M30 ST
M40 ST

System- und Programmierhandbuch MDOS/BASIC

olivetti L1

Druck Nr. 1808/1/7.83

Das Handbuch dient der Information, sein Inhalt ist ohne ausdrückliche schriftliche Vereinbarung nicht Vertragsgegenstand. Technische Änderungen behalten wir uns vor. Die angegebenen Daten sind lediglich Nominalwerte.

System- und Programmierhandbuch MDOS/BASIC

EINLEITUNG

Die Handbücher des Systems M30 ST/M40 ST (unter Betriebssystem MDOS) setzen sich aus 2 Bänden zusammen.

Der erste Band ist als "Benutzerhandbuch" ausgelegt und beschreibt die Hardwarekomponenten, die Bedienung des Systems, alle Dienstprogramme und die wichtigsten Befehle.

Der Ihnen vorliegende zweite Band beschreibt neben der Arbeitsweise des Betriebssystems alle Systembefehle und Dienstprogramme sowie die Programmierung in BASIC. Während sich der erste Band an Kunden mit Standardsoftware richtet, ist der zweite Band als Ergänzung für den Selbstprogrammierer gedacht.

Die Kapitel 1, 2 und 3 sind im vorliegenden "System- und Programmierhandbuch" nicht enthalten. Sie sind im "Benutzerhandbuch" nachzulesen.

INHALTSVERZEICHNIS

ÜBERBLICK ÜBER DIE KAPITEL

KAPITEL 1 bis 3 siehe "BENUTZERHANDBUCH"

4. ARBEITSWEISE DES BETRIEBSSYSTEMS MDOS

5. BIBLIOTHEKEN UND FILES

6. SYSTEMBEFEHLE

7. DIENSTPROGRAMME

8. PROGRAMMIERUNG IN BASIC

9. BASIC-ANWEISUNGEN

10. EINGABE UND EDITING EINES PROGRAMM- ODER TEXTFILES

11. CALCULATOR- UND DEBUGGING MODE

ANHANG

4.	ARBEITSWEISE DES BETRIEBSSYSTEMS MDOS	4.1
4.1	Systemprädisposition	4.1
4.1.1	Festlegung bestimmter Hardwarekonfigurationen	4.1
4.1.2	Festlegung von bestimmten Standardinhalten	4.2
4.2	Betriebsarten des Systems	4.3
4.2.1	Command-Mode	4.4
4.2.2	Running-Mode	4.4
4.2.3	Debugging-Mode	4.7
4.2.4	Calculator-Mode	4.7
4.3	Zustandsdiagramm	4.7
5.	BIBLIOTHEKEN UND FILES	5.1
5.1	Bibliotheken	5.1
5.5.1	Bibliotheken auf Disketten	5.1
5.1.2	Bibliotheken auf Festplatten	5.1
5.1.3	Anlegen von Bibliotheken	5.2
5.1.4	Verwaltung der Bibliotheken	5.4
5.1.5	Arbeiten mit Bibliotheken	5.5
5.1.6	Zugriff auf Bibliotheken und Files	5.6
5.2	Allgemeine Informationen über externe Files	5.7
5.2.1	Allgemeines	5.7
5.2.2	Filenamen	5.8
5.2.3	Erstellen von Files	5.8
5.2.4	Öffnen von Datenfiles	5.11
5.2.5	Schließen eines Files	5.12
5.2.6	Schützen von Files	5.12

6.	SYSTEMBEFEHLE	6.1
6.1	Eingabe von Befehlen über die Tastatur	6.2
6.2	Ausführung von Befehlen mit Prozeduren	6.3
6.2.1	Erstellen einer Prozedur	6.3
6.2.1.1	Eingabe von Systembefehlen und Dienstprogrammen	6.3
6.2.1.2	Eingabe von BASIC-Statements	6.3
6.2.1.3	Spezielle Befehle für Prozeduren	6.4
6.2.1.4	Prozedur als Datenfile	6.4
6.2.2	Aufruf einer Prozedur	6.4
6.3	Liste und Funktionen der Systembefehle	6.5
	AUTO	6.9
	CALCULATOR	6.11
	CATALOG	6.13
	COMPILE	6.17
	CONFIGURE	6.19
	CREATE	6.23
	DATE	6.25
	DCHANGE	6.27
	DECOMPILE	6.29
	DELETE LINE	6.31
	DRAW	6.33
	ENVIRONMENT	6.35
	ERASE	6.37
	EXEC	6.39
	FETCH	6.41
	LBCLOSE	6.43
	LBOPEN	6.45
	LBRESTORE	6.47
	LBSTORE	6.49
	LDIMAGE	6.51
	LDKEYS	6.53
	LINK	6.55
	LIST	6.59
	LVTOC	6.63
	MERGE	6.65
	MODIFY	6.69
	NEW	6.71
	OLD	6.73
	OPTIONS	6.75

	PREPARE	6.77
	PROCEDURE	6.79
	PURGE	6.85
	REPLACE	6.87
	RESEQUENCE	6.89
	REVERSE	6.91
	RUN	6.93
	SAVE	6.97
	SECURE	6.101
	SHIFT	6.103
	SPACE	6.105
	STIMAGE	6.107
	STKEYS	6.109
	TEXT	6.111
	TRANSCODE	6.113
	TRUNCATE	6.117
	VALIDATE	6.119
6.4	Zusätzliche Befehle in Prozeduren	6.121
6.4.1	Automatischer Input	6.121
6.4.2	Bedingte Sprünge	6.121
6.4.3	Fehlermeldungen, Befehle TEST, PRO	6.122
6.5	Befehle im Calculator- und Debugging-Mode	6.123
6.5.1	Liste der Befehle	6.123
6.5.2	Ausführliche Beschreibung der Befehle	6.123
	FKEY	6.125
	SDEG	6.127
	SGRAD	6.127
	SRAD	6.127
	START	6.129
	STOP	6.131
7.	DIENSTPROGRAMME	7.1
7.1	Allgemeine Hinweise	7.1
7.2	Liste der Dienstprogramme	7.1
	DCOPY	7.3
	DINIT	7.5
	FLCOPY	7.7
	FLPRINT	7.11
	LBCREATE	7.13
	LBEMPTY	7.17
	LBPROTECT	7.19
	LBRENAME	7.21
	LBSCRATCH	7.23
	LIBCOPY	7.25
	RESTRUCT	7.29
	VOLLAB	7.31

8.	PROGRAMMIERUNG IN BASIC	8.1
8.1	Einführung in die Sprache BASIC	8.1
8.1.1	Struktur eines BASIC-Programmes	8.1
8.1.2	Funktionelle Zusammenhänge von BASIC-Anweisungen	8.3
8.1.3	BASIC-Zeichen	8.8
8.1.4	Zahlendarstellung	8.9
8.1.5	Konstanten und Variable	8.11
8.1.5.1	Konstanten	8.11
8.1.5.2	Numerische und Stringvariable	8.12
8.1.5.3	Felder (indizierte Variable)	8.13
8.1.6	Ausdrücke und Vergleichsoperatoren	8.16
8.1.6.1	Ausdrücke	8.16
8.1.6.2	Regeln für arithmetische Operatoren	8.16
8.1.7	Vergleichs- und Boole'sche Operatoren	8.18
8.1.7.1	Vergleichsoperatoren	8.18
8.1.7.2	Boole'sche Operatoren	8.18
8.2	Speicheraufteilung	8.19
8.2.1	Gliederung des Anwenderspeichers	8.19
8.2.2	Common-Bereich	8.21
8.3	Ablauforganisation 8.23	
8.4	Verbindung mit Assembler P6066	8.26
8.4.1	Assembler-Sprache	8.26
8.4.2	Verarbeitung durch ein BASIC-Programm	8.27
8.5	Standardformat	8.27
8.5.1	Zahlendarstellung	8.28
8.5.2	Darstellung von Strings	8.29
8.5.3	Stellenkontrolle bei den Anweisungen DISP und PRINT	8.30
8.5.4	Besonderheiten der Anweisung DISP	8.32

9. BASIC-ANWEISUNGEN

9.1 Alphabetisch geordnete Beschreibung der Befehle 9.1

APPEND: 9.1
ASSIGN 9.3
BASSIGN 9.5
BBUILD 9.7
BEEP 9.9
BPAD 9.11
BUILD 9.13
BUILD USING 9.15
CALL 9.17
CHAIN 9.19
COMMON 9.21
CONVERT 9.25
DATA 9.27
DCL 9.29
DEF 9.31
DEF/FNEND 9.35
DELAY 9.39
DEPAD 9.41
DIM 9.43
DISP 9.45
DISP USING 9.51
END 9.55
ERASE 9.57
FILES 9.59
FILE: 9.61
FKEY 9.63
FNEND 9.65
FOR 9.67
GOSUB 9.71
GOTO 9.73
IF...THEN 9.75
IMAGE statement 9.77
INPUT 9.83
INTERRUPT ENABLE 9.87
LET 9.95
NEXT 9.97
ON...GOSUB 9.99
ON...GOTO 9.101
PAD 9.103
PRINT 9.105
PRINT USING 9.109
RANDOMIZE 9.113
READ 9.115
READ: 9.117
REM 9.119
RESTORE 9.121
RESTORE: 9.123
RETURN 9.125

	REVERSE	9.127
	RKB	9.129
	SCRATCH:	9.131
	SETW:	9.133
	STOP	9.135
	TRACE OFF	9.137
	TRACE ON	9.139
	WHERE:	9.141
	WRITE:	9.145
9.2	Standardfunktionen	9.147
9.2.1	Trigonometrische Funktionen	9.148
9.2.2	Mathematische Standardfunktionen	9.139
9.2.3	Numerische Funktionen ohne Argument	9.151
	PI	9.151
	RND	9.151
	DET	9.153
9.2.4	Spezielle numerische Funktionen	9.155
	IOC	9.157
	LEN	9.159
	SCN	9.161
	TAB	9.163
9.2.5	Alphanumerische Funktionen	9.165
	CHR$	9.167
	EXT$	9.169
	BLN$	9.171
	REP$	9.173
9.3	Anweisungen für Matrizenoperationen	9.175
	MAT...=	9.177
	MAT..+-	9.179
	MAT...Skalar*	9.181
	MAT...*	9.183
	MAT...CON	9.185
	MAT...IDN	9.187
	MAT INPUT	9.189
	MAT...INV	9.191
	MAT PRINT	9.193
	MAT PRINT USING	9.195
	MAT READ	9.197
	MAT READ:	9.199
	MAT...TRN	9.201
	MAT WRITE:	9.203
	MAT...ZER	9.205
9.4	I/O-Kanal TASTATUR	9.207
9.4.1	BASIC-Anweisungen für den Kanal TASTATUR	9.207
9.4.2	Bemerkungen	9.212

9.4.3	BASIC-Anweisungen im Detail	9.213
	BUFFER	9.215
	CMD	9.217
	RECEIVE	9.219
	SEND	9.221
	TEST	9.223
	WAIT	9.225
9.4.4	Beispiele	9.227
9.5	Graphisches Arbeiten durch OPTION 'Plot' und 'Graphik-Display' (OPT PLO oder OPT GDI)	9.229
9.5.1	Grundlagen des Plottens	9.229
9.5.2	Plotten über den Thermodrucker PR 2400 mit OPT PLO (ohne Darstellung auf Bildschirm)	9.233
9.5.2.1	Darstellung von Punkten, Linien und Zeichen	9.233
9.5.2.2	Ausführung von Plotprogrammen	9.233
9.5.3	Graphische Darstellung am Bildschirm mit OPT GDI	9.234
9.5.4	Plotten mit einem externen Plotter	9.236
9.5.5	BASIC-Anweisungen der Module OPTION GDI/PLO	9.239
	CPLOT	9.241
	CSIZE	9.243
	CTAB	9.245
	DISP	9.247
	DOT	9.249
	DRAW	9.251
	ERASE	9.253
	EXTERNAL PLOTTER	9.255
	FRAME	9.257
	IDOT	9.259
	INIMAGE	9.261
	IPLOT	9.265
	LDIMAGE	9.267
	MOVE	9.269
	OFFSET	9.271
	PLOT	9.273
	POINTER	9.275
	REVERSE	9.277
	SCALE	9.279
	STIMAGE	9.281
	XAXIS	9.283
	YAXIS	9.285
9.5.6	Systembefehle für den Bildschirm	9.287
	LDIMAGE	9.289
	SITMAGE	9.291
9.5.7	Beispiele	9.293

10.	EINGABE UND EDITING EINES PROGRAMM- ODER TEXTFILES	10.1
10.1	Struktur eines Programmes	10.1
10.2	Struktur eines Textfiles	10.2
10.3	Eingabe eines Programmes	10.4
10.3.1	Vorbereitung des Hauptspeichers	10.4
10.3.2	Zeilennumerierung	10.4
10.3.3	Programmeingabe und Syntaxkontrolle	10.4
10.4	Eingabe eines Textfiles	10.5
10.5	Speichern von Programmen oder Textfiles	10.5
10.6	Korrektur (Editing) eines Programmes oder Textes	10.5
10.6.1	Hilfstasten und Befehle für das Editing	10.5
10.6.2	Editing am Beispiel eines Programmes	10.6
10.6.3	Editing im Display	10.9
10.7	Zusammenfassung der Möglichkeiten beim Erstellen und Editing von Programmen und Texten	10.10
11.	CALCULATOR- UND DEBUGGING-MODE	11.1
11.1	Calculator-Mode	11.1
11.2	Einsetzen des Calculator-Modes	11.1
11.3	Festlegen der Art des Argumentes trigonometrischer Funktionen	11.2
11.4	Befehl zur Belegung der Funktionstasten	11.3
11.5	Lokale Variable und numerische Ausdrücke	11.4
11.5.1	Zahlendarstellung	11.5
11.5.2	Standardfunktionen	11.5
11.6	Verarbeitung der Anweisungen	11.5
11.7	Ausgabe und Ausgabeformat der Ergebnisse	11.6
11.8	Debugging-Mode	11.7
11.8.1	Erreichen und Verlassen des Debugging-Modes	11.7
11.8.2	Operationen im Debugging-Mode	11.8
	Behebbare Fehler	11.8
	Abfrage von Variablenwerten	11.10
	Wertzuweisung an Variable	11.11
11.8.3	Rechnen im Debugging-Mode	11.12
11.8.4	START-, STOP-Befehle im Debugging-Mode	11.13

	EINGABE UND EDITING EINES PROGRAMM- ODER TEXTFILES	10.1
10.1	Struktur eines Programmes	10.1
10.2	Struktur eines Textfiles	10.2
10.3	Eingabe eines Programmes	10.4
10.3.1	Vorbereitung des Hauptspeichers	10.4
10.3.2	Zeilennumerierung	10.4
10.3.3	Programmeingabe und Syntaxkontrolle	10.4
10.4	Eingabe eines Textfiles	10.5
10.5	Speichern von Programmen oder Textfiles	10.5
10.6	Korrektur (Editing) eines Programmes oder Textes	10.5
10.6.1	Hilfstasten und Befehle für das Editing	10.5
10.6.2	Editing am Beispiel eines Programmes	10.6
10.6.3	Editing im Display	10.9
10.7	Zusammenfassung der Möglichkeiten beim Erstellen und Editing von Programmen und Texten	10.10
11.	CALCULATOR- UND DEBUGGING-MODE	11.1
11.1	Calculator-Mode	11.1
11.2	Einsetzen des Calculator-Modes	11.1
11.3	Festlegen der Art des Argumentes trigonometrischer Funktionen	11.2
11.4	Befehl zur Belegung der Funktionstasten	11.3
11.5	Lokale Variable und numerische Ausdrücke	11.4
11.5.1	Zahlendarstellung	11.5
11.5.2	Standardfunktionen	11.5
11.6	Verarbeitung der Anweisungen	11.5
11.7	Ausgabe und Ausgabeformat der Ergebnisse	11.6
11.8	Debugging-Mode	11.7
11.8.1	Erreichen und Verlassen des Debugging-Modes	11.7
11.8.2	Operationen im Debugging-Mode	11.8
	Behebbare Fehler	11.8
	Abfrage von Variablenwerten	11.10
	Wertzuweisung an Variable	11.11
11.8.3	Rechnen im Debugging-Mode	11.12
11.8.4	START-, STOP-Befehle im Debugging-Mode	11.13

A N H A N G

Meldungen des Betriebssystems A.1

Bedienungshinweise A.1

Informationsmeldungen A.2

Fehlermeldungen A.3

Fehlercodes A.5

Liste der Fehlercodes A.7

1. Behebbare Fehler während der Ausführung von BASIC-Programmen A.7

2. Fehler während der Preexecution eines Programmes A.8

3. Nicht behebbare Fehler während der Ausführung eines BASIC-Programmes A.9

4. Fehlermeldungen des BASIC-Compilers A.10

5. Fehlermeldungen im Zusammenhang mit den Diskstationen A.11

6. Nicht behebbare Fehler im Zusammenhang mit peripheren Einheiten A.12

7. Fehler der Eingabe oder Ausführung von Systembefehlen A.13

8. Fehler beim Aufruf oder Ausführung von Dienstprogrammen A.16

9. Sich auf PLOT-OPERATIONEN beziehende Fehlermeldungen A.16

10. Fehlermeldungen für die Interrupt-Behandlung A.17

11. Fehlermeldungen für einen nicht normalen Systemzustand A.18

ISO-Code-Tabelle A.19

Liste der darstellbaren Zeichen und ihre entsprechenden dezimalen Werte A.21

Darstellung der ISO-Zeichen der Spalten 0 und 1 der ISO-Code-Tabelle A.21

4.	ARBEITSWEISE DES BETRIEBSSYSTEMS MDOS	4.1
4.1	Systemprädisposition	4.1
4.1.1	Festlegung bestimmter Hardwarekonfigurationen	4.1
4.1.2	Festlegung von bestimmten Standardinhalten	4.2
4.2	Betriebsarten des Systems	4.3
4.2.1	Command-Mode	4.4
4.2.2	Running-Mode	4.4
4.2.3	Debugging-Mode	4.7
4.2.4	Calculator-Mode	4.7
4.3	Zustandsdiagramm	4.7

4. ARBEITSWEISE DES BETRIEBSSYSTEMS

4.1 Systemprädisposition

Der Anwender kann jederzeit bestimmte Parameter des Betriebssystems ändern. Diese bleiben solange erhalten (sie werden automatisch auf der Systemdisk gespeichert) bis sie vom Anwender selbst geändert werden. Dies betrifft zum einen die Festlegung bestimmter Hardwarekomponenten und die softwaremäßige Festlegung bestimmter Standardinhalte des Systems.

4.1.1 Festlegung bestimmter Hardwarekonfigurationen

Diese Festlegung wird durch Angabe von Parametern der Systembefehle CON und OPT durchgeführt.

Bestimmung der Übertragungsparameter des angeschlossenen Druckers

Das Betriebssystem hat gewisse Parameter für die Übertragung von Ausgabedaten an den Drucker gespeichert, wie z.B. Übertragungsgeschwindigkeit, ParityCheck und Anzahl Stopbits. Diese Standardwerte (auch Defaultwerte genannt) sind auf den Thermodrucker PR 2400 ausgerichtet. Sie werden im Bedarfsfall über den Systembefehl CONFIGURE, siehe Kapitel 6, Seite 6-15, eingegeben.

Als Faustregel gilt, soweit in der jeweiligen Programmbeschreibung nichts anderes ausgesagt wird:

Vor dem ersten Arbeiten mit Programmen, ist je nach angeschlossenem Drucker folgendes über Tastatur einzugeben:

Drucker	Befehl
PR 2400	CON EOL
PR 1450 PR 1472	CON SP=6F7YE2 EOL

Danach arbeitet der angeschlossene Drucker ordnungsgemäß.

Systemmodul für graphisches Arbeiten

Um auf dem System graphische Anwendungen fahren zu können, müssen über den Systembefehl OPT (OPTION) optionale Teile des Betriebssystems geladen werden, die die Graphik steuern. Bei Olivetti-Standardsoftware sind diese Module grundsätzlich schon geladen.

Für das graphische Arbeiten gibt es zwei Möglichkeiten:

OPTION PLO	Der angeschlossene Thermodrucker wird als Plotter benutzt. Die Graphik kann länger als DIN A4 sein.
OPTION GDI (Graphik-display)	der Bildschirm arbeitet als graphischer Schirm. Es kann eine Hard-Copy auf dem Thermodrucker gemacht werden.

Näheres siehe unter Systembefehl OPT, Kapitel 6.

4.1.2 Festlegung von bestimmten Standardinhalten

Vom Anwender können softwareseitig z.B. das Datum oder die Funktionstasten belegt werden. Diese Daten werden auf der Systemdisk gespeichert und bleiben solange erhalten bis sie vom Programm oder dem Anwender selbst wieder geändert werden.

Datierung

Mit dem Systembefehl DATE kann das aktuelle Datum eingegeben werden. Dieses Datum wird dann bei der Erstellung von Files und Bibliotheken gelesen und mit abgespeichert.

Standardbelegung von Funktionstasten

Die Funktionstasten können in verschiedenen Betriebszuständen (z.B. direkt vom Anwender oder vom Programm) mit einem Inhalt belegt werden. Nach dem Ausschalten des Systems ist dieser Inhalt gelöscht. Soll der Inhalt jedoch erhalten bleiben, so kann er mit dem Systembefehl STKEYS auf der Systemdisk gespeichert werden. Bei jedem Neuladen des Systems oder bei Ausführung des Befehles LDKEYS wird dann dieser Inhalt den Funktionstasten erneut zugewiesen.

Festlegung von Standardbibliotheken

Für das Arbeiten mit Files muß die Bibliothek, in der sie enthalten sind, geöffnet sein (siehe Kapitel 5). Damit nun beim Einschalten des Systems oder beim Neuladen des Betriebssystems die benötigten Bibliotheken nicht jedesmal neu geöffnet werden müssen, besteht die Möglichkeit, diese Bibliotheken vom System automatisch öffnen zu lassen.

Soll eine bestimmte Folge von Bibliotheken immer automatisch geöffnet werden, so sind diese einmal zu öffnen (es sei denn, sie sind schon offen) und die so eingegebenen Informationen durch Ausführung des Befehles LBSTORE auf der Systemdisk zu speichern. Die so definierten Bibliotheken können - außer beim Neuladen des Systems - auch durch Ausführung des Befehles LBRESTORE geöffnet werden.

Die Liste der automatisch zu öffnenden Bibliotheken kann durch Ausführung der Systembefehle ENVIRONMENT oder LVT* erhalten werden.

Automatischer Start einer Prozedur

Das System kann so festgelegt werden, daß nach dem Laden automatisch mit der Verarbeitung einer Prozedur begonnen wird (siehe Abschnitt 6.2). Dazu muß in einer automatisch geöffneten Bibliothek eine Prozedur namens *SETUP existieren und der Systembefehl CONFIGURE mit dem Parameter PRO angegeben werden.

4.2 Betriebsarten des Systems

Das Computersystem M40 ST arbeitet auf verschiedenen Systemebenen, die im folgenden Betriebsarten genannt werden.

Dieser Abschnitt zeigt diese unterschiedlichen Zustände in ihrer Bedeutung und ihren Möglichkeiten auf.

Das System M40 ST kennt folgende Betriebsarten:

- COMMAND-MODE
- RUNNING-MODE
- DEBUGGING-MODE
- CALCULATOR-MODE

Der Zusammenhang der einzelnen Betriebsarten wird am Ende dieses Kapitels in einer Graphik verdeutlicht.

4.2.1 Command-Mode

Der Command-Mode erlaubt:

- die Eingabe und Ausführung von Systembefehlen (Kapitel 6),
- Den Aufruf von Dienstprogrammen (Abschnitt 6.4 und Kapitel 7),
- die Eingabe von Programmen und Texten sowie Editing-Operationen,
- den Übergang in den Calculator-Mode (Kapitel 8).

Das System ist im Command-Mode:

- nach dem Laden
- nach der Ausführung eines Anwenderprogrammes,
- nach der Eingabe von Systembefehlen im Calculator-Mode oder nach seiner expliziten Aufhebung durch Drücken der Konsoltaste,
- nach dem Drücken der Tasten CTRL EXIT (BREAK) im Running- oder Debugging-Mode.

Im Command-Mode leuchtet die Signallampe READY konstant. Nach dem Laden des Systems bzw. nach einer Rückkehr in den Command-Mode aus dem Running- oder Debugging-Mode erscheint im Schirm zusätzlich die Meldung "READY" (falls im Befehl SAVE des laufenden Programms der Parameter MSG=0 nicht angegeben wurde).

4.2.2 Running-Mode

Führt das System Dienst- oder Anwenderprogramme aus, so befindet es sich im Running-Mode. Bei der Ausführung von Dienstprogrammen richten sich die Eingriffsmöglichkeiten nach der Art und den Erfordernissen des jeweils aufgerufenen Dienstprogrammes. Die folgende Beschreibung des Running-Modes befaßt sich nur mit der Ausführung von Anwenderprogrammen.

Der Aufruf zur Ausführung eines Programmes erfolgt mit einem zulässigen Format des Befehles RUN.

Wird die Ausführung eines Programmes mit RUN gestartet, so wird, falls erforderlich, eine globale Syntaxprüfung durchgeführt und mit der Verarbeitung des Programmes begonnen. Die Funktionen des Befehles RUN sind in Kapitel 6 detailliert beschrieben.

Bei der Ausführung von Anwenderprogrammen können zwischenzeitlich andere Betriebszustände erreicht oder die Programmausführung durch ein reguläres oder durch Fehler verursachtes Programmende abgebrochen werden.

Möglichkeiten des Verlassens der Programmausführung:

Rückkehr in den Command-Mode:

- Wird die Ausführung eines Programmes durch Erreichen der END-Anweisung regulär beendet, so erfolgt die Rückkehr in den Command-Mode.
- Nach dem Drücken der Tasten CTRL EXIT (BREAK) wird die Verarbeitung des laufenden Programms nach regulärer Beendigung des sich in Verarbeitung befindlichen Statements abgebrochen, eventuell geöffnete Files geschlossen und das Programm für Editing-Operationen wieder zur Verfügung gestellt.

Übergang in den Debugging-Mode

Der Übergang aus dem Running-Mode in den Debugging-Mode erfolgt:

- durch Drücken der Taste EXIT (STEP),
- durch einen STOP-Befehl im Programm,
- durch Ausführung eines im Debugging-Mode eingegebenen STOP-Befehles,
- durch einen behebbaren Fehler
- nach nicht behebbaren Fehlern ohne Möglichkeit der Programmfortsetzung.

Wird während der Ausführung eines Programmes die Taste EXIT (STEP) gedrückt, so wird die Programmverarbeitung am Ende der laufenden Operation unterbrochen. Danach stehen alle Möglichkeiten des Debugging-Modes zur Verfügung (siehe Kapitel 8). Tritt bei der Verarbeitung eines Programmes ein Fehler auf, so meldet das System den Fehlercode und bleibt im Debugging-Mode stehen.

Mit den im Debugging-Mode möglichen Operationen können die Ergebnisse des den Fehler verursachenden Statements erzeugt, die Werte aller im Programm aufgeführten Variablen abgefragt, Rechenoperationen ausgeführt und die Verarbeitung mit dem nächsten oder einem beliebigen anderen Befehl fortgesetzt werden. Die Rückkehr in den Running-Mode erfolgt entweder mit den Tasten SHIFT EXIT (CONTINUE) oder mit dem Befehl START. Die Anweisung "START Zeilennummer" wird vom System nicht akzeptiert, wenn der Programmstart mit "RUN program name" und die Abspeicherung des Programms ohne vorherige Preexecution (d.h. vollständige Syntaxprüfung) erfolgte (siehe Beschreibung der Befehle RUN und PREPARE in Kapitel 6).

Durch das Drücken der Taste EXIT (STEP) wird jeweils eine BASIC-Anweisung ausgeführt und dann in den Debugging-Mode zurückgekehrt.

Inputanforderung

Tritt bei der Programmausführung eine der Anweisungen RKB, INPUT, MAT INPUT oder RECEIVE #n auf ("n" = Adresse des I/O-Kanales Tastatur), so wartet das System auf die Eingabe der Daten in richtiger Anzahl und korrektem Format.

- Nach Abschluß der Eingabe kehrt das System in den Running-Mode zurück.
- Werden die Tasten CTRL EXIT (BREAK) gedrückt, während das System auf eine Eingabe wartet, wird der Programmablauf abgebrochen.
- Wird die Taste EXIT (STEP) gedrückt, so geht das System in den Debugging-Mode. Nach Drücken der Tasten SHIFT EXIT (CONTINUE) oder erneutem Drücken der Taste EXIT (STEP), erscheint im Bildschirm ein Fragezeichen; die Eingabe kann nun erfolgen. Vor der Dateneingabe können zum Beispiel Zwischenrechnungen manuell durchgeführt werden.

Wartet das System auf eine Eingabe, so hört die Signal-Lampe READY mit Blinken auf und leuchtet konstant.

Erfolgt die Programmausführung innerhalb einer Prozedur mit automatischer Inputanforderung, so werden die Eingabedaten bei der Inputanforderung dem Commandfile entnommen und das Programm ohne Unterbrechung fortgesetzt.

Operator Call

Einzelne Dienstprogramme verlangen bestimmte Interventionen des Benutzers, wie etwa den Wechsel von Disks.

Die Aufforderung zum Wechseln einer Disk erfolgt im Bildschirm. Nach Durchführung des Disktausches wird die Verarbeitung des Dienstprogrammes durch Drücken der Taste EXIT (CONTINUE) fortgesetzt.

Das Drücken der Tasten CTRL EXIT (BREAK) wird nur akzeptiert, wenn eine arbeitsfähige Konstellation vorhanden ist.

Behebbare Fehler

Tritt während eines Programmlaufes ein behebbarer Fehler auf (z.B. fehlende Wertzuweisung), so geht das System unter Meldung des Fehlercodes in den Debugging-Mode. Eine Abfrage von Variablen und gegebenenfalls eine Wertzuweisung sind möglich. Der Programmablauf kann durch Drücken der Taste EXIT (CONTINUE) fortgesetzt werden.

Nicht behebbare Fehler in der Ausführungsphase

(Fehlercodes 65 - 97, 162 - 170)

Tritt während eines Programmlaufes ein nicht behebbarer Fehler auf (z.B. RETURN ohne GOSUB), so geht das System unter Meldung des Fehlercodes in einen Quasi-Debugging-Mode, das heißt, eine Abfrage von Variablen ist möglich, der Programmlauf kann aber nicht fortgesetzt werden. Die Tasten CTRL EXIT (BREAK) müssen gedrückt werden, wonach das System in den Command-Mode übergeht.

4.2.3 Debugging-Mode

Das System befindet sich im Debugging-Mode:

- nach Drücken der Taste EXIT (STEP),
- nach Ausführung der Anweisung PREPARE,
- bei einer STOP-Anweisung im Programmablauf,
- nach Meldung eines behebbaren Fehlers,
- nach Auftreten eines nicht behebbaren Fehlers.

Der Debugging-Mode erlaubt das Austesten eines Programmes und die Behebung von Fehlern. Die Arbeitsweise wird in Kapitel 8 beschrieben.

4.2.4 Calculator-Mode

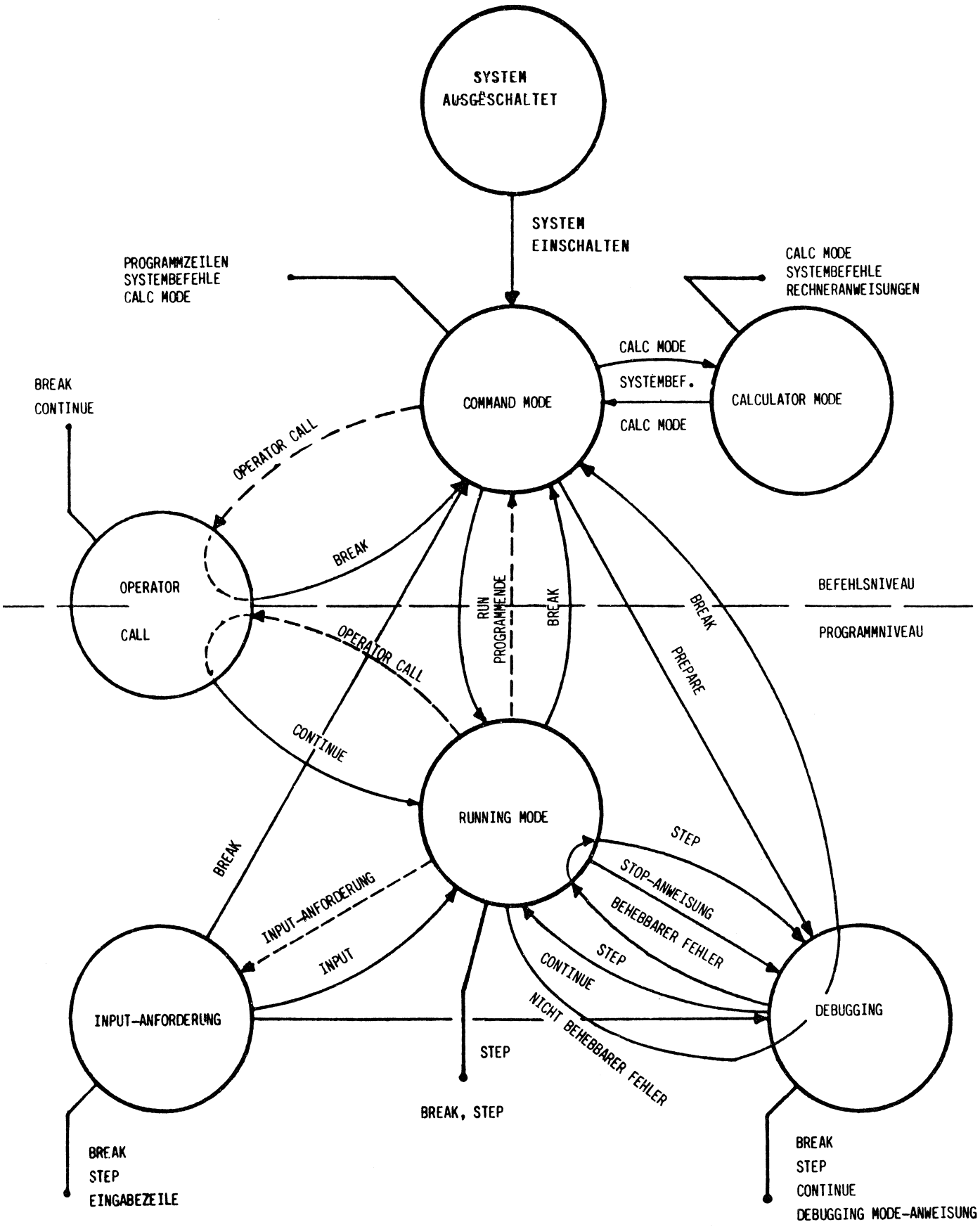
Durch Aktivierung des Calculator-Modes mit Eingabe von CAL, ist das System als Tischrechner zu verwenden. Der Debugging-Mode enthält auch den Calculator-Mode. Die Beschreibung der Möglichkeiten im Calculator-Mode erfolgt in Kapitel 8.

4.3 Zustandsdiagramm

Zum besseren Verständnis des folgenden Zustandsdiagrammes des Betriebssystems beachte man folgende Analogie:

Funktion	Tastendruck
BREAK	CONTROL u. EXIT
STEP	SHIFT u. EXIT
CONTINUE	EXIT

Zustandsdiagramm



5.	BIBLIOTHEKEN UND FILES	5.1
5.1	Bibliotheken	5.1
5.5.1	Bibliotheken auf Disketten	5.1
5.1.2	Bibliotheken auf Festplatten	5.1
5.1.3	Anlegen von Bibliotheken	5.2
5.1.4	Verwaltung der Bibliotheken	5.4
5.1.5	Arbeiten mit Bibliotheken	5.5
5.1.6	Zugriff auf Bibliotheken und Files	5.6
5.2	Allgemeine Informationen über externe Files	5.7
5.2.1	Allgemeines	5.7
5.2.2	Filenamen	5.8
5.2.3	Erstellen von Files	5.8
5.2.4	Öffnen von Datenfiles	5.11
5.2.5	Schließen eines Files	5.12
5.2.6	Schützen von Files	5.12

5. Bibliotheken und Files

Für die permanente Speicherung von Programmen und Datenbeständen stehen bezüglich Charakteristik und Kapazität unterschiedliche Speichermedien zur Verfügung, nämlich:

- zwei integrierte Diskettenstationen oder
- eine integrierte Diskettenstation und eine integrierte Festplatteneinheit.

Um ein zuverlässiges und rasches Wiederauffinden gespeicherter Datenbestände zu erreichen, ist eine logische Organisation dieser Speichermedien erforderlich. Diese Organisation erfolgt in mehreren Stufen:

- Die Datenbestände werden mit einem Namen versehen (Filename).
- Für die Filenamen werden Inhaltsverzeichnisse angelegt (Teilbibliotheken).
- Die Files werden auf spezifizierten Speichermedien an bestimmten Stellen der Bibliotheken gespeichert, in denen auch die Inhaltsverzeichnisse verwaltet werden.

5.1 Bibliotheken

Eine Bibliothek ist ein reservierter, mit einem Namen versehener Speicherbereich auf einem Speichermedium.

5.1.1 Bibliotheken auf Disketten

Eine Standard-Diskette mit 256 KB kann nur eine einzige, dem Anwender zugängliche Bibliothek enthalten. Eine Megafloppy kann zwischen einer und maximal 14 Bibliotheken umfassen.

5.1.2 Bibliotheken auf Festplatte

Die integrierte Festplatte kann in mehrere "logische Platten", sogenannte Daten-Sets eingeteilt werden. Jedes Daten-Set kann mehrere Bibliotheken umfassen.

Die Summe der Bibliotheksgrößen ist durch die Kapazität des Daten-Sets beschränkt. Die Summe der Größen der Daten-Sets ist durch die Kapazität der Festplatte beschränkt.

5.1.3 Anlegen von Bibliotheken

Bibliotheken dienen zur Verwaltung von Files. Allen Files einer Biblothek steht ein gemeinsamer Speicherbereich zur Verfügung. Die Filenamen aller Files einer Bibliothek werden in bis zu drei getrennten Inhaltsverzeichnissen verwaltet.

Die Inhaltsverzeichnisse einer Bibliothek werden "Teilbibliotheken" genannt. Sie unterscheiden sich bezüglich des für sie möglichen Schutzes. Folgende Arten von Teilbibliotheken sind vorhanden:

- Package-Teilbibliothek
- Common-Teilbibliothek,
- User-Teilbibliothek.

Der Schutz einer Teilbibliothek wird durch die Ausführung des Dienstprogrammes LBPROTECT erreicht und bezieht sich immer auf die Gesamtheit der in ihr enthaltenen Files. Der Schutz einzelner Files vor Editing kann über den Systembefehl SECURE erreicht werden.

Für das Anlegen einer Bibliothek sind folgende Angaben erforderlich:

1. Vergabe eines Bibliotheknamens,
2. Festlegung der Einheit, die die Bibliothek aufnehmen soll,
3. Eventuell Vergabe eines Passwords,
4. Festlegung der Größe der Bibliothek,
5. Festlegung der Teilbibliotheken dieser Bibliothek und der Anzahl der in jede Teilbibliothek aufnehmbaren Filenamen.

zu 1. Vergabe eines Bibliotheksnamens

Ein Bibliotheksname kann aus ein bis sechs Großbuchstaben oder Ziffern bestehen. Das erste Zeichen muß ein Buchstabe sein. Bibliotheken, die auf derselben Einheit gespeichert sind, müssen verschiedene Namen tragen.

Beispiele für gültige Bibliotheksnamen:

- BIBL1
- GEO
- ARCHIV

zu 2. Festlegung der Einheit, auf welcher die Bibliothek angelegt wird.

Die Festlegung der für die Speicherung der Bibliothek vorgesehenen Einheit erfolgt durch Angabe des Namens "unit name" dieser Einheit.

Bibliotheksname und "unit name" bilden gemeinsam die "lib ref", die im weiteren die Identifikation der Bibliothek ermöglicht. Das vollständige Format von "lib ref" lautet:

(Bibliotheksname, unit name)

Sie kann auch andere Formen annehmen, deren Format und Gültigkeit in der Beschreibung der Systembefehle und Dienstprogramme jeweils angeführt ist.

zu 3. Vergabe eines Passwords

Bibliotheken können mit einem Password versehen werden. Der Zugriff auf Files einer so geschützten Bibliothek ist nur möglich, wenn das Password angegeben wird. Ohne Kenntnis ihres Passwords kann mit einer Bibliothek nicht gearbeitet werden.

Wurde bei der Installation des Betriebssystems in einer Plattenkonfiguration ein Systempassword vergeben, so ist hat dieses anstelle des Bibliothekspasswords ebenfalls Gültigkeit.

Als "password" ist eine beliebige Folge von maximal sechs Zeichen möglich. Wird kein Password vergeben, so ist der Zugriff auf die Files dieser Bibliothek ohne Angabe eines Passwords möglich.

zu 4. Festlegung der Größe der Bibliothek

Die gewünschte Größe der Bibliothek ist in K-Bytes anzugeben. Dabei ist zu berücksichtigen, daß der für Inhaltsverzeichnisse und Leerplatzverwaltung erforderliche Platzbedarf bei der Angabe der Bibliotheksgröße mit einbezogen werden muß. Wird für die Bibliothek keine Größe angegeben, so wird vom System der Default-Wert genommen. Vergl. dazu das Dienstprogramm LBC (Library Create) in Kapitel 7.

zu 5. Festlegung der Teilbibliotheken

Jede Bibliothek kann in bis zu drei Teilbibliotheken aufgeteilt werden. Jede Teilbibliothek besteht aus dem Inhaltsverzeichnis der in ihr enthaltenen Files.

Die Definition der Größe der Teilbibliothek erfolgt durch Festlegung der Anzahl für das Inhaltsverzeichnis zur Verfügung gestellten Sektoren.

Jeder Sektor kann die Informationen von 27 Files aufnehmen.

Erfolgt keine explizite Festlegung der Größe der Teilbibliotheken, so werden vom Betriebssystem Standardannahmen getroffen:

- Bei Bibliotheken auf Standard-Diskette (256 KB)

  für Package-Teilbibliothek 5 Sektoren (65 Files),
  für Common-Teilbibliothek 4 Sektoren (52 Files),
  Für User-Teilbibliothek 5 Sektoren (65 Files).

- Bei Bibliotheken auf Megafloppies und Festplatten:

  für Package-Teilbibliothek 3 Sektoren (81 Files),
  für Common-Teilbibliothek 2 Sektoren (54 Files),
  für User-Teilbibliothek 3 Sektoren (81 Files).

Die Kapazität einer Bibliothek hängt somit von zwei Vereinbarungen ab:

- Die Summe der einzelnen Filegrößen darf mit dem für die Verwaltung erforderlichen Platz die Gesamtkapazität nicht übersteigen.
- Die Anzahl der in einer Bibliothek speicherbaren Files ist begrenzt durch die Anzahl der für Teilbibliotheken vereinbarten Sektoren.

Das Anlegen einer Bibliothek erfolgt mit dem Dienstprogramm LBCREATE (siehe Kapitel 7).

5.1.4 Verwaltung der Bibliotheken

Bibliotheken können wie in 5.1.3 beschrieben, vereinbart werden. Inhaltsverzeichnisse, die Informationen über die vorhandenen Bibliotheken ausgeben, sind über den Systembefehl LVTOC (Library Volume Table of Contents) erhältlich.

Sollen Bibliotheken oder ihr Inhalt als Gesamtheit gelöscht werden, so stehen mehrere Dienstprogramme zur Verfügung:

LBEMPTY löscht alle in einer Bibliothek gespeicherten Files. Die Bibliothek bleibt jedoch bestehen.

LBSCRATCH löscht eine Bibliothek (und damit auch ihren Inhalt). Um den dadurch frei werdenden Platz wieder verwenden zu können, muß das Dienstprogramm RESTRUCT ausgeführt werden.

DINIT erlaubt das Löschen aller Bibliotheken einer Diskette oder Platte.

Soll der Inhalt einer Bibliothek oder Teilbibliothek in eine andere, bestehende Bibliothek kopiert werden, so ist das Dienstprogramm LIBCOPY auszuführen. Einzelne Files können mit FLCOPY von einer (Teil-)Bibliothek in eine andere (Teil-)Bibliothek kopiert werden.

Soll der Name oder das Password einer Bibliothek geändert werden, steht das Dienstprogramm LBRENAME zur Verfügung.

5.1.5 Arbeiten mit Bibliotheken

Im vorangehenden Abschnitt wurde auf die Möglichkeiten zur Verwaltung von Bibliotheken hingewiesen.

Im folgenden werden nun die zum Arbeiten mit Bibliotheken erforderlichen Bedienungselemente beschrieben.

Um auf in einer Bibliothek enthaltene Files zugreifen zu können, muß dem System bekannt sein, mit welchen Bibliotheken gearbeitet werden soll. Dieser Vorgang wird als "Öffnen einer Bibliothek" bezeichnet und mit dem Systembefehl LBOPEN durchgeführt. Maximal sechs Bibliotheken können gleichzeitig geöffnet sein.

Mit dem Systembefehl LBSTORE wird erreicht, daß bestimmte Bibliotheken bei jedem Laden des Betriebssystems automatisch geöffnet werden. Sie werden "Standardbibliotheken" genannt. Sind keine Bibliotheken als Standardbibliotheken festgelegt, so wird die erste Bibliothek der Systemdisk automatisch geöffnet, sofern sie nicht durch ein Password geschützt ist.

Bemerkungen:

- Das Öffnen bewirkt keine Markierung in der entsprechenden Bibliothek. Im Arbeitsspeicher der Basiseinheit werden Informationen über die momentan verwendeten Bibliotheken gespeichert. Diese Informationen gehen daher beim Ausschalten der Maschine oder beim Neuladen des Betriebssystems verloren.

- Ist die Arbeit mit einer oder allen geöffneten Bibliotheken beendet, so können sie mit dem Befehl LBCLOSE geschlossen werden. Der Zugriff auf den Inhalt dieser Bibliothek(en) ist danach nicht mehr möglich. Sollen anstelle der momentan geöffneten Bibliotheken wieder die Standard-Bibliotheken (LBSTORE) geöffnet werden, steht dafür der Systembefehl LBRESTORE zur Verfügung.

- Für die Anwendung aller Systembefehle und der meisten Dienstprogramme, die sich auf Files oder den Inhalt einer Bibliothek beziehen, muß die Bibliothek geöffnet sein. Für die geöffneten Bibliotheken oder Teilbibliotheken kann mit CATALOG ein Inhaltsverzeichnis ausgegeben und mit SPACE der verfügbare Platz ermittelt werden.

5.1.6 Zugriff auf Bibliotheken und Files

Bibliotheken werden über "lib ref" identifiziert. Fehlt in "lib ref" die Angabe "unit name" und soll auf eine vorhandene, aber nicht geöffnete Bibliothek zugegriffen werden, so wird diese Bibliothek auf der Systemdisk gesucht. Ist in "lib ref" nur "unit name" angegeben, so wird die erste Anwenderbibliothek auf der Einheit dieses Namens angesprochen.

Bei offenen Bibliotheken wird der angegebene Bibliotheksname in der Reihenfolge des Öffnens der Bibliotheken gesucht. Dabei wird immer nach der ersten Übereinstimmung der in "lib ref" vorhandenen Angaben mit der Liste der geöffneten Bibliotheken (siehe LVT*) gesucht.

Bei sich auf Files beziehenden Systembefehlen kann "lib ref" entfallen.

Wird ein bestehendes File verlangt, so werden die Bibliotheken in der Reihenfolge ihres Öffnens durchsucht.

Wird ein neues File erstellt, ohne daß "lib ref" angegeben wird, so wird es in der ersten geöffneten Bibliothek angelegt.

Werden in Programmen Datenfiles verwendet, wo werden die Bibliotheken beim Programmaufruf in der Reihenfolge ihres Öffnens nach diesen Files durchsucht.

Bemerkungen:

- Enthalten verschiedene (offene) Bibliotheken Files mit gleichem Namen, so wird auf das als erstes gefundene File zugegriffen. Bei der Anwendung von Systembefehlen kann durch Angabe der entsprechenden "lib ref" auf jedes gewünschte File zugegriffen werden.

- Die Systembefehle PREPARE und RUN brauchen zur Ausführung Platz in der ersten offenen Bibliothek, wenn sie ohne Angabe eines Filenamens aufgerufen werden. Steht dieser Platz nicht zur Verfügung (ERROR 188), so kann gegebenenfalls durch eine andere Reihenfolge des Öffnens der Bibliotheken dieser Platz geschaffen werden.

5.2 Allgemeine Informationen über externe Files

5.2.1 Allgemeines

Ein File ist eine problemorientierte, organisatorische Zusammenfassung gleichartiger oder gleichbehandelter Daten in einem Speichmedium.

Es werden vier Filetypen unterschieden, und zwar:

1. Programmfiles
2. Textfiles
3. Datenfiles
4. Objectfiles

Die Files heißen "extern", da sie in einer Bibliothek auf einem externen Speicher enthalten sind.

Programm- und Textfiles

Programm- bzw. Textfiles erlauben es, Programme bzw. Texte unter einem Namen zu speichern.

- Ein Programmfile enthält eine mit Zeilennummern versehene Folge ausführbarer BASIC-Anweisungen.
- Ein Textfile besteht aus einer Folge numerierter Zeilen, die beliebige Folgen von ISO-Zeichen enthalten.

Ein Programmfile kann somit nur ein Programm, ein Textfile nur einen Text enthalten. Die Begriffe "Programm" und "Programmfile" (bzw. "Text" und "Textfile") haben also die gleiche Bedeutung.

Für die Behandlung von Programm- und Textfiles ist eine Reihe von Systembefehlen vorgesehen. Textfiles können außerdem in einem BASIC-Programm wie sequentielle Datenfiles gelesen werden.

Datenfiles

Datenfiles werden für die programmunabhängige, permanente Speicherung numerischer und alphanumerischer Daten verwendet.

Objectfiles

Objectfiles enthalten eine ausführbare Assemblerroutine, die von einem BASIC-Programm aufgerufen werden kann.

5.2.2 Filenamen

Jedes File in einer Bibliothek wird durch seinen Namen identifiziert. Über diesen Namen kann das File gesucht oder vor unerlaubtem Zugriff geschützt werden.

Der Filename setzt sich aus einem Kennzeichen der Teilbibliothek (wenn das File einer Package- oder Common-Teilbibliothek angehört) und dem eigentlichen Namen zusammen.

- Der eigentliche Name besteht aus maximal 6 Großbuchstaben oder Ziffern, wobei das erste Zeichen ein Buchstabe sein muß.
- Für das Kennzeichen der Teilbibliothek gilt:
  "*": File einer Package-Teilbibliothek
  "+": File einer Common-Teilbibliothek

Files, die zum Bestand einer User-Teilbibliothek gehören, haben kein Kennzeichen, d.h., ihre Namen beginnen mit einem Großbuchstaben.

	richtige Namen	falsche Namen	
Package-Teilbibliothek	*SINES	*	(kein Alphazeichen)
Common-Teilbibliothek	+G	+8G	(das 1. Zeichen ist kein Buchstabe)
User-Teilbibliothek	GRAPH2	GRAPH66	(mehr als 6 Zeichen)

5.2.3 Erstellen von Files

Erstellen von Programmen und Textfiles

Programme und Texte werden entweder unmittelbar im Arbeitsspeicher kreiert oder sie ergeben sich aus der Umwandlung eines Files mit anderem Typ.

Soll ein Programm über die Tastatur im Arbeitsspeicher erstellt werden, so ist vor der Eingabe der Befehl "NEW" einzugeben (alle in der Folge erwähnten Systembefehle sind in Kapitel 6 beschrieben.

Soll ein Textfile erstellt werden, so lautet der entsprechende Befehl "TEXT".

Wird nach der Erstellung eines Programmes oder Textes der Befehl "SAVE" gegeben, so wird das sich im Arbeitsspeicher befindliche Programm (bzw. der Text) unter seinem Namen in der ebenfalls namentlich angegebenen Teilbibliothek einer Bibliothek gespeichert.

Beispiel:

```
TEXT
AUTO#
10 DAS IST EIN TEXT
20 DER IN DER COMMON-TEILBIBLIOTHEK
30 DER OBEREN FLOPPY-DISK-STATION
40 GESPEICHERT WERDEN SOLL.
```

Nach der Ausführung des Befehles "SAVE +TEXT1,(,F1)" ist der Text in der Common-Teilbibliothek der Diskette der oberen Floppy-Disk-Station enthalten und kann über seinen Namen wieder angesprochen werden (z.B. mit OLD +TEXT1).

Erstellen von Datenfiles

Für die vollständige Vereinbarung eines Datenfiles sind folgende Angaben erforderlich:

1. Name (der auch die Teilbibliothek festlegt).
2. Angabe der für die Erstellung des Files vorgesehenen Bibliothek.
3. Zugriffsart
4. Maximaler Speicherbedarf

Ein Datenfile wird mit dem Befehl "CREATE" erstellt.

Zugriffsarten

Ein Datenfile kann sequentiellen oder random (direkten) Zugriff erlauben.

- Sequentielle Datenfiles:

  Die Daten sind im File in unmittelbarer Folge gespeichert. Das Schreiben von Daten in ein solches File erzeugt eine physische Reihenfolge, die nicht unterbrochen werden kann, und die auch die Reihenfolge des Lesens der Daten aus dem File festlegt.

  Sequentielle Datenfiles können nur auf zwei Arten beschrieben werden:

  - Ab Beginn des File. Der vorherige Inhalt wird dabei vollständig zerstört.

- Durch Anhängen der Daten an den bestehenden Inhalt, der dabei erhalten bleibt. Es ist nicht möglich, ein Datenelement direkt anzusprechen und gegen ein anderes auszutauschen.

