0
001FA2  6700'E05C       3031.           beq carres
001FA6  6100'E058       3032.           bsr co2         * ** AUSGABE z.Z
001FAA  60F2            3033.           bra txtort
                        3034.
001FAC                  3035.  txtber:
001FAC  57 65 72 74     3036.           dc.b 'Wertebereich falsch'.$d.$a.0
        65 62 65 72 65
        69 63 68 20 66
        61 6C 73 63 68
        0D 0A 00
                        3037.
001FC2                  3038.  txtsyn:
001FC2  53 79 6E 74     3039.           dc.b 'Syntax Fehler'.$d.$a.0
        61 78 20 46 65
        68 6C 65 72 0D
        0A 00
                        3040.
001FD2                  3041.  txtadr:
001FD2  41 64 72 65     3042.           dc.b 'Adressmode nicht erlaubt'.$d.$a.0
        73 73 6D 6F 64
        65 20 6E 69 63
        68 74 20 65 72
        6C 61 75 62 74
        0D 0A 00
                        3043.
001FED                  3044.  txtundef:
001FED  55 6E 64 65     3045.           dc.b 'Undefiniertes Symbol'.$d.$a.0
        66 69 6E 69 65
        72 74 65 73 20
        53 79 6D 62 6F
        6C 0D 0A 00
002004                  3046.  txtmult:
002004  4D 65 68 72     3047.           dc.b 'Mehrfach Definiert'.$d.$a.0
        66 61 63 68 20
        44 65 66 69 6E
        69 65 72 74 0D
        0A 00
002019                  3048.  txtadj:
002019  4E 69 63 68     3049.           dc.b 'Nicht auf Wortgrenze'.$d.$a
        74 20 61 75 66
        20 57 6F 72 74
        67 72 65 6E 7A
        65 0D 0A
                        3050.
        00 -- inserted
002030                  3051.           ds 0            * Wortgrenze setzen am Schluss
                        3052.
002030                  3053.           end
    0 Errors detected
```

```
...L68K 1.0  06/09/83   ...Run on 01/22/85
                          1.  ** rev 4.2 Link
                          2.  link grund
                          3.  link crt
                          4.  link flo2       * wegen Initaufruf
                          5.  link util
                          6.  link edit68
                          7.  link assem68
                          8.
000000                    9.  org $0
000000                   10.  section 1
                         11.
007FC6                   12.  end
    0 Errors detected

     Memory Map

00000000  1  :1
00000000  1  :1:GRUND
00000000  1      >>> START
0000002A  @      IOSTAT
0000002E  @      FLIP
0000002F  @      FLIPCNT
00000030  @      FLIP1
00000031  @      FLIP1CNT
00000032  @      PASSFLAG
00000034  @      ERRFLAG
00000036  @      STXTXT
0000003A  @      AKTTXT
0000003E  @      ETXTXT
00000042  @      EOTTXT
00000052  @      PCSTAND
00000056  @      ANFSTAND
0000005A  @      AUSPOI
00000062  @      AUSZAHL
00000070  @      AUSBUF
000000F4  @      EINBUF
000001EE  @      OFFSET
000001F2  @      PCORG
000001F6  @      BEFCODE
000001F8  @      ATTCODE
000001FA  @      WORDBYTE
000001FC  @      ERRCNT
000001FE  @      ERRART
00000200  @      ERRPOI
00000204  @      DEBUGST
00000208  @      DEBUGAK
00000217  @      VIEWPAGE
00000218  @      WRTPAGE
0000021B  @      GROESSE
0000021F  @      ZEILEN
00000220  @      DEBUG
00000221  @      INSL
00000222  @      CURON
00000223  @      CURX
00000224  @      CURY
00000225  @      ESCMERKER
00000226  @      NEUZEICH
00000227  @      TEMPZEICH
00000228  @      OPTFLAG
00000229  @      LINECNT
00000241  @      LINEPTR
00000259  @      SCREEN
000009D8  1      CSTS
00000A00  1      CI
00000A1C  @      SRSTO
00000A1E  @      STEPRATE
00000A1F  @      FEHZA
00000A20  @      CCODE
00000A21  @      DRVAT
00000A22  @      DRVTAB
00000A38  @      PAGECNT
00000A3A  @      PAGEBUF
00000A44  @      COLOR
00000A45  @      XORMODE
00000A4C  @      SYMNEXT
00000A4E  @      SYMTAB
00000A68  1      CIINIT2
00000A76  1      CI2
00000ACC  1      PRTCO2
```