- Randomfiles

  - Wird für ein Datenfile die Zugriffsart "random" festgelegt, so kann auf jedes Datenelement des Files direkt zugegriffen werden, das heißt, die Daten können an jede beliebige Stelle im File geschrieben bzw. ab jeder Position im File gelesen werden. Die Festlegung der Position des Datenelementes kann unabhängig von der Schreib- oder Leseanweisung in einer eigenen Anweisung erfolgen.

Platzbedarf

Die Daten eines Files werden in binärem Format gespeichert. Die kleinste Einheit, auf die in einem Datenfile zugegriffen werden kann, ist das aus 4 Bytes bestehende Wort.

Eine numerische Information belegt im Datenfile in einfacher Genauigkeit ein Wort, in doppelter Genauigkeit zwei Worte.

Für alphanumerische Daten (Strings) errechnet sich der Platzbedarf wie folgt:

- Die Anzahl der Worte ist gleich INT ((n-1)/4+2), wenn "n" die Anzahl der Zeichen im String angibt.

Um im Befehl CREATE die Anzahl der für ein Datenfile zu reservierenden Bytes festzulegen, muß entsprechend der Art und der maximalen Anzahl der im File aufzunehmenden Daten zunächst die Anzahl der Worte und daraus die erforderliche Anzahl Bytes errechnet werden.

Bemerkungen:

- Das Wort ist die kleinste Einheit, auf die in einem Randomfile zugegriffen werden kann.
- Ist die Anzahl der im CREATE-Befehl angegebenen Bytes kein Vielfaches von 128 bzw. 256, so wird diese Anzahl automatisch auf das nächstgrössere Vielfache von 128 bzw. 256 erhöht. Erfolgt keine Längenangabe, so werden 4096 Bytes reserviert.
- Wird das File in der Bibliothek einer Standard-Diskette (256 KB) angelegt, so ist seine Größe immer ein vielfaches von 128 Bytes. Wird es hingegen in einer Bibliothek einer Megafloppy oder Festplatte angelegt, so ist seine Größe immer ein Vielfaches von 256 Bytes.
- Bei Randomfiles ist die aktuelle Länge, unabhängig von der tatsächlichen Belegung mit Daten, immer gleich der maximalen Länge.

Beispiel:

In einer Bibliothek namens "ARCHIV" soll unter dem Namen "SFILE" ein File mit sequentiellem Zugriff erstellt werden. 1000 Strings mit je zehn Zeichen sollen darin Platz finden. Der erforderliche Platzbedarf in Bytes beträgt demzufolge:

1000*4*INT((10-1)/4+2)=4000*4=16000

Der CREATE-Befehl zur Erstellung dieses Files lautet:

CRE SFILE, ARCHIV,S,16000

5.2.4 Öffnen von Daten-Files

Bevor ein Programm auf ein externes Datenfile zugreifen kann, muß dieses durch Angabe seines Namens in einer FILES- oder FILE:-Anweisung geöffnet werden. Jedem angeführten File wird ein logischer Kommunikationskanal zwischen Hauptspeicher und Diskstation zugeordnet. Jeder dieser Kanäle trägt eine Nummer, die das ihm durch seinen Namen zugeordnete File im weiteren Programmablauf identifiziert. Die Zuordnung der Kanäle zu den Files wird durch die Position des Filenamens in der FILES-Anweisung bestimmt. Die daraus entstehende Ordnungszahl heißt "Filedesignator".

Nach dem Öffnen sind Ein- und Ausgabeoperationen mit diesem File möglich, doch muß zwischen Random- und sequentiellem File unterschieden werden.

Sequentielle Files

Nach dem Öffnen eines sequentiellen Files in einer FILES- oder FILE:-Anweisung kann das File ab dessen Beginn gelesen werden. Ist eine Aufzeichnung vorgesehen, so ist dies erst nach der Anweisung SCRATCH: möglich, falls ab Beginn des Files geschrieben werden soll, oder nach APPEND:, falls an ein bestehendes File Daten angefügt werden sollen.

Soll aus dem Schreib-Mode wieder in den Lese-Mode übergegangen bzw. mit dem Lesen erneut begonnen werden, so ist dies ab Filebeginn nach der Anweisung RESTORE: möglich.

Randomfiles

Neben allen von sequentiellen Files in einem Programm gebotenen Einsatzbereichen gibt es bei Files mit Randomzugriff folgende Möglichkeiten:

Nach dem Öffnen des Files in einer FILES- oder FILE:-Anweisung kann sowohl gelesen als auch geschrieben werden. Die Positionierung auf ein bestimmtes Wort im File erfolgt mit einer SETW:-Anweisung.

Es gibt zwei Arten von Randomfiles:

- R-File:
  Wird das File mit Parameter R vereinbart, so enthält es nach der Ausführung von CREATE numerische Daten in einfacher Genauigkeit, die den Wert "nicht initialisiert" haben.

- Z-File:
  Wird das File mit Parameter Z vereinbart, so enthält es nach der Ausführung von CREATE numerische Daten in einfacher Genauigkeit, die den Wert 0 aufweisen.

Textfile

Textfiles können in einem Programm vom Lese-Modus wie sequentielle Datenfiles behandelt werden. Nach dem Öffnen oder nach Ausführung der Anweisung RESTORE: können Textfiles mit READ: gelesen werden. Jede Zeile des Textfiles entspricht einem String, wobei die Zeilennummer Bestandteil dieses Strings ist.

5.2.5 Schließen eines Files

Werden in Programmen bestimmte Operationen mit Datenfiles ausgeführt, so werden diese Files in ihrer Bibliothek als geöffnet gekennzeichnet. Diese Markierung wird nach Beendigung der laufenden Operation oder bei einem regulären Programmende (END oder Funktion BREAK) gelöscht. Bei einem irregulären Programmende (z.B. bei Stromausfall bleibt die Markierung erhalten.

Bei der nächsten Ausführung eines auf ein offen gebliebenes Datenfile zugreifenden Programmes wird ERROR 76 gemeldet. Mit Hilfe des Systembefehls CATALOG können dann die offen gebliebenen Files ermittelt werden. Solche Datenfiles sind vor dem neuerlichen Programmstart mit dem Befehl VALIDATE zu schließen, wobei jedoch zu beachten ist, daß durch ein irreguläres Programmende Informationen verloren gehen können.

5.2.6 Schützen von Files

Das System M40 ST ermöglicht das Schützen von Programm-, Text- und Datenfiles vor bestimmten Operationen. Es ist dadurch zum Beispiel möglich, Programme zu erstellen, die zwar ausgeführt, aber nicht gedruckt oder modifiziert werden können. Der Schutz einzelner Files erfolgt mit dem Befehl SECURE.

6.	SYSTEMBEFEHLE	6.1
6.1	Eingabe von Befehlen über die Tastatur	6.2
6.2	Ausführung von Befehlen mit Prozeduren	6.3
6.2.1	Erstellen einer Prozedur	6.3
6.2.1.1	Eingabe von Systembefehlen und Dienstprogrammen	6.3
6.2.1.2	Eingabe von BASIC-Statements	6.3
6.2.1.3	Spezielle Befehle für Prozeduren	6.4
6.2.1.4	Prozedur als Datenfile	6.4
6.2.2	Aufruf einer Prozedur	6.4
6.3	Liste und Funktionen der Systembefehle	6.5
	AUTO	6.9
	CALCULATOR	6.11
	CATALOG	6.13
	COMPILE	6.17
	CONFIGURE	6.19
	CREATE	6.23
	DATE	6.25
	DCHANGE	6.27
	DECOMPILE	6.29
	DELETE LINE	6.31
	DRAW	6.33
	ENVIRONMENT	6.35
	ERASE	6.37
	EXEC	6.39
	FETCH	6.41
	LBCLOSE	6.43
	LBOPEN	6.45
	LBRESTORE	6.47
	LBSTORE	6.49
	LDIMAGE	6.51
	LDKEYS	6.53
	LINK	6.55
	LIST	6.59
	LVTOC	6.63
	MERGE	6.65
	MODIFY	6.69
	NEW	6.71
	OLD	6.73
	OPTIONS	6.75

	PREPARE	6.77
	PROCEDURE	6.79
	PURGE	6.85
	REPLACE	6.87
	RESEQUENCE	6.89
	REVERSE	6.91
	RUN	6.93
	SAVE	6.97
	SECURE	6.101
	SHIFT	6.103
	SPACE	6.105
	STIMAGE	6.107
	STKEYS	6.109
	TEXT	6.111
	TRANSCODE	6.113
	TRUNCATE	6.117
	VALIDATE	6.119
6.4	Zusätzliche Befehle in Prozeduren	6.121
6.4.1	Automatischer Input	6.121
6.4.2	Bedingte Sprünge	6.121
6.4.3	Fehlermeldungen, Befehle TEST, PRO	6.122
6.5	Befehle im Calculator- und Debugging-Mode	6.123
6.5.1	Liste der Befehle	6.123
6.5.2	Ausführliche Beschreibung der Befehle	6.123
	FKEY	6.125
	SDEG	6.127
	SGRAD	6.127
	SRAD	6.127
	START	6.129
	STOP	6.131

6. SYSTEMBEFEHLE

Das Computersystem M40 ST erlaubt die Lösung technischer und wissenschaftlicher Probleme mit Programmen in der problemorientierten Sprache BASIC. Auch die Kommunikation mit dem System erfordert eine Sprache. Sie besteht aus Systembefehlen, die das Erstellen und Ausführen eines BASIC-Programmes rasch und einfach gestalten. Mit den verfügbaren Befehlen kann man unter anderem:

- Programme erstellen
- Programme ausführen
- Programme modifizieren
- Programme speichern
- Daten- und Textfiles erstellen und abspeichern
- Bibliotheken verwalten

Die Systembefehle ermöglichen außerdem den Austausch von Informationen zwischen dem Hauptspeicher und den externen Einheiten des Systems, nämlich der Floppy-Disk- (FDU) oder einer Platten-Einheit sowie dem Drucker.

In diesem Kapitel wird der Aufbau und die Anwendung dieser Befehle besprochen. Vor der Erläuterung der einzelnen Systembefehle ist es aber notwendig, die allgemeinen Regeln und Elemente dieser Befehle kennenzulernen.

Die folgenden Symbole werden nur verwendet, um das Format eines Befehles zu definieren. Sie sind nicht Bestandteil des Befehles:

-	Bindestrich
_	Unterstreichung
{}	geschweifte Klammer
[]	eckige Klammer
...	Folge von Punkten

Diese Symbole bedeuten:

{}	schließt eine Folge von Parametern ein, von denen mindestens einer vorkommen muß.
[]	schließt eine Folge von optionalen Parametern ein. Diese Parameter können (müssen aber nicht) angegeben werden.
_	Stehen mehrere Parameter zur Auswahl und wird keiner davon angegeben, so setzt das System den unterstrichenen Parameter ein. Ein unterstrichener Parameter muß also nicht explizit eingegeben werden. Kommata, die am Ende von Eingaben trotz Weglassen von Parametern aufgrund der Formate erforderlich wären, entfallen.

... der vorherige Operand kann mehrmals wiederholt werden.

, trennt die Operanden eines Systembefehles voneinander.

Zu beachten ist, daß das Format eines Befehles, insbesondere die Stellung eines Parameters innerhalb desselben, genau eingehalten werden muß.

Beispiel

Richtig:	Falsch:
CAT ,,,F	CAT F

6.1 Eingabe von Befehlen über die Tastatur

Ein Befehl besteht aus einem Schlüsselwort, dem meistens ein oder mehrere Operanden folgen. Die Schlüsselwörter sind englische Begriffe, die die Funktion des Befehles beschreiben.

Die Operanden liefern die zusätzlichen Informationen zur Ausführung der Operation. Für die Schlüsselwörter der am häufigsten verwendeten Befehle sind eigene Tasten vorhanden. Das beschleunigt die Eingabe der Befehle und hilft, Eingabefehler zu vermeiden. Alle Schlüsselwörter können auf die ersten 3 Buchstaben abgekürzt werden, da das System nur die ersten 3 Zeichen zur Bestimmung eines Befehles analysiert.

Wird nach dem dritten Zeichen eines Schlüsselwortes ein falscher Buchstabe eingegeben, wird dieser Name trotzdem richtig interpretiert.

Nach Eingabe des Schlüsselwortes sind die Operanden zeichenweise einzutasten. Die Anweisung wird danach durch Drücken der Taste END OF LINE abgeschlossen. Zwischen dem Schlüsselwort und dem ersten Operanden müssen eine oder mehrere Leerstellen stehen.

Nach dem Drücken der Taste END OF LINE wird der Befehl analysiert. Wenn der Befehl vom System interpretiert werden kann, wird er unmittelbar ausgeführt. Andernfalls erscheint eine Fehlermeldung. Der Befehl kann dann mit den Möglichkeiten der Editing-Operationen im Display korrigiert werden.

Die Eingabe der Systembefehle kann auch mit Hilfe der Funktionstasten F1-F16 erfolgen (siehe FKEY# in den Kapiteln 6.5 und 9.1)

6.2 Ausführung von Befehlen mit Prozeduren

Um eine Vielzahl von Systembefehlen oder andere manuelle Eingaben ohne Eingriff des Benützers ablaufen lassen zu können, bietet das System dem Anwender die Möglichkeit, Katalog-Prozeduren zu definieren.

Zu diesem Zweck werden die entsprechenden Anweisungen und Befehle in einem File - dem COMMAND FILE - gespeichert. Die Ausführung der Befehlsfolge wird durch den Befehl PRO (PROCEDURE) gestartet. Dies ermöglicht dem Benützer, "Programme" auf Systemebene zu erstellen. Häufig verwendete Folgen von Systemanweisungen können also in Commandfiles abgelegt und bei Bedarf abgerufen werden.

6.2.1 Erstellen einer Prozedur

Eine Prozedur wird am einfachsten als Textfile definiert und anschliessend mit TRANSCODE in ein Datenfile umgewandelt.

Zuerst sind die für das Erstellen eines Textfiles erforderlichen Systembefehle einzugeben, also TEXT und normalerweise auch AUTO .

6.2.1.1 Eingabe von Systembefehlen und Dienstprogrammen

Systembefehle und Aufrufe von Dienstprogrammen sind zeilenweise einzugeben.

Beispiel:

```
30 CRE *DFILE, (MAXL, D1), R, 50000
40 EXE LBC,(MORITZ,D2),,SIZE=100,*=2,+=3,NP=5
```

6.2.1.2 Eingabe von BASIC-Statements

Innerhalb einer Prozedur können auch Programme neu erstellt oder vorhandene Programme modifiziert werden. Bei den einzelnen Programmen ist zusätzlich zur Zeilennummer des Textfiles auch die Zeilennummer des BASIC-Statements einzugeben. Wird ein Programm in einer Prozedur neu erstellt, muß der Systembefehl NEW eingetastet und das Programm mit END beendet werden.

Beispiel:

```
10 NEW
20 10 INPUT A
30 20 B=A*PI
40 30 PRINT A,B
50 40 END
60 RUN
```

Bemerkung:

Die Syntax-Kontrolle der BASIC-Anweisungen erfolgt erst bei der Ausführung der Prozedur. Für die Verarbeitung eines so definierten Programmes gelten die üblichen Regeln.

6.2.1.3 Spezielle Befehle für Prozeduren

Zusätzlich zu den allgemeinen Systembefehlen bestehen spezielle Befehle, die nur innerhalb einer Prozedur verwendet werden können. Ihre Anwendung ist in Kapitel 6.4 ausführlicher beschrieben.

IN=	für automatischen Input bei der Verarbeitung von Programmen durch Prozeduren.
IF CC=	für bedingte Sprünge innerhalb einer Prozedur.
TEST,PRO	für spezielle Behandlung von Fehlermeldungen.

6.2.1.4 Prozedur als Datenfile

Das Commandfile einer Prozedur kann direkt als Datenfile erstellt werden. Folgender Aufbau ist zu berücksichtigen:

- Jede Prozedurzeile wird als String abgespeichert (max. 80 Zeichen).
- Die ersten 4 Zeichen jedes Strings enthalten eine 4-stellige Zeilennummer, eventuell mit führenden Nullen.

6.2.2 Aufruf einer Prozedur

Eine Prozedur kann auf mehrere Arten zur Verarbeitung aufgerufen werden:

- Manueller Aufruf mit dem Systembefehl PROCEDURE (siehe Kapitel 6.3).
- Automatischer Aufruf im Anschluss an eine Initialisierungsphase des Systems. In diesem Falle muss die Prozedur unter dem Namen *SETUP abgespeichert sein.
- Automatischer Aufruf durch eine andere Prozedur.

6.3 Liste und Funktion der Systembefehle

Die Systembefehle und ihre Funktionen sind nachstehend in alphabetischer Reihenfolge aufgeführt.

Name	Funktion
AUTO #	bewirkt die automatische Vorgabe von Zeilennummern bei der Eingabe von Text- oder Programmzeilen.
CALCULATOR	ruft den "Calculator-Mode" auf.
CATALOG	druckt das Inhaltsverzeichnis einer offenen Bibliothek.
COMPILE	wandelt ein Textfile in ein ausführbares BASIC-Programm um.
CONFIGURE	erlaubt die Festlegung einer bestimmten Systemkonfiguration.
CREATE	reserviert den Speicherplatz für ein Datenfile in einer Bibliothek.
DATE	speichert das Datum auf der Systemdisk oder -diskette.
DCHANGE	erlaubt den Wechsel einer Diskette. Das Programm im Hauptspeicher wird dadurch nicht gelöscht.
DECOMPILE	wandelt ein ausführbares Programm im Arbeitsspeicher in ein Textfile um.
DELETE LINE	löscht eine oder mehrere Zeilen eines Programms oder eines Textfiles.
DRAW	erstellt eine Hardcopy des Bildschirmininhaltes auf dem Thermodrucker.
ENVIRONMENT	beschreibt die aktuelle Systemkonfiguration.
ERASE	löscht den Bildschirminhalt.
EXEC	lädt ein Dienstprogramm (siehe Kapitel 7) in den Arbeitsspeicher und verarbeitet es.
FETCH	überträgt eine Programm- oder Textfilezeile aus dem Arbeitsspeicher in den Tastaturbuffer.
LBCLOSE	schließt Bibliotheken.
LBOPEN	öffnet Bibliotheken.
LBRESTORE	schließt die momentan offenen Bibliotheken und öffnet an deren Stelle die als "Standard" (LBS) definierten.
LBSTORE	definiert die momentan offenen Bibliotheken als "Standard" für automatisches Öffnen bei jedem Laden des Systems.

LDIMAGE	Ausgabe eines gespeicherten Bildes auf den Bildschirm.
LDKEYS	weist den Funktionstasten den auf der Systemdisk gespeicherten Standardinhalt zu.
LINK	integriert ein als Textfile auf einer Disk oder Diskette gespeichertes Teilprogramm (oder eine mehrzeilige Funktion) in ein Programm im Hauptspeicher.
LIST	druckt eine oder mehrere Zeilen eines sich im Arbeitsspeicher befindlichen Programms oder Textfiles aus.
LVTOC	druckt das Verzeichnis der Bibliotheken einer Einheit oder aller offenen Bibliotheken.
MERGE	kombiniert einen sich im Arbeitsspeicher befindlichen Text mit einem Textfile einer Bibliothek.
MODIFY	ändert den Namen und/oder den reservierten Speicherplatz eines Files.
NEW	ermöglicht die Eingabe eines Programms über die Tastatur. Der Arbeitsspeicher wird gelöscht.
OLD	lädt ein Programm oder Textfile aus einer Bibliothek in den Arbeitsspeicher, dessen alter Inhalt dadurch gelöscht wird.
OPTIONS	lädt zusätzliche Teile des Betriebssystems in den Hauptspeicher. Der für den Anwender verfügbare Teil des Hauptspeichers wird dadurch verkleinert.
PREPARE	analysiert ein Programm syntaktisch und wandelt es in ein (lauffähiges) Objektprogramm um. Nach Ausführung des Befehles befindet sich das System im Debugging-Mode.
PROCEDURE	führt eine Katalogprozedur aus.
PURGE	löscht ein File einer nicht geschützten Bibliothek.
REPLACE	ersetzt ein Programm (oder Textfile) einer Bibliothek durch das sich im Arbeitsspeicher befindliche Programm (oder Textfile) gleichen Namens.
RESEQUENCE	ermöglicht die Neu- bzw. Umnumerierung eines Programms oder Textes im Arbeitsspeicher.
REVERSE	Umkehrung des Bildes auf dem Bildschirm.
RUN	startet die Ausführung eines Programmes.

SAVE	speichert ein sich im Arbeitsspeicher befindliches Programm oder Textfile unter einem Namen in einer Bibliothek.
SECURE	schützt ein Programm oder Datenfile ganz oder teilweise vor dem Ausdruck und vor Änderungen.
SHIFT	ermöglicht das Addieren einer Konstante zu den im Arbeitsspeicher vorhandenen Zeilennummern ab einer bestimmten Stelle.
SPACE	zeigt den verfügbaren Speicherplatz einer Bibliothek im Display an.
STIMAGE	speichert den Bildschirminhalt in einem sequentiellen Datenfile.
STKEYS	speichert den aktuellen Inhalt der Funktionstasten auf der Systemdisk als Standardbelegung.
TEXT	ermöglicht die Eingabe eines Textfiles über die Tastatur. Der Arbeitsspeicher wird dadurch gelöscht.
TRANSCODE	konvertiert ein Datenfile in ein Textfile oder umgekehrt.
TRUNCATE	reduziert den reservierten Speicherplatz eines sequentiellen Datenfiles auf die aktuelle Länge des Files.
VALIDATE	schließt ein offengebliebenes File.

BEFEHL: AUTO # (automatic line numbering)

FUNKTION: Automatische Zuordnung von Zeilennummern zu den Anweisungen eines Programmes oder den Zeilen eines Textes.

FORMAT: AUT[O #][Zeilennummer][,Schrittweite]

"Zeilennummer" positive, ganze Zahl zwischen 1 und 9999, welche die Nummer der nächsten Anweisung oder Textzeile angibt.

"Schrittweite" positive, ganze Zahl, welche die jeweilige Erhöhung zur nächsten Zeilennummer bestimmt.

WIRKUNG:

- Nach Abschluß einer Anweisung oder Textzeile mit der Taste EOL und deren Übertragung in den Arbeitsspeicher, wird die Nummer der nächsten Zeile im Tastaturbuffer generiert und im Display angezeigt.
- Fehlen die Options-Angaben, erhöht das System die zuletzt benutzte Zeilennummer um 10 Einheiten. Steht im Arbeitsspeicher noch keine Programm- oder Textzeile, beginnt die Numerierung mit dem Wert 10.
- Fehlt nur der Parameter "Zeilennummer", wählt das System als Erhöhungseinheit die Schrittweite und erhöht die bisher höchste Zeilennummer im Hauptspeicher entsprechend. Beinhaltet der Hauptspeicher noch keine Programm- oder Textzeile, beginnt die Numerierung mit dem Wert der Schrittweite.
- Ist der Parameter "Schrittweite" nicht definiert, beginnt die Numerierung mit dem unter "Zeilennummer" angegebenen Wert. Die Schrittweite beträgt in diesem Falle 10.

BEMERKUNGEN:

- Die Aufhebung der automatischen Zeilennumerierung erfolgt durch gleichzeitiges Drücken der Tasten [CLEAR] und [SHIFT] bzw. ↑ und ↓.

  Das System befindet sich danach im Command-Mode. Sollen die Zeilen wieder automatisch numeriert werden, ist der Befehl AUTO # erneut zu setzen.
- Das Schlüsselwort des Befehles kann über die Taste [AUTO #] der Befehlstastatur eingegeben werden.

BEISPIELE:

Beispiel 1

Automatische Zuteilung von Zeilennummern während der Programmerstellung.

Automatisch: Anfangswert 10
Schrittweite 10

```
NEW
AUTO#
10 DIM A(20)
20 FOR I=1 TO 20
30 INPUT A(I)
40 NEXTI
50 END
```

Beispiel 2

Zeilennumerierung während der Eingabe eines Textfiles mit vorgewählten Werten für Anfang und Schrittweite.

```
TEX
AUTO# 3,2
3 Dies ist das Beispiel
5 fuer eine Texteingabe.
7 Die Zeilen werden automatisch numeriert.
9 Die erste Zeilennummer ist 3,
11 die Schrittweite betraegt 2.
```

CALCULATOR

BEFEHL: CALCULATOR

FUNKTION: Aufruf des Calculator-Mode um direkte Rechenoperationen auszuführen.

FORMAT: CAL[CULATOR] [{n / FL}]

n ist eine natürliche Zahl zwischen 0 und 13.

FL Ergebnisse sollen in Gleitkommadarstellung angezeigt werden.

WIRKUNG: Das System wird vom Command-Mode in den Calculator-Mode umgeschaltet. Der Operand (n) bestimmt die Anzahl Nachkommastellen, mit denen das Ergebnis einer Rechenoperation dargestellt wird. Der Operand (FL) bestimmt, daß die Darstellung in Gleitkommaformat erfolgen soll. Fehlt der Operand, so wird das Ergebnis im Standardformat angezeigt.

BEMERKUNGEN:

- Durch Eingabe irgendeines gültigen Systembefehles (außer CALCULATOR) kehrt das System vom Calculator-Mode in den Command-Mode zurück.
- Nähere Beschreibung des Calculator Modes erfolgt in Kapitel 8.

BEISPIELE: siehe Kapitel 8

BEFEHL: CATALOG

FUNKTION: Ausdruck des Inhaltsverzeichnisses von Bibliotheken.

FORMAT:

```
CAT[ALOG][ filename[,lib ref]
           [{*}],[lib ref],[{P}][,F]
           [{+}]          [{T}]
           [{:}]          [{D}]
                          [{O}]
```

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Diskstation anzugeben.

"*" = Package-Teilbibliothek.

"+" = Common-Teilbibliothek.

":" = ganze Bibliothek.

"filename" Werden nur Informationen über ein bestimmtes File verlangt, ist "filename" durch den effektiven Namen des Files zu ersetzen.

"P" = Druck des Kataloges der Programmfiles.

"T" = Druck des Kataloges der Textfiles.

"D" = Druck des Kataloges der Datenfiles.

"O" = Druck des Kataloges der Objektfiles (Assembler-Routinen).

"F" = Druck aller Informationen über die Files.

WIRKUNG: Der vollständige Befehl mit allen Operanden (z.B. CATALOG :,(LIB5,D1),P,F) bewirkt den Ausdruck der Informationen über alle durch die ersten 3 Operanden spezifizierten Files:

- Filename
- Art des Files
- Erstellungsdatum des Files
- Datum der letzten File-Modifikation
- Für das File reservierter Speicherplatz in Bytes

- Aktuelle Länge des Files in Bytes
- Identifikationscode des Files
- Anzahl der Abschnitte (max. 4), in die das File innerhalb einer Bibliothek zerlegt ist.

Offen gebliebene Datenfiles werden durch das Wort OPEN gekennzeichnet (siehe Systembefehl VALIDATE).

Für die einen Common-Bereich verwendenden Programme wird die dafür eingesetzte Anzahl Worte mit C=...W angegeben.

Fehlt der 4. Operand (F), werden folgende, gekürzte Informationen über die durch die anderen Operanden spezifizierten Files ausgegeben:

- Filename (ohne Bibliotheksangabe * oder +)
- Art des Files

Fehlt der 3. Operand, erfolgt der Ausdruck der Informationen über Programm-, Text- und Datenfiles (sowie Objektfiles).

Fehlt der 2. Operand, erfolgt der Ausdruck der Informationen über die erste offene Bibliothek (siehe LVT*).

Fehlt der 1. Operand, erfolgt der Ausdruck der Informationen über die Files der User-Teilbibliothek.

BEMERKUNGEN:

- Steht als 1. Operand "filename", ist der 3. und 4. Operand bedeutungslos.
- Die Ausführung des Befehles kann durch Drücken der Tasten CTRL + EXIT abgebrochen werden.
- Ein mit DEAD gekennzeichnetes File ist nicht mehr verwendbar und kann mit PURGE nicht gelöscht werden. Damit der Speicherplatz dieses Files wieder verfügbar wird, sind alle übrigen Files dieser Bibliothek mit LIBCOPY in eine "neue" Bibliothek zu kopieren. Anschließend kann mit LBSCRATCH die "alte" Bibliothek gelöscht werden.
- Die durch "lib ref" angegebene Bibliothek muß offen sein (siehe LBOPEN).
- Beinhaltet der Operand "lib ref" nur den Bibliotheksnamen, so wird der Katalog der ersten offenen Bibliothek gleichen Namens gedruckt.
- Besteht der Operand "lib ref" nur aus dem Namen der Einheit, so wird der Katalog der ersten offenen, auf dieser Einheit gespeicherten Bibliothek ausgedruckt.
- Für jedes File belegt das Betriebssystem einen zusätzlichen Sektor mit Kontrollinformationen, der in der angegebenen Filelänge nicht enthalten ist (siehe SPACE).

BEISPIELE:

CAT :	Druck einer Liste der Namen und Arten aller Files der ersten offenen Bibliothek (LVT*).
CAT +,(LIB1,F1),T,F	Druck eines Kataloges der Textfiles der Common-Teilbibliothek von LIB1 auf der Disk F1.
CAT *,,,F	Ausdruck eines vollständigen Kataloges aller Files der Package-Teilbibliothek der ersten offenen Bibliothek.

Ausdruck aller Teilbibliotheken der oberen Diskettenstation mit vollständiger Information.

```
CAT :,(,F1),,F

     MDOSC -R 2.0    *  VOLLABEL = INTERN  *  LIBRARY = P&FSYS  *  DATE:22-05-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE       EXT

     *STEUER  R  27-06-77  27-06-77   4096       4096                   1
     *OK      P  23-11-79  20-06-78   3712       3712       EDOCEDOC    1
     *VOL     P  27-06-77  20-06-78   0640       0640                   1
     *SEC     P  23-11-79  20-06-78   3712       3712       EDOCEDOC    1
     *CODE    P  23-11-79  20-06-78   3968       3968       EDOCEDOC    1
     *TEXT    P  11-12-78  11-12-78   0768       0768                   1

     +READHX  P                                                          DEAD
     +SEQU    S  22-05-80  22-05-80   1024       0000                   1 OPEN
     +RANDOM  R  22-05-80  22-05-80   1152       1152                   1
     +ZERO    Z  22-05-80  22-05-80   0896       0896                   1

      COPY    P  20-06-78  20-06-78   0384       0384                   1
      D       P  20-06-78  11-12-78   10240      10240                  1
      MIDOC   P  07-12-78  07-12-78   9984       9984                   1
      HEXA    P  11-12-78  11-12-78   1152       1152                   1
      FNK     T  11-12-78  11-12-78   0896       0896                   1
```

Ausdruck der Liste der Files der User-Teilbibliothek der ersten offenen Bibliothek.

```
CAT

     MDOSC -R 2.0    *  VOLLABEL = SYSBTS  *  LIBRARY = P6066   *  DATE:22-05-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE       EXT

      FIGUR1  P
      PROG1   P
      FIGUR2  P
      START   P
      MATRIX  R
      MORGEN  P
      XX      S
```

Ausdruck aller Informationen zum File *STEUER.

```
CAT *STEUER,(,F1)
     *STEUER   R  27-06-77  27-06-77   4096        4096                  1
```

BEFEHL: COMPILE

FUNKTION: Übersetzung eines sich im Arbeitsspeicher befindlichen Textfiles in ein ausführbares BASIC-Programm.

FORMAT: COM[PILE]

WIRKUNG: Ein sich im Arbeitsspeicher befindliches Textfile wird in ein ausführbares BASIC-Programm umgewandelt.

BEMERKUNGEN:
- Das im Arbeitsspeicher vorhandene Textfile muß ein BASIC-Quellenprogramm sein. Ist dies nicht der Fall, erfolgt eine Fehlermeldung.
- Beinhaltet das Textfile syntaktische Fehler, werden nach der vollständigen Übersetzung des Files entsprechende Fehlermeldungen ausgegeben.
- Die Zeilennummern von fehlerhaften Zeilen bleiben erhalten und können für die Ausführung der notwendigen Korrekturen in den Tastaturbuffer gerufen werden (Befehl: FETCH).

BEISPIELE: Beispiel 1

Eingabe des Programmes als Text mit nachfolgender Kompilierung und Ausführung.

```
TEX
AUTO# 5,5
5 INPUT I
10 X=PI+I
15 PRINT X
20 END
COM

LIST
FILE


0005 INPUT I
0010 LET X=PI+I
0015 PRINT X
0020 END

END OF LISTING




RUN
**** FORMALLY CORRECT PROGRAM ****
?
14
 17.141593
```

Beispiel 2:

Fehlerhafte Programmeingabe mit anschließender Korrektur:

```
TEX
AUTO#
10 INPUT I
20 DIEP I
30 GOTO 10
40 END

COM
ERROR 102 IN LINE 20


FETCH 20
0020 DIEP I
0020 DISP I


LIST
FILE


0010 INPUT I
0020 DISP I
0030 GOTO 10
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
?
12354
 12354
?
```

Zeile 20: Falsches Schlüsselwort "DIEP".
Kontrolle der Syntax.
Korrektur der Zeile 20.
Anschließend Ausführung ohne erneute Eingabe von COM.

BEFEHL: CONFIGURE

FUNKTION: Festlegung einer bestimmten Systemkonfiguration

FORMAT: CON[FIGURE][MS=n][,PRO] [,SP= { 0 : 9 } F { 7 8 } { Y N } { E 0 } { 1 2 }]

n Kapazität des Anwenderspeichers in K-Bytes ($15 \leq n \leq 192$)

Pro Automatischer Start der Katalogprozedur *SETUP bei jedem Laden des Betriebssystems.

SP=... Bestimmung der Übertragungsparameter des angeschlossenen Systemdruckers.

WIRKUNG: Der Inhalt des Arbeitsspeichers wird gelöscht und das System neu initialisiert. Die Parameter werden auf der Systemdisk gespeichert und bestimmen die Konfiguration so lange, bis sie durch einen neuen CONFIGURE-Befehl geändert wird.

- Der Parameter MS (Memory-Size) ist angegeben:

  Die Größe des Anwenderspeichers wird auf n K-Bytes reduziert.

- Der Parameter PRO ist angegeben:

  Beim Laden des Betriebssystems wird automatisch die Prozedur mit dem Namen *SETUP gestartet.

- Der Parameter SP=... ist angegeben:

  Der angeschlossene Systemdrucker wird mit den unter SP angegebenen Übertragungsparametern angesteuert. Im folgenden werden die einzelnen Übertragungsparameter erläutert (V24-Schnittstelle)

  {0 : 9} Durch eine ganze Zahl zwischen 0 und 9 wird die Übertragungsgeschwindigkeit (Band-Rate) bestimmt. Die folgende Tabelle gibt die zum jeweiligen Zahlen-Code korrespondierende Band-Rate an:

Zahlen-Code	Band-Rate
0	50
1	110
2	300
3	600
4	1200
5	2400
6	4800
7	9600
8	19200
9	38400

  F Parameter F (Full Duplex) ist obligatorisch und immer anzugeben.

  {7 / 8} Definiert die Anzahl bits pro übertragener Bytes.

  {Y / N} bestimmt, wo ein Parity-Check zu machen ist:

  Y (YES) für "Ja",
  N (No) für "Nein"

  {E / O} definiert die Art des Parity-Checks:

  E (Even) für gerade Parität
  O (Odd) für ungerade Parität

  {1 / 2} bestimmt die Anzahl Stop-Bits nach jedem Zeichen.

BEMERKUNGEN:
- Wird der Parameter MS nicht angegeben, so wird vom System die volle Anwenderkapazität benutzt.

- Wird der Parameter PRO nicht angegeben, so wird eine (vorhandene) Prozedur *SETUP nicht ausgeführt.

- Wird der Parameter SP nicht angegeben, so werden die Defaultwerte SP=8F7YE2 angenommen. Diese entsprechen den Übertragungsparametern des Thermodruckers PR 2400 als Systemdrucker. Bei Nadeldruckern sollte SP=6F7YE2 angegeben werden.

- Wird nur CON (ohne Parameter) eingegeben, so wird das System neu geladen und die folgenden Defaultwerte angenommen:
  - Volle Anwenderkapazität
  - Keine automatische Startprozedur
  - Übertragungsparameter für Thermodrucker PR 2400.

BEISPIELE:

```
CON PRO

CON SP=6F7YE2
```

BEFEHL: CREATE

FUNKTION: Reservieren des für ein Datenfile in einer Bibliothek notwendigen Speicherplatzes auf einer Disk.

FORMAT: CRE[ATE] filename, [lib ref],[{S / R / Z}][,n]

"filename"	Name des zu erstellenden Files.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten-Diskstation anzugeben.
"S"	definiert, daß das File "sequentiell" ist.
"R"	gibt an, daß das File "random" ist. Es enthält numerische Daten in einfacher Genauigkeit, die den Wert "nicht initialisiert" aufweisen.
"Z"	bezeichnet das File als "random". Es beinhaltet numerische Daten in einfacher Genauigkeit mit dem Wert 0 (Null).
"n"	ganze positive Zahl. Sie bestimmt die Anzahl der vom System für das File zu reservierenden Bytes.

WIRKUNG: Der alle Parameter aufweisende Befehl reserviert n Bytes für das durch "filename" bezeichnete File in der durch "lib ref" angegebenen Bibliothek.

- Fehlt der 3. Operand, nimmt das System "S" an.
- Fehlt der 4. Operand, werden 4096 Bytes für das File auf der angegebenen Disk reserviert.

BEMERKUNGEN:
- Die Anzahl Bytes (n) wird vom System automatisch auf ein Vielfaches von 256 aufgerundet.
- Die angesprochene Bibliothek muß offen sein.

- In einer mit LBPROTECT geschützten Package-Teilbibliothek kann kein neues File angelegt werden.
- Ein und derselbe Filename kann auf verschiedenen Disks und auch in mehreren Bibliotheken verwendet werden. Wird jedoch ein Filename ein zweites Mal in der gleichen Bibliothek angegeben, erscheint die Fehlermeldung ERROR 186.
- Eine Bibliothek (bzw. Teilbibliothek) kann maximal die bei der Erstellung des Dienstprogrammes LBCREATE festgelegte Anzahl Files aufnehmen (ERROR 183).

BEISPIELE:

```
CRE *NAME1,C,F1),R,8192
```

In der Package-Teilbibliothek auf der Disk in der linken Station wird ein Randomfile mit 8 K-Bytes erstellt.

```
CRE SFILE
```

In der ersten offenen Bibliothek wird ein sequentielles File (SFILE) mit 4 K-Bytes erstellt.

BEFEHL: DATE

FUNKTION: Speicherung des eingetasteten Datums auf der Systemdisk zwecks automatischer Datierung von Operationen mit Files auf einer Disk.

FORMAT: DAT[E] Datum

"Datum" Folge von 6 beliebigen ISO-Zeichen, mit Ausnahme des Leerzeichens.

WIRKUNG: Bei jedem Laden des Systems (nach dem Einschalten oder nach Ausführung eines OPTIONS- oder CONFIGURE-Befehles) wird auch die letzte, als Datum auf der Systemdisk gespeicherte Zeichenfolge in den Hauptspeicher geladen.

BEMERKUNG: Die normalerweise verwendete Zeichenfolge lautet:

TTMMJJ (= Tag, Monat, Jahr)

Die Ausgabe der Zeichen erfolgt in drei Gruppen zu je zwei durch "-" getrennte Zeichen (z.B. "120379" = 12-03-79 oder "Juli 78" = Ju-li-78)

BEISPIELE:

```
DATE JULI78

     FILE    P  JU-LI-78  JU-LI-78   0256       0256              1
```

```
DATE 120379

     FILE    P  JU-LI-78  12-03-79   0256       0256              1
```

DCHANGE

BEFEHL: DCHANGE (disk change)

FUNKTION: Austauschen von Disks bei geladenem System, ohne dadurch den Inhalt des Arbeitsspeichers zu löschen.

FORMAT: DCH[ANGE] unit name

"unit name" Name der Diskeinheit.

WIRKUNG: Die Bibliotheken in der durch "unit name" angegebenen Einheit werden geschlossen. Das System gibt die Display-Meldung "LOAD DISK ON unit name" aus und erlaubt damit den Austausch der Disk.

Nach dem Austausch ist die Funktion CONTINUE zu betätigen.

Sind weniger als sechs Bibliotheken offen und ist die erste Bibliothek auf der neuen Disk nicht durch ein Password geschützt, wird diese geöffnet und die Display-Meldung "LIBR lib name OPEN ON unit name" ausgegeben.

BEMERKUNGEN:

- Der Systembefehl DCHANGE muß
  . vor jedem Austausch einer Disk
  . und vor jedem Einsatz bisher nicht belegter Stationen
  verwendet werden. Andernfalls erscheint die Fehlermeldung "ABN unit name - DCH OMITTED". Diese Meldung ist mit CLEAR und Funktion CONTINUE zu löschen und DCHANGE nachzuholen.
- Befindet sich das Betriebssystem auf der Disk so ist nur ein Wechsel gleicher Release-Stufe und, bei reduziertem System, mit gleichem Inhalt möglich. Andernfalls muß das System neu eingeschaltet bzw. mit dem entsprechenden BOOTSTRAP neu geladen werden.

DECOMPILE

BEFEHL: DECOMPILE

FUNKTION: Umwandlung eines im Arbeitsspeicher vorhandenen BASIC-Programmes in ein Textfile.

FORMAT: DEC[OMPILE]

WIRKUNG: Das sich im Hauptspeicher befindliche BASIC-Programm wird in ein Textfile umgewandelt.

BEMERKUNGEN:

- Bei der Dekompilierung werden in den Tabellen enthaltene Variablennamen und Zeilennummern, die aufgrund von Änderungen nicht mehr verwendet bzw. angesprochen werden, aus den Tabellen gelöscht.

  Durch Eingabe der Befehle DEC und COM kann also ein Programm verkürzt und im Ablauf beschleunigt werden.
- Mit dem Befehl SECURE geschützte Programme können nicht in ein Textfile konvertiert werden.
- Enthalten zu dekompilierende Zeilen mehr als 76 signifikante Zeichen, so erscheint nach dem Dekompilieren die Fehlermeldung ERROR 213. Die entsprechenden Zeilen sind jedoch richtig dekompiliert.

BEISPIEL:

```
AUTO#
10 FOR I=1 TO 10
20 PRINT SQR(I)
30 NEXT I
40 END


DEC

LIST .,X
FILE


 FOR I=1 TO 10 STEP 1
 PRINT SQR(I)
 NEXT I
 END

END OF LISTING
```

Programmeingabe

Umwandlung des Programmes in einen Text

Ausdruck als Text ohne Zeilennummern

DELETE LINE

BEFEHL: DELETE LINE

FUNKTION: Löschung einer oder mehrerer Programm- oder Textzeilen im Arbeitsspeicher.

FORMAT: DEL[ETE LINE][[Zeilennummer$_1$][, Zeilennummer$_2$]]

"Zeilennummer$_1$" Nummer der zu löschenden Zeile oder der ersten Zeile des zu löschenden Abschnittes.

"Zeilennummer$_2$" letzte Zeile des zu löschenden Abschnittes.

WIRKUNG:

- Der vollständige, mit allen Operanden eingegebene Befehl gibt dem System bekannt, dass alle Zeilen des durch "Zeilennummer1" und "Zeilennummer2" begrenzten Abschnittes des Programmes oder Textfiles im Arbeitsspeicher gelöscht werden sollen.
- Fehlt der 2. Operand, wird nur die durch den 1. Operand definierte Zeile gelöscht.
- Fehlen beide Operanden, wird, je nach der unmittelbar vorher erfolgten Operation, eine der folgenden Zeilen gelöscht:
  1. Die zuletzt über die Tastatur eingegebene Zeile.
  2. Die zuletzt mit dem Befehl FETCH oder mit Hilfe der Taste ↑ oder ↓ in den Tastaturbuffer übertragene Zeile.
  3. Die zuletzt mit dem Befehl LIST ausgedruckte Zeile.
  4. Die letzte Zeile eines mit dem Befehl RUN ausgeführten Programmes.

BEMERKUNGEN:

- Zeilen von mit dem Befehl SECURE geschützten Programmen können nicht gelöscht werden.
- Das Schlüsselwort DELETE LINE kann mit den gleichzeitig gedrückten Tasten DELETE LINE/RUN und SHIFT eingegeben werden.
- Die im Befehl angegebenen Zeilennummern müssen im Programm vorhanden sein.

BEISPIEL:

```
OLD TEST
LIST
FILE      TEST


0010 REM **BEISPIEL "DELETE LINE " **
0020 REM
0030 FOR I=1 TO 5 STEP 1
0040 PRINT I,
0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;
0070 NEXT J
0080 PRINT
0090 NEXT I
0100 END

END OF LISTING



RUN
 1                1  2  3  4
 2                1  2  3  4
 3                1  2  3  4
 4                1  2  3  4
 5                1  2  3  4



DELETE LINE 30,40
DELETE LINE 90
LIST
FILE      TEST


0010 REM **BEISPIEL "DELETE LINE " **
0020 REM
0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;
0070 NEXT J
0080 PRINT
0100 END

END OF LISTING



RUN
 1  2  3  4
```

BEFEHL: DRAW

FUNKTION: Hardcopy des Bildschirminhaltes auf Thermodrucker.

FORMAT: DRA[W] [A]

WIRKUNG: A gibt an, daß vom Inhalt des Bildschirms, der durch den alphanumerischen Controller dargestellt wird, eine Hardcopy erstellt werden soll. Der aktuelle Inhalt des Bildschirmes wird über den Thermodrucker ausgegeben. Der Ausdruck erfolgt im Verhältnis 1:1.12. Ist der Operand A nicht angegeben, so wird der Inhalt des graphischen Controllers ausgegeben.

BEMERKUNGEN:
- Der Ausdruck kann mit der Funktion BREAK oder EXIT abgebrochen werden.
- Das System muß für den Befehl DRAW ohne Operand A mit der Option GDI initialisiert sein.

ENVIRONMENT

BEFEHL: ENVIRONMENT

FUNKTION: Beschreibung der im Einsatz stehenden Konfiguration.

FORMAT: ENV[IRONMENT]

WIRKUNG: Folgende Informationen über die System-Konfiguration werden ausgegeben:

- Release-Code der Systemsoftware
- zuletzt mit DATE gespeichertes Datum
- physische Größe des Anwenderspeichers
- Art des Systemdruckers (alphanumerisch/graphisch)
- vorhandene Controller
- Spezifikationen gemäß zuletzt verwendetem CONFIGURE-Befehl (logische Größe des Arbeitsspeichers)
- momentan geladene Optionen
- Liste der für Disk-Einheiten zulässigen Namen
- Name der Systemdisk
- Liste der zur Zeit offenen Bibliotheken
- Aktuelle Liste der beim Einschalten der Anlage vom Betriebssystem automatisch geöffneten Bibliotheken (siehe auch LBSTORE, LBRESTORE)
- Art des Bildschirms (graphisch)

BEISPIEL:

```
ENV
     MDOSC -R 2.0 *        SYSTEM ENVIRONMENT        * DATE:03-01-83

                  MEMORY PHYSICAL SIZE : 64 K
                  SYSTEM PRINTER : GRAPHIC
                  VIDEO DISPLAY : GRAPHIC
                  I/O INTERFACES :

                  MEMORY LOGICAL SIZE: 64 K
                  SOFTWARE OPTIONS : GDISP

     DISK UNITS:  NAME          PERIPH CODE          PERIPH TYPE
                  F1            C0                   FDU
                  F2            C1                   FDU

                  OPERATING SYSTEM ON DISK UNIT F1

     OPEN LIBRARIES :     NAME            DISK UNIT
                          DATEN           F1
                          P6441           F2
                          HELP            F1
                          D6411           F2
     STORED LIBRARIES :   NAME            DISK UNIT
                          DATEN           F1
                          P6441           F2
```

ERASE

BEFEHL: ERASE

FUNKTION: Löschen des Bildschirminhaltes

FORMAT: ERA[SE] [{A / G}]

WIRKUNG:
A bestimmt den alphanumerischen Controller
G bestimmt den graphischen Controller.

BEMERKUNG: Ist kein Operand angegeben, so wird der gesamte Bildschirminhalt gelöscht. Bei Angabe des Operanden A wird der Inhalt des alphanumerischen Controllers gelöscht. Bei Angabe des Operanden G wird der Inhalt des graphischen Controllers gelöscht.

BEFEHL: EXEC (execute)

FUNKTION: Laden eines Dienstprogrammes in den Arbeitsspeicher und dessen Ausführung.

FORMAT: EXE[C] Dienstprogramm[,Parameter[,Parameter]...]

"Dienst-programm"	Name des Dienstprogrammes.
"Parameter"	Definition der für das jeweilige Dienstprogramm spezifischen Operanden.

WIRKUNG: EXEC verlangt vom System das durch seinen Namen spezifizierte Dienstprogramm. Dieses wird in den Arbeitsspeicher geladen und ausgeführt.

BEMERKUNG: Durch den Befehl EXEC wird der Inhalt des Arbeitsspeichers gelöscht.

Liste der verfügbaren Dienstprogramme:

DCOPY
DINIT
FLCOPY
FLPRINT
LBCREATE
LBEMPTY
LBPROTECT
LBRENAME
LBSCRATCH
LIBCOPY
RESTRUCT
VOLLABEL

FETCH

BEFEHL: FETCH

FUNKTION: Übertragung einer Programm- oder Textzeile aus dem Hauptspeicher in den Tastaturbuffer.

FORMAT: FET[CH][Zeilennummer]

"Zeilennummer" Nummer der in den Tastaturbuffer zu übertragenden Zeile.

WIRKUNG:

- Die der eingegebenen Zeilennummer entsprechende Programm- oder Textzeile wird in den Tastaturbuffer übertragen und in Zeile 0 am Bildschirm angezeigt (vgl.Kap.1.2.3.)

- Fehlt der Operand, wird, je nach der unmittelbar vorher durchgeführten Operation, eine der folgenden Zeilen in den Tastaturbuffer übertragen und am Bildschirm in in Zeile 0 sichtbar gemacht:

  . Die zuletzt über die Tastatur eingegebene Zeile.
  . Die letzte Zeile des mit dem Befehl OLD in den Arbeitsspeicher geladenen Files.
  . Die letzte, mit dem Befehl LIST ausgedruckte Zeile.
  . Die letzte Anweisung des gerade ausgeführten und noch im Arbeitsspeicher stehenden Programmes.
  . Die letzte mit Hilfe der Taste ↑ oder ↓ in den Tastaturbuffer übertragene Zeile.

BEMERKUNGEN:

- Ist die im Befehl FETCH angegebene Zeilennummer im Arbeitsspeicher nicht vorhanden, wird die Zeile mit der nächstkleineren Zeilennummer in den Tastaturbuffer übertragen. Ist die in FETCH vorgegebene Zeilennummer kleiner als die kleinste Zeilennummer im Arbeitsspeicher, wird die Zeile mit der höchsten Zeilennummer in den Tastaturbuffer übertragen.

- Das Format am Bildschirm gezeigten Zeile kann vom Eingabeformat abweichen, da dieses vom System modifiziert wird. Die Eingabe der Anweisung "10 A=B" beispielsweise erscheint aufgrund des Abrufes mit dem Befehl FETCH als "0010 LET A=B" im Display.

Ergibt sich beim Editing durch das System eine Zeile mit mehr als 80 Zeichen, werden alle nicht signifikanten Leerstellen eliminiert. Enthält die Zeile dann immer noch mehr als 80 Zeichen, erscheint beim Aufruf die Fehlermeldung ERROR 213. Die Zeile kann mit RECALL auf den Bildschirm zurückgerufen werden, wobei jedoch nur die ersten 80 Zeichen übernommen werden.

- Zeilen eines mit SECURE geschützten Programmes können mit FETCH nicht abgerufen werden.
- Das Schlüsselwort kann durch Betätigen der Taste FETCH eingegeben werden.

BEISPIELE:

```
OLD TEST
LIST
FILE      TEST


0010 REM **BEISPIEL "DELETE LINE " **
0020 REM
0030 FOR I=1 TO 5 STEP 1
0040 PRINT I,
0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;
0070 NEXT J
0080 PRINT
0090 NEXT I
0100 END

END OF LISTING



FETCH 30
0030 FOR I=1 TO 5 STEP 1



LIST 50,60
FILE      TEST


0050 FOR J=1 TO 4 STEP 1
0060 PRINT J;

END OF LISTING



FETCH
0060 PRINT J;




FETCH 5
0100 END



FETCH 71
0070 NEXT J
```

LBCLOSE

BEFEHL: LBCLOSE (library close)

FUNKTION: Schließen von offenen Bibliotheken.

FORMAT: LBC[LOSE] $\left\{\begin{matrix}\text{lib ref} \\ * \end{matrix}\right\}$

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Diskstation anzugeben.

"*" Schließen aller offenen Bibliotheken.

WIRKUNG: Die durch "lib ref" bezeichnete Bibliothek wird geschlossen. Die Verwendung des Parameters "*" bewirkt die Schliessung aller momentan auf irgendeiner Disk offenen Bibliotheken.

BEMERKUNG: Ist die angegebene Bibliothek auf der spezifizierten Platteneinheit schon geschlossen oder nicht vorhanden, erfolgt eine entsprechende Fehlermeldung.

BEISPIELE:

```
LBC (CH2,D2)
```

Die Bibliothek CH2 auf der Disk F2 wird geschlossen.

```
LBC (CH3,D1)
ERROR 178
```

Auf der Disk F1 ist keine offene Bibliothek des Namens CH3 vorhanden.

BEFEHL: LBOPEN (library open)

FUNKTION: Öffnen einer Bibliothek.

FORMAT: LBO[PEN] lib ref[,password]

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name Diskstation anzugeben.

"password" Ist die Bibliothek mit einem Password versehen, kann sie nur mit dem entsprechenden Kennwort geöffnet werden.

WIRKUNG: Die angegebene Bibliothek wird geöffnet und für die Bearbeitung freigegeben.

BEMERKUNGEN:
- Gleichzeitig können maximal sechs Bibliotheken geöffnet sein. Wird der Befehl LBOPEN gegeben, obwohl schon sechs Bibliotheken offen sind oder die verlangte Bibliothek auf der genannten Platteneinheit nicht vorhanden ist, erfolgt eine Fehlermeldung.
- Befindet sich die Bibliothek auf der Systemdisk, muß die Platteneinheit im Parameter "lib ref" nicht angegeben werden.
- Für alle sich auf ein oder mehrere Files beziehenden Systembefehle muß die entsprechende Bibliothek offen sein (z.B. RUN, SAVE, CATALOG).
- Mit dem Befehl LVT* kann festgestellt werden, welche Bibliotheken offen sind.

BEISPIELE:

```
LBO (HELP,F1)

LBO (HELP1,D1),PASS2
ERROR 217

LBO FRANC
ERROR 218
```

falsches Password

Bibliothek "FRANC" befindet sich nicht auf der Systemdisk

LBRESTORE

BEFEHL: LBRESTORE (library restore)

FUNKTION: Die momentan offenen Bibliotheken werden geschlossen und an deren Stelle die als "Standard" (LBSTORE) definierten Bibliotheken geöffnet.

FORMAT: LBR[ESTORE]

WIRKUNG: Alle offenen Bibliotheken werden geschlossen und die mit LBSTORE festgelegten Bibliotheken an deren Stelle geöffnet. Fehlt eine Standard-Definition, wird die erste Bibliothek auf der Systemdisk geöffnet, sofern diese nicht mit einem Password geschützt ist.

BEMERKUNGEN:
- Existiert weder eine Standard-Festlegung (LBSTORE) noch eine Bibliothek auf der Systemdisk, so sind anschließend keine offenen Bibliotheken mehr vorhanden (gleiche Wirkung wie LBC*).
- Mit LVT* ist feststellbar, welche Bibliotheken mit LBSTORE als "Standard" festgelegt wurden.

BEISPIEL:

```
LVT*
     OPEN LIBRARIES :
                          NAME          DISK UNIT
                          P6FSYS        F1

     STORED LIBRARIES :
                          NAME          DISK UNIT
                          P6FSYS        F2
                          P6FSYS        F1

LBR

LVT*
     OPEN LIBRARIES :
                          NAME          DISK UNIT
                          P6FSYS        F2
                          P6FSYS        F1

     STORED LIBRARIES :
                          NAME          DISK UNIT
                          P6FSYS        F2
                          P6FSYS        F1
```

LBSTORE

BEFEHL: LBSTORE (library store)

FUNKTION: Die Namen der im Moment offenen Bibliotheken werden auf der Systemdisk gespeichert und diese in Zukunft automatisch geöffnet.

FORMAT: LBS[TORE]

WIRKUNG: Die Namen der zur Zeit offenen Bibliotheken sind nach Ausführung dieses Befehles auf der Systemdisk gespeichert. Bei jeder Initialisierung des Systems werden dann die in dieser Liste enthaltenen Bibliotheken automatisch geöffnet. Die Liste der Bibliotheksnamen wird ausgedruckt. Sie werden als Standard-Bibliotheken bezeichnet.

BEMERKUNGEN:

- Das System wird initialisiert durch
  . das Einschalten der Anlage oder Drücken der RESET-Taste,
  . die Ausführung des Befehles CONFIGURE oder OPTIONS.
- Ist bei Ausführung von LBS keine Bibliothek offen, wird auch bei der Initialisierung des Systems keine Bibliothek geöffnet. Wurde LBS noch nie verwendet, so öffnet das System nach einer Initialisierung die erste nicht mit einem Password geschützte Bibliothek.
- Ist eine Bibliothek mit einem Password versehen und wird sie mit LBSTORE als "Standard" definiert, so wird sie in Zukunft ebenfalls automatisch geöffnet, das heißt, die Absicherung mittels Password entfällt.
- Durch den Befehl LVT* kann die Liste der offenen Bibliotheken ausgedruckt werden.

BEISPIEL:

```
LVT*
     OPEN LIBRARIES :
                              NAME            DISK UNIT
                              P6FSYS          F2

     STORED LIBRARIES :
                              NAME            DISK UNIT
                              P6FSYS          F2


LBO (,F1)

LVT*
     OPEN LIBRARIES :
                              NAME            DISK UNIT
                              P6FSYS          F2
                              P6FSYS          F1

     STORED LIBRARIES :
                              NAME            DISK UNIT
                              P6FSYS          F2

LBS
NAME                PERIPHERAL
P6FSYS              F2
P6FSYS              F1

LVT*
     OPEN LIBRARIES :
                              NAME            DISK UNIT
                              P6FSYS          F2
                              P6FSYS          F1

     STORED LIBRARIES :
                              NAME            DISK UNIT
                              P6FSYS          F2
                              P6FSYS          F1
```

LDIMAGE

BEFEHL: LDIMAGE (load image)

FUNKTION: Ausgabe eines gespeicherten Bildes auf den Bildschirm

FORMAT LDI[MAGE] filename [,lib ref]

"filename" Name des das Bild enthaltenden Files.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.

WIRKUNG: Das in einem Datenfile gespeicherte Bild wird auf den Bildschirm übertragen. Der alte Bildschirminhalt wird dadurch nicht gelöscht.

BEMERKUNGEN:
- Fehlt der Parameter "filename", so sucht das System das File mit dem Namen SYSPLO.
- Die Ausführung von LDIMAGE kann durch EXIT oder die Funktion BREAK abgebrochen werden.
- Das System muß mit der Option GDI initialisiert sein.
- LDIMAGE überträgt auch Bilder, die ursprünglich mit der Option PLOT für den Thermodrucker erstellt wurden. Stimmt deren Format jedoch mit den Abmessungen des Bildschirmes nicht überein, entsteht ein reduziertes oder undefiniertes Bild.

LDKEYS

BEFEHL: LDKEYS (load keys)

FUNKTION: Wiederherstellung der Standardbelegung der Funktionstasten.

FORMAT: LDK[EYS]

WIRKUNG: Die individuelle Belegung der Funktionstasten wird wieder durch die Standardbelegung ersetzt (siehe auch STKEY).

BEISPIELE: Beispiel 1

Standardbelegung:	F1: STANDARD F2: BELEGUNG
Systembefehl CAL	Übergang in den Calc.-Mode
`FKEY#1,MODIFIZIERTE`	Die Belegung "STANDARD" der Taste F1 wird durch den String "MODIFIZIERT" ersetzt.
Drücken von F1 und F2:	Im Display steht: MODIFIZIERTE BELEGUNG
Eingabe von LDKEYS	Die Standardbelegung wird wieder hergestellt.
Drücken von F1 und F2:	Im Display steht: STANDARDBELEGUNG

Beispiel 2

Gleiche Standardbelegung wie in Beispiel 1:

```
10 FKEY#1, AENDERUNG
20 FKEY#2, DURCH PROGRAMM
30 END


RUN
```

Eingabe eines Programmes zur Modifikation.

Start der Modifikation

Drücken von F1 und F2:	Im Display steht: ÄNDERUNG DURCH PROGRAMM
LDKEYS	Wiederherstellung der Standard-belegung.
Drücken von F1 und F2:	Im Display steht: STANDARDBELEGUNG

BEFEHL: LINK

FUNKTION: Integration eines Programmteiles oder einer Funktionsdefinition, die als Textfile auf einer Disk oder Diskette gespeichert ist, ins Programm im Hauptspeicher.

FORMAT: LIN[K] filename,[lib ref],Zeilennummer[, α]

"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.
"Zeilennummer"	Zeilennummer, ab welcher eingefügt werden soll.
"α"	Großbuchstabe zur Angabe des Namens, unter dem die Funktionsdefinition ins Programm eingefügt werden soll.

WIRKUNG: Ist der Operand α angegeben, wird die auf einer Disk oder Diskette als Textfile unter "filename" gespeicherte Funktionsdefinition ins Programm im Hauptspeicher eingefügt. Der ersten Zeile der Funktionsdefinition wird die im LINK-Befehl angegebene Zeilennummer zugewiesen. Alle folgenden Zeilennummern werden entsprechend der Schrittweite der Funktionsdefinition umnumeriert. Das System weist der Funktion den Namen FNα oder, bei Stringfunktionen, FNα$ zu.

Ist nur der Operand "Zeilennummer" bekannt, wird der als Textfile unter "filename" auf einer Disk oder Diskette gespeicherte Programmteil ins bestehende Programm eingefügt. Die erste Zeile des Einschubes erhält die im LINK-Befehl angegebene Zeilennummer, die folgenden Zeilen werden entsprechend der Schrittweite des Einschubprogrammes umnumeriert.

Ist n1 die niedrigste Zeilennummer des Textfiles, n2 dessen höchste und m die im LINK-Befehl angegebene Zeilennummer, so darf das Programm im Arbeitsspeicher keine Zeile mit einer Nummer im Bereich von m bis m+n2-n1 enthalten.

BEMERKUNGEN:

- Durch den Befehl SHIFT an der entsprechenden Stelle kann, wenn notwendig, für die Zeilen eines einzufügenden Programmteiles oder einer Funktionsdefinition Platz geschaffen werden.
- Das Programm im Arbeitsspeicher darf nicht mit SECURE gesichert sein.
- Nach Ausführung des LINK-Befehles steht das neue, verarbeitbare Programm im Arbeitsspeicher zur Verfügung.
- Es ist darauf zu achten, daß das Programm nach Ausführung des LINK-Befehles nur ein END-Statement aufweist.

BEISPIELE:

Beispiel 1

Anfügen einer Funktionsdefinition an ein im Hauptspeicher vorhandenes Programm

```
FILE     FUNK1


0100 DEF FNB
0105 PRINT "  I","SQR(I)"
0110 PRINT
0115 FOR I=1 TO N STEP 1
0120 IF INT(SQR(I))<>SQR(I) THEN 130
0125 PRINT I,SQR(I)
0130 NEXT I
0135 FOR I=1 TO 10 STEP 1
0140 PRINT
0145 NEXT I
0150 LET FN*=0
0155 FNEND

END OF LISTING
```

Funktionsdefinition (Textfile)

```
FILE     LINK1


0020 DISP "N=";
0040 INPUT N
0060 LET Y=FNA
0080 GOTO 100
0100 DISP "WEITER";
0120 INPUT X
0140 IF X=1 THEN 100
0160 END

END OF LISTING
```

Hauptprogramm

```
OLD LINK1
LINK FUNK1,(P6FSYS,F2),200,A
```

Zusammenfügen mit LINK

```
FILE     LINK1


0020 DISP "N=";
0040 INPUT N
0060 LET Y=FNA
0080 GOTO 100
0100 DISP "WEITER";
0120 INPUT X
0140 IF X=1 THEN 100
0160 END
0200 DEF FNA
0205 PRINT "  I","SQR(I)"
0210 PRINT
0215 FOR I=1 TO N STEP 1
0220 IF INT(SQR(I))<>SQR(I) THEN 230
0225 PRINT I,SQR(I)
0230 NEXT I
0235 FOR I=1 TO 10 STEP 1
0240 PRINT
0245 NEXT I
0250 LET FN*=0
0255 FNEND

END OF LISTING
```

Neues Hauptprogramm nach Ausführung von LINK

```
DELETE LINE 160
260 END
```

Korrektur der END-Zeile

Beispiel 2

Einfügen eines Unterprogrammes in ein im Hauptspeicher vorhandenes Programm

```
FILE     HAUPT1


0100 DISP "N=";
0110 INPUT N
0120 GOSUB 150
0130 GOTO 160
0150 DISP "WEITER";
0160 INPUT X
0170 IF X=1 THEN 100
0180 END

END OF LISTING
```

Hauptprogramm

```
FILE     FUNK2


0100 PRINT "  I","SQR(I)"
0110 PRINT
0120 FOR I=1 TO N STEP 1
0130 IF INT(SQR(I))<>SQR(I) THEN 150
0140 PRINT I,SQR(I)
0150 NEXT I
0160 FOR I=1 TO 10 STEP 1
0170 PRINT
0180 NEXT I
0190 RETURN

END OF LISTING
```

Unterprogramm (Textfile)

```
OLD HAUPT1
SHIFT 150,1000
LINK FUNK2,(,F2),140
```

Zusammenfügen

```
FILE     HAUPT1


0100 DISP "N=";
0110 INPUT N
0120 GOSUB 1150
0130 GOTO 1160
0140 PRINT "  I","SQR(I)"
0150 PRINT
0160 FOR I=1 TO N STEP 1
0170 IF INT(SQR(I))<>SQR(I) THEN 190
0180 PRINT I,SQR(I)
0190 NEXT I
0200 FOR I=1 TO 10 STEP 1
0210 PRINT
0220 NEXT I
0230 RETURN
1150 DISP "WEITER";
1160 INPUT X
1170 IF X=1 THEN 100
1180 END

END OF LISTING
```

Neues Hauptprogramm nach Ausführung von LINK

BEFEHL: LIST

FUNKTION: Ausdruck einer oder mehrerer Zeilen eines im Arbeitsspeicher vorhandenen Programmes oder Textfiles.

FORMAT:

$$\text{LIS[T][Zeilennummer}_1\text{]}\left[,\left\{\begin{array}{l}\text{[Zeilennummer}_2\text{],X}\\ \text{[Zeilennummer}_2\text{]}\end{array}\right\}\right]$$

"Zeilennummer$_1$" Nummer der zu druckenden Zeile bzw. der ersten Zeile des zu druckenden Absatzes.

"Zeilennummer$_2$" Nummer der letzten Zeile des zu druckenden Absatzes.

"X" unterdrückt den Ausdruck der Zeilennummern bei der Ausgabe eines Textfiles.

WIRKUNG:

- Die im Arbeitsspeicher vorhandenen Programm- oder Textzeilen, deren Zeilennummer nicht kleiner als "Zeilennummer1" und nicht größer als "Zeilennummer2" ist, werden ausgedruckt.
- Fehlen die beiden letzten Operanden, beginnt der Ausdruck der im Arbeitsspeicher vorhandenen Programm- oder Textzeilen ab der im ersten Operanden angegebenen Zeile.
- Fehlen der 1. und der 3. Operand, werden alle Zeilen mit einer kleineren oder gleichen Nummer als im 2. Operanden angegeben, ausgedruckt.
- Fehlen alle Operanden, wird das ganze Programm bzw. Textfile ausgedruckt.
- Der Operand "X" unterdrückt den Ausdruck der Zeilennummern eines Textfiles.

BEMERKUNGEN:

- Mit dem Befehl LIST abgerufene Zeilen werden in einem Standardformat gedruckt. Die Zeilennummer wird vierstellig, mit führenden Nullen, ausgegeben. Die Eingabe "40 A=B" wird beispielsweise als "0040 LET A=B" ausgedruckt. Ergeben sich dadurch Zeilen mit mehr als 80 Zeichen, werden diese komprimiert, indem nicht signifikante Leerstellen unterdrückt werden. Weist die Zeile danach immer noch mehr als 80 Zeichen auf, erfolgt der Ausdruck in zwei Zeilen, wobei die zweite Zeile um vier Stellen (Zeilennummer) eingerückt beginnt.

- Diese Editing-Operationen erfolgen auch, wenn ein Programm mit dem Befehl DECOMPILE in ein Textfile umgewandelt wird.
- Ist ein Programm mit SECURE geschützt, kann es mit LIST nicht ausgedruckt werden.
- Das Schlüsselwort des Befehles kann über die Taste OLD/LIST eingegeben werden.
- Die Ausgabe kann mit BREAK vorzeitig abgebrochen werden.
- Programmfiles können nicht ohne Zeilennummern (Parameter X) ausgedruckt werden.

BEISPIELE:

```
LIST
FILE


0010 ........
0020 BEISPIELE FUER "LIST"
0030 ........
0040
0050 VORLAGE  TEXTFILE
0060
0070 TEILWEISER ODER VOLLSTAENDIGER
0080 AUSDRUCK.
0090
0100 AUSGABE MIT ODER OHNE
0110 ZEILENNUMMER.
0120 (LETZTERES GILT NICHT FUER
0130 PROGRAMME)
0140
0150 ........

END OF LISTING
```

```
LIST 70
FILE


0070 TEILWEISER ODER VOLLSTAENDIGER
0080 AUSDRUCK.
0090
0100 AUSGABE MIT ODER OHNE
0110 ZEILENNUMMER.
0120 (LETZTERES GILT NICHT FUER
0130 PROGRAMME)
0140
0150 ........

END OF LISTING
```

```
LIST 70,110
FILE


0070 TEILWEISER ODER VOLLSTAENDIGER
0080 AUSDRUCK.
0090
0100 AUSGABE MIT ODER OHNE
0110 ZEILENNUMMER.

END OF LISTING
```

```
LIST ,80
FILE


0010 ........
0020 BEISPIELE FUER "LIST"
0030 ........
0040
0050 VORLAGE: TEXTFILE
0060
0070 TEILWEISER ODER VOLLSTAENDIGER
0080 AUSDRUCK.

END OF LISTING
```

```
LIST ,,X
FILE


 ........
 BEISPIELE FUER "LIST"
 ........

 VORLAGE: TEXTFILE

 TEILWEISER ODER VOLLSTAENDIGER
 AUSDRUCK.

 AUSGABE MIT ODER OHNE
 ZEILENNUMMER.
 (LETZTERES GILT NICHT FUER
 PROGRAMME)

 ........

END OF LISTING
```

BEFEHL: LVTOC (library volume table of contents)

FUNKTION: Druck eines Verzeichnisses der Bibliotheken einer Einheit oder aller offenen Bibliotheken.

FORMAT: LVT[OC] [{unit name / *}]

"unit name" Name der entsprechenden Disk-Einheit.

"*" Ausgabe der Liste aller offenen sowie der Standard-Bibliotheken.

WIRKUNG: Durch die Angabe des Parameters "unit name" werden die vorhandenen Bibliotheken mit folgenden Angaben gelistet:

- Überschrift mit Release-Code des Betriebssystems, Volume-Label, Sektorlänge und Datum (Befehl DATE).
- Für jede Bibliothek:
  - Name der Bibliothek (LIBRARY)
  - Erstellungsdatum (CREATE)
  - Physische Anfangs- und Endadresse durch Angabe von Zylinder (erste 3 Ziffern) und Sektor (letzte 3 Ziffern)
  - Anzahl der reservierten Sektoren (SECTORS)
  - Anzahl der noch freien Sektoren innerhalb der Bibliothek (EMPTY SECTORS)
- Gesamtzahl der noch freien Sektoren dieser Einheit (FREE SPACE) nach Ausführung des Dienstprogrammes RESTRUCT.

Bei Verwendung des Parameters "*" werden alle im Moment offenen und die Liste der mit LBS registrierten Standard-Bibliotheken ausgedruckt.

BEMERKUNG:
- Wird kein Parameter angegeben, erfolgt der Druck des Bibliothekskataloges der Systemdisk.

BEISPIEL:

```
LVT F1
     MDOSC -R 2.0 * VOLLABEL=        * TRACK FORMAT=256 BYTE * DATE:03-01-83

     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
     OSLIB    27-10-82    01001         16052        832
     L1ESEF   27-10-82    17001         17014        14
     L1ESES   27-10-82    17015         32024        790
     HELP     03-01-83    32025         52008        1024       34
     DATEN    03-01-83    52009         59044        400        388
FREE SPACE (TOTAL) = 788   SECTORS



LVT F2
     MDOSC -R 2.0 * VOLLABEL=L1/ESE * TRACK FORMAT=256 BYTE * DATE:03-01-83

     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
     D6435    07-06-82    01001         15044        772        15
     D6432    07-06-82    15045         32032        872        23
     P6441    07-06-82    32033         41012        448        11
     D64351   07-06-82    41013         56024        792        40
     D64321   07-06-82    56025         73012        872        23
     D6411    07-10-82    73013         74040        80         0
FREE SPACE (TOTAL) = 12    SECTORS



LVT*
     OPEN LIBRARIES :
                         NAME          DISK UNIT
                         DATEN         F1
                         P6441         F2
                         HELP          F1
                         D6411         F2

     STORED LIBRARIES :
                         NAME          DISK UNIT
                         DATEN         F1
                         P6441         F2
```

MERGE

BEFEHL: MERGE

FUNKTION: Kombination eines sich im Arbeitsspeicher befindlichen Textes mit einem Textfile einer Bibliothek.

FORMAT: MER[GE] filename,[lib ref],[Zeilennummer],[/]

"filename"	Name des Textfiles in der Bibliothek.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.
"Zeilennummer"	Zeilennummer, ab der ein Textfile eingefügt werden soll.
"/"	Überschriebene Zeilen sollen nicht angezeigt werden.

WIRKUNG: Sind alle Parameter angegeben, wird das Textfile der mit "lib ref" spezifizierten Bibliothek in den Text im Arbeitsspeicher integriert. Das einzufügende Textfile beginnt mit der angegebenen Zeilennummer und erhält für die weiteren Zeilennummern die Schrittweite des Originales. Entsteht dabei eine bereits bestehende Zeilennummer, wird diese überschrieben.

Fehlt der letzte Parameter (/), werden überschriebene Zeilen ausgedruckt nach dem Muster:

PREVIOUS LINE
PRESENT LINE

worin #### durch die entsprechende Zeilennummer ersetzt wird.

Fehlt der Parameter "Zeilennummer", wird das Textfile der Bibliothek mit der Originalnumerierung übernommen.

BEISPIELE: Beispiel 1

```
FILE     BASIS


0010 ..........
0020 BEISPIELE FUER 'MERGE'
0030 ..........
0040
0050 ZEILE 1 - BASISPROGRAMM
0060 ZEILE 2 - BASISPROGRAMM
0070 ZEILE 3 - BASISPROGRAMM
0080
0090 ..........

END OF LISTING


FILE     EIN


0005 ----------
0010 ZEILE 1 - EINSCHUB
0015 ZEILE 2 - EINSCHUB
0020 ZEILE 3 - EINSCHUB
0025 ----------

END OF LISTING


OLD BASIS
MERGE EIN,,40


REPLACED LINES:
#### PREVIOUS LINE
#### PRESENT LINE

0040
0040 ----------

0050 ZEILE 1 - BASISPROGRAMM
0050 ZEILE 2 - EINSCHUB

0060 ZEILE 2 - BASISPROGRAMM
0060 ----------



LIST
FILE     BASIS


0010 ..........
0020 BEISPIELE FUER 'MERGE'
0030 ..........
0040 ----------
0045 ZEILE 1 - EINSCHUB
0050 ZEILE 2 - EINSCHUB
0055 ZEILE 3 - EINSCHUB
0060 ----------
0070 ZEILE 3 - BASISPROGRAMM
0080
0090 ..........

END OF LISTING
```

Basisprogramm (Textfile) im Hauptspeicher

Textfile auf Disk oder Diskette

Verknüpfung

Kontrollausdruck der überschriebenen Zeilen

Textfile im Hauptspeicher nach MERGE

Beispiel 2

Verknüpfung ohne Kontrollausdruck überschriebener Zeilen.

```
OLD BASIS
MERGE EIN,,40,/

LIST
FILE      BASIS


0010 ..........
0020 BEISPIELE FUER 'MERGE'
0030 ..........
0040 ----------
0045 ZEILE 1 - EINSCHUB
0050 ZEILE 2 - EINSCHUB
0055 ZEILE 3 - EINSCHUB
0060 ----------
0070 ZEILE 3 - BASISPROGRAMM
0080
0090 ..........

END OF LISTING
```

MODIFY

BEFEHL: MODIFY

FUNKTION: Änderung eines Filenamens und/oder des für ein Datenfile reservierten Speicherplatzes in einer Bibliothek.

FORMAT: MOD[IFY] filename$_1$,[lib ref],[filename$_2$][,n]

"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Diskstation anzugeben.
"filename$_1$"	Name des zu modifizierenden Files.
"filename$_2$"	neuer Filename.
"n"	positive ganze Zahl, die den für das File in der Bibliothek neu zu reservierenden Speicherplatz festlegt.

WIRKUNG: Der vollständige Befehl mit allen vier Operanden bewirkt, daß der alte Name des Files in der Bibliothek durch den neuen Namen ersetzt und der Speicherbereich auf der Disk neu auf n Bytes geändert wird.

- Fehlt der Parameter "n", wird nur der Name des Programm-, Text- oder Datenfiles geändert, nicht aber seine Kapazität.
- Fehlt der Parameter "filename 2", behält das File seinen Namen, doch werden neu n Bytes auf der Disk für das File reserviert.

BEMERKUNGEN:
- Mit MODIFY geänderte Filenamen behalten ihre Zugehörigkeit zu einer bestimmten Teilbibliothek. Daher muß die Kennzeichnung + oder * für "filename 2" nicht eingegeben werden.
- Um zu verhindern, daß Daten verloren gehen, kann der für sequentielle Datenfiles reservierte Platz durch MODIFY nur bis zum bereits belegten Speicherplatz verkleinert werden.

- Da das logische und das physische Ende von Random-Files zusammenfallen, kann der für ein Random-File reservierte Speicherplatz nicht verkleinert werden.
- Zwei Files einer Bibliothek dürfen nie den gleichen Namen tragen.
- Bei mit dem Dienstprogramm LBPROTECT geschützten Files von Package- und Common-Teilbibliotheken kann nur die Größe modifiziert werden.

BEISPIEL: Änderung des Namens und der reservierten Länge eines Random-Files.

```
     MDOSC -R 2.0    *  VOLLABEL =          *  LIBRARY = P6FSYS  *  DATE:07-05-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      QUATSH   P  13-03-80  13-03-80    0384       0384                 1
      TEST     P  13-03-80  13-03-80    0384       0384                 1
      TEST1    P  13-03-80  13-03-80    0384       0384                 1
      SYS      P  15-12-78  20-06-79    2432       2432                 1
      YYY      P  13-03-80  13-03-80    0256       0256                 1  C=1000
      TEXT1    P  13-03-80  13-03-80    0256       0256                 1
      FUNK1    T  07-05-80  07-05-80    0512       0512                 1
   ->DATEN     R  07-05-80  07-05-80    0512       0512                 1


MOD DATEN,(,F2),DATEI1,1000


     MDOSC -R 2.0    *  VOLLABEL =          *  LIBRARY = P6FSYS  *  DATE:07-05-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      QUATSH   P  13-03-80  13-03-80    0384       0384                 1
      TEST     P  13-03-80  13-03-80    0384       0384                 1
      TEST1    P  13-03-80  13-03-80    0384       0384                 1
      SYS      P  15-12-78  20-06-79    2432       2432                 1
      YYY      P  13-03-80  13-03-80    0256       0256                 1  C=1000
      TEXT1    P  13-03-80  13-03-80    0256       0256                 1
      FUNK1    T  07-05-80  07-05-80    0512       0512                 1
   ->DATEI1    R  07-05-80  07-05-80    1024       1024                 1
```

NEW

BEFEHL: NEW

FUNKTION: Vorbereitung des Systems für die Eingabe eines Programmes über die Tastatur.

FORMAT: NEW

WIRKUNG: Der Inhalt des Arbeitsspeichers wird gelöscht. Alle nun eingegebenen BASIC-Zeilen interpretiert das System als Teile eines neuen Programmes und kontrolliert sie auf ihre syntaktische Richtigkeit.

BEMERKUNGEN:

- Da der bestehende Inhalt des Arbeitsspeichers durch die Eingabe des Befehles NEW gelöscht wird, muss vor diesem Befehl der Befehl SAVE oder REPLACE stehen, wenn der alte Inhalt gespeichert werden soll.
- Nach dem Laden (bzw. nach einem OPTIONS- oder CONFIGURE-Befehl) des Systems kann ein Programm ohne den Befehl NEW eingetastet werden.

BFFEHL: OLD

FUNKTION: Laden eines Programmes oder Textfiles ab Diskette oder Disk in den Arbeitsspeicher.

FORMAT: OLD filename[,lib ref]

"filename" Name des in den Arbeitsspeicher zu ladenden Programmes oder Textfiles.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.

WIRKUNG: Es wird geprüft, ob das Programm oder Textfile namens "filename" in einer Bibliothek gespeichert ist. Wenn ja, wird es in den Arbeitsspeicher geladen. Der bisherige Inhalt des Arbeitsspeichers wird dadurch gelöscht.

BEMERKUNG: Der Filename wird in den offenen Bibliotheken aller Platteneinheiten in der Reihenfolge der durch LTV* erhaltenen Liste gesucht. Beinhalten also mehrere Bibliotheken ein File dieses Namens, empfiehlt es sich, den Parameter "lib ref" anzugeben.

OPTIONS

BEFEHL: OPTIONS

FUNKTION: Vorwahlbefehl für die Aktivierung bestimmter Teile des Betriebssystems und Neuladen des Systems.

FORMAT: OPT[IONS] [{ GDI / PLO }]

WIRKUNG: Die im Befehl aufgeführten Module (= Teile) des Betriebssystems werden in den Anwenderteil des Hauptspeichers geladen.

- Parameter PLO:
  wird zur Ausführung von Plotanweisungen für den Systemdrucker benötigt (Thermodrucker).
- Parameter GDI:
  erlaubt die Ausgabe von graphischen Darstellungen auf dem Bildschirm.

Die Eingabe des Befehles ohne Parameter bewirkt, daß die Markierung der bisher geladenen Options auf der Systemdisk gelöscht wird. Das System wird ohne Options neu initialisiert.

BEMERKUNGEN:
- Die bei der Initialisierung des Systems mitzuladenden Options sind auf der Systemdisk gespeichert. Bei jeder Neuinitialisierung, das heißt nach dem Einschalten sowie nach CON, OPT oder einem Fehler mit anschließendem Systemabbruch (ERROR n*A), konfiguriert sich das System entsprechend dem zuletzt ausgeführten OPTIONS-Befehl.
- Durch einen OPTIONS-Befehl wird der Inhalt des Arbeitsspeichers gelöscht.
- Die Options werden nur bei der Ausführung von Anweisungen benötigt. Die Eingabe und die syntaktische Kontrolle kann auch bei nicht geladenen Options erfolgen.

- Die Options benötigen zusätzlichen Platz im Anwenderspeicher gemäß nachfolgender Tabelle:

OPTIONS	PLATZBEDARF 1 K = 1024 bytes
PLO	2 K
GDI	2,5 K

- Die folgende Aufstellung gibt Auskunft, wann die beiden Options benötigt werden:

  PLO - Plot:
  Alle BASIC-Anweisungen zur Verwendung des Thermodruckers als Plotter.

  GDI - Graphic Display:
  Alle BASIC-Anweisungen und Befehle für den Bildschirm DSM, mit Ausnahme der BASIC-Anweisungen DISP, ERASE, REVERSE sowie der Debugging- und Systembefehle ERASE, REVERSE, DRAW.

BEFEHL: PREPARE

FUNKTION: Durchführung der vollständigen, syntaktischen Analyse des Programmes im Arbeitsspeicher und Überprüfung von dessen Ausführbarkeit.

FORMAT: PRE[PARE] [filename[,lib ref]]

"filename" Name eines Programmes auf einer Diskette oder Disk.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.

WIRKUNG:
- Das Programm mit dem angegebenen Namen wird in den Hauptspeicher geladen und seine Anweisungen auf ihre Durchführbarkeit überprüft.
- Fehlen beide Operanden, wird die syntaktische Analyse des sich im Arbeitsspeicher befindlichen Programmes durchgeführt.

BEMERKUNGEN:
- Erscheint nach Eingabe des Befehles keine Fehlermeldung, so befindet sich das Programm in ausführbarer Form im Arbeitsspeicher und das System im Debugging-Mode.
- Ist die Konsoltaste PRINT ALL aktiviert, wird folgende Meldung gedruckt:

  ****FORMALLY CORRECT PROGRAM****
  ::ROOM = X::

  worin "X" die Anzahl der freien Bytes im Anwenderteil des Hauptspeichers angibt. Im Display erscheint die Meldung:

  PROGRAM program name READY TO RUN

- Bewirkte die Ausführung des Befehles PREPARE keine Fehlerdung, kann die Durchführung des sich im Arbeitsspeicher befindlichen Programmes mit der Konsoltaste CONTINUE gestartet werden; die Taste STEP erlaubt die schrittweise Verarbeitung des Programms.
- Erkennt das System während der Durchführung des Befehles PREPARE Fehler im Programm, werden die entsprechenden Fehlermeldungen ausgedruckt. In diesem Falle geht das System in den Command-Mode, damit die notwendigen Korrekturen vorgenommen werden können.
- Nicht ausführbare Anweisungen, wie beispielsweise DCL, DIM oder BUFFER #, werden bereits bei der Ausführung des Befehles berücksichtigt. Im Programm angesprochene Files sind vor der Durchführung von PREPARE zu kreieren.
- Wird das Programm nach Ausführung von PREPARE mit Parameter "filename" ohne weitere Änderung abgespeichert, entfällt die vorgängige Ausführung bei weiteren Aufrufen durch RUN, CHAIN oder PREPARE (siehe auch Systembefehl RUN).
- Wird PREPARE ohne Parameter verwendet, so wird in der ersten offenen Bibliothek Platz benötigt. Ist dieser Platz nicht vorhanden, meldet das System ERROR 188.
- Reduzierte Systeme, die PREPARE nicht enthalten, können nur Programme verarbeiten, deren Preexecution vor der Programmspeicherung mit einem vollständigen System ausgeführt wurde (siehe ERROR 63).

BEISPIEL:

```
PRE TEST
**** FORMALLY CORRECT PROGRAM ****
:: ROOM=44082 ::
```

BEFEHL: PROCEDURE

FUNKTION: Abruf einer Katalog-Prozedur zur Verarbeitung.

FORMAT: PRO[CEDURE] [filename,[lib ref],[A],[Zeilennr$_1$][,Zeilennr$_2$]]

"filename"	Name des Commandfiles.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.
"A"	Definition, dass mit automatischem Input gearbeitet wird.
"Zeilennummer$_1$"	erste zu verarbeitende Zeile der Prozedur.
"Zeilennummer$_2$"	letzte zu verarbeitende Zeile der Prozedur.

WIRKUNG: Sind alle Parameter angegeben, wird das durch "filename" und "lib ref" definierte Commandfile verarbeitet. Die Ausführung beginnt mit "Zeilennummer1" und endet nach der Bearbeitung von "Zeilennummer2". Die zur Ausführung von INPUT- bzw. RKB-Statements notwendigen Eingabewerte werden den entsprechenden "IN="-Zeilen der Prozedur entnommen.

Ist "Zeilennummer1" nicht angegeben, so beginnt die Verarbeitung mit der ersten Zeile der Prozedur.

Fehlt "Zeilennummer2", so endet die Arbeit nach der letzten Zeile der Prozedur.

Fehlt die Angabe "A", so hat die Dateneingabe manuell zu erfolgen. In der Prozedur enthaltene "IN="-Zeilen werden ignoriert.

Ist kein Parameter angeführt, so wird die Verarbeitung einer abgebrochenen Prozedur wieder aufgenommen.

Aufbau und spezielle Befehle einer Prozedur werden in Kapitel 6.2 und 6.4 ausführlich beschrieben.

BEMERKUNGEN:

- Jede Zeile der Prozedur wird im Display angezeigt und, wenn PRINT ALL aktiviert ist, ausgedruckt.
- Nach Beendigung der Prozedur befindet sich das System im Command-Mode.
- Mit BREAK kann die Verarbeitung der Prozedur abgebrochen werden. Das System befindet sich danach im Command-Mode.
- Enthält eine Zeile der Prozedur einen syntaktischen Fehler, erscheint eine Fehlermeldung und die Verarbeitung wird abgebrochen.
  . Ist der Fehler behebbar, kann dieser wie üblich im Debugging-Mode behoben und die Verarbeitung der Prozedur danach mit CONTINUE fortgesetzt werden.
  . Ist der Fehler nicht behebbar, kann der Debugging-Mode nur mit BREAK verlassen werden. Die Prozedur ist damit vorzeitig beendet und das System befindet sich im Command-Mode.
- Befindet sich das System aufgrund eines bei der Ausführung der Prozedur aufgetretenen Fehlers im Command-Mode, kann die Verarbeitung der Prozedur durch Eingabe des Befehles PROCEDURE ohne Operanden wieder aufgenommen werden. Begonnen wird in diesem Falle mit der der fehlerhaften Zeile unmittelbar folgenden Zeile.
- Beinhaltet die Prozedur einen OPTIONS- oder CONFIGURE-Befehl, wird sie nach Ausführung des Befehles nicht fortgesetzt.
- Eine Prozedur kann mit PRO-Befehlen andere Prozeduren aufrufen.
- Enthält die Prozedur eine Zeile mit "IF CC = X, Zeilennummer", worin als "Zeilennummer" ein höherer Wert, als der im Aufruf für "Zeilennummer2" definierte, angegeben ist, wird die Prozedur bei dieser Zeile trotzdem fortgesetzt und beendet, wenn entweder die letzte Zeile der Prozedur oder "Zeilennummer2" erreicht ist.
- Tritt bei der Verarbeitung einer Prozedur ein Fehler auf, kann das Datenfile im Arbeitsspeicher mit

  TRA T, filename, [lib ref],#

  in ein Textfile umgewandelt werden. Nach der Korrektur mit den Editing-Möglichkeiten des Textfiles ist es möglich, das Datenfile nach

  PUR filename, [lib ref]

  mit

  TRA D, filename, [lib ref],#

  in der korrigierten Fassung wieder zu speichern. Die Prozedur kann mit PRO fortgesetzt werden.

- Der Befehl TEST innerhalb einer Prozedur ermöglicht das automatische Übergehen fehlerhafter Zeilen (siehe Kapitel 6.4).
- Wird eine Prozedur unter dem Filenamen *SETUP abgespeichert, kann sie bei der Initialisierung des Systems automatisch gestartet werden (siehe Befehl CONFIGURE PRO).

BEISPIELE: Commandfile CMD 1

```
FILE


0010 OLD PYT
0020 LIST
0030 RUN PYT
0040 IN=2.5
0050 IN=7.4
0060 45PRINT"A=";A,"B=";B
0070 RUN
0080 IN=3.5
0090 IN=5.5
0100 LIST

END OF LISTING
```

Beispiel 1:

Aufruf ohne automatischen Input (PRINT ALL aktiviert)

```
PRO CMD1


OLD PYT
LIST
FILE     PYT


0010 DISP "EINGABE VON A";
0020 INPUT A
0030 DISP "EINGABE VON B"
0040 INPUT B
0050 LET C=SQR(A*A+B*B)
0060 PRINT "C=";C
0070 PRINT
0080 PRINT
0090 END

END OF LISTING



RUN PYT
**** FORMALLY CORRECT PROGRAM ****
EINGABE VON A?
2
EINGABE VON B
?
3
C= 3.6055513


45PRINT"A=";A,"B=";B
RUN
**** FORMALLY CORRECT PROGRAM ****
EINGABE VON A?
2
EINGABE VON B
?
3
A= 2            B= 3
C= 3.6055513


LIST
FILE     PYT


0010 DISP "EINGABE VON A";
0020 INPUT A
0030 DISP "EINGABE VON B"
0040 INPUT B
0045 PRINT "A=";A,"B=";B
0050 LET C=SQR(A*A+B*B)
0060 PRINT "C=";C
0070 PRINT
0080 PRINT
0090 END

END OF LISTING
```

Beispiel 2:

Aufruf mit automatischem Input

```
PRO CMD1,,A
OLD PYT
LIST
FILE     PYT


0010 DISP "EINGABE VON A";
0020 INPUT A
0030 DISP "EINGABE VON B"
0040 INPUT B
0050 LET C=SQR(A*A+B*B)
0060 PRINT "C=";C
0070 PRINT
0080 PRINT
0090 END

END OF LISTING



RUN PYT
**** FORMALLY CORRECT PROGRAM ****
EINGABE VON A?
2.5
EINGABE VON B
?
7.4
C= 7.8108898


45PRINT"A=";A,"B=";B
RUN
**** FORMALLY CORRECT PROGRAM ****
EINGABE VON A?
3.5
EINGABE VON B
?
5.5
A= 3.5          B= 5.5
C= 6.5192024


LIST
FILE     PYT


0010 DISP "EINGABE VON A";
0020 INPUT A
0030 DISP "EINGABE VON B"
0040 INPUT B
0045 PRINT "A=";A,"B=";B
0050 LET C=SQR(A*A+B*B)
0060 PRINT "C=";C
0070 PRINT
0080 PRINT
0090 END

END OF LISTING
```

PURGE

BEFEHL: PURGE

FUNKTION: Löschen eines Files in einer Bibliothek.

FORMAT: PUR[GE] filename [,lib ref]

"filename"	Name des zu löschenden Files.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Diskstation anzugeben.

WIRKUNG: Das System prüft, ob in der durch "lib ref" definierten Bibliothek ein File namens "filename" vorhanden ist. Wenn ja, wird dieses File gelöscht. Der dadurch frei gewordene Platz kann für die Speicherung weiterer Files wieder verwendet werden.

BEMERKUNGEN:
- Ist der Filename in verschiedenen Bibliotheken vorhanden und der Parameter "lib ref" nicht definiert, so wird das File mit dem entsprechenden Namen in der ersten geöffneten Bibliothek (LVT*) gelöscht.
- Für Files von mit dem Dienstprogramm LBPROTECT geschützten Package- oder Common-Teilbibliotheken ist der Befehl unwirksam.
- Die mit "lib ref" angesprochene Bibliothek muß offen sein.

BEISPIEL:

```
PUR TEST1
PUR FUNK1,(,F2)
```

REPLACE

BEFEHL: REPLACE

FUNKTION: Ersetzen eines Programmes oder Textfiles durch den Inhalt des Arbeitsspeichers. Der Name bleibt dabei erhalten.

FORMAT REP[LACE] [lib ref]

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.

WIRKUNG: Kontrolle, ob das Programm oder der Text im Arbeitsspeicher einen Namen hat, und ob in der durch "lib ref" definierten Bibliothek ein Programm oder Textfile unter demselben Namen gespeichert ist. Sind diese beiden Bedingungen erfüllt, wird das gespeicherte Programm oder Textfile durch dasjenige im Arbeitsspeicher ersetzt.

BEMERKUNGEN:
- Der Befehl ermöglicht, ein gespeichertes Programm oder Textfile durch eine beliebig modifizierte Fassung zu ersetzen. Nach Ausführung des Befehles ist die modifizierte Fassung unter dem ursprünglichen Namen des Programmes oder Textfiles gespeichert.
- Das Programm oder Textfile im Arbeitsspeicher hat nur dann einen Namen, wenn es von einer Disk oder Diskette geladen wurde, das heißt, nach Ausführung der Befehle "OLD filename", "RUN filename" und "PREPARE filename".
- Ein neu eingegebenes Programm oder Textfile kann nur mit dem Befehl SAVE abgespeichert werden. Soll ein unter dem Namen "filename" gespeichertes Programm oder Textfile durch ein neu eingegebenes ersetzt werden, heißt die notwendige Befehlsfolge:

  PUR filename [,lib ref]/SAVE filename [,lib ref]

- Ist der Filename in verschiedenen Bibliotheken vorhanden und "lib ref" nicht definiert, so wird durch REPLACE das Programm bzw. Textfile in der ersten offenen Bibliothek ersetzt (LVT*).

- Die mit "lib ref" spezifizierte Bibliothek muß offen sein.
- Soll der im Common-Bereich reservierte Platz geändert werden, muß anstelle von REPLACE die Befehlsfolge "PURGE filename [,lib ref]/SAVE filename, [lib ref][,COM=M]" verwendet werden.
- Wird ein Programm oder Textfile durch die Änderung vergrößert, ist ein REPLACE in folgenden zwei Fällen nicht möglich:

  1. In der angesprochenen Bibliothek ist kein Platz mehr vorhanden.
  2. Die freien Bereiche in der Bibliothek können nur genutzt werden, indem das File in mehr als vier Teilen abgelegt wird (siehe CATALOG). Eine solche Bibliothek kann bei Bedarf mit LIBCOPY neu organisiert werden.

  In beiden Fällen meldet das System "ERROR code FILE filename PURGED", das heißt: das ursprüngliche File auf der Diskette oder Disk ist zerstört. Das geänderte File befindet sich noch im Arbeitsspeicher.

RESEQUENCE

BEFEHL: RESEQUENCE

FUNKTION: Umnumerierung der Zeilen eines im Arbeitsspeicher vorhandenen Programmes oder Textfiles.

FORMAT: RES[SEQUENCE][Zeilennummer][,Schrittweite]

"Zeilennummer" bestimmt die der ersten Zeile des sich im Arbeitsspeicher befindlichen Programmes oder Textfiles zuzuweisende Zeilennummer.

"Schrittweite" positive ganze Zahl, welche die jeweilige Erhöhung der Numerierung bestimmt.

WIRKUNG: Das System weist der ersten Zeile des Programmes oder Textfiles die durch den ersten Operand angegebene Zeilennummer zu. Jede folgende Zeile erhält eine jeweils um die Schrittweite erhöhte Zeilennummer.

- Ist nur der erste Operand angegeben, wird der ersten Zeile des Programmes oder Textfiles die durch diesen spezifizierte Nummer zugeteilt und als Schrittweite 10 angenommen.
- Wird nur der zweite Operand definiert, erhält die erste Zeile des Programmes oder Textfiles den Wert der Schrittweite als Zeilennummer. Jeder folgenden Zeile wird eine jeweils um die Schrittweite erhöhte Zeilennummer zugewiesen.
- Fehlen beide Operanden, erhält die erste Zeile die Zeilennummer 10. Die Schrittweite beträgt ebenfalls 10.

BEMERKUNGEN:
- Bei der Umnumerierung eines Programmes modifiziert das System automatisch auch alle innerhalb von BASIC-Anweisungen vorkommenden Hinweise auf Zeilen (z.B. GOTO, IF...THEN, PRINT USING).
- Enthält ein Programm aufgrund von Korrekturen Hinweise auf gelöschte Zeilen, so wird bei RESEQUENCE die fehlende Zeile intern mitnumeriert, das heißt: in der fortlaufenden Numerierung der effektiv vorhandenen Zeilen entsteht eine Lücke, die nur mit RES nach DEC/COM vermieden werden kann.

BEISPIEL:

```
FILE


0010 DCL 80 A$
0020 DIM P(3,12)
0030 FOR I=1 TO 10 STEP 1
0040 INPUT P(1,I),P(2,I),P(3,I)
0050 PRINT USING 77,P(1,I),P(2,I),P(3,I)
0060 NEXT I
0070 REM
0077 : #####.##   #####.##   ###.###
0080 INPUT P(1,11),P(1,12)
0090 GOSUB 105
0100 REM
0105 REM: UNTERPROGRAMM1
0110 FOR I=1 TO 12 STEP 1



RES

FILE


0010 DCL 80 A$
0020 DIM P(3,12)
0030 FOR I=1 TO 10 STEP 1
0040 INPUT P(1,I),P(2,I),P(3,I)
0050 PRINT USING 80,P(1,I),P(2,I),P(3,I)
0060 NEXT I
0070 REM
0080 : #####.##   #####.##   ###.###
0090 INPUT P(1,11),P(1,12)
0100 GOSUB 120
0110 REM
0120 REM: UNTERPROGRAMM1
0130 FOR I=1 TO 12 STEP 1




RES 5,20

FILE


0005 DCL 80 A$
0025 DIM P(3,12)
0045 FOR I=1 TO 10 STEP 1
0065 INPUT P(1,I),P(2,I),P(3,I)
0085 PRINT USING 145,P(1,I),P(2,I),P(3,I)
0105 NEXT I
0125 REM
0145 : #####.##   #####.##   ###.###
0165 INPUT P(1,11),P(1,12)
0185 GOSUB 225
0205 REM
0225 REM. UNTERPROGRAMM1
0245 FOR I=1 TO 12 STEP 1




END OF LISTING
```

Grundprogramm

Umnumerierung in 10er-Schritten

Umnumerierung mit Intervall 20 ab Zeile 5

REVERSE

BEFEHL: REVERSE

FUNKTION: Umkehrung des Bildes auf dem Bildschirm von positiv in negativ und umgekehrt.

FORMAT: REV[ERSE]

WIRKUNG: Darstellungen auf dem Bildschirm werden von positiv in negativ umgewandelt und umgekehrt.

BEMERKUNGEN:
- Die Hardcopy mit DRAW auf dem Thermoprinter erscheint immer als positiv.
- Der Befehl hat immer eine Umkehrwirkung, gleichgültig, ob von einem Programm ein positives oder negatives Bild erstellt wurde.

BEFEHL: RUN

FUNKTION: Start der Programmausführung.

FORMAT: RUN [filename[,lib ref] | Zeilennummer]

"filename" Name des zu verarbeitenden Files.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Name der Bibliothek, als "unit name" der Name der Diskstation anzugeben.

"Zeilennummer" Zeile, mit welcher die Verarbeitung beginnen soll.

WIRKUNG:
- Wird der Operand "filename" eingegeben, überprüft das System, ob ein File dieses Namens in der durch "lib ref" angesprochenen Bibliothek gespeichert ist. Die Suche nach diesem File beginnt immer in der ersten offenen Bibliothek (LVT*), sofern der Parameter "lib ref" nicht definiert ist. Ist das File ein Programm oder Textfile, wird es in den Arbeitsspeicher geladen. Handelt es sich um ein Programm, wird dieses ausgeführt. Andernfalls erscheint eine Fehlermeldung.
- Besteht die Eingabe aus dem Befehl "RUN" oder "RUN Zeilennummer", wird geprüft, ob sich ein Programm im Arbeitsspeicher befindet. Wenn nicht, erfolgt eine Fehlermeldung. Wenn ja, wird das Programm ab der definierten Zeilennummer ausgeführt.

BEMERKUNGEN:
- Wird "RUN Zeilennummer" eingegeben, beginnt die Ausführung mit der angegebenen Zeile. Die Zeilennummer darf sich jedoch nicht auf eine Anweisung innerhalb einer FOR/NEXT-Schleife oder einer Multi-Line-Funktion beziehen.
- Nach Eingabe eines RUN-Befehles prüft das System, ob eine Preexecution durchzuführen ist. Unter "Preexecution" versteht man eine Prüfung des Programmes durch das System

Eine Preexecution wird nur durchgeführt, wenn:
. ein neu eingegebenes Programm zum ersten Mal ausgeführt werden soll;
. bei der letzten Preexecution ein Fehler festgestellt wurde;
. ein Programm nach dem letzten RUN- oder PREPARE-Befehl geändert worden ist;
. das nach "RUN filename" gestartete Programm ohne vorgängige, fehlerfreie Preexecution gespeichert worden ist.

- Eine erneute Preexecution eines Programmes beim Aufruf durch "RUN filename" unterbleibt, wenn folgende Befehle ausgeführt werden:
  . OLD filename [,lib ref]
  . PREPARE der Funktion
  . Ausführung der Funktion BREAK nach fehlerfreier Preexecution
  . REPLACE [,lib ref]

- Tritt während der Preexecution ein Fehler auf, wird dieser gemeldet und das Programm nicht ausgeführt. Fehlermeldungen, die sich auf fehlende Files, Platzprobleme bezüglich der Disk oder den nötigen Platzbedarf im Anwenderspeicher beziehen, erfolgen erst nach Abschluß der Preexecution, das heißt, die Rückspeicherung nach BREAK ist trotzdem möglich.

- Ist die Taste PRINT ALL aktiviert, wird nach der fehlerfreien Preexecution die Meldung "**** FORMALLY CORRECT PROGRAM****" ausgegeben.

- Wird ein durch die Preexecution bereits als fehlerfrei bezeichnetes Programm erneut gestartet, unterbleibt eine erneute Preexectuion.

- Zwecks Optimierung wird das Programm vor der eigentlichen Verarbeitung in eine Form gebracht, die die Anwendung von Befehlen, wie LIST, FETCH und ähnliche, nicht erlaubt. Daher wird vom System eine Kopie der ursprünglichen Version in der ersten offenen Bibliothek zwischengespeichert und nach der Ausführung oder einem Abbruch mit der Funktion BREAK wieder in den Arbeitsspeicher geladen. Ist in der entsprechenden Bibliothek nicht genügend Platz dafür vorhanden oder keine Bibliothek offen, erscheint die Fehlermeldung ERROR 188.

  Die Zwischenspeicherung unterbleibt, wenn der Befehl in Form von "RUN filename" eingegeben wird. In diesem Falle lädt das System nach Ausführung des Programmes die gespeicherte Version erneut in den Arbeitsspeicher.

- Enthält das Programm eine COMMON-Anweisung, kann mit RUN ohne Parameter nur gestartet werden, sofern das Programm im Arbeitsspeicher schon einen Namen hat.

- Die Meldung "PROGRAM filename RUNNING" kann durch die Eingabe des Parameters "MSG" im Befehl SAVE unterdrückt werden.

- Die Programmausführung kann durch Ausführung der Funktion BREAK abgebrochen werden. Begonnene In-/Outputoperationen werden noch regulär beendet und eventuell geöffnete Files geschlossen. Das System geht in den Command-Mode.
- Mit der Taste EXIT kann die Programmausführung unterbrochen und das System in den Debugging-Mode versetzt werden.

BEISPIEL:

```
OLD TEST
RUN
**** FORMALLY CORRECT PROGRAM ****



RUN



OLD TEST
PRE
**** FORMALLY CORRECT PROGRAM ****
:: ROOM=15942 W ::
REP

RUN TEST
```

Bei der ersten Ausführung wird die Preexecution durchgeführt.

Bei der zweiten Ausführung entfällt die Preexecution.

Abspeichern des Programmes mit durchgeführter Preexecution (d.h., daß diese bei weiteren Programmläufen entfällt).