```
00000AE0  1      CO2
00000D5C  1      WAIT
00000D66  1      CMD
00000D6E  1      ERAPEN
00000D7C  1      SETPEN
00000DA0  1      CLRALL
00000DC0  1      CLRINVIS
00000E14  1      AKTPAGE
00000EE2  1      MOVETO
00000F0C  1      GETXY
00000F4C  1      DRAWTO
00001386  1      TEXTAUS
00001486  1      TEXTEIN
000015FC  1      PRINT4D
00001658  1      PRINT6X
000017B6  1      IGBN
000017C2  1      EXPR
00001ABC  1      CARSET
00001AC2  1      CARRES
00001B2E  1      BUCHECK
00001B56  1      SETUPNAME
00001D0C  1      GETVAL
00001D14  1      SETVAL
00001D1C  1      NEWVAL
00003EA8  1    :1:CRT
00003EA8  1      CURSOREIN
00003EE6  1      CURSORAUS
00003F30  1      AKTCUR
00003FBA  1      CALCCUR
00003FBE  1      GETLINE
00003FEC  1      CLRSCREEN
00004074  1      PRTLINE
0000415A  1      PUTCHAR
000041BA  1      PROGZBE
000042FA  1      GETCODE
000043C6  1      CHARHANDLER
000043FA  1      CHAROSCR
000044DA  1      ERAEOLN
0000454A  1      INSCHAR
000045C2  1      DELCHAR
00004636  1      INSLINE
000046C4  1      DELLINE
00004902  1      CO
0000493A  1    :1:FLO2
0000493A  1      FLINIT
00004958  1      FLOPPY
00004992  1      GETFLOP
00004BE0  1    :1:UTIL
00004BE0  1      SIINIT
00004BEA  1      SI
00004BFA  1      SO
00004C10  1      SISTS
00004C20  1      SOSTS
00004C38  1      GETAD8
00004C68  1      GETAD10
00004C98  1      SETDA
00004CF0  1      GETUHR
00004D2E  1      SETUHR
00004D78  1      SPEAK
00004DC4  1      SPEAK1
00004E00  1      SOUND
00004E1E  1    :1:EDIT68
00004E1E  1      EDIT
00005F96  1    :1:ASSEM68
00006504  1      ASSEMBLE
00007FC5  0      ^0
00007FC5  1      ^1
00007FC5  2      ^2
00007FC5  3      ^3
00007FC5  4      ^4
00007FC5  5      ^5
00007FC5  6      ^6
00007FC5  7      ^7
00007FC5  8      ^8
00007FC5  9      ^9
00007FC5  A      ^A
00007FC5  B      ^B
00007FC5  C      ^C
00007FC5  D      ^D
00007FC5  E      ^E
```

Das Sonderheft von mc zum 68008-System

Mikro-Computer Schritt für Schritt 2

Hier zeigt sich, daß die Grundkonzeption des NDR-Klein-Computers so flexibel gestaltet ist, daß auf der Basis der in Teil 1 erarbeiteten Komponenten nahezu jede beliebige Mikrocomputer-Konfiguration aufgebaut werden kann. Zum einen kann man sich ein Z80-System herstellen, das mit dem Betriebssystem CP/M arbeiten kann. Ein Standard-Computer, dem die unendliche Welt der CP/M-Profi-Software offensteht. Für alle Wißbegierigen, die aus schulischen, beruflichen oder privaten Gründen einen modernen 8/16 Bit-Rechner kennenlernen wollen oder müssen, gibt es den Ausbau mit der 68008-CPU, die mit den anderen Komponenten des Systems problemlos zusammenarbeiten kann. Das NDR-Klein-Konzept bietet so gesehen dem absoluten Anfänger einen sicheren Einstieg, dem Lernenden einen geraden Weg in die Materie und dem Profi in der Industrie ein äußerst wandlungsfähiges Computersystem.

Bezugsmöglichkeiten

bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, in Mikrocomputer-Shops, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder direkt beim Franzis-Verlag gegen

- Voreinzahlung von 30,– DM (28,– DM + 2,– DM Porto) auf unser Postscheckkonto München Nr. 813 75-809 mit Hinweis „Mikrocomputer Schritt für Schritt, Teil 2" (Teil 1 erhalten Sie zum gleichen Preis) oder
- Zusendung eines Schecks (30,– DM)

Franzis'

Franzis-Verlag, Karlstraße 37,
8000 München 2, Telefon 0 89/51 17-2 39

In der Schweiz:
Verlag Thali AG, CH-6285 Hitzkirch

In Österreich: Erb-Verlag Ges.m.b.H.&Co.KG
Mariahilfer Straße 71, A-1061 Wien