BEFEHL: SAVE

FUNKTION: Abspeicherung eines Programmes oder Textes in einer Bibliothek.

FORMAT: SAV[E] filename,[lib ref][,MSG=n][,COM=m]

"filename"	Name, unter welchem das Programm oder Textfile abgespeichert werden soll.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.
"n"	0 oder 1.
"m"	Anzahl Worte für den Common-Bereich (siehe COMMON).

WIRKUNG:

- Es wird geprüft, ob die durch "lib ref" angegebene Bibliothek bereits ein File mit dem Namen "filename" enthält. Wenn nicht, wird kontrolliert, ob in der angesprochenen Bibliothek genügend Speicherplatz verfügbar ist. Wenn ja, wird das Programm oder Textfile unter diesem Namen abgespeichert.

- Parameter "MSG=n" (MSG = Message):

  Er legt fest, welche Meldungen während jedem Programmablauf ausgegeben werden.

  . Wenn n=1, wird die Meldung "PROGRAM file name RUNNIG" unterdrückt.

  . Ist n=0, wird zusätzlich die Meldung "READY" nach Beendigung der Ausführung unterdrückt. Dies ermöglicht die Ausgabe einer Display-Anzeige durch das Programm, in Zeile 0 des Bildschirms, die auch im Command-Mode erhalten bleibt.

  . Ist der Parameter im Befehl nicht spezifiziert, werden alle Meldungen ausgegeben.

- Parameter "COM=n"

  Beim Laden des Programmes werden im Arbeitsspeicher m Worte für den Common-Bereich reserviert (m 65536). Daten im Common-Bereich können an andere Programme weitergereicht worden (vgl. Anweisung COMMON).

BEMERKUNGEN:

- Der Aufbau von "filename" bestimmt die für das Abspeichern gesehene Teilbibliothek. Im einzelnen gilt für das Format der Filenamen:

  1. *name Speicherung in Package-Teilbibliothek
  2. +name Speicherung in Common-Teilbibliothek
  3. name Speicherung in User-Teilbibliothek

  "name" String von 1 - 6 Großbuchstaben oder Ziffern. Er muß mit einem Buchstaben beginnen. Zwischen "*" bzw. "+" und dem ersten Buchstaben darf kein Leerzeichen stehen.

- Ist die Package-Teilbibliothek durch das Dienstprogramm LBPROTECT geschützt, so kann kein weiteres File in dieser Bibliothek gespeichert werden. Ein SAVE-Befehl bewirkt eine Fehlermeldung.
- Beim Katalogausdruck mit dem Parameter "F" wird der Wert von "m" in "COM=m" angegeben.
- Der Common-Bereich wird bei jedem Laden des Programmes durch OLD, RUN, PREPARE oder CHAIN im Arbeitsspeicher reserviert.
- Muß die Größe des Common-Bereiches eines bereits gespeicherten Programmes geändert werden, sind der Reihe nach folgende Befehle auszuführen: OLD-PURGE-SAVE, wobei bei SAVE mit COM=m die neue Größe des Common-Bereiches festzulegen ist.
- Genügt der Platz im Arbeitsspeicher für das Laden von Programm und Common-Bereich nicht, erfolgt eine Fehlermeldung. Fehlt nur der Platz für den Common-Bereich, wird das Programm allein geladen, was mit LIST überprüft werden kann.
- Soll ein Programm bezüglich Platzbedarf und Arbeitsgeschwindigkeit optimiert werden, ist vor SAVE oder REPLACE die folgende Befehlsreihe auszuführen:

  DECOMPILE-COMPILE-PREPARE-BREAK

BEISPIELE:

Beispiel 1:

Ein Programm soll unter dem Namen "+PROG" in der Common-Teilbibliothek von "LIB1" und unter dem Namen "*PROG" in der Package-Teilbibliothek von "LIB2" gespeichert werden. Die Bibliotheken befinden sich auf der Platte "D2".

Die Befehle lauten:

```
SAVE +PROG,[LIB1,D2]
SAVE *PROG,[LIB1,D2]
```

Beispiel 2:

Der Parameter MSG.

```
SAVE *BSP2,,MSG=1
```

Beispiel 3:

Reservierung des Common-Bereiches.

```
SAVE *BSP3,,COM=500
```

Beispiel 4:

Alle Parameter.

```
SAVE *BSP4,(P6FSYS,F2),MSG=0,COM=220
```

Kontrollausdruck der Beispiele 2 - 4:

```
CAT *,,,F

     EFDOSC-R 3.1    *  VOLLABEL =          *  LIBRARY = P6FSYS  *  DATE:07-05-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

     *BSP2     P  07-05-80  07-05-80   0384        0384                 1
     *BSP4     P  07-05-80  07-05-80   0384        0384                 1  C=220
     *BSP3     P  07-05-80  07-05-80   0384        0384                 1  C=500
```

BEFEHL: SECURE

FUNKTION: Verhindert das Listing und Editing von Programmen und das Überschreiben von Datenfiles.

FORMAT: SEC[URE]filename|,lib ref[,n]|

"filename" Name des zu schützenden Files.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.

"n" positive ganze Zahl $1 \leq n \leq 9999$.

WIRKUNG: Das System prüft, ob unter dem Namen "filename" in der durch "lib ref" angegebenen Bibliothek ein Programm oder Datenfile gespeichert ist. Wenn ja, wird dieses gesichert. Ist der Parameter "n" angegeben, erfolgt der Schutz bei Programmfiles erst ab Zeile n.

Folgende Befehle werden nicht mehr ausgeführt:
- DECOMPILE
- RESEQUENCE
- MODIFY
- TRANSCODE
- TRUNCATE
- LINK

Für die Befehle
- DELETE LINE
- FETCH
- ↓
- ↑
- LIST

gilt:
1. Ist der Parameter "n" im Befehl nicht angegeben, wird keiner dieser Befehle ausgeführt.
2. Ist der Wert "n" vorhanden, werden die Befehle ausgeführt, sofern sie sich auf eine Anweisung mit einer kleineren Zeilennummer als "n" beziehen. Bei Datenfiles ist der Parameter "n" wirkungslos, das heißt: es wird immer das ganze File geschützt.

Für gesicherte Datenfiles gilt:
Der Befehl TRANSCODE, das Dienstprogramm FLPRINT und die Anweisung WRITE: werden nicht ausgeführt.

BEMERKUNG: Der geschützte Bereich eines Programmes kann durch einen weiteren SECURE-Befehl erweitert, nicht aber verkleinert werden. Daten von geschützten Datenfiles können nur gelesen werden.

BEISPIELE:

```
0010 DATA 5,7,10,30,45,3
0020 READ N
0030 LET S=0
0040 FOR I=1 TO N STEP 1
0050 READ A
0060 LET S=S+A
0070 NEXT I
0080 PRINT "MITTELWERT";S/N
0090 END



SAVE MITTEL
SEC MITTEL,,20
```

Das so gesicherte Programm erlaubt die Berechnung des arithmetischen Mittels einer Zahlenfolge. Der mit Zeile 20 beginnende Rechenteil ist gesichert. Der Inhalt der DATA-Anweisung kann verändert werden, um zum Beispiel verschiedene Datensätze auszuwerten.

```
OLD MITTEL
LIST
FILE     MITTEL

0010 DATA 5,7,10,30,45,3

SECURED FILE
```

Es wird nur der ungesicherte Teil des Programmes gedruckt.

```
FETCH 20
ERROR  205
```

Zeile 20 kann nicht ins Display geholt werden.

```
20 FOR I=1 TO 20
ERROR  205
```

Zeile 20 kann nicht verändert werden.

SHIFT

BEFEHL: SHIFT

FUNKTION: Erhöhung der Zeilennummer um einen konstanten Wert ab einer beliebigen Zeile eines Programmes oder Textes.

FORMAT: SHI[FT] Zeilennummer, Konstante

"Zeilennummer" Nummer der ersten zu shiftenden Zeile.

"Konstante" positive ganze Zahl, welche die Erhöhung der Zeilennummern festlegt.

WIRKUNG: Die Zeilennummern eines im Arbeitsspeicher vorhandenen Programmes oder Textfiles werden ab der angegebenen Zeilennummer um den Wert der Konstante erhöht.

BEMERKUNGEN:

- Im Gegensatz zu Textfiles werden bei Programmen automatisch auch alle innerhalb von BASIC-Anweisungen vorkommenden Hinweise auf Zeilen (z.B. GOTO, IF...THEN, PRINT USING) korrigiert.
- Die mit "Zeilennummer" angesprochene Zeile muß vorhanden sein.
- Die Schrittweite der Numerierung wird durch SHIFT nicht verändert.
- Ist die Summe von höchster Zeilennummer und angegebener Konstante größer als 9999, erscheint eine Fehlermeldung.

BEISPIEL:

```
FILE


0010 DISP "EINGABE A UND B";
0020 INPUT A,B
0030 PRINT A,B
0040 FOR I=1 TO 20 STEP 4
0050 INPUT A(I)

END OF LISTING



SHI 30,12



LIST
FILE


0010 DISP "EINGABE A UND B";
0020 INPUT A,B
0042 PRINT A,B
0052 FOR I=1 TO 20 STEP 4
0062 INPUT A(I)

END OF LISTING
```

Programm vor Ausführung von SHIFT.

Erhöhung der Zeilennummer ab Zeile 30 um 12.

Programm nach Ausführung des Befehles.

BEFEHL: SPACE

FUNKTION: Abfrage des freien Platzes einer Bibliothek.

FORMAT: SPA[CE] [lib ref]

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Diskstation anzugeben.

WIRKUNG: Der freie Platz in der mit "lib ref" angegebenen Bibliothek wird im Bildschirm angezeigt. Die Angabe erfolgt in Bytes.

BEMERKUNG: Wird in einer Bibliothek ein File angelegt (z.B. durch SAVE, CREATE), so wird vom Betriebssystem außer dem im Katalog ausgewiesenen Platz ein Sektor (256 Bytes) für Kontrollinformationen belegt.

BEISPIEL:

```
SPA (,F1)
SPACE = 142336
```

STIMAGE

BEFEHL: STIMAGE (store image)

FUNKTION: Speicherung des Bildschirminhaltes in einem sequentiellen Datenfile.

FORMAT: STI[MAGE] [filename][,lib ref][,n]

"filename"	Name des Files, in welches das Bild übertragen werden soll.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.
"n"	Buffergröße in K-Bytes.

WIRKUNG: Der Bildschirminhalt wird in das sequentielle Datenfile "filename" übertragen und kann später jederzeit wieder auf den Bildschirm geholt oder mit der Option PLOT für die Ausgabe über den Thermodrucker verwendet werden.

Der Parameter "n" bestimmt die Größe eines Transitbuffers zur Übertragung des Bildes via Arbeitsspeicher. Fehlt der Parameter, wird ein Buffer von 3K gebildet.

BEMERKUNGEN:

- Das System muß mit der Option GDI initialisiert sein.
- Fehlt der Parameter "filename", wird ein File namens SYSPLO gesucht.
- Das mit "filename" bezeichnete Datenfile muß als sequentielles File kreiert sein. Die benötigte Kapazität hängt vom Umfang des Bildinhaltes ab und beträgt im Maximum 35 K-Bytes.
- Wird der Buffer möglichst groß gewählt, so wird zwar die Übertragung beschleunigt, jedoch im Datenfile möglicherweise mehr Platz benötigt, da das Bild als Vielfaches des Buffers abgelegt wird.

- Wird ein mit STIMAGE gespeichertes Bild später wieder in einem Programm als Ausgangsbild für eine Ergänzung benutzt (LDIMAGE), so fehlen die Angaben über SCALE, FRAME usw. und müssen neu definiert werden. Andererseits kann ein Bild am Bildschirm auch mit der Option PLOT weiterverarbeitet werden.

STKEYS

BEFEHL: STKEYS (store keys)

FUNKTION: Speicherung der momentanen Belegung der Funktionstasten als Standardinhalt.

FORMAT: STK[EYS]

WIRKUNG: Die auf den Funktionstasten F1 bis F16 gespeicherten Strings werden als neuer Standardinhalt auf der Systemdisk gespeichert. Dieser Standardinhalt wird bei jeder Neuinitialisierung des Systems oder mit dem Befehl LDKEYS wieder in die Funktionstasten geladen. Er wird durch eine vorübergehend andere Belegung der Funktionstasten nicht geändert.

BEISPIEL:

```
NEW
10 FKEY#1,STANDARD
20 FKEY#2,BELEGUNG
```

Belegung der Funktionstasten durch ein Programm.

Nach

RUN

erhält die Taste F1 den Text STANDARD,
die Taste F2 den Text BELEGUNG.

Nach

STKEYS

wird diese Belegung auf der Systemdisk als Standardbelegung gespeichert. Durch Drücken der Tasten F1 und F2 erscheint der Text STANDARDBELEGUNG im Display.

Konsoltaste CALC MODE aktivieren.

```
FKEY#1,RUN HELP:
FKEY#5,LIST:
```

Belegung der Funktionstasten mittels Calculator-Mode.

Das Setzen eines Doppelpunktes nach den Systembefehlen RUN HELP oder LIST ersetzt das Betätigen der Taste EOL.

Nach Drücken von F1 wird das Programm HELP gestartet und durch Betätigen von F2 ein sich im Arbeitsspeicher befindliches Programm gelistet.

Durch den Befehl STKEYS wird diese Belegung auf der Systemdisk als Standardbelegung gespeichert.

Wird im Calculator-Mode oder innerhalb eines Programmes F1 durch FKEY # 1,AKTUELL geändert, ist nach Eingabe des Befehles LDKEYS oder nach Neuinitialisierung des Systems wieder die Standardbelegung gültig, das heißt: nach Eingabe von LDKEYS wird durch Drücken von F1 das Programm HELP gestartet.

BEFEHL: TEXT

FUNKTION: Das System wird für die Eingabe eines Textes über die Tastatur vorbereitet.

FORMAT: TEX[T]

WIRKUNG:
- Textzeilen müssen, wie Programmzeilen, eine Zeilennummer enthalten.
- Die Zeilennummer einer Textzeile ist nicht Bestandteil des Textes. Sie kann durch RESEQUENCE oder SHIFT verändert werden.

BEISPIEL:

```
TEXT
AUTO#
10 BEISPIEL FUER TEXTFILE
20
30 BELIEBIGE EINGABE MIT
40 GROSSBUCHSTABEN ODER
50 Kleinbuchstaben.
60
70 ZEILENNUMMER NICHT VERGESSEN!


RES 5,5
LIST
FILE


0005 BEISPIEL FUER TEXTFILE
0010
0015 BELIEBIGE EINGABE MIT
0020 GROSSBUCHSTABEN ODER
0025 Kleinbuchstaben.
0030
0035 ZEILENNUMMER NICHT VERGESSEN!

END OF LISTING
```

BEFEHL: TRANSCODE

FUNKTION: Konvertierung von Textfiles in Datenfiles und umgekehrt.

FORMAT: TRA[NSCODE] $\left\{\begin{matrix}T\\D\end{matrix}\right\}$,filename,[lib ref][,#]

"filename"	Name eines Datenfiles.
"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.
"T"	Umwandeln in ein Textfile.
"D"	Umwandeln in ein Datenfile.

WIRKUNG:

- Parameter "T":
  Es wird geprüft, ob in der durch "lib ref" angegebenen Bibliothek ein Datenfile namens "filename" existiert. Ist das File geschützt, erscheint eine Fehlermeldung. Andernfalls wird kontrolliert, ob alle Datenelemente alphanumerisch (Strings) sind. Wenn ja, wird jedes einzelne Datenelement in eine Textzeile umgewandelt und im Arbeitsspeicher abgelegt.

  . Mit Angabe von Parameter "#":
  Kontrolle, ob das Datenfile den Aufbau eines konvertierten Textfiles hat und ob jeder String eine eigene Zeilennummer aufweist. Wenn ja, werden die ursprünglichen Zeilennummern beibehalten.

  . Ohne Parameter "#":
  Das Textfile hat als erste Zeilennummer den Wert 1 und wird mit Schrittweite 1 numeriert. Eventuell abgespeicherte Zeilennummern werden als Bestandteil des Textes interpretiert. Das Textfile befindet sich nach der Umwandlung im Arbeitsspeicher.

- Parameter "D":
  Es wird geprüft, ob im Arbeitsspeicher ein Textfile vorhanden ist. Des weiteren wird kontrolliert, ob in der durch "lib ref" angegebenen Bibliothek schon ein File namens "filename" existiert. Wenn ja, erscheint eine Fehlermeldung. Andernfalls folgt die Kontrolle, ob in der durch "lib ref" bezeichneten Bibliothek genügend freier Speicherplatz vorhanden ist. Ist dies der Fall, wird ein sequentielles Datenfile mit dem Namen "filename" kreiert.

  In einem so geschaffenen Datenfile wird jede Zeile des Textfiles als ein alphanumerisches Datenelement (String) abgespeichert.

  . Mit Angabe von Parameter "#":
    Die Zeilennummer wird ebenfalls gespeichert und die Zeile als String aufgezeichnet.

  . Ohne Parameter "#":
    Die Zeilennummern werden nicht gespeichert.

BEMERKUNGEN:

- Ist der Arbeitsspeicher für die Aufnahme des komplett konvertierten Datenfiles zu klein, werden nur so viele Datenelemente in Textzeilen umgewandelt, wie im Arbeitsspeicher Platz finden.
- Enthält ein Datenelement mehr als 76 Zeichen (ohne Zeilennummer), so wird die Konvertierung mit dieser Zeile abgebrochen (ERROR 213).
- Während der Ausführung des Befehles (Parameter "T") ist die Tastatur nicht verwendbar.
- Der Befehl kann für mit SECURE/Parameter T und/oder LBPROTECT geschützte Files mit Parameter D nicht eingesetzt werden.
- Die durch "lib ref" spezifizierte Bibliothek muß offen sein.

BEISPIELE:

```
LIST
FILE


0005 BEISPIEL FUER TEXTFILE
0010
0015 BELIEBIGE EINGABE MIT
0020 GROSSBUCHSTABEN ODER
0025 Kleinbuchstaben.
0030
0035 ZEILENNUMMER NICHT VERGESSEN!

END OF LISTING
```

Eingabe eines Textfiles.

```
TRA D,DATA1,(,F2),#
TRA T,DATA1,(,F2)

LIST
FILE


00010005 BEISPIEL FUER TEXTFILE
00020010
00030015 BELIEBIGE EINGABE MIT
00040020 GROSSBUCHSTABEN ODER
00050025 Kleinbuchstaben.
00060030
00070035 ZEILENNUMMER NICHT VERGESSEN!

END OF LISTING
```

Umwandlung in ein Datenfile DATA1.

Rückwandlung in ein Textfile: Doppelte Zeilennumerierung, da Parameter # fehlt.

```
TRA T,DATA1,(,F2),#

LIST
FILE


0005 BEISPIEL FUER TEXTFILE
0010
0015 BELIEBIGE EINGABE MIT
0020 GROSSBUCHSTABEN ODER
0025 Kleinbuchstaben.
0030
0035 ZEILENNUMMER NICHT VERGESSEN!

END OF LISTING
```

Rückwandlung in ein Textfile mit "alter" Zeilennummer.

TRUNCATE

BEFEHL: TRUNCATE

FUNKTION: Verkleinern des für ein sequentielles Datenfile reservierten Speicherplatzes auf den effektiv belegten Platz.

FORMAT: TRU[NCATE] filename[,lib ref]

"filename" Name des sequentiellen Datenfiles.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disketten- oder Diskstation anzugeben.

WIRKUNG: Es wird kontrolliert, ob ein nicht geschütztes Datenfile namens "filename" in der durch "lib ref" angegebenen Bibliothek existiert. Wenn ja, und wenn es sich um ein sequentielles Datenfile handelt, wird der reservierte Speicherplatz auf den effektiv belegten Platz verkleinert (logisches Ende des Files = physisches Ende).

BEMERKUNGEN:
- Bei Randomfiles bewirkt der Befehl eine Fehlermeldung, da die effektive Länge von Randomfiles immer der reservierten Länge entspricht.
- Die maximale Länge des Datenfiles beträgt nach Ausführung des Befehles immer ein ganzzahliges Vielfaches von 128 bei Disketten, bzw. 256 bei Disks.
- Der freigewordene Speicherplatz kann für andere Files wieder eingesetzt werden.
- Die durch "lib ref" angesprochene Bibliothek muß offen sein.
- Der Befehl kann für mit SECURE geschützte Files nicht verwendet werden.

BEISPIEL:

```
CAT ,,,F

     EFDOSC-R 3.1    *  VOLLABEL =          *  LIBRARY = P6FSYS  *  DATE:07-05-80
     FILE   TYPE  CREAT    LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      TEST    S  07-05-80  07-05-80   1024       0080                1

TRU TEST

CAT ,,,F

     EFDOSC-R 3.1    *  VOLLABEL =          *  LIBRARY = P6FSYS  *  DATE:07-05-80
     FILE   TYPE  CREAT    LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      TEST    S  07-05-80  07-05-80   0128       0080                1
```

BEFEHL: VALIDATE

FUNKTION: Schließen eines offen gebliebenen Datenfiles.

FORMAT: VAL[IDATE] ,filename[,lib ref]

"filename" Name des zu schließenden Files.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Diskstation anzugeben.

WIRKUNG: Das aufgrund eines nicht regulären Programmendes (z.B. Stromausfall) offen gebliebene Datenfile namens "filename" wird geschlossen.

BEMERKUNGEN:
- Wurde das File beschrieben, so können einige der zuletzt eingegebenen Daten eines Randomfiles zerstört sein. Bei sequentiellen Files ist nach dem Schließen die vor dem Öffnen vorhandene aktuelle Länge gespeichert. Die Daten des Files sind nur entsprechend der gespeicherten, effektiven Länge des Files verfügbar, das heißt, daß alle Werte verloren sind, die zwischen dem letzten Programmstart und dessen irregulärem Abbruch eingegeben wurden.
- Offen gebliebene Files werden im Katalog mit "OPEN" gekennzeichnet.

BEISPIEL:

```
CAT TEST
      TEST     S  07-05-80  07-05-80   1024        0000                  1  OPEN

VAL TEST

CAT TEST
      TEST     S  07-05-80  07-05-80   1024        0000                  1
```

6.4 Zusätzliche Befehle in Prozeduren

Wie unter 6.2.1.3 beschrieben, bestehen vier spezielle Befehle, die nur innerhalb von Prozeduren erlaubt sind und folgende Möglichkeiten bieten:

6.4.1 Automatischer Input

Bei der Ausführung von Programmen durch eine Prozedur können die Daten zu INPUT-, MATINPUT- und RKB-Statements auf zwei Arten eingegeben werden:

- bei Ausführung der entsprechenden Zeile manuell über die Tastatur!
- durch feste Aufnahme in die Prozedur. Sie werden dann während der Ausführung des Programmes automatisch als Input zugewiesen.

Für die automatische Datenübernahme ist folgendes Format zu verwenden:

$$\text{IN} = \left\{\begin{matrix}\text{num. Konstante}\\ \text{Stringkonst.}\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{num. Konstante}\\ \text{Stringkonst.}\end{matrix}\right\}\ldots\right]$$

Für jedes auszuführende INPUT-/RKB-Statement ist eine "IN="-Zeile in der Prozedur vorzusehen. Darin erfolgt die Dateneingabe in der gleichen Form wie bei der manuellen Eingabe.

Fehlen innerhalb einer "IN="-Zeile bei der Ausführung der Prozedur Daten, erscheinen zwei Fragezeichen, und die fehlenden Werte können über die Tastatur nachgeliefert werden. Enthält die "IN="-Zeile zu viele Daten, werden die nicht benötigten ignoriert.

Die Anzahl der "IN="-Zeilen sollte mit der Anzahl auszuführender INPUT-Statements übereinstimmen. Sind zu viele "IN="-Zeilen vorgesehen, werden die nicht verwendeten überlesen. Sind zu wenig "IN="-Zeilen vorhanden, tritt ein unbehebbarer Fehler auf, der durch

ERROR 204 IN LINE Zeilennummer

angezeigt wird. Das System verläßt die Prozedur und befindet sich im Command-Mode (siehe Befehl PRO).

Der automatische Input kommt nur zum Einsatz, wenn dies beim Aufruf der Prozedur durch den Parameter A verlangt wird. Andernfalls werden die "IN="-Zeilen ignoriert.

6.4.2 Bedingte Sprünge

Prozeduren können auch Zeilen enthalten, die einen bedingten Sprung innerhalb der Prozedur bewirken. Der Sprung erfolgt in Abhängigkeit von einer vorher definierten Kondition namens CC.

Das Format einer solchen Sprunganweisung lautet:

IF CC = X,Zeilennummer

"X" numerische Konstante zwischen 0 und 9.

"Zeilennummer" Zeilennummer der Prozeduranweisung, bei der die Verarbeitung fortgesetzt werden soll.

- Zwischen IF und CC muß eine Leerstelle eingegeben werden.
- Der Wert der Kondition CC wird durch das zuletzt ausgeführte Programm über eine Stringvariable der deklarierten Länge 1 festgelegt. Diese Stringvariable muß an der ersten Stelle des Common-Bereiches stehen. Um einen Sprung zu bewirken, muß diese Stringvariable das entsprechende ISO-Zeichen "0" bis "9" enthalten.

6.4.3 Fehlermeldungen, Befehle TEST, PRO

Stellt das System bei der Verarbeitung einer Prozedur Fehler fest (Prozedurfehler oder Fehler in einem durch die Prozedur aufgerufenen Programm), so bestehen folgende Fortsetzungsmöglichkeiten:

- Behebbare Fehler innerhalb einer Programmverarbeitung: Gleiche Möglichkeiten wie bei der direkten Verarbeitung eines Programmes.
- Nicht behebbare Fehler während der Programmausführung: Die Programmausführung wird abgebrochen und das System befindet sich im Quasi-Debugging-Mode, der als Fortsetzung nur die Funktion BREAK erlaubt. Die Ausführung der Prozedur wird dadurch abgebrochen, kann jedoch durch Eingabe von PRO (EOL) mit der nächsten Prozedurzeile fortgesetzt werden.
- Fehler bei der Ausführung von Systembefehlen: Das System befindet sich im Command-Mode. Die Prozedur kann abgebrochen oder durch die Eingabe von PRO (EOL) mit der nächsten Prozedurzeile fortgesetzt werden. Im letzteren Fall besteht die Möglichkeit einer automatischen Prozedurfortsetzung mittels TEST. Wird in einer Prozedur der Befehl TEST ON eingefügt, so wird beim Auftreten nicht behebbarer Fehler automatisch mit der nächsten Prozeduranweisung weitergearbeitet. Mit TEST OFF als Prozedur-Befehl kann diese automatische Fehlerbehandlung jederzeit wieder abgebrochen werden. Wird anstelle von TEST OFF der Befehl TEST# verwendet, wird gleichzeitig mit dem Abbruch der automatischen Fehlerbehandlung die Anzahl festgestellter Fehler seit dem Befehl TEST ON ausgedruckt.

Bemerkungen:

- Soll beim Auftreten von nicht behebbaren Programmfehlern die automatische Weiterverarbeitung einer Prozedur gesichert werden, ist folgendes vorzusehen:

Eine Funktion für den internen Interrupt, die im Falle von nicht behebbaren Programmfehlern mit CHAIN ein Dummy-Programm aufruft, wird ins Progamm eingefügt, das dann regulär beendet wird.

- Fehler, die während der Preexecution eines Programmes auftreten, bewirken auch nach TEST ON einen Abbruch der Prozedurausführung.

6.5 Befehle im Calculator- und Debugging-Mode

Eine Anzahl Systembefehle sind nur im Calculator- oder Debugging-Mode einsetzbar. Die Systembefehle für den Bildschirm sind zusätzlich im Command-Mode verwendbar.

6.5.1 Liste der Befehle:

Name	Funktion
FKEY	Belegung einer Funktionstaste mit einer Stringkonstante.
SDEG	Vorwahl der Maßeinheit "Winkeldarstellung in Altgrad".
SGRAD	Vorwahl der Maßeinheit "Winkeldarstellung in Neugrad".
SRAD	Vorwahl der Maßeinheit "Winkeldarstellung im Bogenmaß".
START	Fortsetzung der Programmausführung ab der angegebenen Zeile.
STOP	Programmstop bei der angegebenen Zeilennummer.

Befehle für den Bildschirm:

DRAW	Hardcopy des Bildschirminhaltes auf Thermodrucker.
ERASE	Löschen des graphischen und/oder alphanummerischen Bildschirminhaltes.
LDIMAGE	Ausgabe eines gespeicherten Bildes auf den Bildschirm.
REVERSE	Umkehrung des Bildes auf dem Bildschirm von positiv auf negativ und umgekehrt.
STIMAGE	Speicherung des Bildschirminhaltes in einem sequentiellen Datenfile.

6.5.2 Ausführliche Beschreibung der Befehle

Für die Bildschirmbefehle gemäß Liste unter Abschnitt 6.5.1 siehe Kapitel 6.3 der allgemeinen Systembefehle.

BEFEHL: FKEY # (funktion key)

FUNKTION: Zuweisung eines Inhaltes an eine Funktionstaste.

FORMAT: FKEY # n, Stringkonstante [:]

"n" ganze Zahl zwischen 1 und 16.

WIRKUNG: Einer durch den Wert des Identifikators "n" bestimmten Taste wird der Inhalt einer Stringkonstante zugewiesen.

BEMERKUNGEN:

- Durch jedes Drücken der Funktionstaste wird die der Taste zugewiesene Zeichenfolge in den Tastaturbuffer übertragen. Folgt der Stringkonstante ein Doppelpunkt, wird der Inhalt des Tastaturbuffers direkt (d.h. ohne EOL) ans System übermittelt. Den 16 Funktionstasten können insgesamt maximal 238 Zeichen zugewiesen werden.
- Der Inhalt der Funktionstaste bleibt erhalten, bis er durch einen neuen Befehl FKEY mit gleichem n überschrieben oder durch den Befehl LDKEYS bzw. durch eine Neuinitialisierung des Systems wieder durch den Standardinhalt ersetzt wird. Durch Eingabe von STKEYS wird der so definierte Inhalt zum Standardinhalt. Die letzten beiden Befehle sind im Debugging-Mode nicht möglich, bzw. verlassen den Calculator-Mode und erreichen automatisch den Command--Mode.
- Der Inhalt einer Funktionstaste entspricht dann einer gültigen Eingabe, wenn auch dessen zeichenweises Eintasten über die Tastatur eine gültige Eingabe bilden würde.

BEISPIEL: Siehe STKEYS (Seite 6.109) und FKEY als BASIC-Anweisung (Kapitel 9).

SDEG
SGRAD
SRAD

BEFEHL: SDEG, SGRAD, SRAD

FUNKTION: Festlegung der Masseinheit trigonometrischer Funktionen.

FORMAT: SDE[G]
SGR[AD]
SRA[D]

WIRKUNG: Nach dem Einschalten des Calculator-Mode bzw. Erreichen des Debugging-Mode wird SRAD definiert, das heißt: die Maßeinheit für die Winkeldarstellung ist das Bogenmaß. Bei Verwendung von SDEG oder SGRAD sind die Winkel in dezimaler Form einzugeben bzw. werden die Resultate von Winkeln dezimal ausgedruckt.

BEMERKUNG: Durch Aufleuchten einer kleinen, gelben Konsollampe wird dem Anwender angezeigt, wenn nicht im Bogenmaß gerechnet wird, das heißt, wenn mit SDEG oder SGRAD umgeschaltet wurde.

BEISPIEL:

```
SDEG

SIN(30)
 0.500000

SGRAD

SIN(33.33)
 0.499955

SRAD

SIN(PI/6)
 0.500000
```

BEFEHL: START

FUNKTION: Start eines unterbrochenen Programmablaufes ab einer bestimmten Zeilennummer.

FORMAT: STA[RT] Zeilennummer

"Zeilennummer" positive Zahl zwischen 1 und 9999.

WIRKUNG: Dieser Befehl löst die Weiterverarbeitung eines unterbrochenen Programmlaufes ab der durch den Parameter "Zeilennummer" angegebenen Programmzeile aus.

BEMERKUNGEN:
- Um den Befehl ausführen zu können, muß sich das System im Debugging-Mode befinden.
- Der Parameter "Zeilennummer" ist so zu wählen, daß eine sinnvolle Verarbeitung des Programmes möglich ist. Das Sprungziel darf nicht innerhalb von
  - FOR/NEXT-Schleifen,
  - Unterprogrammen,
  - mehrzeiligen Funktionsdefinitionen

  liegen.
- Die als Parameter angegebene Zeilennummer muß im Programm vorhanden sein.

BEISPIEL:

```
LIST
FILE     TEST2


0010 INPUT A
0020 INPUT B
0030 PRINT A↑B
0040 STOP
0050 PRINT
0060 END

END OF LISTING



RUN TEST2
?
2
?
4
 16
STOP       IN LINE 40
START 20
?
5
 32
STOP       IN LINE 40
STA 10
?
3
?
5
 243
STOP       IN LINE 40
```

STOP

BEFEHL: STOP

FUNKTION: Unterbrechen des Programmablaufes vor Ausführung einer bestimmten Zeile.

FORMAT: STO[P] Zeilennummer

"Zeilennummer" = positive ganze Zahl zwischen 1 und 9999

WIRKUNG: Die Verarbeitung des Programmes wird unmittelbar vor der angegebenen Zeile unterbrochen. Das System befindet sich danach im Debugging-Mode.

BEMERKUNGEN:

- Um den Befehl ausführen zu können, muß sich das System im Debugging-Mode befinden.
- Das Programm muß mit der Funktion CONTINUE oder START fortgesetzt oder durch Drücken von EXIT schrittweise ausgeführt werden.
- Der Stop-Befehl bleibt bis zum Ende der Programmausführung oder bis zur Eingabe eines neuen STOP-Befehles gültig.
- Die als Parameter angegebene Zeilennummer muß im Programm vorhanden sein.

BEISPIEL:

```
LIST
FILE     TEST3


0010 INPUT A
0020 INPUT B
0030 PRINT "EINGABEN:";A,B
0040 PRINT A+B
0050 STOP
0060 PRINT
0070 END

END OF LISTING



RUN TEST3
?
3
?
2
EINGABEN: 3       2
 5
STOP      IN LINE 50

STO 40
STA 20
?
3
EINGABEN: 3       3
STOP      IN LINE 40
```

7.	DIENSTPROGRAMME	7.1
7.1	Allgemeine Hinweise	7.1
7.2	Liste der Dienstprogramme	7.1
	DCOPY	7.3
	DINIT	7.5
	FLCOPY	7.7
	FLPRINT	7.11
	LBCREATE	7.13
	LBEMPTY	7.17
	LBPROTECT	7.19
	LBRENAME	7.21
	LBSCRATCH	7.23
	LIBCOPY	7.25
	RESTRUCT	7.29
	VOLLAB	7.31

7. DIENSTPROGRAMME

7.1 Allgemeine Hinweise

Im folgenden Abschnitt wird die Funktion der im Betriebssystem enthaltenen Dienstprogramme beschrieben.

Der Aufruf eines Dienstprogrammes erfolgt unter Angabe des Namens und der erforderlichen Parameter mit dem Befehl EXEC.

Im Gegensatz zur Ausführung von Systembefehlen wird bei der Ausführung von Dienstprogrammen der Arbeitsspeicher benötigt. Soll der bisherige Inhalt des Arbeitsspeichers erhalten bleiben, ist vor Aufruf eines Dienstprogrammes der Inhalt des Arbeitsspeichers mit SAVE oder REPLACE abzuspeichern.

Bemerkung:

Der Begriff "Systemdisk" steht im folgenden für diejenige Disk, die das aktuelle Betriebssystem enthält.

7.2 Liste der Dienstprogramme

DCOPY	Kopieren von Disks.
DINIT	Initialisieren einer Disk.
FLCOPY	Kopieren von Files.
FLPRINT	Ausdruck eines Datenfiles.
LBCREATE	Anlegen einer Bibliothek auf einer Disk.
LBEMPTY	Löschen aller Files einer Bibliothek.
LBPROTECT	Schützen von Teilbibliotheken.
LBRENAME	Ändern eines Bibliotheksnamens.
LBSCRATCH	Löschen einer Bibliothek.
LIBCOPY	Kopieren einer Bibliothek.
RESTRUCT	Neuorganisation des Diskinhaltes nach dem Löschen einer Bibliothek.
VOLLABEL	Kennzeichnung einer Disk mit dem Volume-Label.

DCOPY

DIENST-
PROGRAMM: DCOPY (disk copy)

FUNKTION: Kopieren von Disks.

FORMAT: EXE[C] DCO[PY], unit name$_1$, unit name$_2$[,[volume label][,V]]

"unit name$_1$" Input-Einheit (Original).

"unit name$_2$" Output Einheit (Kopie).

"volume label" Kennzeichnung der Disk.

"V" "volume label" ebenfalls kopieren.

WIRKUNG:

- Der gesamte Inhalt der Disk wird ohne Änderung der logischen Struktur kopiert. Die Kopie kann nur auf artgleiche Platten erfolgen.
- Ist der Parameter "volume label" angegeben, so wird geprüft, ob die Output-Disk den im Aufruf angegebenen Volume-Label enthält. Bei Nichtübereinstimmung wird keine Kopie erstellt.
- Ist der Parameter "V" spezifiziert, so wird der Volume-Label des Originals übernommen.

MELDUNGEN:

- Nach dem Aufruf des Dienstprogrammes "DCOPY" erscheint im Display die Meldung "DISK ON unit name1 unit name2". Sie erlaubt eine nochmalige Kontrolle, ob die Wahl der Platteneinheiten richtig war:
  - . richtig: SHIFT EXIT (FUNKTION CONTINUE)
  - . falsch: CONTROL EXIT (FUNKTION BREAK)
- Nach Beendigung des Kopiervorgangs erscheint die Anzeige "RESTORE DISKS". Bei Bedarf können nun die am Kopiervorgang beteiligten Disks ausgewechselt werden. Mit CONTINUE wird der Command-Mode wieder erreicht.
- Die Meldung "LOAD DISK ON unit name" zeigt an, daß die Disk zur Aufnahme der Kopie in der genannten Station fehlt. Die Disk kann eingelegt und das Dienstprogramm mit der Funktion CONTINUE fortgesetzt werden.

- ERROR n [-RESTORE DISK]
  Der Kopiervorgang wurde aufgrund des Fehlers n abgebrochen (siehe Liste der Fehlermeldungen). Erscheint gleichzeitig die Meldung "RESTORE DISKS", so können die für den Kopiervorgang verwendeten Disks auf ihre Funktionsfähigkeit geprüft und allenfalls ausgetauscht werden. Der Kopiervorgang ist nicht durchgeführt.

ACHTUNG: Betriebssysteme werden immer mitkopiert. Erfolgt eine Kopie auf eine Disk, die das aktuelle Betriebssystem enthält, so wird dieses gelöscht. Es muß dann entweder eine andere Systemdiskette eingelegt oder das System neu geladen werden.

BEISPIELE: Kopieren von Diskettenstation F2 auf F1.

```
EXE DCO,F2,F1
DISK ON F1      F1
RESTORE DISKS

END OF LISTING
```

Kopieren bei Monofloppyeinheit (im Beispiel mit Stationsbezeichnung F1)

```
EXE DCO,F1,F1
DISK ON F1      F1
RESTORE DISKS

END OF LISTING
```

DINIT

DIENST-
PROGRAMM: DINIT

FUNKTION: Initialisieren einer Disk.

FORMAT: EXE[C] DIN[IT], unit name [,[volume label] [, L]]

"unit name" Name der Einheit, die die zu initialisierende Disk enthält.

"volume label" gewünschte Kennzeichnung der zu initialisierenden Disk.

"L" logische Initialisierung.

WIRKUNG:

- Ist der Parameter "L" angegeben, werden alle Bibliotheken der Disk gelöscht.
- Ist der Parameter "volume label" angegeben, so wird dieser auf die entsprechende Disk geschrieben.
- Fehlt der Operand "volume label", erhält die entsprechende Disk den Standardnamen "MDOS".

MELDUNG: Nach dem Abruf des Dienstprogrammes, jedoch vor Beginn der Initialisierung, erscheint die Display-Meldung "ACTION ON UNIT unit name". Die Initialisierung kann darauf mit der Funktion CONTINUE gestartet oder mit der Funktion BREAK verhindert werden. Das Ende der Initialisierung wird durch "READY" angezeigt.

BEMERKUNGEN:

- Die aktuelle Systemdisk kann mit DINIT nicht gelöscht werden.
- Jede neue Disk muß vor Ihrer Verwendung auf dem System mit DINIT initialisiert werden.

- Mit DINIT kann ein nicht mehr benötigtes Betriebssystem auf einer Disk gelöscht werden.

BEISPIEL:

```
EXE DIN,F1,DISK1,L
ACTION ON UNIT F1     ?
READY



LUT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT= 256 BYTE * DATE:07-05-80

     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
FREE SPACE (TOTAL) = 1898  SECTORS
MAXIMUM BLOCK OF FREE SPACE    = 1898  SECTORS
```

FLCOPY

DIENST-
PROGRAMM: FLCOPY (file copy)

FUNKTION: Kopieren eines Files aus einer (Teil-)Bibliothek in eine andere (oder in die gleiche) Bibliothek.

FORMAT: EXE[C] FLC[OPY], IN=[lib ref$_1$],[password$_1$],filename$_1$,

OUT=[lib ref$_2$],[password$_2$]$\left[\left[\begin{matrix}\text{filename}_2\\ *\\ +\end{matrix}\right\}\right]$

"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.
"IN=lib ref$_1$"	gibt die das zu kopierende File enthaltende (Teil-)Bibliothek an.
"OUT=lib ref$_2$"	bestimmt die (Teil-)Bibliothek, in die das File kopiert werden soll.
"filename$_1$"	Name des zu kopierenden Files.
"filename$_2$"	Name, den die Kopie erhalten soll.
"*"	spezifiziert die Package-Teilbibliothek.
"+"	spezifiziert die Common-Teilbibliothek.
"password$_1$"	Password für die Bibliothek mit dem Original-file.
"password$_2$"	Password für die Bibliothek, die die Kopie aufnimmt.

WIRKUNG:
- Werden alle Parameter angegeben, so gelten für den letzten Parameter folgende Regeln:
  . Ist "filename2" spezifiziert, so wird geprüft, ob in der Empfängerbibliothek bereits ein File gleichen Namens existiert. Wenn ja, erscheint eine Fehlermeldung. Andernfalls wird kopiert und die Kopie erhält den "filename2".

. Bei Angabe von "*" wird das File unter seinem bestehenden Namen in die Package-Teilbibliothek kopiert.
. Durch Eingabe von "+" wird das File unter seinem Originalnamen in die Common-Teilbibliothek kopiert.

- Fehlt der letzte Parameter, wird das File unter seinem alten Namen in die User-Teilbibliothek aufgenommen.
- Ist "lib ref1"="lib ref2", wird das File in die Bibliothek kopiert, die auch das Original enthält.
- Die übrigen Wirkungen von "lib ref" sind aus den nachstehenden Tabellen ersichtlich:

IN=	"filename" wird gesucht in:	
ohne lib-ref	1. Bibliothek der Systemdisk	1)
lib name	der entsprechenden Bibliothek der Systemdisk	
,unit name	allen offenen Bibliotheken in der Einheit "unit name"	1)
lib name, unit name	der angegebenen Bibliothek in der Einheit "unit name"	

OUT=	File wird kopiert in:	
ohne lib ref	1. Bibliothek der Systemdisk	1)
lib name	die entsprechende Bibliothek auf der Systemdisk	
,unit name	1. Bibliothek in der Einheit "unit name"	1)
lib name, unit name	die angegebene Bibliothek in der Einheit "unit name"	

1) Die Suche erfolgt in der mit LTV* angegebenen Reihenfolge.

BEMERKUNGEN:
- Die spezifizierten Bibliotheken müssen nicht offen sein.
- Aus oder in eine geschützte Package-Teilbibliothek (LBPROTECT) kann nicht kopiert werden.

- Ist das Originalfile mit SECURE geschützt, wird dieser Schutz mitkopiert.
- Soll ein offenes File (siehe CATALOG) kopiert werden, wird es zuerst geschlossen und erst dann dupliziert.
- Soll ein mit DEAD gekennzeichnetes File (siehe CATALOG) kopiert werden, so erfolgt eine Fehlermeldung.

FLPRINT

DIENST-
PROGRAMM: FLPRINT (file print)

FUNKTION: Ausdruck des Inhaltes eines Datenfiles.

FORMAT: EXE[C] FLP[RINT], filename[,lib ref]

"filename" Name des Datenfiles.

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
- (,unit name)

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.

WIRKUNG: Es wird kontrolliert, ob ein ungeschütztes Datenfile mit dem Namen "filename" in der mit "lib ref" angegebenen Bibliothek existiert. Wenn ja, wird sein Inhalt über den Drucker im Standardformat ausgegeben. Der Ausdruck kann mit BREAK abgebrochen werden. Die angegebene Bibliothek muß offen sein.

BEISPIEL:

```
CAT DATA7
      DATA7   R  07-05-80  07-05-80   2048       2048                1

EXE FLP,DATA7

 0               TEXT1              .41839390       0               TEXT2
 .88649157       ABCDEFGHIJKLMNOP                   -6.0511000E+62   0
 .98707879        0                 .95457924       0                .55713896
 0                .81827105         0               4.7693357E-02    0
 .68675523        0                 .17611750       0                3.5848356E-02
 0                .50296996         0               .62662024        0
 .88834013        0                 .85254312       0                .73719855
 0                .70058045         0               .16484721        0
 .23854511        0                 .35330382       0                .43174324
 0                .95315603         0               .49297749        0
 8.9233064E-02    0                 .19927847       0                .45305855
 0                .87681069         0               .96187680        0
 .41639730        0                 .28043841       0                .86194278
READY
```

LBCREATE

DIENST-
PROGRAMM: LBCREATE (library create)

FUNKTION: Anlegen einer Bibliothek auf einer Disk.

FORMAT: EXE[C] LBC[REATE],lib ref,[password][,SIZE=Z][,*=n1]
[,+=n2][,NP=n3]

"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.
"SIZE=Z"	gewünschte Kapazität (Z) in K-Bytes.
"password"	Kennwort mit maximal 6 Zeichen.
"n1, n2, n3"	positive ganze Zahlen ≤99 zur Platzreservierung für die Filenamen in den Teilbibliotheken.
"*"	Package-Teilbibliothek.
"+"	Common-Teilbibliothek.
"NP"	User-Teilbibliothek.

WIRKUNG:

- Auf der mit "lib ref" angegebenen Disk oder Diskette wird die spezifizierte Bibliothek mit den gewünschten Teilbibliotheken eingerichtet. Die Zahlen n1, n2 und n3 legen fest, wieviele Directory-Sektoren den einzelnen Teilbibliotheken zur Verfügung gestellt werden. Da in jedem Sektor 27 Filenameneinträge möglich sind, wird die Zahl der in den einzelnen Teilbibliotheken speicherbaren Files implizit festgelegt.
- Die gewünschte Größe der Bibliothek wird durch den Parameter "SIZE=Z" in Kilobytes angegeben.
- Die Größe "Z" einer Bibliothek errechnet sich aus der für die Datenspeicherung benötigten Kapazität, den Sektoren für die Eintragung der Teilbibliotheken (n1+n2+n3) und 4 Sektoren (1 K-Byte) für Kontrolleintragungen des Systems.

Somit läßt sich für die Größe "Z" (in K-Bytes) folgende Formel anwenden:

$$Z=K+INT[(n1+n2+n3)/4]+2$$

worin "K" die gewünschte Kapazität in der Bibliothek für Datenspeicherung (in K-Bytes) ist und für n1, n2, n3 die im Befehlsformat verwendeten Werte bzw. deren Angaben aus der Default-Tabelle einzutragen sind.

BEMERKUNGEN:

- Fehlen die Parameter für die Kapazität und/oder die Teilbibliotheken, werden folgende Default-Werte angenommen:

Typ des Datenträgers	SIZE (Kapazität der Bibliothek)	Parameter der Teilbibliotheken		
		*	+	NP
Megafloppy (256 Bytes pro. Sektor)	262144 Bytes (1024 Sektoren)	3	2	3
Standardfloppy 8 Zoll (256 KB, 128 Bytes pro Sektor)	Gesamtkapazität der Diskette (ca. 240 KB)	5	4	5
Minifloppy (320 KB, 256 Bytes pro Sektor)	262144 Bytes (1024 Sektoren)	3	2	3

- Die Parameter der Teilbibliotheken in einer Bibliothek auf einer Festplatte sind dieselben wie bei einer Megafloppy.
- Soll eine neue Bibliothek mit LBCREATE auf einer bereits benutzten Diskette eingerichtet werden, muß die Diskette vorher mit DINIT oder LBSCRATCH gelöscht werden.
- Neue Disketten müssen mit DINIT initialisiert werden. Um auf eine neu angelegte Bibliothek zugreifen zu können, muß sie mit LBO geöffnet werden.
- Die maximale Anzahl der Bibliotheken beträgt für die Megafloppy 14 und für die Standard-Diskette 1. Die Anzahl der Bibliotheken auf einer Festplatte hängt von der Anzahl und der Größe der angelegten "Daten-Set's" ab.

BEISPIEL:

```
LVT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT=256 BYTE * DATE:07-05-80


     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
FREE SPACE (TOTAL) = 1898  SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 1898  SECTORS

EXE LBC,(P6FSYS,F1),,NP=5,*=2,+=0
READY

LVT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT=256 BYTE * DATE:07-05-80


     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
     P6FSYS   07-05-80   001001        073026        1898       1887
FREE SPACE (TOTAL) = 0     SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 0     SECTORS
```

LBEMPTY

DIENST-
PROGRAMM: LBEMPTY (library empty)

FUNKTION: Löschen aller Files einer Bibliothek.

FORMAT: EXE[C] LBE[MPTY], lib ref[,password]

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.

"password" Password der entsprechenden Bibliothek.

WIRKUNG: Alle in der mit "lib ref" spezifizierten Bibliothek enthaltenen Files werden gelöscht. Name und Größe der Bibliothek bleiben jedoch erhalten. Der frei gewordene Speicherplatz kann wieder verwendet werden.

BEMERKUNGEN:
- Ist unter "lib ref" nur "lib name" angegeben, wird die entsprechende Bibliothek auf der Systemdisk angesprochen.
- Ist die durch "lib ref" spezifizierte Bibliothek mit einem "password" geschützt, muß dieses im Aufruf angegeben werden.

DIENST-
PROGRAMM: LBPROTECT (library protect)

FUNKTION: Schützen von Teilbibliotheken vor der Anwendung bestimmter Systembefehle.

FORMAT: EXE[C]LBP[ROTECT],[lib ref],[password] $\left[,\left\{\begin{matrix}*\\+\end{matrix}\right\}\right]$

"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.
"password"	Password der entsprechenden Bibliothek.
"*"	Eine Package-Teilbibliothek soll geschützt werden.
"+"	Die Common-Teilbibliothek soll geschützt werden.

WIRKUNG: Wird als letzter Parameter "*" angegeben, so werden für die Package-Teilbibliothek der mit "lib ref" angegebenen Bibliothek folgende Systembefehle und Dienstprogramme nicht ausgeführt:

CREATE
MODIFY
PURGE
SAVE
TRANSCODE
TRUNCATE
EXE FLCOPY

Für das Dienstprogramm LIBCOPY gilt:
Es wird nur kopiert, wenn die Package-Teilbibliothek der Empfängerbibliothek leer ist. Wird das Dienstprogramm ausgeführt, so ist auch die Kopie geschützt.

Wird als letzter Parameter "+" angegeben, so werden für die Common-Teilbibliothek der durch "lib ref" bezeichneten Bibliothek folgende Systembefehle nicht ausgeführt:

MODIFY
PURGE

Wird als letzter Parameter weder "*" noch "+" angegeben, so werden sowohl die Package-Teilbibliothek als auch die Common-Teilbibliothek geschützt.

BEMERKUNGEN:

- Wird im Parameter "lib ref" nur "lib name" angegeben, so wird die Bibliothek mit dem Namen "lib name" auf der Systemdisk geschützt.
- Wird im Parameter "lib ref" nur "unit name" angegeben, so wird die erste Bibliothek (LVT*) der Einheit mit dem Namen "unit name" geschützt.
- Fehlt der Parameter "lib ref", so wird die erste Bibliothek (LVT*) der Systemdisk geschützt.
- Ist die durch "lib ref" bestimmte Bibliothek mit einem Password versehen, so muß es beim Aufruf des Dienstprogrammes angegeben werden.
- Der durch dieses Dienstprogramm erreichte Schutz kann nicht mehr aufgehoben werden.

LBRENAME

DIENST-
PROGRAMM: LBRENAME (library rename)

FUNKTION: Änderung des Namens einer Bibliothek und/oder des Passwords.

FORMAT: EXE[C] LBR[ENAME],lib ref,[passw.$_1$],$\left\{\begin{matrix}\text{new lib name[,passw.}_2\text{]}\\ \text{,passw.}_2\end{matrix}\right\}$

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name

Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.

"password$_1$" altes Password.

"password$_2$" neues Password.

"new lib name" neuer Name für die Bibliothek.

WIRKUNG: Werden alle Parameter angegeben, so wird der durch "lib ref" bestimmten Bibliothek der Name "new lib name" und das neue "password2" zugewiesen.

Wird "password1" angegeben, aber "password2" nicht, so ist die Bibliothek in Zukunft nicht mehr durch ein Password geschützt.

Fehlt der Parameter "new lib name", so wird nur das Password der Bibliothek geändert.

BEMERKUNGEN:
- Ein Password ist ein String mit 1 bis 6 beliebigen Zeichen (außer Leerzeichen).
- Wird für "lib ref" nur "lib name" angegeben, so wird die Bibliothek auf der Systemdisk gesucht.
- Eine offene Bibliothek ist auch nach Ausführung des Dienstprogrammes noch offen.
- Die Liste der bei jeder Systeminitialisierung automatisch geöffneten Bibliotheken wird nicht geändert und muß allenfalls mit LBSTORE korrigiert werden.

- Soll eine bisher nicht mit einem Password geschützte Bibliothek ein Password erhalten, so muß als "password1" das Systempassword angegeben werden. Wurde kein Systempassword vorgegeben, so kann eine Bibliothek nachträglich kein Password mehr erhalten.
- Da bei EFDOS-Systemen kein Systempassword möglich ist, ist die nachträgliche Vergabe eines Passwords an eine Bibliothek nur durch Kopieren mit LIBCOPY zu erreichen.

BEISPIEL:

```
LUT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT=256 BYTE * DATE:01-08-80


     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
     P6FSYS   07-05-80   001001        073026        1898       1887
FREE SPACE (TOTAL) = 0     SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 0     SECTORS

EXE LBR,(P6FSYS,F1),,LIB1
READY

LUT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT=256 BYTE * DATE:01-08-80


     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
     LIB1     07-05-80   001001        073026        1898       1887
FREE SPACE (TOTAL) = 0     SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 0     SECTORS
```

LBSCRATCH

DIENST-
PROGRAMM: LBSCRATCH (library scratch)

FUNKTION: Löschen einer Bibliothek.

FORMAT: EXE[C] LBS[CRATCH], lib ref[,password]

"lib ref" kann folgende Formen annehmen:
- (lib name,unit name)
- lib name
Als "lib name" ist der Bibliotheksname, als "unit name" der Name der Disk anzugeben.

"password" Password der zu löschenden Bibliothek.

WIRKUNG: Die mit "lib ref" bestimmte Bibliothek wird gelöscht.

BEMERKUNGEN:
- Ist die zu löschende Bibliothek mit einem Password versehen, so muß dieses angegeben werden.
- Ist in "lib ref" nur "lib name" angegeben, so wird die entsprechende Bibliothek auf der Systemdisk gesucht.
- Wird eine Bibliothek auf einer Disk gelöscht, so ist der freigewordene Speicherplatz erst nach Ausführung von RESTRUCT wieder verwendbar.

  Bei der Ausgabe der Bibliotheksübersicht mittels LVT erfolgt deshalb die Ausgabe des freien Speicherplatzes getrennt nach gelöschten und nicht gelöschten Bibliotheken.
- Ist eine mit diesem Dienstprogramm gelöschte Bibliothek in der Liste der bei jeder Systeminitialisierung automatisch geöffneten Bibliotheken, so wird diese Liste nicht geändert. Sie ist gegebenenfalls mit LBSTORE neu festzulegen.

BEISPIEL:

```
LUT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT=256 BYTE * DATE:01-08-80


     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
     LIB1     07-05-80   001001        073026        1898       1887
FREE SPACE (TOTAL) = 0     SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 0     SECTORS


EXE LBS,(LIB1,F1)
READY

LUT F1
     MDOSC -R 2.0 * VOLLABEL=DISK1  * TRACK FORMAT=256 BYTE * DATE:01-08-80


     LIBRARY  CREAT      BEG OF EXT    END OF EXT    SECTORS    EMPTY SECTORS
FREE SPACE (TOTAL) = 1898  SECTORS
MAXIMUM BLOCK OF FREE SPACE   = 1898  SECTORS
```

DIENST-
PROGRAMM: LIBCOPY (library copy)

FUNKTION: Kopieren einer (Teil-)Bibliothek in eine andere Bibliothek.

FORMAT:

EXE[C] LIB[COPY], IN=[lib ref],[password$_1$],[{*, +, :}],
OUT=[lib ref],[password$_2$][,filename]

"lib ref"	kann folgende Formen annehmen: - (lib name,unit name) - lib name - (,unit name) Als "lib name" ist der Name der Bibliothek, als "unit name" der Name Diskstation anzugeben.
"IN=lib ref"	bestimmt die Bibliothek, aus der eine oder mehrere Teilbibliotheken kopiert werden.
"password$_1$"	Password der mit "IN=lib ref" bestimmten Bibliothek.
"*"	Eine Package-Teilbibliothek soll kopiert werden.
"+"	Eine Common-Teilbibliothek soll kopiert werden.
":"	Alle Teilbibliotheken sollen kopiert werden.
"OUT=lib ref"	bestimmt die Empfängerbibliothek.
"password$_2$"	Password der Empfängerbibliothek.
"filename"	bestimmt, ab welchem File mit Kopieren begonnen werden soll (siehe CATALOG).

WIRKUNG: Die spezifizierten Teilbibliotheken der mit "IN=lib ref" bestimmten Bibliothek werden in die mit "OUT=lib ref" definierte Bibliothek kopiert. Die Files der Teilbibliotheken werden in der gleichen Reihenfolge kopiert, wie sie in der durch CATALOG erzeugten Liste aufgeführt sind. Ist der Parameter "filename" angeführt, so wird der Kopiervorgang erst mit diesem File gestartet.

BEMERKUNGEN:

- Die angesprochenen Bibliotheken müssen nicht offen sein.
- Wird in einem Operand für "lib ref" nur "lib name" gesetzt, bezieht sich das Dienstprogramm auf die entsprechende Bibliothek der Systemdisk.
- Wird in einem Operand für "lib ref" nur "unit name" spezifiziert, so bezieht sich das Dienstprogramm auf die erste Bibliothek (siehe LVT) der angegebenen Einheit.
- Fehlt "lib ref" im Parameter "IN=", so wird aus der ersten Bibliothek der Systemdisk kopiert.

  Fehlt "lib ref" im Parameter "OUT=", so erfolgt die Kopie in die erste Bibliothek der Systemdisk.
- Sind die angesprochenen Bibliotheken mit einem Password versehen, so ist dieses anzugeben.
- Fehlt der dritte Operand (*/+/:), wird die User-Teilbibliothek kopiert.
- Durch diesen Kopiervorgang werden alle Files in einem Abschnitt abgespeichert (siehe CATALOG, Kolonne EXT). Eine Ausnahme bildet das Kopieren auf User-Disketten, wo allenfalls ein File in maximal zwei Teilen gespeichert sein kann.
- Eine mit LBPROTECT geschützte Package-Teilbibliothek kann nur kopiert werden, wenn die Empfängerbibliothek noch keine Files enthält. Ein nur teilweises Kopieren unter Verwendung des Parameters "filename" ist nicht möglich.

MELDUNGEN:

RESTORE DISKS
Der Kopiervorgang ist abgeschlossen. Disketten können gewechselt werden. Die Taste SHIFT EXIT (CONTINUE) führt in den Command-Mode zurück.

LOAD DISK ON unit name1 [unit name2]
Die angegebene Bibliothek (ev. beide) ist in der Einheit "unit name" nicht vorhanden. Der Kopiervorgang ist aufgeschoben. Die entsprechenden Disketten können gewechselt werden. Der Kopiervorgang wird mit CONTINUE aktiviert oder mit BREAK endgültig abgebrochen (z.B. falscher Bibliotheksname).

ERROR n - RESTORE DISKS
Der Kopiervorgang ist aufgrund der Fehlermeldung n abgebrochen worden. Je nach Fehlercode (siehe Anhang) sind Disketten auszuwechseln. Mit SHIFT EXIT (CONTINUE) wird der Command-Mode wieder erreicht.

ERROR n ON SUBLIB $\left\{\begin{matrix}*\\:\\NP\end{matrix}\right\}$

Der Kopiervorgang in die angegebene Teilbibliothek (NP steht für User-Teilbibliothek) mußte abgebrochen werden, da in der Directory nicht genügend Platz für die Eintragung aller Filenamen vorhanden war. Werden im Aufruf des Dienstprogrammes noch Kopien weiterer Teilbibliotheken gefordert, so wird die Arbeit mit der nächsten Teilbibliothek fortgesetzt.

filename NOT COPIED
Der Kopiervorgang wurde für die das File "filename" enthaltende Teilbibliothek abgebrochen, da in der Empfängerbibliothek kein Platz mehr vorhanden war. Das File "filename" und alle Files, die gemäß CATALOG in der Fileliste der entsprechenden Teilbibliothek nachfolgend angeführt sind, wurden nicht kopiert.

NO LIBRARY lib name ON UNIT unit name
Die Liste der Bibliotheken, die bei jeder Systeminitialisierung automatisch geöffnet werden, stimmt mit den vorhandenen Bibliotheken nicht mehr überein, da das Dienstprogramm für den Kopiervorgang einen Diskettenwechsel verlangt hatte.

BEISPIEL:

Inhalt der Bibliothek LIB1:

```
CAT :,(,F1),,F

     EFDOSC-R 3.1    *  VOLLABEL = DISK1   *  LIBRARY = LIB1    *  DATE:01-08-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      FILE1    S  01-08-80  01-08-80   50048       0000                1
      FILE2    R  01-08-80  01-08-80   150016      150016              1
      FILE77   S  01-08-80  01-08-80   35072       0000                2
```

Inhalt der Bibliothek P6FSYS:

```
CAT :,(,F2),,F

     EFDOSC-R 3.1    *  VOLLABEL =         *  LIBRARY = P6FSYS  *  DATE:01-08-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      TEST     S  07-05-80  07-05-80   1024        0000                1
      TEST2    P  07-05-80  07-05-80   0384        0384                1
      TEST3    P  07-05-80  07-05-80   0384        0384                1
      DATA7    R  07-05-80  01-08-80   5120        5120                1
```

Kopieren der Bibliothek P6FSYS (alle Teilbibliotheken) in die Bibliothek LIB1. Das File DATA7 hat keinen Platz in LIB1!

```
EXE LIB,IN=(,F2),,:,OUT=(,F1)
 DATA7   NOT COPIED
ERROR  188
```

Neuer Inhalt der Bibliothek LIB1:

```
CAT :,(,F1),,F

     EFDOSC-R 3.1    *  VOLLABEL = DISK1   *  LIBRARY = LIB1    *  DATE:01-08-80

     FILE    TYPE  CREAT     LAST MOD  MAX SIZE  USED SIZE   CODE      EXT

      FILE1    S  01-08-80  01-08-80   50048       0000                1
      FILE2    R  01-08-80  01-08-80   150016      150016              1
      FILE77   S  01-08-80  01-08-80   35072       0000                2
      TEST     S  07-05-80  07-05-80   1024        0000                1
      TEST2    P  07-05-80  07-05-80   0384        0384                1
      TEST3    P  07-05-80  07-05-80   0384        0384                1
```

Kontrolle: Freier Platz in LIB1 kleiner als File DATA7.

```
SPA (,F1)
SPACE = 2944
```

RESTRUCT

DIENST-
PROGRAMM: RESTRUCT

FUNKTION: Neuorganisation eines Diskinhaltes nach dem Löschen von Bibliotheken.

FORMAT: EXE[C] RES[TRUCT][,unit name]

"unit name" Name der Disk

WIRKUNG: Der Speicherplatz von mit LBSCRATCH gelöschten Bibliotheken einer Disk wird wieder freigegeben. Dabei entsteht eine neue, lückenlose Bibliotheksorganisation.

BEMERKUNGEN:
- Fehlt der Parameter "unit name", ist die Systemdisk angesprochen.
- Offene Bibliotheken bleiben offen, jedoch wird die Liste der bei einer Initialisierung des Systems automatisch geöffneten Bibliotheken gelöscht. Sie muß mit LSBSTORE neu definiert werden.

MELDUNGEN: LIBRARY lib name OK
Der Name jeder bei der neuorganisation korrekt übernommenen Bibliothek wird in obiger Form ausgelistet.

ERROR n LIBRARY lib name DELETED
Die Ausführung des Dienstprogrammes wurde aufgrund des Fehlercodes n (siehe Anhang) gestoppt. Die mit "lib name" angegebene Bibliothek ist zerstört. RESTRUCT kann mit der Funktion CONTINUE fortgesetzt oder mit der Funktion BREAK endgültig abgebrochen werden.

VOLLABEL

DIENST-
PROGRAMM: VOLLABEL (volume label)

FUNKTION: Kennzeichnung einer Disk.

FORMAT: EXE[C] VOL[LABEL], unit name, volume label

"unit name" Name der Einheit, deren Disk oder Disketten einen "volume label" erhalten soll.

"volume label" Name aus 1 bis 6 Großbuchstaben oder Ziffern. Das erste Zeichen muß ein Buchstabe sein.

WIRKUNG: Die in der mit "unit name" angesprochenen Einheit enthaltene Disk erhält den spezifizierten Volume-Label.

BEISPIEL:

```
CAT

     MDOSC -R 2.0    *  VOLLABEL =          *  LIBRARY = P6FSYS  *  DATE:01-08-80
     FILE    TYPE  CREAT    LAST MOD  MAX SIZE  USED SIZE   CODE      EXT




EXE VOL,F2,DISK2
READY

CAT

     MDOSC -R 2.0    *  VOLLABEL = DISK2   *  LIBRARY = P6FSYS  *  DATE:01-08-80
     FILE    TYPE  CREAT    LAST MOD  MAX SIZE  USED SIZE   CODE      EXT
```

8.	PROGRAMMIERUNG IN BASIC	8.1
8.1	Einführung in die Sprache BASIC	8.1
8.1.1	Struktur eines BASIC-Programmes	8.1
8.1.2	Funktionelle Zusammenhänge von BASIC-Anweisungen	8.3
8.1.3	BASIC-Zeichen	8.8
8.1.4	Zahlendarstellung	8.9
8.1.5	Konstanten und Variable	8.11
8.1.5.1	Konstanten	8.11
8.1.5.2	Numerische und Stringvariable	8.12
8.1.5.3	Felder (indizierte Variable)	8.13
8.1.6	Ausdrücke und Vergleichsoperatoren	8.16
8.1.6.1	Ausdrücke	8.16
8.1.6.2	Regeln für arithmetische Operatoren	8.16
8.1.7	Vergleichs- und Boole'sche Operatoren	8.18
8.1.7.1	Vergleichsoperatoren	8.18
8.1.7.2	Boole'sche Operatoren	8.18
8.2	Speicheraufteilung	8.19
8.2.1	Gliederung des Anwenderspeichers	8.19
8.2.2	Common-Bereich	8.21
8.3	Ablauforganisation 8.23	
8.4	Verbindung mit Assembler P6066	8.26
8.4.1	Assembler-Sprache	8.26
8.4.2	Verarbeitung durch ein BASIC-Programm	8.27
8.5	Standardformat	8.27
8.5.1	Zahlendarstellung	8.28
8.5.2	Darstellung von Strings	8.29
8.5.3	Stellenkontrolle bei den Anweisungen DISP und PRINT	8.30
8.5.4	Besonderheiten der Anweisung DISP	8.32

8. PROGRAMMIERUNG IN BASIC

8.1 Einführung in die Sprache BASIC

Die Programmiersprache BASIC ist die in der mittleren Datentechnik am weitesten verbreitete Sprache. Der Grund dafür dürfte nicht zuletzt in der leichten Programmierbarkeit liegen.

Das System bietet ein sehr komfortables BASIC, das in Form eines "inkrementellen Compilers" implementiert ist. Dieser hat gegenüber einem BASIC-Interpreter den Vorteil, daß neben der Syntax einer BASIC-Anweisung auch Kontrollen bezüglich der Verträglichkeit der verschiedenen Anweisungen durchgeführt werden, und zwar vor Ausführung des Programmes. Zum Beispiel wird ein fehlendes Sprungziel bereits vor dem Programmstart erkannt. Dank dieser Methode werden bei der Programmerstellung extrem lange Testzeiten vermieden.

Bevor in Abschnitt 8.1.2 auf die funktionellen Zusammenhänge von BASIC-Anweisungen eingegangen wird, soll Abschnitt 8.1.1 die Struktur eines BASIC-Programmes aufgezeigen. Die Eingabe und das Editing von Programmen wird in Kapitel 10 beschrieben.

8.1.1 Struktur eines BASIC-Programmes

Ein BASIC-Programm besteht aus einer Folge von Anweisungen, die mit einer vierstelligen Zeilennummer beginnen. Es wird zwischen ausführbaren und nicht ausführbaren Anweisungen unterschieden:

- Die ausführbaren Anweisungen bewirken eine Aktion während des Programmlaufes, wie zum Beispiel die Zuweisung eines Wertes an eine Variable oder die Ausgabe von Daten über den Drucker oder auf den Bildschirm.
- Nicht ausführbare Anweisungen beschreiben die für das Programm notwendigen Informationen, bewirken jedoch während des Programmlaufes keine sichtbare Aktion. Da sie vor dem Programmlauf in der Preexecution behandelt werden, können sie an beliebiger Stelle innerhalb des Programmes stehen. (Ausnahme: DCL innerhalb einer mehrzeiligen Funktion, siehe auch BASIC-Anweisung DEF).

  Als Beispiel können folgende, nicht ausführbare Anweisungen angeführt werden:

COMMON	(common)	Definition des Common-Bereiches.
DCL	(declare)	Vereinbarung von Variablen.
DIM	(dimension)	Dimensionierung von indizierten Variablen.

Ein Programm wird sinnvollerweise in drei Teile gegliedert. Im ersten Teil werden die nicht ausführbaren Anweisungen aufgeführt, das heißt, der Common-Bereich wird definiert, die Variablen mit Länge, Dimension und Genauigkeit werden bestimmt und die zu benützenden Datenfiles festgelegt. Im zweiten Teil wird die eigentliche Programmroutine geschrieben und im dritten Teil alle verwendeten Subroutinen und mehrzeiligen Funktionen angeführt.

Die Anzahl Anweisungen eines Programmes ist einerseits beschränkt durch die Speicherkapazität des Systems und andererseits durch die Zeilennummern, die ganze Zahlen zwischen 1 und 9999 sein müssen. Jede Anweisung eines BASIC-Programmes entspricht einer Zeile. Durch die Zeilennummer wird (bei ausführbaren Anweisungen) die Reihenfolge der Verarbeitung im Programm festgelegt. Ausnahmen bilden Sprünge, die durch Anweisungen, wie GOTO, GOSUB, IF...THEN, FOR...NEXT, INTERRUPT, oder Funktionsaufrufe ausgelöst werden. Die Reihenfolge der Eingabe der Programmzeilen spielt dabei keine Rolle.

Eine Programmzeile kann inkl. Zeilennummer maximal 80 Zeichen enthalten.

```
FILE


0010 REM *** BEISPIEL EINER PROGRAMMGLIEDERUNG ***
0020 REM
0030 REM *** SPEICHERORGANISATION ***
0040 FILES DATEN1,SFILE2
0050 DCL S(A,D2,Y)
0060 DCL ↑0A$,64(X$,X1$)
0070 DIM R(8,12)
0080 REM
      :
1000 REM *** EINGABE/VERARBEITUNG ***
1100 INPUT A,Y
1200 GOSUB 9010
1300 INPUT D2
1310 GOSUB 9100
1320 REM
1330 REM
1520 RKB X$
1530 GOSUB 9510
1540 FOR I=1 TO 8 STEP 1
1550 FOR J=1 TO 12 STEP 1
1560 INPUT R(I,J)
1570 REM
      :
5000 REM *** AUSGABE ***
5010 PRINT USING 8010,A,B2,Y
5020 PRINT
5030 PRINT USING 8400,EXT$(X$,D2,P),A$
5040 REM
5050 REM
      :
8000 REM *** PRINT-FORMATE ***
8010 : ###.###    #####.##    ####.######
8020 REM
8400 :    > 'LLLLLLLLLLLLLLLLL < BIS > 'LLLLLLLLLLLLLLLLLL' <
8500 REM
8510 REM

9000 REM *** UP1 ***
9010 REM
9020 REM
9100 REM *** UP2 ***
9999 END

END OF LISTING
```

8.1.2 Funktionelle Zusammenhänge von BASIC-Anweisungen

Im folgenden Verzeichnis werden alle BASIC-Anweisungen in ihrem Zusammenhang aufgelistet und durch eine knappe Beschreibung erläutert.

Vereinbarungen für Speicherplatzzuweisungen

(nicht ausführbare Anweisungen)

DCL	legt fest, welche Variablen mit einfacher Genauigkeit verarbeitet werden sollen und bestimmt die maximale Länge von Strings.
DIM	legt die Anzahl Elemente für numerische und alphanumerische Felder fest.
COMMON	bestimmt den Common-Bereich. Variableninhalte im COMMON-Bereich des Arbeitsspeichers bleiben erhalten, wenn mit CHAIN ein anderes Programm von Disks geladen wird.

Zuweisung

LET	weist das Ergebnis eines Ausdruckes einer numerischen oder einer Stringvariablen zu.

Programmverzweigungen

FOR/NEXT	dient zur Bildung von Schleifen in Abhängigkeit von einer Laufvariablen.
GOTO	unbedingter Sprung zu einer Zeilennummer.
IF/THEN	bedingter Sprung zu einer Zeilennummer.
ON...GOTO	bedingter Sprung zu einer Zeilennnummer in Abhängigkeit vom Wert eines Ausdruckes.

Unterprogramme und definierte Funktionen

CALL	Aufruf einer Assembler-Routine
DEF/FNEND	dient zur Vereinbarung mehrzeiliger Funktionen.
GOSUB	Aufruf eines Unterprogrammes.
ON...GOSUB	Aufruf von Unterprogrammen in Abhängigkeit vom Wert eines Ausdruckes.
RETURN	markiert das logische Ende eines Unterprogrammes.

Anweisungen zur Ein-/Ausgabe und für interne Files

DATA	baut ein internes Datenfile auf.
DISP	Ausgabe von Daten im Standardformat über das Display.
DISP USING	Ausgabe von Daten in definiertem Format über das Display.
ERASE	Löschen des Bildschirminhaltes.

: (IMAGE)	Festlegung eines Formates in einer Zeile, die dann mit "...USING Zeilennummer" aufgerufen wird.
INPUT	ermöglicht die Eingabe von numerischen und/oder alphanumerischen Daten über die Tastatur.
PRINT	Ausgabe über den Drucker im Standardformat.
PRINT USING	Ausgabe über den Drucker in definiertem Format.
READ	Lesen von Daten aus einem internen File.
RESTORE	Setzen des Pointers auf das erste Element des internen Datenfiles.
REVERSE	Umkehren des Bildschirminhaltes von positiv in negativ und umgekehrt.
RKB	Eingabe einer beliebigen Zeichenfolge über die Tastatur, wobei alle Zeichen akzeptiert werden.

Anweisungen für Strings

ASSIGN	Die in einem String durch einen Delimiter begrenzten Teilstrings werden einer Liste von numerischen oder alphanumerischen Variablen zugeordnet.
BASSIGN	In einem String in internem Format enthaltene Daten werden auf eine Liste von numerischen oder Stringvariablen aufgeteilt.
BBUILD	Zuweisung der Ergebnisse einer Liste von numerischen oder Stringausdrücken an eine Stringvariable in internem Format.
BPAD	Eine Stringvariable wird mit binären Zeichen bis zur deklarierten Länge aufgefüllt.
BUILD	Die Werte einer Liste von Ausdrücken werden einer Stringvariablen im Standardformat zugewiesen, wobei ein Trennzeichen eingefügt werden kann.
BUILD USING	Die Werte einer Liste von Ausdrücken werden einer Stringvariablen in definiertem Format zugewiesen.
CONVERT	1. Die ISO-Codes der Zeichen eines Stringausdruckes werden der Reihe nach den Elementen eines Vektors zugewiesen. 2. Die Werte eines Vektors werden als ISO-Zeichen interpretiert und der Reihe nach einer Stringvariablen zugewiesen.
DEPAD	entfernt die Füllzeichen aus einer Stringvariablen.
PAD	füllt eine Stringvariable bis zur deklarierten Länge mit ISO-Zeichen auf.

Anweisungen für externe Files

APPEND	erlaubt das Anfügen von Daten an ein bestehendes sequentielles Datenfile.
CHAIN	beendet die Programmausführung und startet ein anderes Programm.
FILES	legt die maximale Anzahl Files fest, die bei der Ausführung des Programmes gleichzeitig geöffnet sein können und öffnet die, deren Namen in der Anweisung aufgeführt sind.
FILE:	schließt das dem Filedesignator zugeordete File und erlaubt das Öffnen eines anderen Files unter Beibehaltung des Filedesignators.
READ:	erlaubt das Lesen von Daten aus einem Text- oder Datenfile.
RESTORE:	setzt in einem Text- oder sequentiellen Datenfile den Pointer auf das erste Element und erlaubt anschließend das Lesen ab Beginn.
SCRATCH:	setzt den Pointer in einem sequentiellen Datenfile an den Beginn und versetzt das File in den Schreib-Mode.
SETW:	setzt den Pointer eines Randomfiles (R- oder Z-File) auf das angegebene Wort.
WHERE:	Abfrage der aktuellen Position, des Typs des adressierten Datenelementes und der Länge eines Strings.
WRITE:	schreibt Daten in das angegebene Datenfile.

Spezielle Anweisungen

BEEP	programmierbares, akustisches Signal.
DELAY	bewirkt die Unterbrechung der Programmausführung während einer bestimmten Zeit.
END	bezeichnet das physische Ende eines Programmes.
FKEY	erlaubt die Belegung einer Funktionstaste mit einer Zeichenfolge.
INTERRUPT ENABLE	ermöglicht die Fehlerbehandlung durch den Anwender.
RANDOMIZE	Bei Aufruf der Funktion RND wird eine von der Standardfolge verschiedene Folge von Zufallszahlen erzeugt.
REM	ermöglicht das Einfügen von Kommentaren in ein Programm.
STOP	unterbricht die Programmausführung und bewirkt den Übergang in den Debugging-Mode.

TRACE ON	bewirkt den Ausdruck der Zeilennummern in der Reihenfolge ihrer Verarbeitung.
TRACE OFF	hebt die Wirkung der Anweisung TRACE ON und der Konsoltaste TRACE auf.

Anweisungen für Matrizenoperationen

MAT...=	elementweise Zuweisung der Werte einer Matrix an eine andere.
MAT...±	Matrizenaddition oder -subtraktion.
MAT...*	Multiplikation einer Matrix mit einem Skalar oder Multiplikation zweier Matrizen.
MAT...CON	setzt alle Elemente einer Matrix auf 1.
MAT...IDN	erzeugt eine Einheitsmatrix.
MAT INPUT	zeilenweise Eingabe der Werte einer Matrix über die Tastatur
MAT...INV	bildet die Inverse einer Matrix.
MAT READ	elementweise Zuweisung von Daten aus einem internen File an eine oder mehrere Matrizen.
MAT READ:	elementweise Zuweisung von Daten aus einem externen File an eine oder mehrere Matrizen.
MAT PRINT	Ausdruck einer oder mehrerer Matrizen im Standardformat.
MAT PRINT USING	Ausdruck einer oder mehrerer Matrizen in definiertem Format.
MAT...TRN	bildet die Transponierte einer Matrix.
MAT WRITE:	Schreibt die Elemente einer oder mehrerer Matrizen in ein Datenfile.
MAT...ZER	erzeugt eine Nullmatrix.

Anweisungen der Option GDI/PLOT

CPLOT	Ausgabe der Ergebnisse von Stringausdrücken über den Plotter oder Bildschirm.
CSIZE	Definition der Größe und Schreibrichtung für die Ausgabe von Strings über Plotter oder Bildschirm.
CTAB	Verschiebung um eine bestimmte Zeichen- und Zeilenanzahl für die Ausgabe von Strings.
DOT	Zeichnen eines durch Koordinaten definierten Punktes.
DRAW	Ausgabeanweisung für das Bild über den integrierten Drucker.
ERASE	Löschen des Bildschirminhaltes (gilt nicht für OPT PLO).

EXTERNAL PLOTTER	Vorwahl für einen externen peripheren Plotter.
FRAME	Festlegung der Größe der Zeichenfläche und des Rahmens für das Bild.
IDOT	Zeichnen eines Punktes durch Angabe der Relativkoordinaten dx und dy.
INIMAGE	Angabe des Namens des für die Speicherung des Bildes vorgesehenen Datenfiles (nur OPT PLO), Aufbau des Buffers im Arbeitsspeicher und Laden der für die Plotoperation notwendigen Parameter.
IPLOT	Ziehen einer Verbindungslinie zu dem durch die Relativkoordinaten dx und dy erreichten Punktes.
LDIMAGE	Definition eines Plotfiles, Übernahme des letzten gespeicherten Bildes mit den Parametern in den Arbeitsspeicher oder auf den Bildschirm.
MOVE	Positionieren auf einen durch Koordinaten definierten Punkt.
OFFSET	Verschieben des Koordinatenursprunges im Koordinatensystem.
PLOT	Ziehen einer Linie zu dem durch Koordinaten definierten Punkt.
POINTER	Übernahme von Koordinaten vom Bildschirm durch manuellen Eingriff (gilt nicht für OPT PLO).
REVERSE	Umkehrung des Bildschirminhaltes von positiv in negativ und umgekehrt (gilt nicht für OPT PLO).
SCALE	Festlegen der Maßstäbe im Koordinatensystem durch Minimum und Maximum der beiden Achsen. Dadurch werden die Maßstabfaktoren impliziert und der Koordinatenursprung definiert.
STIMAGE	Speicherung des Bildschirminhaltes in einem sequentiellen Datenfile (gilt nicht für OPT PLO).
XAXIS	Zeichnen einer parallel zur X-Achse liegenden Achse.
YAXIS	Zeichnen einer parallel zur Y-Achse liegenden Achse.

Anweisungen zur Programmierung von peripheren Einheiten

BUFFER	Im Arbeitsspeicher wird für einen I/O-Kanal ein Buffer für den Datenaustausch mit einer peripheren Einheit reserviert.
CMD	Senden von Steuer- und Prüfbefehlen.
INTERRUPT ENABLE	externer Interrupt infolge einer Kondition eines peripheren Kanales.
RECEIVE	String-Input von der peripheren Einheit in den Arbeitsspeicher.
SEND	String-Output aus dem Arbeitsspeicher an eine Ausgabeeinheit.
TEST	Übertragung des Status der peripheren Einheit ins Arbeitsregister. Laufende I/O-Operationen mit dieser Einheit werden nicht unterbrochen.
WAIT	Das Ende der laufenden I/O-Operation wird abgewartet und der Status der Peripherie ins Arbeitsregister übertragen.

8.1.3 BASIC-Zeichen

Die Sprache BASIC hat einen bestimmten Zeichen- oder Wortvorrat und unterliegt gewissen Syntaxregeln. Unter diesem Aspekt sind alle verwendeten Anweisungen, Daten und Variablen zu sehen.

Die in BASIC vorkommenden Zeichen sind unterteilt in:

- alphabetische,
- numerische und
- Sonderzeichen.

Alphabetische Zeichen:

Dazu gehören die Großbuchstaben des lateinischen Alphabets.

Numerische Zeichen:

sind die Ziffern 0 - 9.

Sonderzeichen:

sind in folgender Tabelle zusammengefaßt:

	Name		Name
	Leerzeichen (blank)	;	Strichpunkt
=	Gleichheitszeichen oder Zuweisung	.	Schlußpunkt oder Dezimalpunkt
+	Additionszeichen	:	Doppelpunkt
-	Subtraktionszeichen	%	Prozentzeichen
*	Sternchen oder Multiplikationszeichen	?	Fragezeichen
		<	größer als
/	Schrägstrich oder Divisionszeichen	>	kleiner als
		,	Komma
↑	Potenzierung	"	Anführungszeichen
(	öffnende Klammer	#	Nummernzeichen
)	schließende Klammer	$	Dollarzeichen

Bemerkung

Leerzeichen (blanks) können in einem Programm beliebig gesetzt werden, um die Lesbarkeit zu erhöhen. Es ist jedoch zu beachten, daß Leerzeichen in folgenden Fällen signifikant sind:

- innerhalb von alphanumerischen Konstanten (Strings);
- in Formatanweisungen, die das Format der Ausgabedaten auf Drucker und Display spezifizieren.

Nicht zulässig sind Leerzeichen an folgenden Stellen:

- innerhalb einer Zeilennummer,
- innerhalb von BASIC-Wörtern,
- innerhalb von Variablennamen und Funktionen,
- innerhalb von numerischen Konstanten.

8.1.4 Zahlendarstellung

Der numerische Wert der Zahlen wird im Dezimalsystem dargestellt.

Zahlenbereich

Unter der Größe einer Zahl ist ihr Absolutwert zu verstehen. Der Zahlenbereich des Systems umfaßt alle Zahlen, die

größer oder gleich 10^{-99} und

kleiner oder gleich $9.999999999999*10^{99}$ sind.

Genauigkeit

Die Genauigkeit einer Zahl bedeutet die maximale Anzahl signifikanter Ziffern, aus denen die Zahl bestehen kann. Das System kann mit einfacher oder doppelter Genauigkeit arbeiten. Wird für die Darstellung der Zahl die allgemeine Form "$n.10^m$" verwendet, so gilt:

einfache Genauigkeit (6 signifikante Ziffern)	doppelte Genauigkeit (13 signifikante Ziffern)
$1 \leq n \leq 999999$	$1 \leq n \leq 9999999999999$
$-63 \leq m \leq +63$	$-99 \leq m \leq +99$

Wird nichts anderes vereinbart, arbeitet das System generell mit doppelter Genauigkeit (Festlegung der einfachen Genauigkeit siehe Anweisung DCL, Kapitel 9).

Externe Darstellung von Zahlen

Zahlen können als

- ganze Zahlen,
- Dezimalzahlen in Festkommadarstellung oder
- Dezimalzahlen in Gleitkommadarstellung

ein- oder ausgegeben werden. Die Wahl der Darstellung ist abhängig von der Größe der Zahlen und der erforderlichen Genauigkeit. Die Zahlen können positiv oder negativ sein. Negative Zahlen werden durch ein Minuszeichen vor der Zahl dargestellt, positiven Zahlen kann ein Pluszeichen vorangestellt werden.

- Ganze Zahlen:
  Sie können bis zu 13 Ziffern aufweisen und mit einem Vorzeichen versehen sein. Bei positiven Zahlen muß das Vorzeichen nicht gesetzt werden.

  Beispiele: 0
  +4038
  -80

- Dezimalzahlen in Festkommadarstellung:
  Sie können bis zu 13 Stellen aufweisen und mit einem Vorzeichen versehen sein. Dem ganzzahligen Teil folgen ein Dezimalpunkt und bis zu 12 Nachkommastellen. Ein positives Vorzeichen kann entfallen.

  Beispiele: 58.6
  20
  -.408
  +.10
  +32.28147

-	9999999999999	0.0000000000001
+	0.0000000000001	9999999999999

- Dezimalzahlen in Gleitkommadarstellung:
  Sie bestehen aus einem Vorzeichen, gefolgt von einer ganzen oder einer Dezimalzahl (Mantisse), welcher der Buchstabe E (Exponent) angehängt wird. Die Zahl hinter dem Buchstaben E gibt die Zehnerpotenz an, mit der die Mantisse multipliziert wird, und besteht aus maximal zwei Ziffern, die mit einem Vorzeichen versehen sein können. Die Mantisse kann maximal 13 signifikante Ziffern enthalten. Sowohl bei der Mantisse als auch beim Exponenten kann ein positives Vorzeichen entfallen.

  Beispiele:

Gleitkomma	Festkomma
.99E-5	0.0000099
+1.0E+10	10000000000
9E-10	0.0000000009

Wahl der Zahlendarstellung

Eine Zahl kann in jeder der drei beschriebenen Zahlendarstellungen über die Tastatur eingegeben werden. Die Zahl "9 Millionen" kann beispielsweise wie folgt dargestellt werden:

```
9 000 000
9 000 000 . 000
9E +6
```

Der Zahlenbereich des Absolutbetrages liegt zwischen 1E-99 und 9.999999999999E+99. Werden mehr als 13 signifikante Stellen eingegeben, so erscheint im Display eine Fehlermeldung.

Die Zahlendarstellung der Daten auf dem Display und auf der Ausgabeeinheit kann im Programm festgelegt werden (siehe auch Befehle DISP USING, PRINT USING).

8.1.5 Konstanten und Variable

8.1.5.1 Konstanten

- Numerische Konstanten
  Eine numerische Konstante ist eine ganze oder eine Dezimalzahl in Fest- oder Gleitkommadarstellung, deren Wert während der Programmausführung unverändert bleibt. Sie wird immer mit doppelter Genauigkeit behandelt.

  Beispiele:
  ```
  X = Y/95
  Z = 84.9
  Y = 2*X-1.2E+4
  ```

  worin 95, 84.9, 2 und 1.2E+4 Konstanten sind.

- Konstante π:
  Die Zahl π = 3.141592 654590 ist als interne Konstante in doppelter Genauigkeit vorhanden. Sie kann mit dem Namen PI aufgerufen und im Zusammenhang mit numerischen Ausdrücken verwendet werden.

Beispiele: PI*2
SIN (PI/4)

- Stringkonstanten
  Eine Stringkonstante besteht aus einer Folge von Zeichen, die von Anführungszeichen eingeschlossen werden. Das Anführungszeichen selbst ist nicht Bestandteil der Konstante. Zugelassen sind alle Zeichen der ISO-Code-Tabelle mit Ausnahme des Anführungszeichens.

  Beispiele: "DIE FLÄCHE DES TRAPEZES BETRÄGT"
  "0 1 2 3 4"
  " VOLUMEN"

  Unter der Länge einer Stringkonstante versteht man die Anzahl Zeichen innerhalb der Anführungszeichen. Die maximale Länge wird durch die maximale Länge der BASIC-Zeile limitiert und beträgt 74 Zeichen.

8.1.5.2 Numerische und Stringvariable

- Numerische Variable
  Sie sind mit Namen bezeichnete Größen, deren Wert während der Programmausführung verändert werden kann. Numerische Variable werden durch einen beliebigen Großbuchstaben dargestellt, dem eine Ziffer (0-9) folgen kann.

  Beispiele: Z
  Z9
  Z1

  Eine Variable, der noch kein Wert zugewiesen wurde, hat einen "nicht definierten" Wert. Sie wird "nicht initialisierte Variable" genannt. Wird eine solche Variable in einem arithmetischen Ausdruck verwendet, erfolgt eine Fehlermeldung, es sei denn, die Anweisung INTERRUPT ENABLE werde für die Fehlerbehandlung benutzt.

  Wird ohne Eingabe eines anderen Wertes weitergerechnet (durch Drücken der Konsoltaste CONTINUE), erhält die Variablevom System den Wert 0 zugewiesen. Sie gilt aber weiterhin als "nicht definiert".

  Im Maximum können 123 numerische Variable im gleichen Programm verwendet werden.

- Stringvariable
  Stringvariable enthalten eine Folge von zulässigen Zeichen und können während dem Programmablauf verändert werden. Namen von Stringvariablen bestehen aus einem Großbuchstaben des Alphabets (A-Z), gefolgt von einem Dollarzeichen ($), oder aus einem Großbuchstaben, gefolgt von einer Ziffer und dem Dollarzeichen. Die Zeichen können über die Tastatur eingegeben (siehe Anweisungen INPUT, RKB und RECEIVE) oder vom Programm zugewiesen werden (siehe READ, DATA).

Eine Variable ohne Wertzuweisung wird mit dem Wert "nicht definiert" initialisiert und erhält bei der Ausführung des Programmes den Wert "Nullstring". Außerdem wird eine Fehlermeldung ausgegeben.

Stringvariable können bis zu 1023 Zeichen enthalten. Ihre Länge kann mit dem Befehl DCL festgelegt werden (siehe Kapitel 9).

In einem Programm können maximal 256 Stringvariable verwendet werden.

Ist die Länge einer Stringvariablen nicht angegeben, wird sie automatisch auf 16 Zeichen limitiert. Wird einer Stringvariablen ein String zugewiesen, der länger ist als die deklarierte Länge, erfolgt eine Fehlermeldung. Wird das Programm trotzdem mit CONTINUE (Taste SHIFT EXIT) fortgesetzt, wird der String rechts abgeschnitten.

8.1.5.3 Felder (indizierte Variable)

Indizierte Variable bezeichnen ein Feld von Variablen.
Es kann sich dabei sowohl um numerische als auch um Stringvariable handeln.

Ein Feld kann ein- oder zweidimensional sein.

- Ein eindimensionales Feld (Vektor) kann als eine natürliche Folge von Elementen betrachtet werden.
- Ein zweidimensionales Feld ist eine Matrix, bestehend aus Zeilen und Spalten.

Ein Feldname besteht bei numerischen Variablen aus einem Großbuchstaben (A-Z), bei Stringvariablen aus einem Großbuchstaben, gefolgt von einem Dollarzeichen ($).

Ein Feldelement wird bestimmt durch den Namen des Feldes, zusammen mit einem Index bei einem Vektor, mit zwei Indizes bei einer Matrix. Die Indizes geben die Position des Elementes im Feld an.

Beispiele:

- Ist A der Name eines Vektors der Dimension 6, heißen die Elemente:

  A(1), A(2), A(3), A(4), A(5), A(6)

  A(4) ist also die vierte Komponente des Vektors A.

- Bezeichnet Z eine 3x4-Matrix, heißen die Elemente:

Z(1,1)	Z(1,2)	Z(1,3)	Z(1,4)
Z(2,1)	Z(2,2)	Z(2,3)	Z(2,4)
Z(3,1)	Z(3,2)	Z(3,3)	Z(3,4)

  Das Element Z(2,3) ist also das dritte Element der zweiten Zeile.

- Ist Z$ ein Stringvektor der Dimension 3, heißen die Komponenten (Elemente):

  Z$(1), Z$(2), Z$(3).

- Ist A$ eine 2x2 Matrix, heißen die Elemente:

  A$(1,1) A$(1,2)
  A$(2,1) A$(2,2)

Die Indizes können beliebige arithmetische Ausdrücke sein, die einen ganzzahligen Wert zwischen eins und der oberen Feldgrenze ergeben, wobei bei einem nicht ganzzahligen Ergebnis gerundet wird.

Die Dimension eines Feldes (Anzahl Elemente) wird durch den Befehl DIM festgelegt.

Eine Variable und ein Feld können in einem Programm den gleichen Namen tragen; so kann B eine Variable, aber auch ein Feld bezeichnen. Es ist jedoch nicht erlaubt, in einem Programm einem Vektor und einer Matrix denselben Namen zuzuordnen. Der Begriff "initialisierte Variable" bezieht sich bei indizierten Variablen auf die einzelnen Elemente. Diese werden wie einfache Variable behandelt.

Feldvereinbarung

Unter Feldvereinbarung ist die Angabe der Felddimension (ein- oder zweidimensional) und der Anzahl Elemente zu verstehen. Es ist zwischen expliziten und impliziten Feldvereinbarungen zu unterscheiden.

Eine Feldvereinbarung wird mit Hilfe der BASIC-Anweisung DIM getroffen. Fehlt die Vereinbarung für bestimmte Felder im Programm, so werden vom System folgende Standardvereinbarungen angenommen:

	eindimensionales Feld	zweidimensionales Feld
numerisches Feld	10	10x10
Stringfeld	10	5x5

Tabelle: Anzahl Komponenten von Feldern im Standardformat.

Der Speicherplatz der Daten während der Programmausführung ist in der nachfolgenden Tabelle dargestellt:

Datentyp	Platzbelegung			Bemerkungen
	Arbeits-speicher	externer Speicher		
	Bytes	Bytes	Worte	
Numerische Konstante	$4+\frac{N+1}{2}$	8	2	1. N=Anzahl Ziffern der numerischen Konstante. 2. Der Wert des Ausdrucks $\frac{N+1}{2}$ wird auf die nächste ganze Zahl aufgerundet.
Numerische Variable	10			Numerische, einfach genau deklarierte Variable werden schneller bearbeitet als doppelt genaue.
Wert einer numerischen Variablen		4	1	Einfach genau.
		8	2	Doppelt genau.
Matrix oder Vektor	N*4+10			Einfach genau. N=Anz.Elemente der Matrix od. des Vektors
	N*8+10			Doppelt genau.
String-konstante	N+2	4.INT $\frac{N-1}{4}+2$	INT $\frac{n-1}{4}+2$	1. N=Anz. Zeichen des Strings. 2. INT=ganzzahligen Anteil des Ausdrucks bilden.
Stringvariable	N+9			N=deklarierte Länge des Strings.
Wert einer Stringvariablen		siehe Stringkonstante		
Stringarray	13+(L+2)*N			1. L=deklarierte Länge der Elemente. 2. N=Anzahl Elemente des Arrays.

Tabelle: Speicherbedarf von Daten

8.1.6 Ausdrücke und Vergleichsoperatoren

8.1.6.1 Ausdrücke

Ein Ausdruck besteht aus einer Verknüpfung von Konstanten, (indizierten) Variablen und/oder Funktionen.

Für numerische Ausdrücke sind sowohl die Standardfunktionen als auch ein- oder mehrzeilige, vom Anwender definierte Funktionen zulässig (z.B. SIN(X), INT(Y/Z)). Die Verknüpfungsoperatoren für numerische Ausdrücke sind:

Symbol	Bedeutung
↑	Potenzierung
*	Multiplikation
/	Division
+	Addition
-	Subtraktion

Für Stringausdrücke sind sowohl die Standard-Stringfunktionen (wie z.B. REP$, CHR$) als auch einfache oder mehrzeilige, vom Anwender definierte Funktionen zulässig. Der Verknüpfungsoperator für Stringausdrücke ist ausschließlich die Stringaddition (Symbol +).

Beispiele: Numerische Ausdrücke:

```
A1
X/Y+11.5-73*Z1
(X-10)*F
SIN(X*Y)
-97.4
A↑B+INT(ABS(Z))
A-FNB(3,X/2)
```

Stringausdrücke:

```
A$+"MÜLLER"+Z1$
"TEXT: "+R$
```

8.1.6.2 Regeln für arithmetische Operatoren

- Potenzierung:

  B↑E Die Basis B wird zur Potenz E erhoben.

  Sonderfälle:

Basis B	Exponent E	Ergebnis B↑E
0	0	1
0	<0	ERROR 12
0	>0	0
<0	<>INT(E)	ERROR 11
<>0	0	1

- Multiplikation und Addition:

A*B	A wird mit B multipliziert
A+B	B wird zu A addiert.

Regeln:

Es gilt das kommutative Gesetz:

A*B = B*A
A+B = B+A

Das transitive Gesetz gilt nicht uneingeschränkt. Durch die Reihenfolge der Operationen können minimale Abweichungen entstehen:

A*(B*C) ≃ (A*B)*C
A+(B+C) ≃ (A+B)+C

- Division und Subtraktion:

A/B	A wird durch B dividiert.
A-B	B wird von A subtrahiert.

Die Division durch Null führt zur Fehlermeldung ERROR 3 (Overflow).

Erlaubte Vorzeichenkombinationen sind zum Beispiel:

-B+(-A)+C-(-Z)
A+-B
A*-B
A++B

- Prioritätsregeln:

Die numerischen Operatoren haben in nachstehender Reihenfolge Priorität:

↑	*	+
	/	-
höchste Priorität		niedrigste Priorität

Die Operatoren * und / sowie + und - haben jeweils die gleiche Priorität. Operatoren gleicher Priorität werden von links nach rechts verarbeitet, es sei denn, daß durch Klammern etwas anderes definiert wird; Klammern durchbrechen die Prioritätsregeln.

Beispiele: -3↑2=-9 aber (-3)↑2=9
8/4*2=4 aber 8/(4*2)=1

8.1.7 Vergleichs- und Boole'sche Operatoren

8.1.7.1 Vergleichsoperatoren

Operator	Bedeutung
=	gleich
<>, ><	ungleich
>=, =>	größer gleich
<=, =<	kleiner gleich
>	größer
<	kleiner

Allgemeine Form von Vergleichen:

"Ausdruck 1 Vergleichsoperator Ausdruck 2"

Dabei wird der Inhalt zweier Ausdrücke verglichen. Die Ausdrücke 1 und 2 können beide numerische oder beide Stringausdrücke sein.

Werden Zeichenketten verglichen, so erfolgt der Vergleich Zeichen für Zeichen von links nach rechts, entsprechend der Reihenfolge der Zeichen in der ISO-Code-Tabelle.

Werden zwei Zeichenketten verschiedener Länge verglichen, wird vom System der kürzere String mit dem ISO-Zeichen NULL bis zur Länge der längeren Zeichenkette aufgefüllt und danach der Vergleich ausgeführt.

Wird bis zur signifikanten Länge des kürzeren Strings Gleichheit festgestellt, so gilt der längere Ausdruck als der größere.

Beispiele: (Strings)

```
"AB"   ist groesser als "AAAAA"
"2500" ist kleiner als "3"
"1234" ist gleich "1234"
"2x4"  ist gleich "2x4    "
```

8.1.7.2 Boole'sche Operatoren

Ein Vergleich liefert als Ergebnis einen Wert "wahr" oder "unwahr". Die Wahrheitswerte von zwei Vergleichen können durch die Boole'schen Operatoren

AND
OR

verknüpft werden und liefern wieder einen Wahrheitswert. Das allgemeine Format lautet:

(Vergleich 1) $\left\{\begin{matrix}\text{AND}\\\text{OR}\end{matrix}\right\}$ (Vergleich 2)

Die beiden Vergleichsoperatoren "Vergleich 1" und "Vergleich 2" müssen in Klammern eingeschlossen sein und das in 8.1.7.1 beschriebene Format haben.

Die Verknüpfung der beiden Vergleiche mit AND liefert den Wahrheitswert "wahr", wenn sowohl "Vergleich 1" als auch "Vergleich 2" den Wert "wahr" hat.

Die Verknüpfung der beiden Vergleiche mit OR liefert den Wahrheitswert "wahr", wenn mindestens einer der beiden Vergleiche "Vergleich 1" oder "Vergleich 2" den Wert "wahr" hat.

Wird für "wahr" 1 und für "unwahr" 0 geschrieben, sieht die Wahrheitstafel für AND und OR wie folgt aus:

AND

Vergleich 1	1	1	0	0
Vergleich 2	1	0	1	0
(1) AND (2)	1	0	0	0

OR

Vergleich 1	1	1	0	0
Vergleich 2	1	0	1	0
(1) OR (2)	1	1	1	0

Beispiele:

(A>B) OR (C>D) — Der Ausdruck ist "wahr", wenn A größer als B und/oder C größer als D ist.

(A>B) AND (C>D) — Der Ausdruck ist "wahr", wenn sowohl A größer als B als auch C größer als D ist.

8.2 Speicheraufteilung

8.2.1 Gliederung des Anwenderspeichers

Zum besseren Verständnis des Ablaufes eines Programmes soll hier die Speicheraufteilung des Anwenderspeichers kurz skizziert werden.

Beispiel der Aufteilung kurz vor dem Start, aber nach der Preexecution:

I	Options und Configure	Anwender-
II	Common-Bereich	speicher
III	Programm-Code	(z.B. 48K)
IV	Buffer und Datenbereich	
V	Stack	

Der Anwenderspeicher ist in mindestens drei Teilbereiche unterteilt, nämlich:

- Programmcode,
- Buffer und Datenbereich,
- Stack.

Gegebenenfalls werden zwei weitere Speicherabschnitte benötigt, zum Beispiel für Options oder die Verwendung eines Common-Bereichs.

Zu I: In diesen Bereich werden die vom Anwender über den Befehl OPT gewünschten Options (STR, GDI, RS2...) oder die durch den CON-Befehl definierten Prädispositionen als Moduls geladen. Beim Neuladen des Betriebssystems oder bei einem Programmwechsel bleibt diese Zone stets erhalten. Sie kann erst durch Eingabe von OPT und CON wieder freigegeben werden.

Zu II: In dieser Speicherzone werden für die in einem BASIC-Programm als Common definierten Variablen Speicherplätze reserviert. Bei einem Programmwechsel wird dieser Bereich, im Gegensatz zu Teil III, IV und V, nicht gelöscht, es sei denn, das Folgeprogramm enthält keinen Common-Bereich.

Zu III: Teil III beinhaltet den Programmcode, der von der Diskette eingelesen und in eine ausführbare Form gebracht wird.

Zu IV: In diesem Bereich befinden sich die von der Preexecution ermittelten Speicherplätze für die Buffer von Datenfiles (gemäß FILES-Anweisung), die In-/Output-Buffer für die Peripherie und die Speicherplätze für die im Programm definierten Variablen (z.B. gemäß DIM- und DCL-Anweisung).

Zu V: In dieser, STACK genannten Zone, die keine feste Länge besitzt, werden während des Programmablaufes Speicherplätze für die Dauer der jeweiligen Bearbeitung benötigt. So zum Beispiel für Rücksprungadressen aus Subroutinen, Berechnung bestimmter Stringfunktionen, den Plot-Buffer bei eingesetzter Option PLO und andere Operationen. Die obige Skizze ist abstrakt, da der Stack nicht grundsätzlich die Differenz zwischen dem Gesamtanwenderspeicher und dem durch die Teilbereiche I bis IV besetzten Speicher belegt. Der Stack ist vielmehr als "Keller-Speicher" zu betrachten, der während der Programmausführung zu verschiedenen Zeiten unterschiedlich groß ist.

Reicht der Speicherplatz für die in obiger Skizze dargestellten Bereiche nicht aus, kommt es zu folgenden Fehlermeldungen:

ERROR 65 Stack-Overflow (tritt während des Programmablaufes auf).

ERROR 117 Programm-Code-Overflow (erscheint schon beim Laden des Programmes; OLD, RUN).

ERROR 181 Overflow des Buffer- und Datenbereiches.

8.2.2 Common-Bereich

Der Common-Bereich ist die einzige der in 8.2.1 dargestellten Speicherzonen, die vom Betriebssystem nicht automatisch verwaltet werden kann, sondern direkt vom Anwender zu definieren und in der Struktur zu organisieren ist.

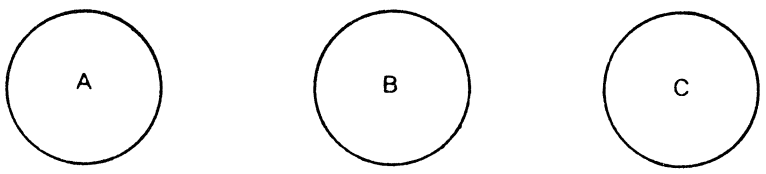
Daher soll hier kurz auf die Verwendung des Common-Bereiches eingegangen werden. Formelle Informationen sind der BASIC-Anweisung COMMON in Kapitel 9.1 zu entnehmen.

Der Sinn des Common-Bereiches liegt im Datenaustausch zwischen Programmen, ohne daß die Daten in einem externen Datenfile zwischengespeichert werden müssen. Dazu wird im Arbeitsspeicher ein "geschützter Bereich" reserviert, dessen Inhalt bei der Beendigung eines Programmablaufes oder bei Programmstarts erhalten bleibt.

Bei der Definition des Common-Bereiches über die BASIC-Anweisung sind die gewünschten Variablen zu spezifizieren. Diese müssen in verschiedenen Programmen bezüglich Typ und Deklaration übereinstimmen, was durch die Anweisungen DCL und DIM erreicht wird. Der für die angeführten Variablen benötigte Speicherplatz ist zu berechnen und mittels eines speziellen Parameters beim Abspeichern des Programmes anzugeben.

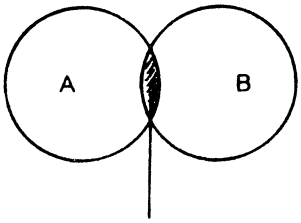
Anhand eines Beispieles soll die Festlegung des Common-Bereiches für die drei sich folgenden Programme PA, PB und PC erläutert werden:

Betrachtet man die Datenbereiche als Mengen, so ergibt sich folgende Darstellung:

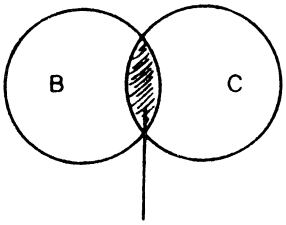


Datenbereich des Programmes PA | Datenbereich des Programmes PB | Datenbereich des Programmes PC

Die gemeinsamen Daten von PA und PB bilden den Durchschnitt der Mengen A und B, jene von PB und PC den Durchschnitt der Mengen B und C:



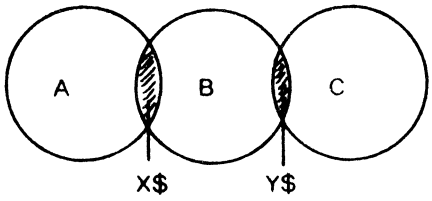
Durchschnitt 1



Durchschnitt 2

Es bestehen nun zwei Möglichkeiten:

- Durchschnitt 1 und Durchschnitt 2 haben keine gemeinsamen Daten:



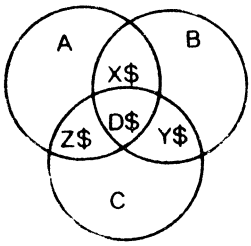
  Die COMMON-Zeilen lauten wie folgt:

```
Programm PA                    COMMON X$[,Y$]
Programm PB                    COMMON X$,Y$
Programm PC                    COMMON X$,Y$
```

  Da Y$ in PA und PB nicht gemeinsam enthalten ist, kann es in der COMMON-Zeile des Programmes PA weggelassen werden (eckige Klammer).

  Obwohl im Programm PC die Variable X$ nicht benötigt wird, muß sie als Platzhalter angegeben werden.

- Durchschnitt 1 und Durchschnitt 2 haben einen Durchschnitt:



  Die COMMON-Zeilen ergeben sich wie folgt:

```
Programm PA                    COMMON D$,X$,Y$,Z$
Programm PB                    COMMON D$,X$,Y$[,Z$]
Programm PC                    COMMON D$,X$,Y$,Z$
```

  Für das Programm PA ist Y$, für PC ist X$ Platzhalter. Im Programm PB ist die Variable Z$ nicht notwendig, jedoch wird durch ein Weglassen kein Speicherplatz gespart, da die reservierte Anzahl Bytes für den Common-Bereich diese Variable einschließen muß, um vom Programm PC verarbeitet werden zu können.

Allgemein läßt sich folgende Regel formulieren:

Ist für drei Folgeprogramme die COMMON-Anweisung und der Common-Bereich anzugeben bzw. zu bestimmen, so gilt:

- Als Variablenliste ist die Vereinigung der Durchschnitte der Mengen A, B, C und der dafür benötigte Speicherplatz zu ermitteln.
- Beim ersten und letzten Programm können die letzten Variablen weggelassen werden, wenn sie von diesem nicht benötig werden. Für dieses Programmm reduziert sich dann auch die Anzahl zu reservierender Bytes.
- Beim mittleren Programm können die letzten Variablen auch weggelassen werden, wenn sie nicht benötigt werden. Trotzdem ist der volle Speicherplatz zu reservieren, wenn diese Variablen vom dritten Programm noch benötigt werden.

Am einfachsten ist es, pauschal die Vereinigung der Durchschnitte von A, B und C zu verwenden.

8.3 Ablauforganisation eines Programmes

Ein Programm durchläuft normalerweise alle Zeilen von oben nach unten, von der kleinsten Zeilennummer bis zur größten. Dabei werden nicht ausführbare Anweisungen nicht berücksichtigt. Die Verarbeitung in aufsteigender Reihenfolge der Zeilennummern wird selbstverständlich durch Sprunganweisungen (wie GOTO, GOSUB), Schleifen oder Funktionsaufrufe unterbrochen.

Ein manueller Eingriff in ein Programm ist nur über die Anweisung INPUT oder RKB möglich, wobei je nach Eingabe in bestimmte Programmroutinen gesprungen werden kann.

Mit Hilfe der Anweisungen RECEIVE...AND GO und INTERRUPT ENABLE lassen sich u.a. manuelle Eingriffe von aussen programmieren, die zum einen eine überlappende Arbeitsweise erlauben und zum anderen einen Eingriff zu jedem beliebigen Zeitpunkt gewährleisten (wobei die sich jeweils in Bearbeitung befindliche BASIC-Anweisung zuerst beendet wird).

Eine weitere Besonderheit ist das Arbeiten im Modus "free running" über die V24- oder Current Loop-Schnittstelle. Dabei ist es zum Beispiel möglich, für jeden V24-Kanal einen Buffer - den sogenannten Leitungsbuffer - zu definieren, der Daten von der Peripherie empfangen kann, während das Programm andere Routinen verarbeitet. Die Daten gehen somit nicht verloren. Von bis zu vier Geräten können gleichzeitig Daten empfangen werden (Näheres dazu siehe "Handbuch für Peripherie und Datenübertragung").

Da für alle externen und einige integrierte Peripherien die Programmierung überlappender Abläufe möglich ist, können im System mehrere parallele Abläufe programmiert werden. Mehrere Organisationsformen stehen dafür zur Verfügung. Zur Veranschaulichung sind die beiden folgenden Beispiele angeführt:

1. Überlappende Arbeitsweise über den Tastaturkanal

a) normale Arbeitsweise

```
100 DISP "Eing.Anschrift";
110 RKB A$

150 \
160  | Plausibilitäts-
 .   | kontrolle und
 .   | Speichern auf
300 / Diskette

320 GOTO 100
```

b) überlappende Arbeitsweise

```
 90 A$=Anfangswert
100 DISP "Eing.Anschrift"
110 RECEIVE #32,A$ AND GO
120 IF A$=Anf.wert THEN

150 \
160  | Plausibilitäts-
 .   | kontrolle und
 .   | Speichern auf
300 / Diskette

310 WAIT # 32
320 GOTO 100
```

Im Fall a) wartet das System in Zeile 110, bis alle Daten eingegeben wurden und führt erst dann die Zeilen 150 bis 300 aus.

Im Fall b) arbeitet das Programm während der Eingabe weiter, das heißt, simultan zur Dateneingabe für die Anschrift I-te werden die Daten der Anschrift I-1 (vorherige Anschrift) kontrolliert und geschrieben.

2. Programmierung paralleler Abläufe

Die Abläufe werden in eine Folge von Unterprogrammen aufgeteilt, deren Verarbeitung von Steuervariablen überwacht wird (z.B. mit ON...GOSUB). Liegt eine Eingabe für einen Ablauf nicht vor, so wird die entsprechende Steuervariable nicht verändert, das Programm führt aber die Unterprogramme der nicht wartenden Abläufe aus. Verlangt eines der Unterprogramme eine Eingabe oder Daten von einer Peripherie, so sollte diese Eingabe am Ende des Unterprogrammes stehen. Dadurch bleibt bis zum vollständigen Eintreffen dieser Daten Zeit, die Unterprogramme der anderen Abläufe auszuführen.

Nach dem Ende eines ausgeführten Unterprogrammes wird jeweils mit TEST abgefragt, ob die Daten schon vorliegen. Ist dies der Fall, wird auch dieser Ablauf wieder in die wechselweise Verarbeitung der Unterprogramme einbezogen.

Bemerkungen:

- Solche Programmstrukturen können nur innerhalb eines Programmes gewählt werden. Bei einem Programmwechsel werden die Datenkanäle aufgelöst, wodurch Informationen verloren gehen können. Parallele Abläufe können nur dort programmiert werden, wo alle Abläufe aller Programmteile zur gleichen Zeit im Speicher sind.
- Durch die Steuerung der Datenkanäle geht CPU-Zeit verloren, die sich auf die Verarbeitungsgeschwindigkeit des Programmes auswirkt. Bei einer größeren Anzahl zu überwachender Kanäle (insbesondere V24) kann das zu einer spürbaren Verlangsamung der Programmabläufe führen.

3. Programmeingriff zu beliebigem Zeitpunkt mittels Interrupt

Eine große Anzahl Datensätze ist in Verarbeitung. Der Anwender möchte nun zu einem beliebigen Zeitpunkt wissen, welcher Datensatz gerade bearbeitet wird oder aber den Inhalt eines schon verarbeiteten Datensatzes ausgegeben haben.

```
FILE      INTER

0010 BUFFER #32,80
0020
0030


0100 RECEIVE #32,A$ AND GO
0110 FOR I=1 TO 1000 STEP 1
0120 INTERRUPT ENABLE (E,"A","0010000000000000",1)
0130
0140


0430 NEXT I
0440
0450


6000 DEF FNA(Y)
6010 WAIT #32
6030 ASSIGN A$,X;32
6040 IF X>I THEN 6070
6050 PRINT "Datensatz";X;"lautet";Z$(X)
6060 GOTO 6100
6070 PRINT "Es wurden erst";I;"Datensaetze bearbeitet"
6080 GOTO 6100
6090 PRINT "Es wird gerade Datensatz";I;"bearbeitet"
6100 LET FN*=0
6110 RECEIVE #32,A$ AND GO
6120 FNEND
6130 END

END OF LISTING
```

Folgende Eingriffsmöglichkeiten bestehen:

- Wird zum beliebigen Zeitpunkt die Taste EOL gedrückt, so wird vom Programm die Funktion FNA aufgerufen und über die Zeile 6080 ausgegeben, daß im Moment der Datensatz I bearbeitet wird.
- Wird die Nummer eines Datensatzes eingegeben, so gibt es zwei Möglichkeiten:
  . entweder ist die eingegebene Nummer größer als die Nummer des gerade bearbeiteten Satzes: dann wird der Text der Zeile 6060 ausgegeben;
  . oder die eingegebene Nummer ist kleiner oder gleich der Nummer des gerade bearbeiteten Satzes: dann wird der Text in Zeile 6040 ausgegeben. Z$(X) stellt den Inhalt des Datensatzes X dar.

Man beachte, daß der Programmablauf durch die obige Programmierung nicht abgebrochen wird. Vielmehr wird durch den Interrupt über die Tastatur eine vom Anwender definierte Routine (mehrzeilige Funktion) ausgeführt und dann der normale Programmablauf sofort wieder fortgesetzt (Näheres dazu siehe Kapitel 9.1 und 9.3.4).

8.4 Verbindung mit Assembler (P 6066)

8.4.1 Assembler-Sprache

Zur besseren Ausnutzung des Systems können Assemblerprogramme des Computer-Systems Olivetti P 6066 unter dem Betriebssystem MDOS ausgeführt werden. Eine Erstellung von Assembler-Programmen ist nur über das System P 6066 möglich. Es handelt sich dabei um ein Subset des IBM 360-Assemblers, doch stehen die Gleitkommabefehle nicht zur Verfügung. Der Zweck des Assemblereinsatzes liegt vor allem darin, die Verarbeitungsgeschwindigkeit bestimmter Operationen, wie Plausibilitätskontrollen, Sortierroutinen, Code-Umwandlungen, In-/Output- und Read-/Write-Operationen, zu erhöhen. So wird zum Beispiel die Suche nach der 1000sten Komponente eines Stringvektors durch den Assembler auf einen Zehntel der Zeit verkürzt (Näheres zur Programmierung in Assembler siehe Programmierhandbuch "Assembler-Sprache").

8.4.2 Verarbeitung durch ein BASIC-Programm

Mit Hilfe der BASIC-Anweisung CALL kann eine Assembler-Routine von einem BASIC-Programm aufgerufen und verarbeitet werden. Anschließend wird das Programm in der der Anweisung CALL folgenden Zeile fortgesetzt.

Die Assembler-Routine muß für die Verarbeitung durch das BASIC-Programm als Objektfile vorliegen (Typ 0 im CATALOG-Befehl).

In einem BASIC-Programm können mehrere CALL-Anweisungen enthalten sein. Die gleiche Assembler-Routine kann mehrmals aufgerufen werden. Ist das der Fall, so bleibt diese Routine so lange im Arbeitsspeicher, bis entweder eine andere Assembler-Routine aufgerufen wird oder das System einen internen Modul laden muß.

Assembler-Routinen können sowohl vom Disketten- als auch vom Platten-Betriebssystem aufgerufen werden.

In der Anweisung CALL können numerische und Stringvariable, aber auch Felder, als Parameter definiert werden. Die Assembler-Routine kann dann auf diese zugreifen (Näheres dazu ist dem Handbuch "Assembler-Betriebssystem" bzw. der BASIC-Anweisung CALL in Kapitel 9 zu entnehmen).

8.5 Standardformat

Bei Anweisungen des Typs

- BUILD
- DISP
- PRINT
- MAT PRINT

werden die Ergebnisse von numerischen oder alphanumerischen Ausdrücken im Standardformat dargestellt bzw. übergeben. Des weiteren erfolgt beim Rechnen im Debugging-Mode und bei Stellung ST des Dezimalstellenrades beim Rechnen im Calculator-Mode die Ausgabe der Werte im Standardformat.

Die Darstellung von Zahlen und Strings erfolgt in all diesen Fällen im Standardformat. Bei den Anweisungen DISP und PRINT ergeben sich durch die Stellenbeschränkung und die Verwendung verschiedener Trennzeichen zusätzliche Möglichkeiten, die getrennt behandelt werden.

8.5.1 Zahlendarstellung

Darstellung ganzer Zahlen

Ganze Zahlen werden linksbündig entsprechend ihrer Stellenzahl ausgegeben. Das erste Zeichen ist bei positiven Zahlen das Leerzeichen, bei negativen Zahlen das Vorzeichen "-". Am Ende der Zahl steht immer ein Leerzeichen. Ist die Zahl intern in doppelter Genauigkeit dargestellt, so erfolgt die Ausgabe im Standardformat mit maximal 8 Ziffern. Ist sie in einfacher Genauigkeit dargestellt, so erfolgt die Ausgabe im Standardformat mit maximal 6 Ziffern. Ganze Zahlen, die in doppelter Genauigkeit mehr als 8 Stellen, in einfacher Genauigkeit mehr als 6 Stellen haben, werden im Gleitkommaformat dargestellt. Dabei erfolgt bei Zahlen in doppelter Genauigkeit eine Rundung auf 8 Stellen. Bei Zahlen in einfacher Genauigkeit erfolgt keine Rundung, da auch intern nicht mehr als 6 signifikante Stellen vorhanden sind.

Dezimalzahlen im Festkommaformat

- Darstellung von Dezimalzahlen in doppelter Genauigkeit:

  Die Darstellung von Dezimalzahlen erfolgt mit maximal 8 Ziffern. Zusätzlich werden eine Stelle vor der Zahl für das Vorzeichen ("-" bei negativen, Leerzeichen bei positiven Zahlen), eine Stelle für den Dezimalpunkt und eine Leerstelle am Ende der Zahl benötigt.

  Führende Nullen werden unterdrückt. Nullen am Ende der Zahl werden im Dezimalteil nur dann dargestellt, wenn sie das Ergebnis einer Rundung sind.

  Die Anzahl Nachkommastellen in der Darstellung richtet sich nach der Anzahl Vorkommastellen, wobei die Summe aus Vor- und Nachkommastellen maximal 8 beträgt. Ist die Summe aus signifikanten Vor- und Nachkommastellen kleiner als 8, so erfolgt die Darstellung mit entsprechend weniger Stellen. Hat die Zahl 8 Vorkommastellen, so wird zwar der Dezimalpunkt, jedoch keine Nachkommastelle ausgegeben. Hat die darzustellende Zahl einen Absolutbetrag, der kleiner ist als 1, so ist die erste Stelle nach dem Vorzeichen der Dezimalpunkt.

- Darstellung von Dezimalzahlen in einfacher Genauigkeit:

  Für die Darstellung von Dezimalzahlen in einfacher Genauigkeit gelten sinngemäß die gleichen Regeln wie für Zahlen in doppelter Genauigkeit. Die maximale Anzahl Ziffern ist jedoch auf 6 begrenzt. Die letzten Stellen werden nicht gerundet, da auch in der internen Darstellung nicht mehr als 6 signifikante Ziffern vorhanden sind.

- Zahlendarstellung im Gleitkommaformat:

  Zahlen haben in Gleitkommadarstellung folgendes Format:

  - Vorzeichen Mantisse ("-" oder Leerstelle)
  - Mantisse
    . 1 Vorkommastelle (ungleich Null)
    . Dezimalpunkt
    . 7 Nachkommastellen (bei doppelt genauer Darstellung wird die letzte Ziffer gegebenenfalls gerundet, bei einfach genauer Darstellung sind nur 5 Stellen signifikant).
  - E (Kennzeichen für den Exponenten zur Basis 10)
  - Vorzeichen des Exponenten ("+" oder "-")
  - Exponent (2 Stellen), führende Nullen werden ausgedruckt
  - Leerzeichen nach der Zahl

  Insgesamt erfordert die Gleitkommadarstellung also 15 Stellen. Der Übergang zur Gleitkommadarstellung erfolgt dann, wenn mehr signifikante Stellen vorhanden sind, als in der Festkommadarstellung ausgegeben werden können.

Beispiele zur Zahlendarstellung

GLEITKOMMADARSTELLUNG	DOPPELT GENAU	EINFACH GENAU
1.000000000000E+00	1	1
1.000000000000E+01	10	10
1.000000000000E+02	100	100
1.000000000000E+03	1000	1000
1.000000000000E+04	10000	10000
1.000000000000E+05	100000	100000
1.000000000000E+06	1000000	1.0000000E+06
1.000000000000E+10	1.0000000E+10	1.0000000E+10
1.000000000000E-01	.1	.1
1.000000000000E-02	.01	.01
1.000000000000E-03	.001	.001
1.000000000000E-04	.0001	.0001
1.000000000000E-07	.0000001	.0000001
1.000000000000E-11	1.0000000E-11	1.0000000E-11
1.100000000000E+00	1.1	1.1
1.123450000000E+00	1.12345	1.12345
1.123456700000E+00	1.1234567	1.12345
1.123456780000E+00	1.1234568	1.12345
-1.000000000000E+00	-1	-1
-1.234567800000E+00	-1.2345678	-1.23456
-1.234567890000E+02	-123.45679	-123.456
-1.000000000000E-01	-.1	-.1
-9.999990000000E+00	-9.99999	-9.99999
-9.999999900000E+07	-99999999	-9.9999900E+07
-9.999999910000E+07	-99999999.	-9.9999900E+07

8.5.2 Darstellung von Strings

Strings werden zeichenweise linksbündig dargestellt. Die Anzahl der angegebenen Zeichen im Beispiel entspricht der aktuellen Länge des Strings.

Beispiel:

```
FILE


0010 DCL 80A$
0020 DISP "STRING";
0030 RKB A$
0040 PRINT A$
0050 GOTO 20
0060 END

END OF LISTING



Strings werden mit ihrer aktuellen Laenge gedruckt.
@⊣÷↑←↓≡↖
Dieser String wird mit '*' bis zur maximalen Laenge aufgefuellt:***************
```

8.5.3 Stellenkontrolle bei den Anweisungen DISP und PRINT

Für das Display und den Drucker steht je ein Buffer mit 80 Zeichen zur Verfügung, in dem die Ausgabezeile aufbereitet wird. Für jeden Buffer existiert ein Pointer, der die Stelle des Buffers anzeigt, an der mit der Darstellung des nächsten Ausgabeelementes begonnen werden kann.

Die Stellung des Pointers ist abhängig von den bisher dargestellten Ausgabeelementen, doch kann seine Stellung durch die Wahl der Trennzeichen "," oder ";" oder durch Verwendung der Funktion TAB verändert werden.

Ist die vollständige Darstellung im Buffer nicht möglich, so wird der bisherige Inhalt des Buffers ausgegeben, sein Inhalt gelöscht und mit der weiteren Darstellung der Ausgabeelemente beim ersten Zeichen im Buffer wieder begonnen. Die Darstellung von Strings mit einer Länge von mehr als 80 Zeichen in einer Zeile ist nicht möglich.

Das Trennzeichen Komma ","

Enthält die Liste der Ausgabeelemente das Trennzeichen Komma, so wird der Buffer in fünf Zonen zu je 16 Zeichen unterteilt. Diese Zonen beginnen bei den Positionen 1, 17, 33, 49 und 65. Ist das dem zuletzt dargestellten Ausgabeelement folgende Trennzeichen das Komma, so wird der Wert des Pointers auf das nächsthöhere ganzzahlige Vielfache von 16+1 erhöht. Ist dieser Wert größer oder gleich 80, wird der Inhalt des Buffers ausgegeben und erneut mit der Aufbereitung des Bufferinhaltes ab der ersten Stelle begonnen.

Trennzeichen Strichpunkt ";"

Das Trennzeichen ";" bewirkt keine Änderung der Stellung des Pointers. Durch ";" getrennte Ausgabeelemente werden also unmittelbar aneinander anschließend ausgegeben. Der Inhalt des Buffers wird ausgegeben, sobald durch ein weiteres Ausgabeelement 80 Zeichen erreicht oder überschritten werden. Dieses Ausgabeelement kommt dann wieder als erstes Element in den Buffer.

Funktion TAB (num.Ausdruck)

Die Funktion TAB erlaubt es, eine beliebige Position im Buffer direkt anzulaufen. Ist diese Position bereits mit Daten belegt, so wird der Inhalt des Buffers ausgegeben und der Pointer auf die Stelle im Buffer gesetzt, die durch die Funktion TAB bestimmt ist.

Ist der Wert n, der dem gerundeten Ergebnis des arithmetischen Ausdruckes entspricht, kleiner als 1, so erfolgt eine Fehlermeldung. Ist der Wert n >80, so wird, falls es sich um die Anweisung PRINT handelt, n=n-INT((n-1)/80)*80 und eine Zeilenschaltung ausgeführt. Um durch das Trennzeichen keine weitere Tabulation zu bewirken, muß nach Aufruf der Funktion TAB das Trennzeichen Strichpunkt ";" gesetzt werden.

In einer PRINT- oder DISP-Anweisung sind auch Kombinationen der Trennzeichen und der Funktion TAB möglich.

Trennzeichen am Ende der Anweisung DISP oder PRINT

Wird hinter das letzte Element einer Ausgabeliste ein Trennzeichen (";" oder ",") gesetzt, so wird der Inhalt des Buffers nur dann ausgegeben, wenn 80 Zeichen erreicht sind. Ausgabeelemente nachfolgender PRINT- bzw. DISP-Anweisungen werden an den bestehenden Inhalt des Buffers angefügt. Dadurch ist es möglich, die Elemente mehrerer PRINT- oder DISP-Anweisungen in einer Zeile auszugeben.

Fehlt am Ende der Ausgabeliste einer PRINT- bzw. DISP-Anweisung das Trennzeichen, so wird der Inhalt des Buffers ausgegeben und der Pointer an dessen erste Stelle gesetzt. Folgt einer durch ein Trennzeichen abgeschlossenen PRINT- bzw. DISP-Anweisung eine PRINT- oder DISP-Anweisung ohne Ausgabeelemente, so wird der bestehende Inhalt des Buffers ausgegeben und der Pointer auf dessen erste Stelle zurückgesetzt.

Hat der Druckbuffer keinen Inhalt, bewirkt eine PRINT-Anweisung ohne Ausgabeelement eine Leerzeile. Ist der Display-Buffer leer, so wird die Anzeige im Display durch eine DISP-Anweisung ohne Ausgabeelement gelöscht.

8.5.4 Besonderheiten der Anweisung DISP

In der Display-Zeile des Bildschirms (Zeile 0) können bis zu 80 Zeichen angezeigt werden.

Zur Ausgabe von Texten auf dem Bildschirm in definierten Zeilen und Spalten existiert ein spezielles Format der Display-Anweisung, die als erstes Zeichen das ISO-Zeichen ESCAPE (ESC, ISO-Code 27) überträgt (nähere Einzelheiten siehe Abschnitt 9.3.3).

Beispiele:

```
FILE


0010 REM ** WIRKUNG DER TRENNZEICHEN ',' UND ';' **
0020 REM
0030 DCL 20B$
0040 PRINT "TRENNZEICHEN ',' → STANDARD-TABULLATION:"
0050 PRINT
0060 READ A,A$,B,C,B$
0070 PRINT A,A$,B,C,B$
0080 DATA 13.4567,String 1,1626.345,10101,**********
0090 END

END OF LISTING



RUN
TRENNZEICHEN ',' → STANDARD-TABULLATION:

 13.4567        String 1          1626.345        10101
**********
```

```
FILE


0010 REM ** WIRKUNG DER TRENNZEICHEN ',' UND ';' **
0020 REM
0030 DCL 20B$
0040 PRINT "TRENNZEICHEN ';' → KEINE TABULLATION:"
0050 PRINT
0060 READ A,A$,B,C,B$
0070 PRINT A;A$;B;C;B$
0080 DATA 13.4567,String 1,1626.345,10101,**********
0090 END

END OF LISTING



RUN
TRENNZEICHEN ';' → KEINE TABULLATION:

 13.4567 String 1 1626.345  10101 **********
```

```
FILE


0010 REM ** WIRKUNG DER FUNKTION 'TAB' **
0020 REM
0030 DCL 20B$
0040 PRINT "TABULLATION MITTELS 'TAB':"
0050 PRINT
0060 READ A,A$,B,C,B$
0070 PRINT A;TAB(15);A$;TAB(40);B;TAB(30);C;TAB(50);B$
0080 DATA 13.4567,String 1,1626.345,10101,**********
0090 END

END OF LISTING



RUN
TABULLATION MITTELS 'TAB':

 13.4567       String 1                  1626.345
                                 10101               **********
```

```
FILE


0010 REM ** BEISPIEL 'PRINT' **
0020 REM
0030 REM  ** AUFGESCHOBENER AUSDRUCK MITTELS ',' ODER ';' **
0040 REM
0050 TRACE ON
0060 REM
0070 FOR I=1 TO 7 STEP 1
0080 PRINT RND,
0090 NEXT I
0100 PRINT
0110 PRINT
0120 FOR I=1 TO 7 STEP 1
0130 PRINT RND*100;
0140 NEXT I
0150 END

END OF LISTING



RUN
#70
#80
#90
#80
#90
#80
#90
#80
#90
#80
 .63829321        .41839390        .97356004        .88649157        .21507853
#90
#80
#90
#80
#90
#100
 4.7279296E-02    .98707879
#110

#120
#130
#140
#130
#140
#130
#140
#130
#140
#130
#140
#130
#140
#150
 95.457924   55.713896   81.827105   4.7693357   68.675523   17.611750   3.5848356
```

9. BASIC-ANWEISUNGEN

9.1 Alphabetisch geordnete Beschreibung der Befehle

	9.1
APPEND:	9.1
ASSIGN	9.3
BASSIGN	9.5
BBUILD	9.7
BEEP	9.9
BPAD	9.11
BUILD	9.13
BUILD USING	9.15
CALL	9.17
CHAIN	9.19
COMMON	9.21
CONVERT	9.25
DATA	9.27
DCL	9.29
DEF	9.31
DEF/FNEND	9.35
DELAY	9.39
DEPAD	9.41
DIM	9.43
DISP	9.45
DISP USING	9.51
END	9.55
ERASE	9.57
FILES	9.59
FILE:	9.61
FKEY	9.63
FNEND	9.65
FOR	9.67
GOSUB	9.71
GOTO	9.73
IF...THEN	9.75
IMAGE statement	9.77
INPUT	9.83
INTERRUPT ENABLE	9.87
LET	9.95
NEXT	9.97
ON...GOSUB	9.99
ON...GOTO	9.101
PAD	9.103
PRINT	9.105
PRINT USING	9.109
RANDOMIZE	9.113
READ	9.115
READ:	9.117
REM	9.119
RESTORE	9.121
RESTORE:	9.123
RETURN	9.125

	REVERSE	9.127
	RKB	9.129
	SCRATCH:	9.131
	SETW:	9.133
	STOP	9.135
	TRACE OFF	9.137
	TRACE ON	9.139
	WHERE:	9.141
	WRITE:	9.145
9.2	Standardfunktionen	9.147
9.2.1	Trigonometrische Funktionen	9.148
9.2.2	Mathematische Standardfunktionen	9.139
9.2.3	Numerische Funktionen ohne Argument	9.151
	PI	9.151
	RND	9.151
	DET	9.153
9.2.4	Spezielle numerische Funktionen	9.155
	IOC	9.157
	LEN	9.159
	SCN	9.161
	TAB	9.163
9.2.5	Alphanumerische Funktionen	9.165
	CHR$	9.167
	EXT$	9.169
	BLN$	9.171
	REP$	9.173
9.3	Anweisungen für Matrizenoperationen	9.175
	MAT...=	9.177
	MAT..+	9.179
	MAT...Skalar*	9.181
	MAT...*	9.183
	MAT...CON	9.185
	MAT...IDN	9.187
	MAT INPUT	9.189
	MAT...INV	9.191
	MAT PRINT	9.193
	MAT PRINT USING	9.195
	MAT READ	9.197
	MAT READ:	9.199
	MAT...TRN	9.201
	MAT WRITE:	9.203
	MAT...ZER	9.205
9.4	I/O-Kanal TASTATUR	9.207
9.4.1	BASIC-Anweisungen für den Kanal TASTATUR	9.207
9.4.2	Bemerkungen	9.212

9.4.3	BASIC-Anweisungen im Detail	9.213
	BUFFER	9.215
	CMD	9.217
	RECEIVE	9.219
	SEND	9.221
	TEST	9.223
	WAIT	9.225
9.4.4	Beispiele	9.227
9.5	Graphisches Arbeiten durch OPTION 'Plot' und 'Graphik-Display' (OPT PLO oder OPT GDI)	9.229
9.5.1	Grundlagen des Plottens	9.229
9.5.2	Plotten über den Thermodrucker PR 2400 mit OPT PLO (ohne Darstellung auf Bildschirm)	9.233
9.5.2.1	Darstellung von Punkten, Linien und Zeichen	9.233
9.5.2.2	Ausführung von Plotprogrammen	9.233
9.5.3	Graphische Darstellung am Bildschirm mit OPT GDI	9.234
9.5.4	Plotten mit einem externen Plotter	9.236
9.5.5	BASIC-Anweisungen der Module OPTION GDI/PLO	9.239
	CPLOT	9.241
	CSIZE	9.243
	CTAB	9.245
	DISP	9.247
	DOT	9.249
	DRAW	9.251
	ERASE	9.253
	EXTERNAL PLOTTER	9.255
	FRAME	9.257
	IDOT	9.259
	INIMAGE	9.261
	IPLOT	9.265
	LDIMAGE	9.267
	MOVE	9.269
	OFFSET	9.271
	PLOT	9.273
	POINTER	9.275
	REVERSE	9.277
	SCALE	9.279
	STIMAGE	9.281
	XAXIS	9.283
	YAXIS	9.285
9.5.6	Systembefehle für den Bildschirm	9.287
	LDIMAGE	9.289
	SITMAGE	9.291
9.5.7	Beispiele	9.293

APPEND:

ANWEISUNG: APPEND:

FUNKTION: Anfügen von Daten an ein sequentielles File.

FORMAT: APPEND: filedesignator

"file-
designator" numerischer Ausdruck.

WIRKUNG: In dem durch "filedesignator" angegebenen, externen File wird der Pointer auf das erste freie Element des Files gesetzt. Bei einer nachfolgenden WRITE:-Anweisung mit dem gleichen Filedesignator werden die Daten an die bereits bestehenden angefügt. Ergibt die Berechnung des Ausdruckes für den Filedesignator keinen ganzzahligen Wert, wird dieser gerundet.

BEMERKUNG: Der Wert des Filedesignators muß größer sein als Null und kleiner oder gleich der Anzahl Filenamen in der Anweisung FILES.

BEISPIEL:

```
FILE     APPEND


0010 FILES SDAT
0020 REM DIESER TEIL BESCHREIBT DAS FILE VON BEGINN AN
0030 SCRATCH :1
0040 FOR I=1 TO 10 STEP 1
0050 WRITE :1,I
0060 NEXT I
0070 REM ES WERDEN DI ERSTEN 5 DATEN GELESEN
0080 RESTORE :1
0090 FOR I=1 TO 5 STEP 1
0100 READ :1,A
0110 PRINT A;
0120 NEXT I
0130 PRINT
0140 REM NEUE DATEN WERDEN HINZUGEFUEGT
0150 REM
0160 REM
0170 APPEND :1
0180 FOR I=11 TO 15 STEP 1
0190 WRITE :1,I
0200 NEXT I
0210 REM LESEN DES GESAMTEN FILES
0220 RESTORE :1
0230 READ :1,A EOF 260
0240 PRINT A;
0250 GOTO 230
0260 PRINT
0270 PRINT "FILENDE"
0280 END

END OF LISTING



 1  2  3  4  5
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15
FILENDE
```

ASSIGN

ANWEISUNG: ASSIGN

FUNKTION: Zuweisung von Werten aus einem Stringausdruck an eine oder mehrere numerische oder alphanumerische Variable. Als Datentrennzeichen dient ein fixer Delimiter.

FORMAT: ASSIGN Stringausdr., {num.Var. / Stringvar.} [, {num.Var. / Stringvar.} ...] ;delimiter d

"delimiter d" numerischer Wert eines beliebigen ISO-Zeichens, 0 < d < 255.

WIRKUNG: Das System berechnet den Stringausdruck. Entsprechend der Zahl d wird die entstandene Zeichenkette in verschiedene Strings aufgeteilt, die mit ihrem numerischen oder alphanumerischen Wert den Variablen der Variablenliste zugewiesen werden.

BEMERKUNG: Einer numerischen Variablen in der Variablenliste muß ein numerischer Wert als Ergebnis zugewiesen werden. Die Anzahl der Strings, in die die Zeichenkette aufgeteilt wird und die durch den Delimiter d getrennt sind, muß größer oder gleich der Anzahl der Variablen in der Variablenliste sein.

BEISPIEL:

```
FILE     ASSIGN

0010 DCL 30(A$,V$,Z$)
0020 DISP "BITTE VOR UND ZUNAME EINGEBEN";
0030 RKB A$
0040 REM TRENNUNG DES NAMENS IN VOR UND ZUNAME
0050 REM
0060 ASSIGN A$,V$,Z$;32
0070 PRINT "VORNAME: ";V$;TAB(35);"  ZUNAME: ";Z$
0080 GOTO 20
0090 END

END OF LISTING


RUN
VORNAME: ANDRE                          ZUNAME: DEUTSCH
VORNAME: RODOLFO                        ZUNAME: SUTTER
VORNAME: SUSANNA                        ZUNAME: DRESSLER
VORNAME: FRANZISKA                      ZUNAME: BOECKLI
```

BASSIGN

ANWEISUNG: BASSIGN (binary assign)

FUNKTION: Zuweisung, in internem Format, des Inhaltes eines Stringausdruckes an eine oder mehrere Variable.

FORMAT: BASSIGN Stringausdr., {num. Var. / Stringvar.}[,{num. Var. / Stringvar.}]....

WIRKUNG: Der Stringausdruck wird berechnet und die entstandene Zeichenkette als eine Liste von Daten in internem Format aufgefasst. Die Daten werden den entsprechenden Variablen in der Variablenliste zugewiesen.

BEMERKUNG: Die Anzahl der Daten im Stringausdruck muss grösser oder gleich sein wie die Anzahl der Variablen in der Variablenliste. Einer numerischen Variablen muss ein numerischer Wert, einer Stringvariablen ein String zugewiesen werden.

BEISPIEL:

```
FILE     BASIGN

0010 REM BEISPIEL FUER BASSIGN/BBUILD
0020 REM
0030 DISP "EINGABE VON A UND B";
0040 INPUT A,B
0050 BBUILD A$,A,B
0060 PRINT "A=";A;"B=";B;TAB(40);"BBUILD: A$=";A$
0070 BASSIGN A$,X,Y
0080 PRINT "A$=";A$;TAB(40);"BASSIGN: X=";X;" Y=";Y
0090 PRINT
0100 GOTO 30
0110 END

END OF LISTING


RUN
A= 5 B= 5                                  BBUILD: A$=■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■                    BASSIGN: X= 5  Y= 5

A= 1.2345 B=-1.2345                        BBUILD: A$=■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■                    BASSIGN: X= 1.2345  Y=-1.2345

A= 100000 B= .00001                        BBUILD: A$=■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■                    BASSIGN: X= 100000  Y= .00001

A= 5.555 B= 6.666                          BBUILD: A$=■■■■■■■■■■■■■■■■■■■■
A$=■■■■■■■■■■■■■■■■■■■■                    BASSIGN: X= 5.555  Y= 6.666
```

ANWEISUNG: BBUILD (binary build)

FUNKTION: Übertragung, in internem Format, eines Ergebnisses einer Liste mit arithmetischen oder Stringausdrücken an eine Stringvariable.

FORMAT: BBUILD Stringvar., {num.Ausdruck / Stringausdr.} [, {num.Ausdruck / Stringausdr.}] .....

WIRKUNG: Die angegebenen Ausdrücke werden berechnet und ihre Werte der Stringvariablen in Binärdarstellung zugewiesen. Strings sind in internem Format dargestellt.

BEMERKUNG: Eine Zahl belegt in einfacher Genauigkeit 4 Bytes (1 Wort), in doppelter Genauigkeit 8 Bytes. Jeder String mit N Zeichen benötigt INT((N-1)/4+2)* 4 Bytes. BBUILD verwendet damit die gleiche Darstellungsart wie die Anweisung WRITE: für externe Datenfiles.

BEISPIEL:

```
FILE     BASIGN


0010 REM BEISPIEL FUER BASSIGN/BBUILD
0020 REM
0030 DISP "EINGABE VON A UND B";
0040 INPUT A,B
0050 BBUILD A$,A,B
0060 PRINT "A=";A;"B=";B;TAB(40);"BBUILD: A$=";A$
0070 BASSIGN A$,X,Y
0080 PRINT "A$=";A$;TAB(40);"BASSIGN: X=";X;" Y=";Y
0090 PRINT
0100 GOTO 30
0110 END

END OF LISTING



RUN
A=-1 B=-3.25                              BBUILD: A$=■■■■■■■$■■%■■■■■$
A$=■■■■■■■$■■%■■■■■$                      BASSIGN: X=-1  Y=-3.25

A=-.1 B= 10                               BBUILD: A$=■■■■■■■$■■■■■■■■■
A$=■■■■■■■$■■■■■■■■■                      BASSIGN: X=-.1  Y= 10

A= 1234568.0 B= 987654.23                 BBUILD: A$=■■■#E9■■■■■■■eB0■■■■
A$=■■■#E9■■■■■■■eB0■■■■                   BASSIGN: X= 1234568.0  Y= 987654.23
```

ANWEISUNG: BEEP

FUNKTION: Ausgabe eines akustischen Signals.

FORMAT: BEEP

WIRKUNG: Für ca. 0,2 Sekunden ertönt ein Signal.

BEISPIELE: 1.

```
0050 DISP "ZAHL ZWISCHEN 1 UND 10 EINGEBEN";
0060 INPUT I
0070 IF (I>=1) AND (I<=10) THEN 100
0080 BEEP
0090 GOTO 50
```

2. Erzeugung eines Dauersignals:

```
0110 FOR I=1 TO 20 STEP 1
0120 BEEP
0130 NEXT I
```

ANWEISUNG: BPAD (binary pad)

FUNKTION: Füllen einer Stringvariablen mit binären Füllzeichen bis zur deklarierten Länge.

FORMAT: BPAD Stringvariable

BEMERKUNGEN:
- Das binäre Füllzeichen entspricht der Binärstellung von 255 : 11111111.

BEISPIEL:

```
FILE     BPAD

0010 DCL 30A$
0020 DISP "BITTE STRING EINGEBEN";
0030 RKB A$
0040 PRINT "A$:";A$
0050 BPAD A$
0060 PRINT "A$:";A$
0070 GOTO 20
0080 END

END OF LISTING



RUN
BITTE STRING EINGEBEN?
BEISPIEL BPAD
A$:BEISPIEL BPAD
A$:BEISPIEL BPAD■■■■■■■■■■■■■■■■■■■■■■■■
BITTE STRING EINGEBEN?
```

BUILD

ANWEISUNG: BUILD

FUNKTION: Übertragung, im Standardformat, des Ergebnisses einer Liste mit arithmetischen und/oder Stringausdrücken an eine Stringvariable.

FORMAT: BUILD Stringvar.,{num.Ausdr.|Stringausd.}[,{num.Ausdr.|Stringausd.}.....][;delim.d]

WIRKUNG: Die Ausdrücke werden berechnet und die Ergebnisse im Zeichenformat an die Stringvariable übertragen. Die Elemente werden durch das dem angegebenen Delimiter in der ISO-Tablle entsprechende Zeichen getrennt. Fehlt die Delimiter-Angabe, weist die Stringvariable kein Trennzeichen zwischen zwei Strings aus, es verbleibt jedoch jeweils ein Leerzeichen (ISO-Code 32) vor und nach einem numerischen Wert.

BEMERKUNGEN:
- Der Wert eines numerischen Ausdruckes wird im Standardformat (siehe Abschnitt 8.4) an die Stringvariable übergeben.
- Ist das Resultat eines Ausdruckes eine Zeichenkette, werden die einzelnen Zeichen ohne Veränderung an die Stringvariable übertragen.

BEISPIEL:

```
FILE     BUILD

0010 REM BEISPIEL 'BUILD'
0020 REM
0030 DCL 30 A$
0040 DISP "A UND B EINGEBEN";
0050 INPUT A,B
0060 LET C=A↑B
0070 BUILD A$,A,"HOCH",B,"=",C;32
0080 PRINT A$
0090 GOTO 40
0100 END

END OF LISTING


 2  HOCH  3  =  8
 2.5  HOCH  4.36  =  54.327340
 10  HOCH  9  =  1.0000000E+09
```

BUILD USING

ANWEISUNG: BUILD USING

FUNKTION: Übertragung, in spezifiziertem Format, des Ergebnisses einer Liste mit arithmetischen und/oder Stringausdrücken an eine Stringvariable.

FORMAT:

BUILD USING {Zeilennr. / Stringvar.},Stringvar.,{num.Ausdr. / Stringausdr.}[,{num.Ausdr. / Stringausdr.}].....

Die Zeilennummer entspricht der Nummer derjenigen Programmzeile, die das Format der Ergebnis-Stringvariablen festlegt. Anstelle der Zeilennummer kann der Name einer das Format definierenden Stringvariablen stehen.

WIRKUNG: Die Ausdrücke werden berechnet und die Resultate an die Stringvariable übertragen. Das Format ist jenes, das in der Programmzeile mit der in der Anweisung angegebenen Nummer steht.

Steht in einer Anweisung anstelle der Zeilennummer der Name einer Stringvariablen, werden die Ergebnisse in dem darin angegebenen Format übertragen.

BEMERKUNG: Die für die Formatspezifikation von Zeichenstrings verwendbaren Zeichen sind in der Beschreibung "Formatspezifikation" aufgeführt.

BEISPIELE:

Beispiel 1:

Formatangabe für BUILD USING in einer Image-Zeile.

```
FILE     BUILD1


0010 REM BEISPIEL 'BUILD USING'
0020 REM
0030 DCL 40 A$
0040 DISP "A UND B EINGEBEN",
0050 INPUT A,B
0060 LET C=A↑B
0070 BUILD USING 100,A$,A,B,C
0080 PRINT A$
0090 GOTO 40
0100 :#####.### HOCH #####.### = #######.###
0110 END

END OF LISTING



    2.000 HOCH      3.000 =        8.000
    2.500 HOCH      4.360 =       54.327
   10.000 HOCH      9.000 = ***********
```

Beispiel 2:

Aufbereitung des Formates einer Stringvariablen.

```
FILE     BUILD2


0010 REM BEISPIEL 'BUILD USING'
0020 REM
0030 DCL 40(A$,B$)
0040 DISP "A UND B EINGEBEN";
0050 INPUT A,B
0060 LET C=A↑B
0070 LET B$="#####.### HOCH #####.### = #######.###"
0080 BUILD USING B$,A$,A,B,C
0090 PRINT A$
0100 GOTO 40
0120 END

END OF LISTING



    2.000 HOCH      3.000 =        8.000
    2.500 HOCH      4.360 =       54.327
   10.000 HOCH      9.000 = ***********
```

CALL

ANWEISUNG: CALL

FUNKTION: Laden einer Assembler-Routine (Objektfile) in den Arbeitsspeicher und Beginn der Verarbeitung.

FORMAT: CALL Stringausdr.1 [,{num.Ausdr. / Stringausdr} ...]

"Stringausdr.1"	Name eines Assembler-Objektfiles, das in einer (offenen) Bibliothek gespeichert ist.
"num.Ausdr."	numerischer Ausdruck, dessen Ergebnis von der Assembler-Routine übernommen wird.
"Stringausdr."	Stringausdruck, dessen Ergebnis von der Assembler-Routine übernommen wird.

WIRKUNG: Die durch "Stringausdr.1" definierte Assembler-Routine wird in den Arbeitsspeicher geladen. Dadurch übergibt das BASIC-Programm die Kontrolle an die Assembler-Routine, mit deren Ausführung sofort begonnen wird.

BEMERKUNGEN:

- Die Anzahl der Parameter innerhalb der CALL-Anweisung wird nur durch die Anzahl möglicher Zeichen in einer Zeile beschränkt.
- Nach Ausführung der Assembler-Routine wird das BASIC-Programm mit der der CALL-Anweisung folgenden Anweisung fortgesetzt.
- Während der Ausführung von Assembler-Routinen auftretende Fehler bewirken folgende Fehlermeldung:

  ABEND n IN FILE filename

  worin 0 < n < 14 und "filename" der Name des Objektfiles ist.
- Mit dem System können Assembler-Routinen nur abgearbeitet werden. Sie müssen an der P 6066 erstellt und abgearbeitet werden. Solche Assembler-Routinen sind in CATALOG dann als Objekt-Files(0) gekennzeichnet.

CHAIN

ANWEISUNG: CHAIN

FUNKTION: Unterbrechung der Verarbeitung eines Programmes und Ausführung eines anderen Programmes, ohne dass ein manueller Eingriff nötig ist.

FORMAT: CHAIN Stringausdr.

"Stringausdr." Das Ergebnis des Stringausdruckes muß der Name eines in einer offenen Bibliothek gespeicherten Programmes sein.

WIRKUNG: Die Verarbeitung des laufenden Programmes wird beendet und alle in diesem Programm verwendeten externen Files geschlossen. Das Programm mit dem unter "Stringausdr." angegebenen Namen wird aufgerufen und mit seiner Ausführung begonnen.

BEMERKUNG: Alle im durch CHAIN aufgerufenen Programm verwendeten externen Files müssen mit der Anweisung FILES geöffnet werden, auch wenn im aufrufenden Programm mit den gleichen Files gearbeitet wurde. Da über die Files mit dem Filedesignator 1, 2, 3 oder 4 Informationen im Hauptspeicher zurückbehalten werden, wird die Ausführung von CHAIN beschleunigt, wenn die von beiden Programmen verwendeten externen Files in der Anweisung FILES auf den Positionen 1-4 aufgeführt sind.

BEISPIEL:

```
FILE     HAUPT


0010 FILES SFILE
0020 PRINT "ANFANG HAUPTPROGRAMM 'HAUPT'"
0030 SCRATCH :1
0040 WRITE :1,"HAUPT"
0050 CHAIN "UP1"
0060 END

END OF LISTING



FILE     UP1


0010 FILES SFILE
0020 PRINT "ANFANG UNTERPROGRAMM 'UP1'"
0030 DISP "WELCHES PROGRAMM SOLL RECHNEN";
0040 RKB A$
0050 APPEND :1
0060 WRITE :1,"UP1"
0070 CHAIN A$
0080 END

END OF LISTING



FILE     UP2


0010 FILES SFILE
0020 PRINT "ANFANG UNTERPROGRAMM 'UP2'"
0030 PRINT "FOLGENDE PROGRAMME HABEN GERECHNET:"
0040 READ :1,A$ EOF 70
0050 PRINT A$
0060 GOTO 40
0070 PRINT "UP2"
0080 END

END OF LISTING



RUN HAUPT
**** FORMALLY CORRECT PROGRAM ****
ANFANG HAUPTPROGRAMM 'HAUPT'
**** FORMALLY CORRECT PROGRAM ****
ANFANG UNTERPROGRAMM 'UP1'
WELCHES PROGRAMM SOLL RECHNEN?
UP2
**** FORMALLY CORRECT PROGRAM ****
ANFANG UNTERPROGRAMM 'UP2'
FOLGENDE PROGRAMME HABEN GERECHNET:
HAUPT
UP1
UP2
```

COMMON

ANWEISUNG: COMMON

FUNKTION: Definition der Variablen in einem BASIC-Programm, deren Werte in verschiedenen Programmen Verwendung finden sollen.

FORMAT:

```
COMMON {num. Feld ()      }[  {num. Feld ()      }]
       {einf. Stringvar.  }[, {einf. Stringvar.  }]...
       {Stringfeld ()     }   {Stringfeld ()     }
```

"num. Feld"	Bezeichnung eines numerischen Feldes, dessen Speicherplatz im Common-Bereich reserviert werden soll.
"einf. Stringvar."	Name einer einfachen Stringvariablen, deren Speicherplatz im Common-Bereich reserviert werden soll.
"Stringfeld"	Bezeichnung eines Stringfeldes, dessen Speicherplatz im Common-Bereich reserviert werden soll.

WIRKUNG: Die COMMON-Anweisung legt für das laufende Programm die Variablen fest, die ihre Werte an ein anderes Programm übergeben oder von einem anderen Programm eine Wertzuweisung erhalten sollen.

Der Speicherplatz für die Variablen der COMMON-Anweisung wird in der dort festgelegten Reihenfolge mit den gültigen Deklarationen und Dimensionen im Common-Bereich reserviert.

BEMERKUNGEN:

- Um den gemeinsamen Zugriff zu ermöglichen, reserviert das System einen Common-Bereich, der durch das Ende der Programmausführung nicht gelöscht wird. Die Größe (in Worten, 1 Wort=4 Bytes) des Common-Bereiches wird vom Anwender im Systembefehl SAVE durch Definition des Parameters "COM=n" festgelegt.
- Wird ein mit Angabe des Parameters "COM=n" gespeichertes Programm in den Arbeitsspeicher geladen, reserviert das System die ersten n Worte des Arbeitsspeichers für den Common-Bereich. Benutzte also das unmittelbar vorher geladene Programm einen gleich großen oder kleineren Common-Bereich, bleiben die darin gespeicherten Daten unverändert erhalten und stehen im laufenden Programm zur Verfügung.

- In einem BASIC-Programm darf nur eine COMMON-Anweisung vorhanden sein.

- Ein eine COMMON-Anweisung enthaltendes BASIC-Programm kann nur ausgeführt werden, wenn es unter Angabe des Parameters "COM=n" mit dem Befehl SAVE gespeichert wurde. Dabei ist n der Speicherbedarf in Worten für im Common-Bereich zu speichernde Variablenwerte.

- Wird ein Programm von Disk geladen und dort nachträglich eine COMMON-Anweisung eingefügt, so muß es anschließend mit "PURGE" auf der Disk gelöscht werden, bevor es mit "SAVE" und Parameter "COM=..." abgespeichert werden kann.

- Die Durchführung eines Programmes, das einen Common-Bereich verwendet, kann mit "CHAIN", "RUN filename" oder "OLD filename","RUN" aufgerufen werden.

- Die Variablen, die Werte an den Common-Bereich übergeben und jene, die diese Werte dem Common-Bereich entnehmen, müssen nicht den gleichen Namen tragen, jedoch in Typ und Deklaration übereinstimmen.

- Der Platzbedarf des Common-Bereiches läßt sich wie folgt berechnen:
  . Jedes Element eines numerischen Feldes in einfacher Genauigkeit benötigt 4 Bytes = 1 Wort.
  . Jedes Element eines numerischen Feldes in doppelter Genauigkeit benötigt 8 Bytes = 2 Worte.
  . Jedes Element eines Stringfeldes belegt K+2 Bytes, wobei K die Länge (Anzahl Bytes) dieses Elementes ist.

- Werte von Elementen eines Feldes können in einem Folgeprogramm auf mehrere Felder aufgeteilt werden, wobei jedoch die Summe der Elemente der Felder im zweiten Programm der Anzahl Elemente des ursprünglichen Feldes entsprechen muß. Bei numerischen Feldern muß außerdem die Genauigkeit (doppelt oder einfach), bei alphanumerischen Feldern die deklarierte Länge übereinstimmen.

- Der Inhalt einer einfachen Stringvariablen kann nicht in Teilstrings zerlegt und in einem Folgeprogramm Elementen eines Stringfeldes zugewiesen werden.

- Der für die Variablen einer COMMON-Anweisung benötigte Platz darf die im Systembefehl SAVE durch Parameter "COM=n" reservierte Anzahl Bytes nicht übersteigen.

- Der für den Common-Bereich reservierte Platz kann in den verschiedenen Programmen unterschiedlich groß sein. Beträgt der Common-Bereich im die Daten zur Verfügung stellenden Programm n Bytes, im Folgeprogramm jedoch m Bytes, so gilt:
  . m < n: Die in den letzten (n-m) Bytes gespeicherten Daten gehen verloren.
  . m > n: Alle Daten bleiben erhalten. Die im neu reservierten Teil des Common-Bereiches liegenden Variablen haben jedoch keinen Inhalt, bis ihnen vom Programm ein Wert zugewiesen wird.

- Der Common-Bereich enthält keine signifikanten Daten, bis den Variablen erstmals ein Wert zugewiesen wird.

BEISPIEL:

```
FILE     PROG1


0010 COMMON X(),A$(),B$
0020 DIM X(2,2),A$(4)
0030 DCL SX(),6A$(),20B$
0040 LET X(1,1)=1
0050 LET X(1,2)=2
0060 LET X(2,1)=3
0070 LET X(2,2)=4
0080 LET A$(1)="EINS"
0090 LET A$(2)="ZWEI"
0100 LET A$(3)="DREI"
0110 LET A$(4)="VIER"
0120 LET B$=" in Worten lautet "
0130 REM
0140 REM
0150 CHAIN "PROG2"
1000 END

END OF LISTING


FILE     PROG2


0010 COMMON X(),A$(),B$
0020 DIM X(2,2),A$(4),Z(4)
0030 DCL SX(),6A$(),20B$
0040 LET Z(1)=X(1,1)
0050 LET Z(2)=X(1,2)
0060 LET Z(3)=X(2,1)
0070 LET Z(4)=X(2,2)
0080 FOR I=1 TO 4 STEP 1
0090 PRINT Z(I);B$,A$(I)
0100 NEXT I
0110 END

END OF LISTING


RUN PROG1
**** FORMALLY CORRECT PROGRAM ****
**** FORMALLY CORRECT PROGRAM ****
 1   in Worten lautet            EINS
 2   in Worten lautet            ZWEI
 3   in Worten lautet            DREI
 4   in Worten lautet            VIER
```

CONVERT

ANWEISUNG: CONVERT

FUNKTION: Umwandlung einer Zeichenkette in die entsprechenden numerischen Codes der ISO-Tabelle oder umgekehrt.

FORMAT: 1) CONVERT Stringausdr. TO num.Vektor LENGTH num.Variable

2) CONVERT num.Vektor TO Stringvar. LENGTH arithm.Ausdruck

WIRKUNG: 1. Format:

Der Stringausdruck wird berechnet. Jedes Zeichen des sich daraus ergebenden Strings erhält den entsprechenden numerischen Code aus der ISO-Tabelle (siehe Anhang) und wird der Reihe nach den Elementen des Vektors aus der Anweisung zugewiesen. Die "Längen"-Variable enthält die Anzahl der umgewandelten Zeichen. Dieser Wert wird vom System automatisch ermittelt.

2. Format:

Der arithmetische Ausdruck wird berechnet und das Ergebnis auf die nächste ganze Zahl n gerundet. Die Werte der ersten n Elemente des Vektors aus der Anweisung werden ebenfalls auf die nächste ganze Zahl gerundet und dann in die entsprechenden Zeichen der ISO-Tabelle umgewandelt. Der daraus entstehende Zeichenstring wird der entsprechenden Stringvariablen zugewiesen.

BEISPIEL:

```
FILE     CONVER


0010 REM *PROGRAMMBEISPIEL FUER DIE ANWEISUNG 'CONVERT'
0020 REM
0030 DCL 32A$
0040 DIM A(32)
0050 PRINT
0060 PRINT
0070 DISP "STRING EINGEBEN";
0080 RKB A$
0090 CONVERT A$ TO A LENGTH L
0100 PRINT "A$= '";A$,"'"
0110 PRINT "LAENGE";L
0120 PRINT "ISO CODE";
0130 FOR I=1 TO L STEP 1
0140 PRINT A(I);
0150 NEXT I
0160 PRINT
0170 PRINT
0180 DISP "LAENGE DES VEKTORS";
0190 INPUT N
0200 PRINT "LAENGE";N
0210 PRINT "ISO-CODE";
0220 FOR I=1 TO N STEP 1
0230 DISP "ELEMENT";I;
0240 INPUT A(I)
0250 PRINT A(I);
0260 NEXT I
0270 PRINT
0280 CONVERT A TO A$ LENGTH N
0290 PRINT "A$= '";A$,"'"
0300 GOTO 50
0310 END

END OF LISTING



RUN


STRING EINGEBEN?
OLIVETTI
A$= 'OLIVETTI'
LAENGE 8
ISO CODE 79  76  73  86  69  84  84  73

LAENGE DES VEKTORS?
8
LAENGE 8
ELEMENT 1 ?
79
ELEMENT 2 ?
76
ELEMENT 3 ?
73
ELEMENT 4 ?
86
ELEMENT 5 ?
69
ELEMENT 6 ?
84
ELEMENT 7 ?
84
ELEMENT 8 ?
73
ISO-CODE 79  76  73  86  69  84  84  73
A$= 'OLIVETTI'
```

DATA

ANWEISUNG: DATA

FUNKTION: Erzeugen eines internen Files von Daten, die anschliessend den Variablen aus den READ- und MATREAD-Anweisungen zugewiesen werden.

FORMAT: DATA { num. Konst. / Stringkonst. } [, { num. Konst. / Stringkonst. } .....]

WIRKUNG: Bei Beginn der Programmausführung wird im Hauptspeicher eine Tabelle erzeugt, die alle Konstanten aller DATA-Anweisungen des Programmes enthält. Ein Pointer zeigt auf das erste Element der Tabelle. Die Konstanten der Tabelle werden den Variablen aus den READ- und MATREAD-Anweisungen zugewiesen. Nach jeder Zuweisung zeigt der Pointer auf das jeweils nächste Element der Tabelle. (Mit der Anweisung RESTORE kann der Pointer wieder auf das erste Element zurückgesetzt werden.)

BEMERKUNGEN:

- Stringkonstanten, die am Anfang oder am Ende ein Komma (,) oder ein Leerzeichen enthalten, müssen in Anführungszeichen stehen.
- Die DATA-Anweisungen können an jeder Stelle im Programm stehen. Jede Stringkonstante muss einer Stringvariablen zugewiesen werden. Jeder numerischen Variablen einer READ- oder MATREAD-Anweisung muss eine numerische Konstante aus der Tabelle entsprechen.

BEISPIELE:

Beispiel 1:

```
FILE     DATA


0010 PRINT
0020 PRINT "IHR TYP:"
0030 PRINT
0040 RESTORE
0050 FOR I=1 TO 5 STEP 1
0060 READ A$
0070 DISP A$;
0080 RKB B$
0090 PRINT A$;":",B$
0100 NEXT I
0110 GOTO 10
0120 DATA "ALTER","GROESSE","GEWICHT","HAARFARBE","BES.KENNZ."
0130 END

END OF LISTING



RUN

IHR TYP:

ALTER:          30
GROESSE:        1.80
GEWICHT:        72.5
HAARFARBE:      BRAUN
BES.KENNZ.:     BART
```

Beispiel 2:

```
FILE


0010 DATA "EINS",1,"ZWEI",2,"DREI",3,"VIER",4
0020 DCL 5A$
0030 FOR I=1 TO 4 STEP 1
0040 READ A$,Z
0050 PRINT A$;" in Zahlen lautet ";Z
0060 NEXT I
0070 END

END OF LISTING



RUN
EINS in Zahlen lautet  1
ZWEI in Zahlen lautet  2
DREI in Zahlen lautet  3
VIER in Zahlen lautet  4
```

ANWEISUNG: DCL (declare)

FUNKTION: Festlegung der mit einfacher Genauigkeit zu verarbeitenden Variablen und der maximal vorgesehenen Längen für Strings.

FORMAT:

```
DCL { { S( num.Var.    )[,num.Var.    ]...)[,S(num.Var.    )...] }
    { { n( Stringvar.  )[,Stringvar.  ]...)[,n(Stringvar.  )...] }...
    { SINGLE                                                     }
```

n ist eine ganze Zahl zwischen 1 und 1023. Nach einem Feldnamen muss ein leeres Klammernpaar folgen. Beziehen sich S und n auf eine einzige numerische oder Stringvariable, können die Klammern wegfallen.

WIRKUNG: Die Werte der numerischen Variablen zwischen den Klammern hinter dem S werden in einfacher Genauigkeit dargestellt, desgleichen auch die Elemente von Feldern. Wird bei einer DCL-Anweisung der Parameter SINGLE angegeben, werden alle Werte der numerischen Variablen in einfacher Genauigkeit gerechnet. Den Stringvariablen, auf die sich die Zahl n bezieht, werden im Hauptspeicher n Bytes zugewiesen. Auch für jedes Feldelement eines Stringfeldes werden im Hauptspeicher n Bytes reserviert.

BEMERKUNGEN:

- Bezieht man sich in einem Programm mehrmals, durch verschiedene DCL-Anweisungen, auf dieselbe Variable, hat die Anweisung mit der höchsten Zeilennummer Gültigkeit. Bei mehreren Deklarationen innerhalb einer Anweisung ist die jeweils letzte gültig.
- Ergebnisse von numerischen Ausdrücken sind immer in doppelter Genauigkeit gerechnet.

BEISPIELE:

```
30 DCL SINGLE
```

```
90 DCL 30A$
```

```
20 DCL 30(A$,B$,C$(),D$),S(I,J,K)
```

```
10 DCL S(I,I1,I2),20 (A$,E3$),10(B$,D$)
```

ANWEISUNG: DEF

FUNKTION: Nicht ausführbare Anweisung, die innerhalb einer Zeile eine numerische oder eine Stringfunktion definiert (Definition einer einzeiligen Funktion).

FORMAT: DEF { FNα (Parameterliste) = num. Ausdruck / FNα$ (Parameterliste) = Stringausdruck }

"α" A...Z, Parameterliste = einfache numerische oder alphanumerische Variable. Für "α" muß ein Großbuchstabe des Alphabetes stehen.

"FNα" Name einer numerischen Funktion, die als Ergebnis einen numerischen Wert liefert. Parameter können numerische oder Stringvariable sein.

"FNα$" Name einer Stringfunktion. Als Ergebnis wird ein String geliefert. Parameter können numerische oder Stringvariable sein.

Die in einer DEF-Anweisung innerhalb der Klammern stehenden Parameter sind Pseudovariable, die keinen Bezug zu gleichnamigen Variablen außerhalb der Funktion haben. Beim Funktionsaufruf sind diese Parameter durch entsprechende Variable zu ersetzen, wodurch der Pseudovariablen der aktuelle Wert der an entsprechender Stelle im Aufruf stehenden Variablen zugewiesen wird. Die Pseudovariablen werden "formale Parameter" genannt.

Auf der rechten Seite der Anweisung können außer den Parameternamen auch andere Variable, sogenannte globale Variable, stehen. Diese müssen aber schon vor dem Funktionsaufruf einen Wert besitzen.

WIRKUNG: Die Funktion FNα oder FNα$ wird durch den Ausdruck auf der rechten Seite definiert. Die Funktion wird in einem Programm so oft ausgeführt, wie ihr Name FNα oder FNα$ zusammen mit den aktuellen Parametern vorkommt.

BEMERKUNGEN:

- Eine Funktion kann an jeder Stelle im Programm stehen, darf jedoch nur ein einziges Mal definiert werden und in dieser Form nur eine Zeile umfassen.
- Auch direkte rekursive Aufruffolgen (z.B. 10 DEF FNA=FNA) o. indirekte rekursive Aufruffolgen (z.B. 10 DEF FNA=X+FNB
  50 DEF FNB=FNA+Y)
  sind nicht erlaubt.
- Maximal 15 Parameter können verwendet werden. Formale Parameter haben keinerlei Beziehung zu Variablen mit gleichem Namen.
- Tritt während einem Programmlauf die Anweisung DEF auf, wird die Verarbeitung in der folgenden Programmzeile fortgesetzt.
- Die aktuellen Parameter müssen in Typ und Anzahl mit den formalen übereinstimmen. Maximal 26 Funktionen dürfen in einem Programm definiert werden.
- Wird eine Funktionsdefinition und alle ihre Aufrufe gelöscht, so ist für eine fehlerfreie Preexecution DECOMPILE und COMPILE auszuführen.

BEISPIELE:

Beispiel 1

Einzeilige numerische Funktion.

```
FILE     DEF1


0010 DEF FNB(X)=PI/180*X
0020 DEF FNC(X)=COS(FNB(X))
0030 DEF FNS(X)=SIN(FNB(X))
0040 DEF FNT(X)=TAN(FNB(X))
0050 DEF FNA(X)=180*ATN(X)/PI
0060 PRINT "  X        SIN        COS        TAN        ATN"
0070 PRINT
0080 DISP "VON GRAD?,BIS GRAD?,SCHRITTWEITE";
0090 INPUT A,B,C
0100 FOR I=A TO B STEP C
0110 PRINT USING 150,I,FNS(I),FNC(I),FNT(I),FNA(I)
0120 NEXT I
0130 PRINT
0140 GOTO 80
0150 :###  ##.######  ##.######  ##.######  ####.####
0160 END

END OF LISTING


  X        SIN        COS        TAN        AIN

  1   0.017452   0.999848   0.017455    45.0000
  2   0.034899   0.999391   0.034921    63.4349
  3   0.052336   0.998630   0.052408    71.5651
  4   0.069756   0.997564   0.069927    75.9638
  5   0.087156   0.996195   0.087489    78.6901

 40   0.642788   0.766044   0.839100    88.5679
 45   0.707107   0.707107   1.000000    88.7270
 50   0.766044   0.642788   1.191754    88.8542
```

Beispiel 2:

Einzeilige alphanumerische Funktion.

```
FILE


0010 DEF FNA$(A$,A,B)=EXT$(A$,A,B)
0020 DISP "STRING";
0030 RKB X$
0040 DISP "VON....BIS";
0050 INPUT X,Y
0060 PRINT X$;TAB(25);"EXT$(";X$;",";X;",";Y;")=";FNA$(X$,X,Y)
0070 GOTO 20
0080 END

END OF LISTING



RUN
TELESKOPHALS              EXT$(TELESKOPHALS, 1 , 8 )=TELESKOP

TELESKOPHALS              EXT$(TELESKOPHALS, 9 , 12 )=HALS


OLIVETTI P6066            EXT$(OLIVETTI P6066, 10 , 14 )=P6066
OLIVETTI P6066            EXT$(OLIVETTI P6066, 4 , 8 )=VETTI
```

DEF/FNEND

ANWEISUNG: DEF/FNEND (define/function end)

FUNKTION: Nicht ausführbare Anweisungen zur Definition einer mehrzeiligen numerischen oder Stringfunktion.

FORMAT:

```
DEF {FNα [(Parameterliste)][Liste d. lokalen Variablen]}
    {FNα$[(Parameterliste)][Liste d. lokalen Variablen]}
     .
     .
     .
    FN*   = num. Ausdruck
    FN*$  = Stringausdruck          mind. eine Zuweisung
     .
     .
     .
    FNEND
```

"FNα"	Name einer numerischen Funktion; dem Funktionsnamen wird der errechnete Wert zugewiesen.
FNα$	Name einer Stringfunktion; dem Funktionsnamen wird als Ergebnis ein Zeichenstring zugewiesen.
"Parameterliste"	Einfache numerische oder alphanumerische Variable.
"lokale Variable"	Im Funktionsprogramm verwendete numerische oder alphanumerische Variable, die zu gleichnamigen Variablen im Hauptprogramm keine Beziehung haben.
"FN*"/"FN*$"	Pseudovariable, welchen vor dem Rücksprung aus der Funktion FNα oder FNα$ der jeweilige Funktionswert zugewiesen wird.

In arithmetischen oder Stringausdrücken können außer Parametern und lokalen Variablen auch globale Variable auftreten.

Zwischen den Programmzeilen DEF und FNEND können neben den Anweisungen für die Berechnung der Werte der Pseudovariablen (FN* und FN*$) beliebige BASIC-Anweisungen stehen.

WIRKUNG: Die Funktion FNα oder FNα$ wird durch alle Anweisungen zwischen den Programmzeilen DEF FNα oder DEF FNα$ und der Zeile FNEND definiert. Sie wird in einem Programm so oft ausgeführt, wie ihr Name FNα oder FNα$ zusammen mit den aktuellen Parametern im Hauptprogramm vorkommt.

BEMERKUNGEN:

- Eine sich über mehrere Zeilen erstreckende Funktion kann an jeder Stelle des Programmes definiert werden.
- Funktionsdefinitionen sind physisch in sich abgeschlossene Programmteile. Es darf deshalb in eine Funktionsdefinition weder verzweigt noch diese durch eine Sprunganweisung verlassen werden (z.B. IF...THEN, READ...EOF). Funktionsdefinitionen dürfen auch keine STOP-Anweisungen enthalten.
- Verzweigungen innerhalb einer Funktion sind zulässig.
- Mit der Anweisung DCL ist es möglich, die Länge der Pseudostringvariablen FN*$ und der lokalen Stringvariablen auch innerhalb einer Funktion festzulegen.
- Parameter und lokale numerische Variable arbeiten mit doppelter Genauigkeit.
- Der Wert einer globalen Größe wird innerhalb einer Funktion verändert, wenn die Funktion eine Zuweisung an die globale Variable enthält. Im Inneren einer Funktion sind Wertzuweisungen an Parameter gestattet, der Wert des entsprechenden Argumentes bleibt jedoch unverändert. Die aktuellen Parameter müssen in Typ und Anzahl den formalen entsprechen. Tritt während der Verarbeitung eines Programmes die Anweisung DEF auf, wird die Verarbeitung in der dem FNEND folgenden Programmzeile fortgesetzt.
- Parameter haben keinerlei Beziehung zu im Programm vorkommenden Variablen mit gleichem Namen.
- Lokalen Variablen muß innerhalb der Funktion ein Wert zugewiesen werden. Namen von lokalen Variablen haben keinerlei Beziehung zu etwaigen, gleichlautenden Namen von Variablen im Programm, das heißt, die Variablen im Hauptprogramm werden durch eine Wertzuweisung an eine gleichnamige lokale Variable nicht verändert. Insgesamt dürfen maximal 15 Parameter und lokale Variable verwendet werden.
- In einem Programm dürfen maximal 26 Funktionen vorkommen.
- Wird eine Funktionsdefinition und all ihre Aufrufe gelöscht, ist für eine fehlerfreie Preexecution DECOMPILE und COMPILE auszuführen.
- Fehler, die während der Ausführung von Funktionen auftreten, werden mit der Zeilennummer der aufrufenden Zeile gemeldet.

BEISPIELE:

Beispiel 1

Mehrzeilige numerische Funktion.

```
FILE     FNEND1


0010 DEF FNA(I,J)L
0020 LET L=I-INT(I/J)*J
0030 IF L=0 THEN 70
0040 LET I=J
0050 LET J=L
0060 GOTO 20
0070 LET FN*=J
0080 FNEND
0090 DISP "EINGABE A,B";
0100 INPUT A,B
0110 PRINT "DER GROESSTE GEMIENSAME TEILER VON";A;"UND";B;"IST";FNA(A,B)
0120 GOTO 90
0130 END

END OF LISTING



DER GROESSTE GEMIENSAME TEILER VON 85 UND 63 IST 1
DER GROESSTE GEMIENSAME TEILER VON 125 UND 35 IST 5
DER GROESSTE GEMIENSAME TEILER VON 95478 UND 24 IST 6
```

Beispiel 2

Mehrzeilige alphanumerische Funktion.

```
FILE     FNEND2


0010 DEF FNA$(X$)I,A$
0020 LET A$=""
0030 FOR I=LEN(X$) TO 1 STEP -1
0040 LET A$=A$+EXT$(X$,I,I)
0050 NEXT I
0060 LET FN*$=A$
0070 FNEND
0080 DISP "EINGABE STRING";
0090 RKB A$
0100 PRINT USING 110,A$,FNA$(A$)
0110 : 'RRRRRRRRRRR I 'LLLLLLLLLLLL
0120 GOTO 80
0130 END

END OF LISTING



          EIN I NIE
        NEGER I REGEN
          MIT I TIM
      GAZELLE I ELLEZAG
```

DEPAD

ANWEISUNG: DEPAD

FUNKTION: Eliminieren der Füllzeichen am Ende einer Stringvariablen.

FORMAT: DEPAD Stringvariable, n

"n" ganze Zahl zwischen 0 und 255.

WIRKUNG: Aus Stringvariablen werden die Zeichen entfernt, die in der ISO-Tabelle der Zahl n entsprechen. Sie werden rechts beginnend eliminiert, und zwar so lange, bis das erste, nicht dem ISO-Code n entsprechende Zeichen auftritt.

BEMERKUNGEN:
- Sollen die binären Füllzeichen einer mit BPAD aufgefüllten Stringvariablen entfernt werden, ist n = 255 einzusetzen.

BEISPIEL:

```
FILE     DEPAD


0010 REM *PROGRAMMBEISPIEL 'PAD' UND 'DEPAD'*
0020 REM
0030 DCL 32A$
0040 PRINT
0050 PRINT
0060 DISP "STRING EINGEBEN";
0070 RKB A$
0080 PRINT "EINGABE: '";A$;"'"
0090 PAD A$,42
0100 PRINT "A$ NACH 'PAD': '";A$;"'"
0110 DEPAD A$,42
0120 PRINT "A$ NACH 'DEPAD': '";A$;"'"
0130 GOTO 40
0140 END

END OF LISTING




EINGABE: 'OLIVETTI'
A$ NACH 'PAD': 'OLIVETTI************************'
A$ NACH 'DEPAD': 'OLIVETTI'


EINGABE: 'BASIC  P6066'
A$ NACH 'PAD': 'BASIC  P6066********************'
A$ NACH 'DEPAD': 'BASIC  P6066'
```

DIM

ANWEISUNG: DIM (dimension)

FUNKTION: Festlegung der Dimension eines oder mehrerer Felder.

FORMAT: DIM Feldname(Zeilen[,Spalten])[,Feldname(Zeilen[,Spalten])]

WIRKUNG: Folgt einem Feldnamen nur eine Zahl r zwischen Klammern, so handelt es sich um einen Vektor mit r Elementen. Alle im Programm verwendeten Feldindizes müssen kleiner oder gleich r sein. Folgt einem Feldnamen ein Zahlenpaar zwischen Klammern (r,c), so handelt es sich um eine Matrix mit r Zeilen und c Spalten. Die Indizes dürfen die jeweiligen Feldgrenzen nicht überschreiten ($\leq$r und $\leq$c). Die Indizes einer Feldvereinbarung müssen größer 0 sein.

BEMERKUNGEN:

- Wird ein eindimensionales Feld nicht durch DIM deklariert, erhält es vom System 10 Elemente zugewiesen.
- Wird ein zweidimensionales Feld nicht durch DIM deklariert, erhält es vom System 10 Zeilen und 10 Spalten bei numerischen, 5 Zeilen und 5 Spalten bei alphanumerischen Feldern zugewiesen.
- Der vom Compiler zugelassene Höchstwert für eine Dimension beträgt 65536. Bei zwei Dimensionen muß das Produkt kleiner sein als 65536. Die tatsächlich höchstzulässigen Werte für die Dimensionen hängen von der Größe des Hauptspeichers und seiner Belegung durch das Programm ab.
- Die Anweisung DIM kann an jeder Stelle in einem Programm stehen.
- Bezieht man sich in einem Programm mehrmals, durch verschiedene DIM-Anweisungen, auf dieselbe Variable, hat die Anweisung mit der höchsten Zeilennummer Gültigkeit. Bei mehreren Dimensionierungen innerhalb einer Anweisung ist die jeweils letzte gültig.
- Ein eindimensionales Feld darf nicht den gleichen Namen haben wie ein zweidimensionales.
- Jedes Feldelement wird zu Beginn mit dem Wert "nicht definiert" belegt.

BEISPIEL:

```
FILE     DIM


0010 REM BEISPIEL 'DIM'
0020 REM
0030 DCL SA()
0040 DIM A(2,3)
0050 FOR I=1 TO 2 STEP 1
0060 FOR J=1 TO 3 STEP 1
0070 LET A(I,J)=RND*100
0080 NEXT J
0090 NEXT I
0100 DISP "WELCHES ELEMENT";
0110 INPUT I,J
0120 PRINT "A(";I;",";J;") =";A(I,J)
0130 GOTO 100
0140 END

END OF LISTING



A( 1 , 1 ) = 63.8293
A( 2 , 2 ) = 21.5078
A( 1 , 3 ) = 97.356
A( 1.5 , 2.1 ) = 41.8393
A( 1 , 2 ) = 41.8393
```

ANWEISUNG: DISP (display)

FUNKTION: Ausgabe von Daten an beliebigen Positionen auf dem Bildschirm und in der zweiten Systemzeile.

FORMAT 1: DISP "0" + CHR$ (Operationscode)
+ CHR$ (30 + Zeile) + CHR$ (31 + Spalte)
+ Stringausdruck

- 'Operationscode', 'Zeile' und 'Spalte' können numerische Variable, numerische Konstanten oder das Ergebnis einer numerischen Operation bzw. Funktion sein.
- Die Wirkung der Stringfunktion CHR$ ist auf Seite 9.157 erläutert.
- 'Zeile' gibt die jeweilige Zeile an, in der 'Stringausdruck' am Bildschirm ausgegeben werden soll. Für 'Zeile' ist einer der Werte zwischen 2 und 24 (einschließlich) zulässig (vgl. Kap. 1.2.3).
- 'Spalte' gibt die jeweilige Spalte an, ob 'Stringausdruck' am Bildschirm erscheinen soll. Die zulässigen Werte liegen zwischen 1 und 80 (einschließlich) (vgl. Kap. 1.2.3).
- 'Stringausdruck' ist die Zeichenkette, die am Bildschirm ausgegeben werden soll.

WIRKUNG: Durch den Operationscode wird das Ausgabe-Format bestimmt und der angegebene String wird an der definierten Stelle am Bildschirm angezeigt.

BEMERKUNGEN: Das Steuerzeichen Θ wird erzeugt, indem die Taste CONTROL festgehalten
und die Taste [*/+] bzw. [£/C] (ASCII-Tastatur) gedrückt wird.
Dieses Zeichen muß als erste Stringkonstante nach dem Schlüsselwort stehen. Mit diesem Zeichen wird dem System mitgeteilt, daß der 'Stringausdruck' nicht in der Display-Zeile (Zeile 0) am Bildschirm, sondern an anderen Zeilen angezeigt werden soll. Der dezimale Wert des ESCAPE-Zeichens ist 27.

Paßt ein 'Stringausdruck' nicht in eine Zeile, so wird er an der Stelle hinter dem Zeichen abgeschnitten, das an der 80. Position steht. Der restliche Teil des 'Stringausdrucks' wird nicht angezeigt (auch nicht in der nächsten Zeile). Der 'Stringausdruck' kann eine Stringkonstante, eine Stringvariable oder das Ergebnis einer Stringfunktion bzw. -operation sein.

Zum 'Operationscode' : Mit diesem Code wird festgelegt, in welchem Format (z.B. blinkend, in negativer Schrift) der 'Stringausdruck' am Bildschirm angezeigt wird.

Die am Bildschirm ausgegebenen Stringausdrücke bleiben am Bildschirm erhalten, wenn das Programm beendet oder abgebrochen worden ist. Während eines Programmlaufs können die Stringausdrücke mit einer weiteren BASIC-Anweisung DISP oder mit der BASIC-Anweisung ERASE gelöscht werden. Nach Programmende oder -abbruch können sie mit dem Systembefehl ERASE, ggf. mit Parameter A gelöscht werden.

Fast alle Formate werden von dem alphanumerischen Controler verwaltet, durch den der Bildschirm in 23 Zeilen (+ 2 Zeilen Display-Meldungen und für Systemmeldungen) à 80 Zeichen eingeteilt wird.

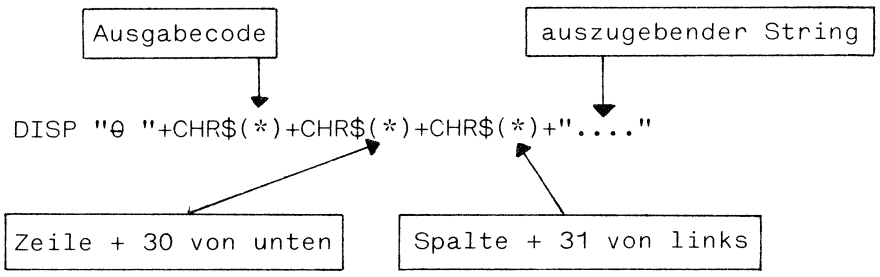
Einige weitere Formate werden von dem graphischen Controler verwaltet. Dieser teilt den Bildschirm in 39 Zeilen (+ 2 Zeilen) à 80 Zeichen ein.

Zunächst werden die 'Operationscodes' erläutert, die zu dem Alpha-Mode (23 Zeilen à 80 Zeichen) gehören. Der 'Operationscode' ist eine Zahl, die der unten stehenden Tabelle entnommen werden kann und vom Anwender zu bestimmen ist. Die Tabelle ist folgendermaßen zu verstehen: Soll in dem 'Stringausdruck' jedes Zeichen umrahmt und blinkend angegeben werden, so ist als 'Operationscode' die Zahl 31 zu wählen.

Textdarstellung auf dem Bildschirm im Graphik-Mode

(39 Zeilen à 80 Zeichen)

Beispiel Buchst. A		normal	blinking	highlight	blink+high	reverse	blink+rev.
Alpha-Schirm mit 23 Zeilen	A	0	16	32	48	64	80
	A	1	17	33	49	-	81
	A	2	18	34	50	-	82
	A	3	19	35	51	67	-
	A	4	20	36	52	68	84
	A	5	21	37	53	69	85
	A	6	22	38	54	70	86
	A	7	23	39	55	71	-
	A	8	24	40	56	72	-
	A	9	25	41	57	73	-
	A	10	26	42	58	74	-
	A	11	27	43	59	75	91
	A	12	28	44	60	76	92
	A	13	29	45	61	77	93
	A	14	30	46	62	78	94
	A	15	31	47	63	79	95



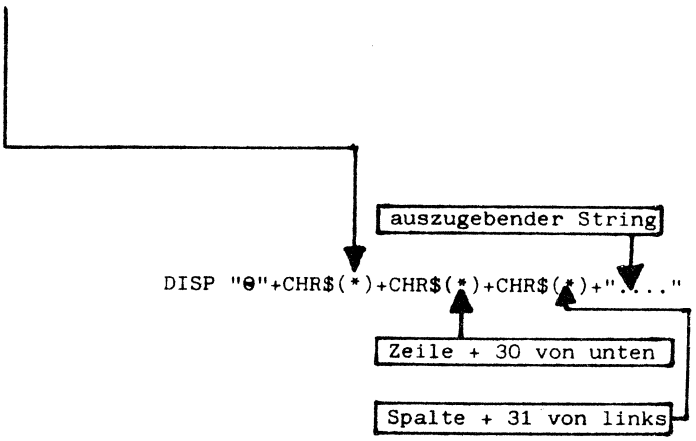
Bemerkung: Als Ausgabecode ist einer der in der Tabelle dargestellten Dezimalzahlen einzutragen.

Es folgen die 'Operationscodes' und deren Bedeutung, die mit dem Graphik-Mode (39 Zeilen à 80 Zeichen) verwaltet werden.

Textdarstellung auf dem Bildschirm im Graphik-Mode (39 Zeilen à 80 Zeichen)

Graphik - Controller

	Dezimal-Code		Iso-Zeichen	Bedeutung
Grafik - Schirm mit 39 Zeilen	65	≙	A	Aufhebung von Code B
	66	≙	B	Nachfolgende PRINT-Anweisung nicht auf Schirm dargestellt
	83	≙	S	Alle Zeilen um eine Zeile nach oben verschoben
	87	≙	W	Auszugebender String ersetzt alle anderen Zeichen in norm. Schrift
	88	≙	X	Auszugebender String erscheint zusätzlich in negativer Schrift
	89	≙	Y	Auszugebender String erscheint zusätzlich in normaler Schrift
	90	≙	Z	Auszugebender String ersetzt alle anderen Zeichen in neg. Schrift



Der 'Operationscode' ist stets der maximale Wert (oben: Dezimal-Code). Statt CHR$ ('Operationscode' kann auch das entsprechende ISO-Code-Zeichen angegeben werden, also z.B. "A" statt CHR$ (65).

BEISPIELE:

Beispiel 1

Mit dem folgenden Programm wird die Stringkonstante "M3040" in Zeile 12 ab Position 38 ausgegeben. Dabei ist jedes Zeichen eingerahmt.

```
FILE

0010 ERASE
0020 DISP "Θ"+CHR$(15)+CHR$(30+12)+CHR$(31+38)+"M30/M40"
0090 END
```

Beispiel 2

Mit dem folgenden Programm erfolgt die Ausgabe in derselben Position in derselben Zeile wie im obigen Programm.
Die Darstellung erfolgt blinkend und in Negativschrift

```
FILE

10 ERASE
20 P=80
30 Z=30+12
40 S=31+38
50 M$="M30 M40"
60 DISP "θ"+CHR$(P)+CHR$(Z)+CHR$(S)+M$
999 END
```

Beispiel 3

Mit dem Programm *DISP werden die Zeilen- und Spaltenpositionen am Bildschirm angezeigt:

```
FILE     *DISP

0010 DISP "ZEILEN-NR.,POSITION";
0020 INPUT A,B
0025 LET P=0
0030 IF (A<=1) OR (A>=41) THEN 10
0040 IF (B<=0) OR (B>=81) THEN 10
0045 BUILD A$,A,B;32
0050 DISP "θ"+CHR$(P)+CHR$(30+A)+CHR$(31+B)+A$
0999 END
```

Beispiel 4

```
FILE     *DISPU

0010 DISP "ZEILEN-NR.,POSITION";
0020 INPUT A,B
0030 IF (A<=1) OR (A>=41) THEN 10
0040 IF (B<=0) OR (B>=81) THEN 10
0045 A=A+30
0046 B=B+31
0050 DISP USING 70,"θ"+CHR$(0)+CHR$(A)+CHR$(B),A,B
0060 GOTO 10
0070 :ZEILE:###, POS.:###
0999 END
```

Das Zeichen a hat den dezimalen Wert 60, d.h. der Text steht in Zeile 30.

Das Zeichen b hat den dezimalen Wert 41, d.h. der Text beginnt in Position 10 von Zeile 30 (vgl. Erläuterung zum Befehl DISP).

FORMAT 2:

DISP [{ num.Ausdruck / Stringausdruck / TAB (num.Ausdr.) } [{ ; / , } { num.Ausdruck / Stringausdruck / TAB (num.Ausdr.) }] ... [{ ; / , }]

"," / ";" Trennzeichen mit einer bestimmten Bedeutung für die Ausgabe (siehe Kapitel 8.5)

WIRKUNG: Die Ergebnisse der Ausdrücke werden im Standardformat dargestellt und im Display sichtbar gemacht.

Die Position der Ausdrücke in einer Zeile hängt sowohl von der erforderlichen Länge der Darstellung als auch von den verwendeten Trennzeichen (Komma oder Strichpunkt) und der Funktion TAB ab.

Stellenkontrolle der Zeichen im Display

Die Ergebnisse eines Ausdruckes werden durch die Anweisung DISP in einem Register, dem Display-Buffer, in Form einer Folge von Zeichen dargestellt und in der zweiten Systemzeile angezeigt.

BEMERKUNGEN:

- Zusätzliche Möglichkeiten der Anweisung DISP für das graphische Display sind im Kapitel 9.3.3 aufgeführt.
- Die Anweisung DISP ohne Angabe von parametern löscht den Inhalt des Buffers und setzt den Pointer an die erste Stelle.
- Das Standardformat ist in Abschnitt 8.5 genau beschrieben.

BEISPIEL: PRINT ALL-Taste aktiviert.

```
FILE     DISP

0010 DISP "WERTE VON X UND Y";
0020 INPUT X,Y
0030 PRINT X,Y,X↑Y
0040 GOTO 10
0050 END

END OF LISTING



RUN
WERTE VON X UND Y?
3,5
 3                5                243
WERTE VON X UND Y?
2,31
 2                31               2.1474836E+09
WERTE VON X UND Y?
```

DISP USING

ANWEISUNG: DISP USING

FUNKTION: Darstellung von formatiertem Text auf dem Bildschirm

FORMAT 1:

DISP USING n, "Θ"+CHR$(Code)[+CHR$(a)+CHR$(b)][,{num.Ausdr. / Stringausdr.}][,{num.Ausdr. / Stringausdr.}]

mit n = Zeilennummer der Formatzeile

Θ = Zeichen 27 der ISO-Code-Tabelle

Code = dezimaler Wert einer Dualzahl, die das Format des darzustellenden Ausdrucks bestimmt (vgl. Anweisung DISP)

a = Zeichen der ISO-Code-Tabelle zwischen 32 und 70

b = Zeichen der ISO-Code-Tabelle zwischen 32 und 111

num.Ausdr. ist ein beliebiger numerischer Ausdruck, dessen Ergebnisstring mit dem in Zeile n definierten Format ausgegeben werden soll.

WIRKUNG: entspricht dem Befehl DISP. (siehe dort)

BEMERKUNG: Wie DISP kann auch DISP USING in dieser Form für das alphanumerische Arbeiten mit dem Bildschirm verwendet werden.

Der Aufbau der Formatzeile ist unter "IMAGE statement" beschrieben.

Es ist möglich, die vier Steuerzeichen ganz oder teilweise fest in die Formatzeile zu übernehmen.

FORMAT 2: DISP USING {Zeilennr. / Stringvar.}, {num. Ausdr. / Stringausdr.} [, {num. Ausdr. / Stringausdr.}]...

"Zeilennummer" bezieht sich auf die Programmzeile mit der Formatspezifikation (siehe Anweisung IMAGE).

"Stringvar." enthält eine Formatspezifikation.

WIRKUNG: Die Ergebnisse der Ausdrücke werden der Reihe nach in dem durch die Formatanweisung spezifizierten oder durch die Stringvariable festgelegten Format auf dem Display ausgegeben.

Jede auf dem Display dargestellte Größe wird von links nach rechts, gemäß dem jeweiligen Abbildungszeichen des Formatfeldes, ausgegeben.

BEMERKUNGEN:

- Jede Display-Ausgabe durch die Anweisung DISP USING erfolgt ab der ersten Stelle, auch wenn eine vorhergehende DISP-Anweisung mit "," oder ";" endete. Es dürfen nicht mehr Größen ausgegeben werden, als durch die Formatspezifikation angegeben wird. Sind es weniger, so haben die restlichen Formatelemente keine Wirkung.
- Die Ausgabeelemente der DISP USING-Anweisung müssen dem spezifizierten Format entsprechen.

BEISPIELE: Beispiel 1

PRINT ALL-Taste aktiviert

```
FILE     DISPU1


0010 REM *BEISPIEL 'DISPUSING'
0020 REM
0030 DCL 30A$
0040 LET A$="I=####.###  J=####.###"
0050 DISP "I,J";
0060 INPUT I,J
0070 DISP USING A$,I,J
0080 DELAY 20
0090 GOTO 50
0100 END

END OF LISTING



RUN
I,J?
777,999
I= 777.000  J= 999.000
I,J?
4.158,59,75
TOO MUCH INPUT-EXCESS IGNORED
I=   4.158  J=  59.000
I,J?
3.14,19.2333333333
I=   3.140  J=  19.233
I,J?
14.2,999999.33
I=  14.200  J=********
I,J?
```

Beispiel 2

PRINT ALL-Taste aktiviert

```
FILE     DISPU2


0010 REM BEISPIEL 'DISP USING'
0020 REM
0030 DISP "IHR NAME";
0040 RKB A$
0050 DISP USING 80,A$
0060 DELAY 50
0070 GOTO 30
0080 :******** 'CCCCCCCCCCCC ********
0090 END

END OF LISTING



RUN
IHR NAME?
PETER WALKER
******** PETER WALKER  ********
IHR NAME?
```

END

ANWEISUNG: END

FUNKTION: Die Anweisung gibt das physische Ende eines Programmes an.

FORMAT: END

WIRKUNG: Die Programmausführung wird beendet. Die Werte von Variablen sind nicht mehr definiert und externe Files werden geschlossen.

BEMERKUNGEN:
- Die END-Anweisung muß in jedem Programm genau einmal vorkommen.
- Die END-Anweisung muß in der Zeile mit der höchsten Zeilennummer stehen.
- Bei der Programmerstellung mit automatischer Zeilennumerierung ist nach Eingabe von END die Numerierung mit SHIFT und CLEAR zu unterbrechen.

BEISPIEL:

```
0500 REM ***  DIE LETZTE ANWEISUNG     ***
0510 REM ***  IN EINEM BASIC-PROGRAMM  ***
0520 REM ***     IST IMMER:             ***
0530 REM
0540 REM              END
0550 REM
0560 REM
```

ERASE

ANWEISUNG: ERASE

FUNKTION: Löschen des Bildschirmes.

FORMAT: ERASE

WIRKUNG: Der Bildschirm wird von Zeile 3 bis Zeile 40 gelöscht. Der Inhalt des Display- und des Tastaturbuffers bleibt unberührt.

FILES

ANWEISUNG: FILES

FUNKTION: Festlegung der Anzahl und gegebenenfalls der Namen der Files auf Disks oder Disketten, auf die ein Programm zugreift.

FORMAT: FILES $\left\{\begin{matrix}\text{filename}\\ *\end{matrix}\right\} \left[;\left\{\begin{matrix}\text{filename}\\ *\end{matrix}\right\}\right] \ldots$

WIRKUNG: Alle in der Anweisung mit Namen angegebenen Files werden geöffnet und sequentielle sowie Textfiles hierbei in den Lese-Modus versetzt.

Jedem Filenamen wird eine Zahl (Designator) zugewiesen, die der Reihenfolge in der Liste entspricht. Der erste Filename erhält die Zahl 1, der zweite die Zahl 2 usw. Wird anstelle eines Filenamens "*" gesetzt, werden Platz und Filedesignator für ein File reserviert, das in einer nachfolgenden FILE:-Anweisung geöffnet wird.

BEMERKUNGEN:
- Bei jeder sich auf ein File beziehenden Operation muß als erster Operand der Filedesignator angegeben werden.
- Die Anweisung FILES darf in einem Programm nur einmal vorkommen, doch kann sie an jeder beliebigen Stelle stehen.
- Jedes File kann mit einer FILE:-Anweisung durch ein anderes File ersetzt werden (siehe Anweisung FILE:).
- Ein angeführtes File muß in einer offenen Bibliothek enthalten sein.

BEISPIEL:

```
FILE     FILES


0010 DCL 20A$
0020 REM *** BEISPIEL FUER 'FILES' UND 'FILE:'
0030 REM
0040 REM *** OEFFNEN VON 5 FILES ***
0050 REM
0060 FILES SDAT;TDAT;UDAT;VDAT;WDAT
0070 REM
0080 REM *** OEFFNEN FUER SCHREIBEN ***
0090 REM
0100 FOR I=1 TO 5 STEP 1
0110 SCRATCH :I
0120 READ A$(I)
0130 REM *** BESCHREIBEN DER FILES ***
0140 REM
0150 WRITE :I,"DAS IST FILE "+A$(I)
0160 REM
0170 REM *** SCHLIESSEN DER FILES ***
0180 FILE : I,*
0190 NEXT I
0200 DATA SDAT,TDAT,VDAT,UDAT,WDAT
0210 DISP "WELCHES FILE";
0220 INPUT I
0230 REM *** OEFFNEN DES ANGEGEBENEN FILES ***
0240 REM
0250 FILE : I,A$(I)
0260 REM
0270 REM *** LESEN VOM FILE ***
0280 REM
0290 READ :I,A$
0300 PRINT "FILE";I;": ";A$
0310 GOTO 210
0320 END

END OF LISTING



FILE 3 : DAS IST FILE UDAT
FILE 5 : DAS IST FILE WDAT
FILE 1 : DAS IST FILE SDAT
FILE 4 : DAS IST FILE VDAT
FILE 2 : DAS IST FILE TDAT
```

FILE:

ANWEISUNG: FILE:

FUNKTION: Zugriff auf ein File, dessen Name nicht in einer FILES-Anweisung spezifiziert ist und Schließen von Files vor Programmende.

FORMAT: FILE: arithm. Ausdruck, {Stringausdruck / *}

Der arithmetische Ausdruck wird berechnet und das Ergebnis gerundet. Die so erhaltene ganze Zahl n ist der Filedesignator des Filenamens in der Anweisung.

WIRKUNG: Der Ergebnisstring des Stringausdruckes muß ein Filename sein. Das so durch seinen Namen angegebene File ersetzt das File, das bisher durch den Filedesignator n bezeichnet wurde.

Das letztere File wird geschlossen und an seiner Stelle das File mit dem angegebenen Filenamen unter dem gleichen Filedesignator geöffnet.

Wird anstelle des Filenamens "*" angegeben, so wird das File mit dem Filedesignator n geschlossen, ohne daß ein anderes File geöffnet wird.

BEMERKUNGEN:
- Der Filename in der Anweisung muß sich von allen Namen der bereits geöffneten Files unterscheiden.
- Der Filedesignator n muß größer sein als Ø und darf höchstens gleich der Anzahl Files in der FILES-Anweisung sein.

BEISPIEL: Siehe Anweisung FILES.

FKEY#

ANWEISUNG: FKEY # (funktion key)

FUNKTION: Zuweisung eines Inhaltes an eine Funktionstaste.

FORMAT: FKEY # n, Zeichenfolge [:]

"n" ganze Zahl zwischen 1 und 16.

WIRKUNG: Einer durch den Wert des Identifikators n bestimmten Taste wird eine Zeichenfolge zugewiesen.

BEMERKUNGEN:

- Die 'Zeichenfolge' ist eine beliebige Folge von druckbaren Zeichen. Sie muß nicht durch Anführungszeichen begrenzt sein, wie das bei einer Stringkonstanten der Fall ist.
- Durch jedes Drücken der Funktionstaste wird die der Taste zugewiesene Zeichenfolge in den Tastaturbuffer übertragen. Folgt der Zeichenfolge ein Doppelpunkt, wird der Inhalt des Tastaturbuffers direkt (d.h. ohne EOL) ans System übermittelt. Den 16 Funktionstasten können insgesamt maximal 238 Zeichen zugewiesen werden.
- Der Inhalt der Funktionstaste bleibt erhalten, bis er durch eine neue Anweisung FKEY mit gleichem n überschrieben, oder durch die Anweisung LDKEYS bzw. durch eine Neuinitialisierung des Systems wieder durch den Standardinhalt ersetzt wird. Durch Eingabe von STKEYS wird der so definierte Inhalt zum Standardinhalt.
- Der Inhalt einer Funktionstaste entspricht dann einer gültigen Eingabe, wenn auch dessen zeichenweises Eintasten über die Tastatur eine gültige Eingabe bilden würde.
- Die Funktionstastenbelegung kann nicht nur über das Programm, sondern auch im Calculator- und Debugging-Mode erfolgen.

BEISPIEL:

```
FILE     FKEY


0010 DCL 80A$
0020 FKEY #1,DIES
0030 FKEY #2,IST
0040 FKEY #3,EIN
0050 FKEY #4,TEST
0060 FKEY #5,BEISPIEL
0070 FKEY #6,1.11111:
0080 FKEY #7,2.22222:
0090 FKEY #8,ENDE:
0100 DISP "EINGABE STRING  (F8=ENDE)";
0110 RKB A$
0120 PRINT "STRING: ";A$
0130 IF A$="ENDE" THEN 190
0140 DISP "EINGABE NUMERISCHER WERT";
0150 INPUT A
0160 PRINT "NUMERISCHER WERT";A
0170 PRINT
0180 GOTO 100
0190 FKEY #1,RUN FKEY:
0200 PRINT
0210 PRINT
0220 PRINT "*** DRUECKEN SIE DIE TASTE F1 ZUM ERNEUTEN START DES PROGRAMMES ***"
0230 PRINT
0240 PRINT
0250 END

END OF LISTING



STRING: DIES IST EIN TESTBEISPIEL
NUMERISCHER WERT 2.22222

STRING: EIN BEISPIEL IST DIES
NUMERISCHER WERT 1.11111

STRING: ENDE


*** DRUECKEN SIE DIE TASTE F1 ZUM ERNEUTEN START DES PROGRAMMES ***


STRING: TEST
NUMERISCHER WERT 2.22222

STRING: ENDE


*** DRUECKEN SIE DIE TASTE F1 ZUM ERNEUTEN START DES PROGRAMMES ***
```

FNEND

ANWEISUNG: FNEND (function end)

FUNKTION: Kennzeichnung des Endes einer mehrzeiligen Funktionsdefinition.

FORMAT: FNEND

WIRKUNG: siehe DEF/FNEND.

BEMERKUNG: Jede mehrzeilige Funktionsdefinition muß mit der Anweisung FNEND enden.

FOR

ANWEISUNG: FOR

FUNKTION: Kennzeichnung des Beginnes einer Schleife.

FORMAT: FOR Laufvar.=num.Ausdr. TO num.Ausdr. [STEP num.Ausdr.]

```
      .
      .
      .
      beliebige BASIC-Instruktionen
      .
      .
      .
NEXT Laufvariable
```

"Laufvariable" einfache numerische Variable. Die arithmetischen Ausdrücke in der Laufanweisung geben den Anfangswert der Laufvariablen, deren Endwert und die Schrittweite an, um die sich die Laufvariable nach jedem Durchlauf der Schleife erhöht.

WIRKUNG: Die Folge von Anweisungen zwischen FOR und NEXT wird solange ausgeführt, bis der Wert der Laufvariablen den angegebenen Endwert übersteigt. Ist der Anfangswert der Laufvariablen größer (kleiner bei Angabe einer negativen Schrittweite) als der Endwert, so wird die Schleife nicht ausgeführt. Der Wert der Laufvariablen bleibt unverändert und das Programm wird mit dem ersten Befehl nach NEXT fortgesetzt. Ist der Anfangswert kleiner (bei negativer Schrittweite größer) als der Endwert, so wird die Schleife durchlaufen. Bei jeder NEXT-Anweisung wird der Wert der Laufvariablen um die Schrittweite erhöht. Ist der neue Wert der Laufvariablen kleiner oder gleich (bei negativer Schrittweite größer oder gleich) als der Endwert, so wird die Schleife von neuem durchlaufen, und zwar solange, als der Wert der Laufvariablen den Endwert nicht überschreitet (bzw. unterschreitet bei negativer Schrittweite). Nach Beendigung der Schleife fährt das Programm mit dem auf NEXT folgenden Programmschritt fort.

BEMERKUNGEN:

- Fehlt die Angabe der Schrittweite, wird diese impliziert als 1 angenommen. Zwei oder mehrere Schleifen können geschachtelt werden. Sie dürfen sich jedoch nicht überschneiden.

Richtig:	Falsch:
FOR A = 1 TO 10	FOR A = 1 TO 10
FOR B = 1 TO 5	FOR B = 1 TO 5
NEXT B	NEXT A
NEXT A	NEXT B

- Durch Schrittweite Null wird eine Endlos-Schleife gebildet, wenn der Wert der Laufvariablen in der Schleife nicht verändert wird.
- Eine FOR/NEXT-Schleife kann durch Sprunganweisungen (GO TO, ON...GO TO, IF...THEN) vorzeitig beendet werden, wenn das Sprungziel außerhalb der Schleife liegt.
- Bei einem Sprung aus der Schleife behält die Laufvariable ihren letzten Wert bei. Von außerhalb kann jedoch nicht in die Schleife gesprungen werden.
- Folgende Elemente können Bestandteil einer Schleife sein:

  GOSUB
  ON ... GOSUB

- Jeder Anweisung FOR muß ein NEXT entsprechen. Sind mehrere Schleifen geschachtelt, müssen sie verschiedene Laufvariablen aufweisen.
- Der Wert und die Schrittweite einer Laufvariablen dürfen innerhalb der Schleife verändert werden, nicht aber der Endwert.
- Bis zu 15 Schleifen können geschachtelt werden.

BEISPIEL:

```
FILE     FOR


0010 REM BEISPIEL 'FOR/NEXT'
0020 REM
0030 DISP "1. SCHLEIFE: ANFANGSWERT,ENDWERT,SCHRITTWEITE";
0040 INPUT A1,A2,A3
0050 DISP "2. SCHLEIFE: ANFANGSWERT,ENDWERT,SCHRITTWEITE";
0060 INPUT B1,B2,B3
0070 PRINT ,"ANFANGSWERT","ENDWERT","SCHRITTWEITE"
0080 PRINT
0090 PRINT "I-SCHLEIFE",A1,A2,A3
0100 PRINT "J-SCHLEIFE",B1,B2,B3
0110 PRINT
0120 FOR I=A1 TO A2 STEP A3
0130 PRINT "I-SCHLEIFE: I=";I,"J-SCHLEIFE: ";
0140 FOR J=B1 TO B2 STEP B3
0150 PRINT "J=";J;
0160 NEXT J
0170 PRINT
0180 NEXT I
0190 PRINT
0200 PRINT "I=";I;TAB(15);"(LETZTER WERT(";I-A3;") + SCHRITTWEITE(";A3;"))"
0210 PRINT "J=";J;TAB(15);"(LETZTER WERT(";J-B3;") + SCHRITTWEITE(";B3;"))"
0220 PRINT
0230 PRINT
0240 GOTO 30
0250 END

END OF LISTING


                  ANFANGSWERT      ENDWERT          SCHRITTWEITE

I-SCHLEIFE         1                3                1
J-SCHLEIFE         1                1.5              .2

I-SCHLEIFE: I= 1                   J-SCHLEIFE: J= 1 J= 1.2 J= 1.4
I-SCHLEIFE: I= 2                   J-SCHLEIFE: J= 1 J= 1.2 J= 1.4
I-SCHLEIFE: I= 3                   J-SCHLEIFE: J= 1 J= 1.2 J= 1.4

I= 4            (LETZTER WERT( 3 ) + SCHRITTWEITE( 1 ))
J= 1.6          (LETZTER WERT( 1.4 ) + SCHRITTWEITE( .2 ))


                  ANFANGSWERT      ENDWERT          SCHRITTWEITE

I-SCHLEIFE        -1.2             -1.9             -.3
J-SCHLEIFE         1                3                2

I-SCHLEIFE: I=-1.2                 J-SCHLEIFE: J= 1 J= 3
I-SCHLEIFE: I=-1.5                 J-SCHLEIFE: J= 1 J= 3
I-SCHLEIFE: I=-1.8                 J-SCHLEIFE: J= 1 J= 3

I=-2.1          (LETZTER WERT(-1.8 ) + SCHRITTWEITE(-.3 ))
J= 5            (LETZTER WERT( 3 ) + SCHRITTWEITE( 2 ))
```

GOSUB

ANWEISUNG: GOSUB

FUNKTION: Bewirkt den Sprung zu einer bestimmten Anweisung, bei der ein Unterprogramm beginnt.

FORMAT: GOSUB Zeilennummer

WIRKUNG: Das Programm fährt bei der durch die Zeilennummer definierten Anweisung fort. Die zuletzt ausgeführte Anweisung des Unterprogrammes muß RETURN sein, damit das Programm in die Zeile nach GOSUB (Zeile mit der nächsthöheren Zeilennummer) zurückspringt.

BEMERKUNGEN:
- In einem Unterprogramm können auch mehrere RETURN-Anweisungen vorkommen. Ein Unterprogramm kann auch GOSUB-Anweisungen enthalten. Durch RETURN wird jedesmal in die Zeile nach dem letzten GOSUB (Zeile mit der nächsthöheren Zeilennummer) gesprungen. Die Anzahl der möglichen Schachtelungen von Unterprogrammen ist abhängig vom bei der Ausführung des Programmes im Arbeitsspeicher zur Speicherung der Rücksprungadressen freibleibenden Platz.
- Der rekursive Aufruf von Unterprogrammen ist möglich. Ein Unterprogramm darf nur mit RETURN verlassen werden.

BEISPIEL:

```
FILE     GOSUB

0010 REM *BEISPIEL FUER REKURSIVEN UNTERPROGRAMMAUFRUF
0020 REM
0030 LET A=0
0040 PRINT "DAS UNTERPROGRAMM RUFT SICH SELBST AUF:"
0050 PRINT
0060 GOSUB 100
0070 PRINT
0080 PRINT "ENDE DER UNTERPROGRAMMAUFRUFE"
0090 GOTO 170
0100 LET A=A+1
0110 IF A>5 THEN 160
0120 PRINT "Ausfuehrung";A;"des Unterprogrammes"
0130 GOSUB 100
0140 LET A=A-1
0150 PRINT "Ruecksprung von RETURN zu GOSUB";A
0160 RETURN
0170 END

END OF LISTING



DAS UNTERPROGRAMM RUFT SICH SELBST AUF:

Ausfuehrung 1 des Unterprogrammes
Ausfuehrung 2 des Unterprogrammes
Ausfuehrung 3 des Unterprogrammes
Ausfuehrung 4 des Unterprogrammes
Ausfuehrung 5 des Unterprogrammes
Ruecksprung von RETURN zu GOSUB 5
Ruecksprung von RETURN zu GOSUB 4
Ruecksprung von RETURN zu GOSUB 3
Ruecksprung von RETURN zu GOSUB 2
Ruecksprung von RETURN zu GOSUB 1

ENDE DER UNTERPROGRAMMAUFRUFE
```

GOTO

ANWEISUNG: GOTO

FUNKTION: Sprung zu einer bestimmten Zeile des Programmes.

FORMAT: GOTO Zeilennummer

WIRKUNG: Die Verarbeitung des Programmes wird mit der in GOTO bezeichneten Zeile fortgesetzt.

BEMERKUNGEN:
- Sprünge in mehrzeilige Funktionen und FOR/NEXT-Schleifen sind nicht erlaubt.
- Bei GOTO innerhalb einer Schleife muß das Sprungziel immer zwischen FOR und NEXT liegen, falls die Schleife nicht vorzeitig beendet werden soll.

BEISPIEL:

```
FILE     GOTO

0010 REM *DATAROGRAMMBEISPIEL 'GOTO'
0020 REM
0030 REM *BEISPIEL FUER EINE ENDLOSSCHLEIFE
0040 REM
0050 PRINT "Das ist eine Endlosschleife ....................................."
0060 PRINT "Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden"
0070 PRINT
0080 GOTO 50
0090 END

END OF LISTING


Das ist eine Endlosschleife .....................................
Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden

Das ist eine Endlosschleife .....................................
Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden

Das ist eine Endlosschleife .....................................
Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden

Das ist eine Endlosschleife .....................................
Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden

Das ist eine Endlosschleife .....................................
Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden

Das ist eine Endlosschleife .....................................
Das Programm kann nur mit der Taste 'BREAK' abgebrochen werden
```

IF...THEN

ANWEISUNG: IF ... THEN

FUNKTION: Bedingte Verzweigungen in einem Programm.

FORMAT:

$$\text{IF}\left\{\begin{array}{l}\text{Vergleich}\\ \text{(Vergleich 1)}\ \left\{\begin{array}{l}\text{AND}\\ \text{OR}\end{array}\right\}\ \text{(Vergleich 2)}\end{array}\right\}\ \text{THEN Zeilennummer}$$

$$\text{mit: Vergleich} = \left\{\begin{array}{l}\text{num.Ausdr.}\\ \text{Stringausdr.}\end{array}\right\}\ \text{Vergl. Operator}\ \left\{\begin{array}{l}\text{num.Ausdr.}\\ \text{Stringausdr.}\end{array}\right\}$$

WIRKUNG: Die Ausdrücke werden berechnet, die Ergebnisse miteinander verglichen und ihr Wahrheitswert gebildet. Sind zwei Vergleiche mit dem Boole'schen Operator AND oder OR verknüpft, wird der Wahrheitswert der Verknüpfung gebildet.

Liefert der gesamte Ausdruck den Wahrheitswert "wahr", wird zur angegebenen Zeilennummer verzweigt.

Bei einer Verknüpfung mit AND ist der Wahrheitswert des Ausdruckes "wahr", wenn beide Vergleichsbedingungen erfüllt sind.

Bei einer Verknüpfung mit OR ist der Wahrheitswert des gesamten Ausdruckes "wahr", wenn mindestens ein Vergleich den Wahrheitswert "wahr" liefert.

Liefert der Vergleich oder die Verknüpfung von zwei Vergleichen den Wahrheitswert "falsch", wird das Programm mit der nächsten Zeile fortgesetzt.

BEMERKUNG: Befindet sich die Anweisung IF...THEN in einer mehrzeiligen Funktion, muß das Sprungziel der Verzweigung innerhalb der Funktion liegen. Dasselbe gilt auch, wenn die Anweisung in einer Schleife vorkommt und die Schleife nicht aufgrund der Bedingung vorzeitig verlassen werden soll.

ANWEISUNG: : (image)

FORMATSPEZIFIKATIONEN

FUNKTION: Diese Spezifikationen bestimmen das Format, in welchem die Ausgabegrößen (Zahlen und Strings) dargestellt werden (siehe PRINT USING, DISP USING und BUILD USING).

FORMAT: : { Text / imagefield } [[{ Text / imagefield }]] .....

":" Kennzeichen für Formatspezifikation.

WIRKUNG: Beziehen sich die Operanden der Anweisung PRINT USING, MAT PRINT USING, DISP USING oder BUILD USING auf eine Zeile mit einer Formatspezifikation, werden die Ausgabegrößen für den Drucker, das Display bzw. die Stringvariable dem Format entsprechend generiert. Die Ausgabe der Werte erfolgt gemäß der Beschreibung im Abschnitt "Ausgabe von Werten bei PRINT USING, DISP USING, BUILD USING" (siehe Seite 9.73).

FORMATFELDER (IMAGE FIELDS)

Die Formatfelder einer Formatspezifikation müssen für numerische Werte numerisch und für Stringausdrücke Stringformatfelder sein.

Die Formatfelder für die Zahlenausgabe können sein:

- Felder für ganzzahlige Größen
- Felder für Dezimalzahlen
- Felder für Zahlen in Exponentialdarstellung
- Felder mit dem Zeichen $ als Symbol vor der Zahl

Formatfeld für ganzzahlige Größen: # [#]...[#]

Es besteht aus einer Folge von für Ziffern stehenden Symbolen.

Größe: Minimum 2, Maximum 20 Zeichen.

Formatfeld für Dezimalzahlen: # #.# ...[#]

Es besteht aus einer Folge von für Dezimalzahlen stehenden Symbolen.

Größe: Minimum 3, Maximum 26 Zeichen inkl. Dezimalpunkt.

Formatfeld für Exponentialdarstellung:

Für die Mantisse gelten die Regeln für Dezimalzahlen, zusätzlich ist für das Exponentenfeld ↑↑↑↑ einzugeben.

Größe: Minimum 7, Maximum 30 Zeichen (3# -Zeichen,↑↑↑↑)

Bemerkung: Das Symbol ↑↑↑↑ ist immer das letzte Zeichen des Formatfeldes.

Formatfeld mit $ als Symbol:

$ [$] ... [$] Formatfeld für Ganzzahlgrößen
Formatfeld für Dezimalgrößen

Es besteht aus einer Folge von $-Zeichen.

Größe: Minimum 2, Maximum 20 Zeichen.

Formatfeld für Stringausdrücke:

'L L ... L
'R R ... R
'C C ... C

bestehen aus einem Hochkomma und eventuell einem oder mehreren Buchstaben L, R oder C.

Größe: Minimum 1, Maximum 74 Zeichen.

BEISPIELE: Beispiel 1

```
FILE


0010 REM *** FORMATFELDER ***
0020 REM
0030 PRINT "Zur Kontrolle der Druckpositionen:"
0040 PRINT
0050PRINT"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789"
0060 PRINT
0070 :     **********.####,          $$$$$$$,        ****.## SFr,     ###.#%
0080 LET A=123.4567890123
0090 LET B=1000
0100 LET C=256.12
0110 LET D=10.5
0120 PRINT USING 70,A,B,C,D
0130 END

END OF LISTING


Zur Kontrolle der Druckpositionen:

12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789

     12345678.9E-05,            $1000,         256.12 SFr,      10.5%
```

Beispiel 2

```
FILE


0010 REM *** FORMATFELDER ***
0020 REM
0030 PRINT "Zur Kontrolle der Druckpositionen:"
0040 PRINT
0050PRINT"123456789012345678901234567890123456789012345678901234567890123456789"
0060 FOR I=1 TO 3 STEP 1
0070 PRINT
0080 :'LLLLLLLLLLLL
0090 :'RRRRRRRRRRRR
0100 :'CCCCCCCCCCCC
0110 :'LLLLLLLLLLLL
0120 :'LLLLLLL  ↑↑↑↑    $$$$   ####      43HJN!"#$%&'**)(<>/JWTI
0130 PRINT I
0140 NEXT I
0150 :'LLLLLLLL    ::::    'CCCCCCCC    ::::    'RRRRRRRR
0160 PRINT USING 80,"OLIVETTI","P6066"
0170 PRINT USING 90,"Olivetti","P6066"
0180 PRINT USING 100,"OLIVETTI","P6066"
0190 PRINT USING 120,"MAERZ",125,350
0200 PRINT USING 150,"HUBERT","MARIA","ROLF","SUSANNA"
0210 PRINT USING 150,"ENDE"
0220PRINT"123456789!123456789!123456789!123456789!123456789!123456789!12345678"
0230 END

END OF LISTING



Zur Kontrolle der Druckpositionen:

123456789012345678901234567890123456789012345678901234567890123456789

 1

 2

 3
OLIVETTI
P6066
      Olivetti
         P6066
   OLIVETTI
    P6066
MAERZ      ↑↑↑↑     $125     350       43HJN!"#$%&'**)(<>/JWTI
HUBERT       ::::       MARIA      ::::          ROLF
SUSANNA      ::::                  ::::
ENDE         ::::                  ::::
123456789!123456789!123456789!123456789!123456789!123456789!12345678
```

Ausgabe von Werten mit den Anweisungen

PRINT USING, MAT PRINT USING, DISP USING, BUILD USING

Die Ausgabe der Daten erfolgt nach folgenden Regeln:

1. Bei numerischen Größen in einem Formatfeld für ganzzahlige Werte entspricht jeder Ziffer ein Symbol # . Für die Ausgabe des Vorzeichens ist zusätzlich zur maximalen Anzahl Stellen ein weiteres Symbol vorzusehen. Die Zahl wird rechtsbündig ins Formatfeld übertragen. Ist die Zahl nicht ganzzahlig, wird der gebrochene Anteil abgeschnitten. Bei einer positiven Zahl steht vor der ersten Ziffer eine Leerstelle, bei einer negativen Zahl ein Minuszeichen. Bei Formatüberschreitung werden anstelle der Ziffern * (Sternchen) ausgegeben.

2. Bei numerischen Größen in einem Formatfeld für Dezimalzahlen entspricht jeder Ziffer ein Symbol # .

   Sind für Bruchteile der Zahl nicht genügend Symbole vorgesehen, wird die Zahl gerundet und der letzte Teil rechts abgeschnitten. Bei positiven Zahlen ist das letzte Zeichen vor der ersten Ziffer eine Leerstelle, bei negativen Zahlen ein Minuszeichen. Sind für den ganzzahligen Teil der Zahl nicht genügend Symbole # vorgesehen, wird für jedes Zeichen des Formatfeldes ein * ausgegeben.

3. Bei numerischen Größen in einem Formatfeld für Zahlen in Exponentialdarstellung entspricht jeder Ziffer ein Symbol # , analog zu 2. Anstelle der Zeichen ↑↑↑↑ wird der Buchstabe E, ein Minus- oder Leerzeichen und 2 darauffolgende Ziffern gesetzt. Diese Ziffern bilden den Exponenten zur Basis 10. Das Format muß einen Dezimalpunkt enthalten.

4. Der numerische Wert wird auf ein Formatfeld mit einem $-Zeichen als Symbol abgebildet, wobei nur ein einziges $-Zeichen links im Formatfeld stehen muß.

BEMERKUNGEN:

- In allen Formatfeldern für die Ausgabe numerischer Werte muß eine Stelle für das Vorzeichen vorgesehen sein. Bei der Ausgabe einer positiven Zahl wird die erste Stelle des Formatfeldes (# oder $) durch ein Leerzeichen ersetzt, bei der Ausgabe einer negativen Zahl wird die erste Stelle des Formatfeldes für die Ausgabe des Minus-Zeichens verwendet.
- Zahlen, deren Absolutbetrag kleiner ist als 1, werden mit 0 vor dem Dezimalpunkt dargestellt. Ein Formatfeld für Dezimalzahlen muß daher vor dem Dezimalpunkt zwei #-Zeichen aufweisen.

INPUT

ANWEISUNG: INPUT

FUNKTION: Zuweisung von Werten an Variable während des Programmablaufes.

FORMAT: INPUT $\left\{\begin{matrix}\text{num.Var.}\\\text{Stringvar.}\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{num.Var.}\\\text{Stringvar.}\end{matrix}\right\}.....\right]$

WIRKUNG: Der Programmablauf wird angehalten und in der Display-Zeile erscheint ein Fragezeichen. Der Operator kann nun die Werte, getrennt durch Kommas, eingeben. Sie werden den Variablen der INPUT-Anweisung der Reihe nach zugewiesen. Die Zuweisung erfolgt erst, wenn der Programmablauf nach dem Eintasten aller Werte durch Drücken der Taste EOL fortgesetzt wird.

BEMERKUNG: Zwei Fragezeichen (??) in der Display-Zeile geben an, daß noch nicht allen Variablen Werte zugewiesen wurden. Das System wartet auf die Eingabe der restlichen Daten. Die Eingabewerte müssen mit dem Typ der entsprechenden Variablen übereinstimmen. Die Anzahl der eingegebenen Werte darf nicht größer sein als die Anzahl vorhandener Variablen. Ein String muß in Anführungszeichen (") gesetzt werden, falls er folgende Zeichen enthält:

- Komma
- Leerzeichen am Anfang oder Ende
- Eingabe des Nullstrings

Die Zuweisung der Werte kann entweder durch Eingabe über die Tastatur oder durch Ausführung von Prozeduren mit automatischer Input-Übernahme erfolgen (siehe Abschnitt 6.4).

MELDUNGEN: ?

Das Fragezeichen gibt an, daß eine Eingabe verlangt ist. Es kann aber auch Bestandteil einer vorangehenden Display-Anzeige im Bildschirmzeile 0 sein, falls diese durch das Trennzeichen ";" abgeschlossen wurde.

??

Zwei Fragezeichen zeigen an, daß noch nicht allen Variablen der Variablenliste der Anweisung INPUT ein Wert zugewiesen wurde. Die Eingabe ist fortzusetzen, bis alle Variablen einen Wert aufweisen.

"TOO MUCH INPUT-EXCESS IGNORED"

Diese Meldung gibt an, daß die Anzahl eingegebener Werte die Anzahl Elemente der Variablenliste übersteigt. Die überzähligen Eingaben werden ignoriert.

"INCORRECT FORMAT-RETYPE LINE"

Diese Meldung erscheint, wenn numerischen Variablen alphanumerische Daten zugewiesen werden. Zahlen, die alphanumerischen Variablen übergeben werden, werden als String interpretiert und ohne Fehlermeldung übernommen.

BEISPIELE: Beispiel 1

```
LIST
FILE


0010 DISP "ZAHL,STRING,ZAHL";
0020 INPUT A,S$,B
0030 PRINT S$,A,B
0040 GOTO 10
0050 END

END OF LISTING



RUN
ZAHL,STRING,ZAHL?
-.254838,XXXXXX,12E-32
XXXXXX          -.254838          1.2000000E-31
ZAHL,STRING,ZAHL?
XXXXX,3.33,YYYYY
INCORRECT FORMAT-RETYPE LINE
3.33,XXXXX,-5.5555
XXXXX             3.33            -5.5555
ZAHL,STRING,ZAHL?
1,2,3
2                 1                 3
ZAHL,STRING,ZAHL?
1,"XX,XX,XX,XX",2,3,4
TOO MUCH INPUT-EXCESS IGNORED
XX,XX,XX,XX       1                 2
ZAHL,STRING,ZAHL?
```

Beispiel 2

```
LIST
FILE


0010 LET K=I=9
0020 INPUT I,B(I)
0030 PRINT "Altes I=";K;"  ,Neues I=";I;"   ,B(";K;") =";B(K)
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
?
2.18.6
Altes I= 9   ,Neues I= 2    ,B( 9 ) = 18.6
```

INTERRUPT ENABLE

ANWEISUNG: INTERRUPT ENABLE

FUNKTION: Behandlung von internen und externen Interrupts durch das Programm.

FORMAT:
1. INTERRUPT ENABLE (I, fkt.name)
2. INTERRUPT ENABLE (E, fkt.name,intmsk[,Priorität])

"I"	interner Interrupt
"E"	externer Interrupt
"fkt.name"	Stringausdruck, der entweder den Namen einer mehrzeiligen Funktion oder das Zeichen "*" enthält.
"intmsk, Priorität"	siehe "Handbuch für Peripherie und Datenfernübertragung".

1. INTERNER INTERRUPT

WIRKUNG: Die folgenden Ausführungen beziehen sich nur auf den internen Interrupt (erstes Format).

Die Anweisung INTERRUPT ENABLE erlaubt die Behandlung von während des Programmablaufes auftretenden Fehlern. Enthält der Parameter "fkt.name" den Namen einer mehrzeiligen Funktion, wird diese bei Auftreten eines Fehlers aufgerufen. Enthält der Parameter "fkt.name" das Zeichen "*", wird der Fehler vom System behandelt, das heißt, die Interrupt-Bedingung ist aufgehoben.

BEMERKUNGEN:
- Die durch die INTERRUPT-Anweisung definierte Fehlerbehandlung bleibt für alle nachfolgenden Zeilen gültig, bis eine neue INTERRUPT-Anweisung auftritt oder das Programm beendet ist.
- Die DEF-Anweisung der Funktion muß als Parameter zwei einfache numerische Variable enthalten. Der ersten Variablen wird die Nummer des aufgetretenen Fehlers übergeben. Die zweite Variable enthält die Nummer der Zeile, in der der Interrupt (Fehler) aufgetreten ist. Aufgrund dieses Wertes kann der Anwender definieren, wie der Fehler behandelt werden soll.

- Die mehrzeilige Funktion muß der Pseudovariablen FN* einen Wert zuweisen. Bei FN* = 0 ist der Fehler durch die definierte Funktion zu behandeln. Bei FN* ≠ 0 wird der Fehler nach Verlassen der Funktion vom System behandelt. In diesem Falle wird die Programmausführung unterbrochen und die Meldung "ERROR Fehlernummer IN LINE Zeilennummer" ausgegeben. Ist der Fehler behebbar, kann die Programmverarbeitung fortgesetzt werden.

- Liefert die Funktion als Ergebnis FN* = 0 und ist der Fehler behebbar, wird die Programmausführung mit der Zeile fortgesetzt, die der den Interrupt hervorrufenden Anweisung folgt.

- Liefert die Funktion als Ergebnis FN* = 0 und ist der Fehler nicht behebbar, wird die den Interrupt hervorrufende Anweisung wiederholt und das Programm fortgesetzt. Kann die Fehlerursache in der Funktion nicht behoben werden, entsteht eine Endlosschleife.

- Die Anweisung INTERRUPT ENABLE wird bei folgenden, nicht behebbaren Fehlern ignoriert:

  ERROR 65 — Überschreitung der Kapazität des Anwenderspeichers während der Programmausführung.

  ERROR 74 — Überschreitung der maximal zulässigen Anzahl Aufrufen (255) anderer oder derselben Funktion innerhalb einer Funktion.

- Tritt ein Fehler während der Ausführung der durch den Interrupt aufgerufenen Funktion auf, ist keine Interruptbehandlung möglich; es erscheint die Fehlermeldung 282.

- Es ist zu beachten, daß sich die Definition "behebbarer" und "nicht behebbarer" Fehler auf die Fehlerbehandlung ohne Einsatz von INTERRUPT bezieht. Als "behebbar" werden die Fehler bezeichnet, für die das System eine Standardbehandlung vorsieht (z.B. Zuweisung von Null bei nicht initialisierten numerischen Variablen). Durch Verwendung der INTERRUPT-Anweisung wird es jedoch möglich, die Ursache von "nicht behebbaren" Fehlern zu beseitigen, ohne daß das Programm abgebrochen wird.

BEISPIELE: Beispiel 1

Das folgende, einfache Beispiel (Programm INTER) illustriert die Fehlerbehandlung mittels INTERRUPT bei manueller Eingabe der Namen von Datenfiles. Dadurch ist es möglich, einen Programmabbruch aufgrund der Eingabe eines falschen Filenamens zu vermeiden.

Erläuterungen zum Beispiel:

Zwischen den Zeilen 20 und 110 erfolgende Interrupts bedingen den Aufruf der Funktion FNA. Da FN* = 0, erscheinen keine Fehlermeldungen des Systems.

Da FILE2 nicht existiert bzw. kein Randomfile ist, verursacht die Anweisung in Zeile 50 einen Interrupt und damit den Aufruf der Funktion FNA. Dort wird mittels Parameter x der Fehler 187 bzw. 207 festgestellt und zwecks erneuter Eingabe des Filenamens in Zeile 1070 gesprungen. Da FN* = 0 und die Fehler 187 bzw. 207 nicht behebbar sind, wird die Anweisung von Zeile 50 nach dem Verlassen der Funktion erneut ausgeführt. Durch das falsche Format der Eingabe (7WERT) erfolgt wiederum ein Interrupt und die Funktion wird erneut aufgerufen. Erst bei der Eingabe des Filenamens DATEN wird das Programm ohne Interrupt fortgesetzt.

```
LIST
FILE      INTER


0010 FILES *
0015 DCL 20B$
0020 INTERRUPT ENABLE (I,"A")
0030 DISP "NAME DATEN-FILE";
0040 RKB A$
0050 FILE : 1,A$
0055 PRINT
0060 PRINT "Inhalt der Datei ";A$;":"
0070 PRINT
0075 FOR I=1 TO 4 STEP 1
0080 READ :1,B$
0090 PRINT B$
0100 NEXT I
0110 GOTO 1120
1000 REM*******************************
1010 DEF FNA(X,Y)
1020 IF (X=187) OR (X=207) THEN 1050
1030 PRINT "Fehler";X;",Dateiname hat unzulaessiges Format"
1040 GOTO 1070
1050 PRINT "Datei ";A$;" existiert nicht oder ist kein Random-File"
1070 DISP "NAME DATEN-FILE";
1080 RKB A$
1090 LET FN*=0
1110 FNEND
1120 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
NAME DATEN-FILE?
FILE2
Datei FILE2 existiert nicht oder ist kein Random-File
NAME DATEN-FILE?
7WERT
Fehler 107 ,Dateiname hat unzulaessiges Format
NAME DATEN-FILE?
DATEN

Inhalt der Datei DATEN:

BEISPIEL
FUER DEN GEBRAUCH
DER INTERRUPT-
ANWEISUNG
```

Beispiel 2

Das Programm OVERFL zeigt die Verwendung der INTERRUPT-Anweisung bei behebbaren Fehlern am Beispiel eines Overflows (Division durch Null).

Im ersten Programmlauf ist für den Parameter "fkt.name" in der INTERRUPT-Anweisung das Zeichen "*" gesetzt. Das bedeutet, dass die normale Fehlerbehandlung vom System übernommen wird.

Im zweiten Programmlauf wird für "fkt.name" der Name B angegeben. Das heisst, dass ein eventueller Interrupt von der mehrzeiligen Funktion behandelt wird. Wie das Beispiel zeigt, ist im ersten Fall die Intervention des Operators notwendig. Im zweiten Fall wird das Programm automatisch bis zum Ende durchlaufen (wichtig für Nachtverarbeitungen).

```
LIST
FILE      OVERFL


0010 FILES DAT
0020 LET Z=5
0030 DISP "INTERRUPT JA=1, NEIN=0";
0040 INPUT R
0050 LET Z$="*"
0060 IF R=0 THEN 80
0070 LET Z$="B"
0080 INTERRUPT ENABLE (I,Z$)
0110 PRINT
0120 PRINT "POSITION         X-WERT                Y=5/X"
0130 PRINT
0135 PRINT
0140 FOR I=1 TO 10 STEP 1
0150 LET S=0
0160 READ :1,X
0170 LET Y=Z/X
0180 IF S=1 THEN 200
0190 PRINT USING 292,I,X,Y
0200 LET S=1
0210 NEXT I
0220 PRINT
0222 PRINT
0225 PRINT "----------------------------------------"
0240 DEF FNB(A,B)
0250 IF A<>3 THEN 280
0260 LET S=1
0270 PRINT USING 295,I,X
0280 LET FN*=0
0290 FNEND
0292 :###          #####.###            #####.###
0295 :###          #####.###          O V E R F L O W
0300 END

END OF LISTING



RUN
INTERRUPT JA=1, NEIN=0?
0

POSITION         X-WERT                Y=5/X


  1              2.360                 2.119
  2              4.800                 1.042
  3              6.890                 0.726
  4              7.300                 0.685
  5              0.250                20.000
  6              1.600                 3.125
  7              0.600                 8.333
ERROR 3   IN LINE 170
RUN
INTERRUPT JA=1, NEIN=0?
1

POSITION         X-WERT                Y=5/X


  1              2.360                 2.119
  2              4.800                 1.042
  3              6.890                 0.726
  4              7.300                 0.685
  5              0.250                20.000
  6              1.600                 3.125
  7              0.600                 8.333
  8              0.000           O V E R F L O W
  9              4.500                 1.111
 10              8.310                 0.602


----------------------------------------
```

2. EXTERNER INTERRUPT

Die vollständigen Möglichkeiten dieser Anweisung werden im Handbuch "Programmierung von Peripherie und Datenfernübertragung" dargestellt. In diesem Abschnitt wird nur die Verwendung des externen Interrupts beim Tastaturkanal dargestellt.

Die Tastatur des Systems kann als periphere Einheit programmiert werden (siehe Kapitel 9.4). Ein externer Interrupt liegt dann vor, wenn die Taste END OF LINE gedrückt wird.

FORMAT: INTERRUPT ENABLE (E,funname,"0010000000000000",1)

"funname" Buchstabe, der den Namen der aufzurufenden Funktion festlegt (als Stringkonstante oder -variable).

WIRKUNG: Ist der Tastaturkanal aktiviert und wird die Taste END OF LINE gedrückt, wird die durch "funname" spezifizierte, mehrzeilige Funktion aufgerufen und ausgeführt.

Nach Ausführung dieser Funktion wird das Programm mit der Zeile fortgesetzt, die ohne Auftreten des Interrupts als nächste verarbeitet worden wäre.

BEMERKUNGEN:

- Durch die Programmierung eines externen Interrupts kann die Eingabe über die Tastatur parallel zu anderen Verarbeitungen erfolgen.
- Die mehrzeilige Funktion muß mit einem formalen Parameter definiert werden. Ihm wird als aktueller Parameter beim Auftreten des Interrupts die Zeilennummer der Zeile übergeben, bei der der Interrupt aufgetreten ist. Der an FN* zugewiesene, numerische Wert ist nicht relevant.
- Die Anweisung INTERRUPT ENABLE für den Tastaturkanal kann parallel zu Anweisungen für die Behandlung interner oder externer Interrupts für andere Kanäle aktiviert sein.
- Der letzte Parameter "Priorität" gibt die Priorität zu anderen INTERRUPT ENABLE-Anweisungen für andere Peripherien an.

BEISPIEL: Eingabe von Strings und ihre Speicherung in einem sequentiellen Datenfile, parallel zum Plotten der Funktion X.

$(X = -5(0.1)5,\ n=1(1)^{10})$

```
FILE     INTERR

0010 REM * BEISPIEL FUER EXTERNEN INTERRUPT ANHAND DES TASTATURKANALES
0020 REM
0030 REM VEREINBARUNGEN
0040 FILES DATEN
0050 DCL 80A$
0060 BUFFER #32,80
0070 SCRATCH :1
0080 INTERRUPT ENABLE (E,"T","0010000000000000",1)
0090 REM *ERSTMALIGES AKTIVIEREN DES TASTATURKANALS
0100 DISP "EINGABE, ENDE='END'"
0110 RECEIVE #32,A$ AND GO
0120 LET H=0
0130 REM ** BEGINN DES ZEICHENPROGRAMMES
0140 INIMAGE
0150 ERASE
0160 FRAME 8,5.6
0170 SCALE -8,8,-11.2,11.2
0180 XAXIS 0,2,-6,6
0190 YAXIS 0,1,-11,11
0200 FOR I=1 TO 10 STEP 1
0210 FOR N=1 TO 5 STEP 1
0220 MOVE -5,(-5)↑N
0230 FOR X=-5 TO 5 STEP 0.1
0240 PLOT X,X↑N
0250 NEXT X
0260 NEXT N
0270 IF INT(I/2)=I/2 THEN 310
0280 IF H=1 THEN 350
0290 ERASE ON
0300 GOTO 320
0310 ERASE OFF
0320 NEXT I
0330 REM*  WARTEN AUF ABSCHLUSS DER EINGABEN
0340 IF H=0 THEN 340
0350 DISP "DRUCK DATEN=1,HARDCOPY=2,ENDE=0";
0360 INPUT I
0370 ON I+1 GOTO 560,390,430
0380 GOTO 350
0390 RESTORE :1
0400 READ :1,A$ EOF 350
0410 PRINT A$
0420 GOTO 400
0430 DRAW
0440 GOTO 350
0450 REM *INTERRUPTFUNKTION
0460 DEF FNT(L)
0470 WAIT #32
0480 IF A$="END" THEN 530
0490 WRITE :1,A$ EOF 530
0500 DISP "EINGABE, ENDE='END'"
0510 RECEIVE #32,A$ AND GO
0520 GOTO 540
0530 LET H=1
0540 LET FNT=0
0550 FNEND
0560 END

END OF LISTING
```

ANWEISUNG: LET

FUNKTION: Zuweisung der Werte von Ausdrücken an eine oder mehrere Variable eines Programmes.

FORMAT: [LET] n.V.[=n.V.][=...]=n.A.
S.V.[=S.V.][=...]=S.A.

n.V. = num. Variable
S.V. = Stringvariable
n.A. = num. Ausdruck
S.A. = Stringausdruck

WIRKUNG: Der Ausdruck rechts vom letzten Gleichheitszeichen wird berechnet und das Ergebnis den Variablen auf der linken Seite zugewiesen.

BEMERKUNGEN:
- Das Schlüsselwort LET muß nicht eingegeben werden.
- Die aktuelle Länge der Stringvariablen in einer Zuweisung ist gleich der Anzahl Zeichen, die sich als Resultat des Stringausdruckes ergeben.
- Ist die vereinbarte Länge der Stringvariablen kleiner, so erfolgt eine Fehlermeldung. Nach dem Drücken der Taste CONTINUE wird der Ergebnisstring entsprechend der Länge der Variablen abgeschnitten.

BEISPIELE: Beispiel 1

```
0010 REM ***Beispiel 1 fuer  'LET'***
0020 LET A=B=C=D=E=F=G=H=I=K=L=M=A1=X9=Z0=J1=G8=B5=77
0030 PRINT A,B,C,G,
0040 LET A=0
0050 LET A=A+1
0060 PRINT A
0070 IF A<=4 THEN 50
0080 PRINT "A1=",A1
0090 LET A1=PI*(G8-76)+SQR(Z0+23)/SIN(X9*PI/180)
0100 PRINT "A1=",A1
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 77              77              77              77
 2
 3
 4
 5
A1= 77
A1= 13.404634
```

Beispiel 2

```
LIST
FILE


0010 PRINT
0020 PRINT
0030 LET A$="Die Kapitel"
0040 LET B$=" 4 "
0050 LET C$=" 5"
0060 LET D$=" 6"
0070 LET E$=" 7 bis 9 "
0080 DCL 80(F$,G$,H$)
0090 LET F$="beschreiben das System."
0100 LET G$="beschreiben die Sprache."
0110 LET H$="  "+A$+B$+","+C$+"und"+D$+F$
0120 PRINT "H$=";H$
0130 LET H$=A$+E$+G$
0140 PRINT "H$=";H$
0150 LET S$=""
0160 FOR I=1 TO 9 STEP 1
0170 LET S$=S$+CHR$(I)
0180 NEXT I
0190 PRINT "S$=";S$
0200 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****


H$=  Die Kapitel 4 , 5und 6beschreiben das System.
H$=Die Kapitel 7 bis 9 beschreiben die Sprache.
S$=
```

NEXT

ANWEISUNG: NEXT

FUNKTION: Ende einer Schleife.

FORMAT: NEXT Laufvariable

WIRKUNG: Siehe Anweisung FOR/NEXT.

ON ...GOSUB

ANWEISUNG: ON ... GOSUB

FUNKTION: Sprung in ein Unterprogramm. Das Sprungziel ist abhängig vom Wert eines Ausdruckes.

FORMAT: ON arithm. Ausdruck GOSUB Zeilennr.[,Zeilennr.]...

"Zeilennummer" gibt das Sprungziel an.

WIRKUNG: Der arithmetische Ausdruck wird berechnet und das Ergebnis gerundet.

Diese ganze Zahl gibt an, zu welcher der Zeilennummern rechts von GOSUB gesprungen werden soll. Jede Zeilennummer in der Anweisung gibt die erste Zeile des jeweiligen Unterprogrammes an. Die letzte Zeile jedes Unterprogrammes enthält die Anweisung RETURN, die den Rücksprung in die Zeile nach ON ... GOSUB bewirkt.

BEMERKUNGEN:
- In einem Unterprogramm können auch mehrere RETURN-Anweisungen vorkommen.
- Ergibt der arithmetische Ausdruck nach der Rundung eine Zahl, die kleiner als 1 oder grösser als die Anzahl Zeilennummern in der Anweisung ON ... GOSUB ist, wird das Programm mit der nächsten Anweisung nach ON ... GOSUB fortgesetzt.

BEISPIEL:

```
0010 REM**** ON...GOSUB *****
0030 DISP "WAEHLEN SIE EIN UNTERPROGRAMM";
0040 INPUT I
0050 PRINT
0060 PRINT "I=";I
0070 PRINT
0080 ON I GOSUB 140,160,180,200
0090 IF I<.5 THEN 120
0100 IF I<4.5 THEN 30
0110 IF I=9 THEN 220
0120 PRINT "KEIN UNTERPROGRAMM UNTER DIESER NUMMER"
0130 GOTO 30
0140 PRINT "DAS IST DAS ERSTE UNTERPROGRAMM"
0150 RETURN
0160 PRINT "DAS IST DAS ZWEITE UNTERPROGRAMM"
0170 RETURN
0180 PRINT "DAS IST DAS DRITTE UNTERPROGRAMM"
0190 RETURN
0200 PRINT "DAS IST DAS VIERTE UNTERPROGRAMM"
0210 RETURN
0220 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
WAEHLEN SIE EIN UNTERPROGRAMM?
3

I= 3

DAS IST DAS DRITTE UNTERPROGRAMM
WAEHLEN SIE EIN UNTERPROGRAMM?
2.1

I= 2.1

DAS IST DAS ZWEITE UNTERPROGRAMM
WAEHLEN SIE EIN UNTERPROGRAMM?
-12

I=-12

KEIN UNTERPROGRAMM UNTER DIESER NUMMER
WAEHLEN SIE EIN UNTERPROGRAMM?
.499

I= .499

KEIN UNTERPROGRAMM UNTER DIESER NUMMER
WAEHLEN SIE EIN UNTERPROGRAMM?
.501

I= .501

DAS IST DAS ERSTE UNTERPROGRAMM
WAEHLEN SIE EIN UNTERPROGRAMM?
9

I= 9
```

ON ...GOTO

ANWEISUNG: ON ... GOTO

FUNKTION: Sprung in eine bestimmte Programmzeile, in Abhängigkeit vom Wert eines arithmetischen Ausdruckes.

FORMAT: ON arithm. Ausdruck GOTO Zeilennr.[,Zeilennr.]...

"Zeilennummer" gibt das Sprungziel an.

WIRKUNG: Der arithmetische Ausdruck wird berechnet und das Ergebnis auf die nächste ganze Zahl gerundet. Diese Zahl gibt an, zu welcher Zeilennummer rechts von GOTO gesprungen werden soll.

BEMERKUNG: Ergibt der arithmetische Ausdruck nach der Rundung eine Zahl, die kleiner als 1 oder größer als die Anzahl Zeilennummern der Anweisung ON ... GOTO ist, wird kein Sprung ausgeführt, sondern das Programm mit der nächsten Anweisung fortgesetzt.

BEISPIEL:

```
LIST
FILE


0010 REM*** ON....GOTO *****
0020 REM
0030 FOR I=-1 TO 5 STEP 1
0040 ON I GOTO 70,90,110,130
0050 PRINT "KEIN GUELTIGES I"
0060 GOTO 140
0070 PRINT "I=1"
0080 GOTO 140
0090 PRINT "I=2"
0100 GOTO 140
0110 PRINT "I=3"
0120 GOTO 140
0130 PRINT "I=4"
0140 NEXT I
0150 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
KEIN GUELTIGES I
KEIN GUELTIGES I
I=1
I=2
I=3
I=4
KEIN GUELTIGES I
```

ANWEISUNG: PAD

FUNKTION: Auffüllen einer Stringvariablen mit Füllzeichen.

FORMAT: PAD Stringvariable, n

"n" ganze Zahl zwischen 0 und 255 (ISO-Zeichen).

WIRKUNG: Der zwischen aktueller und deklarierter Länge liegende Teil einer Stringvariablen wird mit dem der Zahl n in der ISO-Tabelle entsprechenden Zeichen aufgefüllt.

BEISPIEL: Siehe Anweisung DEPAD.

ANWEISUNG: PRINT

FUNKTION: Druck von Zahlen und Strings im Standardformat.

FORMAT:

PRINT [{Stringausdruck / num.Ausdruck / TAB(num.Ausdr)}[{,/;}{Stringausdruck / num.Ausdruck / TAB(num.Ausdr)}]...[{,/;}]]

"," / ";" Trennzeichen der Ausgabeelemente.

WIRKUNG: Die Ergebnisse der Ausdrücke werden berechnet und im Standardformat von links nach rechts der Reihe nach gedruckt.

Die Position der Zeichen in der Druckzeile kann mit den Anweisungselementen

- TAB (arithmetischer Ausdruck)
- "," Komma
- ";" Strichpunkt

beliebig festgelegt werden (siehe Kapitel 8.5).

PRINT ohne Angabe von Ausgabeelementen bewirkt die Ausgabe einer Leerzeile.

Stellenkontrolle der Zeichen in der Druckzeile

Die PRINT-Anweisung erzeugt eine dem ausgegebenen String entsprechende Folge von Zeichen in einem Buffer.

Ein Pointer weist auf die erste freie Stelle des Buffers. Sobald der Buffer gefüllt ist, wird sein Inhalt als eine Druckzeile ausgegeben und der Pointer auf die erste Stelle des Buffers zurückgesetzt.

Mit den Elementen ",", ";" und TAB (arithmetischer Ausdruck) kann das Ausgabeformat einer Druckzeile festgelegt werden.

BEMERKUNGEN:

- Durch Verwendung des Kommas (,) als Trennzeichen wird der Buffer des Druckers in 5 Druckzonen zu je 16 Stellen aufgeteilt.
- Die Funktion TAB bewirkt die Ausgabe des nächsten Elementes ab einer bestimmten Position.
- Die Darstellung von Daten im Standardformat ist im Kapitel 8.5 beschrieben.

BEISPIELE: Beispiel 1

```
LIST
FILE


0010 PRINT "Es folgen 10 Leerzeilen"
0020 FOR I=1 TO 10 STEP 1
0030 PRINT
0040 NEXT I
0050 PRINT "Das waren 10 Leerzeilen"
0060 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
Es folgen 10 Leerzeilen









Das waren 10 Leerzeilen
```

Beispiel 2

```
LIST
FILE


0010 PRINT "OLIVETTI"
0020 PRINT
0030 PRINT TAB(10);"P6066"
0040 PRINT 1,2,3,4,5
0050 PRINT 1;2;3;4;5
0060 PRINT 1,2,3,4,"DIESER STRING IST ZU LANG"
0070 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
OLIVETTI

          P6066
 1             2             3             4             5
 1  2  3  4  5
 1             2             3             4
DIESER STRING IST ZU LANG
```

Beispiel 3

```
LIST
FILE


0010 DCL 256A$
0020 LET A$=""
0030 FOR I=0 TO 255 STEP 1
0040 LET A$=A$+CHR$(I)
0050 NEXT I
0060 PRINT A$
0070 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
■┌┴┘≀8⁄⌂↖→≡↓≹←00800000≠∧⊣8↑⸮800000 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]↑_`abcdefghijklmnopqrstuvwxyz{|}‾▒$▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
```

PRINT USING

ANWEISUNG: PRINT USING

FUNKTION: Ausdruck von Daten in definiertem Format.

FORMAT: PRINT USING $\left\{\begin{matrix}\text{Zeilennr.}\\\text{Stringvar.}\end{matrix}\right\}$,$\left\{\begin{matrix}\text{num.Ausdr.}\\\text{Stringausdr}\end{matrix}\right\}\left[,\left\{\begin{matrix}\text{num.Ausdr.}\\\text{Stringausdr}\end{matrix}\right\}\right]\ldots$

"Zeilennummer" bezeichnet die Zeile mit der Formatspezifikation (Image-Zeile).

"Stringvar." ist der Name der die Formatspezifikation enthaltenden Stringvariablen.

WIRKUNG: Die Werte der Ausdrücke werden in dem Format gedruckt, welches in der Formatspezifikation aufbereitet oder in der Stringvariablen angegeben ist. Jeder Wert entspricht einem Formatfeld, von links beginnend (siehe IMAGE-Anweisung).

BEMERKUNG: Jede Anweisung PRINT USING bewirkt, dass die Werte in einer neuen Zeile ausgedruckt werden, auch wenn die vorhergehende Anweisung PRINT mit "," oder ";" endet. Ist die zu druckende Anzahl Werte grösser als die im Formatstring vorgesehene, werden die restlichen Werte in der nächsten Zeile im gleichen Format ausgegeben. Sind weniger Ausgabeelemente als Formatelemente vorhanden, werden anstelle der restlichen Formatelemente Leerzeichen gesetzt. Die Ausgabeelemente und Formatfelder müssen in ihrem Typ übereinstimmen.

BEISPIELE: Beispiel 1

```
LIST
FILE     PRIUS1


0010 REM*** PRINT USING ****
0020 REM
0030 REM*** NUMERISCHE FELDER ***
0040 DISP "NUMERISCHER WERT";
0050 INPUT A
0060 PRINT USING 110,A,A,A
0070 PRINT USING 120,A
0080 PRINT USING 130,A,A
0090 PRINT
0100 GOTO 40
0110 :MIT / OHNE NACHKOMMASTELLEN: #####, #####.# UND #####.####
0120 :EXPONENTIALDARSTELLUNG:                ##.##^^^^
0130 :DOLLAR FELDER:                       $$$$$.# UND $####.####
0140 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
NUMERISCHER WERT?
123
MIT / OHNE NACHKOMMASTELLEN:    123,    123.0 UND    123.0000
EXPONENTIALDARSTELLUNG:                  1.23E+02
DOLLAR FELDER:                       $123.0 UND $ 123.0000

NUMERISCHER WERT?
123.456
MIT / OHNE NACHKOMMASTELLEN:    123,    123.5 UND    123.4560
EXPONENTIALDARSTELLUNG:                  1.23E+02
DOLLAR FELDER:                       $123.5 UND $ 123.4560

NUMERISCHER WERT?
-123.456
MIT / OHNE NACHKOMMASTELLEN:   -123,   -123.5 UND   -123.4560
EXPONENTIALDARSTELLUNG:                 -1.23E+02
DOLLAR FELDER:                      $-123.5 UND $-123.4560

NUMERISCHER WERT?
0.5
MIT / OHNE NACHKOMMASTELLEN:      0,      0.5 UND      0.5000
EXPONENTIALDARSTELLUNG:                  5.00E-01
DOLLAR FELDER:                       $0.5 UND $   0.5000

NUMERISCHER WERT?
-0.5
MIT / OHNE NACHKOMMASTELLEN:     -0,     -0.5 UND     -0.5000
EXPONENTIALDARSTELLUNG:                 -5.00E-01
DOLLAR FELDER                       $-0.5 UND $  -0.5000

NUMERISCHER WERT?
```

Beispiel 2

```
LIST
FILE


0010 REM*** PRINT USING ****
0020 REM
0030 REM*** ALPHANUMERISCHE FELDER **
0040 DCL 80A$
0050 DISP "STRING";
0060 RKB A$
0070 PRINT "STRING: ";A$
0080 PRINT
0090 PRINT USING 150,A$
0100 PRINT USING 160,A$
0110 PRINT USING 170,A$
0120 PRINT
0130 PRINT
0140 GOTO 50
0150 !RECHTSBUENDIGE AUSGABE: 'RRRRRRRRRRRRRRRRRRR
0160 !LINKSBUENDIGE AUSGABE:  'LLLLLLLLLLLLLLLLLLL
0170 !ZENTRIERTE AUSGABE:     'CCCCCCCCCCCCCCCCCCC
0180 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
STRING: A

RECHTSBUENDIGE AUSGABE:                        A
LINKSBUENDIGE AUSGABE:  A
ZENTRIERTE AUSGABE:                 A


STRING: OLIVETTI P6066

RECHTSBUENDIGE AUSGABE:          OLIVETTI P6066
LINKSBUENDIGE AUSGABE:  OLIVETTI P6066
ZENTRIERTE AUSGABE:        OLIVETTI P6066


STRING: COMPUTERSYSTEM OLIVETTI P6066

RECHTSBUENDIGE AUSGABE: COMPUTERSYSTEM OLIVET
LINKSBUENDIGE AUSGABE:  COMPUTERSYSTEM OLIVET
ZENTRIERTE AUSGABE:     COMPUTERSYSTEM OLIVET
```

RANDOMIZE

ANWEISUNG: RANDOMIZE

FUNKTION: Bei Aufruf der Anweisung RANDOMIZE in Verbindung mit der Funktion RND wird eine von der Standardfolge verschiedene Folge von Zufallszahlen erzeugt.

FORMAT: RAN[DOMIZE]

BEMERKUNGEN:

- Nach dem Einschalten der Maschine wird beim Erzeugen von Zufallszahlen jeweils mit der gleichen Basiszahl begonnen.
- Mit Hilfe von RANDOMIZE wird bei jedem Programmlauf bei eingeschalteter Maschine eine andere Folge von Zufallszahlen erzeugt.

BEISPIELE: Beispiel 1

```
LIST
FILE


0010 REM** ERZEUGUNG VON ZUFALLSZAHLEN ***
0020 REM
0030 REM   OHNE RANDOMIZE
0040 FOR I=1 TO 20 STEP 1
0050 PRINT RND,
0060 NEXT I
0070 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105
 4.7693357E-02    .68675523        .17611750        3.5848356E-02    .50296996
 .62662024        .88834013        .85254312        .73719855        .70058045
```

Beispiel 2

```
LIST
FILE


0010 REM** ERZEUGUNG VON ZUFALLSZAHLEN ***
0020 REM
0030 REM   MIT RANDOMIZE
0035 RANDOMIZE
0040 FOR I=1 TO 20 STEP 1
0050 PRINT RND,
0060 NEXT I
0070 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
 .46928183        .93340150        .73981330        .93170842        .66550775
 4.9468396E-02    .41167724        .16474361        4.9257438E-02    .45449761
 .29001342        .73709782        1.0500667E-02    .92415374        .38637937
 .26806888        .15637405        7.7169896E-03    .18172449        3.9252707E-03
RUN
 .30027045        .84385646        .14483391        .59239684        .91177673
 .51377738        .21853162        .35957285        .80906657        .55912861
 .96844639        .33748557        .87592708        .94394928        .14757729
 .35964649        .44423446        .63348580        .61190501        .13459401
RUN
 .13125906        .14975876        .45703308        .24691844        .52625347
 .70871824        .81387396        .89417735        .15268155        .50867605
 .49596029        .44052650        .92023354        .46784937        .41861974
 .59421705        .61918537        .90043514        .44113692        .74452268
```

READ

ANWEISUNG: READ

FUNKTION: Zuweisung von Werten aus dem internen File an Variable. Die Werte werden in einer internen Tabelle mit Hilfe von DATA-Anweisungen generiert (siehe Anweisung DATA).

FORMAT: READ {num. Var. / Stringvar.} [,{num. Var. / Stringvar.}]...

WIRKUNG: Die Werte der internen Tabelle werden der Reihe nach den Variablen in der READ-Anweisung zugewiesen. Mit dem Lesen der Tabelle wird an der durch den Pointer bezeichneten Stelle begonnen (siehe Anweisung DATA).

Der Pointer kann mit der Anweisung RESTORE auf das erste Element der Tabelle zurückgesetzt werden.

BEMERKUNGEN:

- Die Werte werden den Indizes von Feldern erst beim Aufruf innerhalb der READ-Anweisung zugewiesen; somit kann eine Variable der READ-Anweisung als Index eines Feldelementes der gleichen READ-Anweisung verwendet werden. Die Werte müssen im Typ mit den Variablen übereinstimmen.
- READ kann nur dann verwendet werden, wenn im Programm mindestens eine DATA-Anweisung vorkommt. In einer READ-Anweisung dürfen nur so viele Werte verlangt werden, wie in der internen Tabelle noch vorhanden sind.

BEISPIELE: Beispiel 1

```
LIST
FILE


0010 REM*** READ ***
0020 REM
0030 DATA 1,2.54
0040 READ I,A(I)
0050 PRINT "I=";I,"A(I)=";A(I)
0060 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
I= 1            A(I)= 2.54
```

Beispiel 2

```
LIST
FILE


0010 REM*** READ ****
0020 REM
0030 DATA 1,2,3,"STRING",5,6,7,8,9,10
0040 READ A,B,C,A$,D
0050 PRINT A$,A,B,C,D
0060 READ F,G,H
0070 PRINT F,G,H
0080 READ X,Y,Z
0090 PRINT X,Y,Z
0100 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
STRING            1                 2                 3                 5
 6                7                 8
ERROR 88  IN LINE 80
```

READ:

ANWEISUNG: READ:

FUNKTION: Zuweisung von Daten aus einem externen File an die Variablen des Programmes.

FORMAT: READ: filedesign,{num.Var. / Stringvar}[,{num.Var. / Stringvar}]...[EOF Zeilennr.]

"filedesignator" arithmetischer Ausdruck.

WIRKUNG: Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl n gerundet. Diese Zahl n bezeichnet das File in der FILES-Anweisung, dessen Elemente gelesen werden sollen. Der Lesevorgang beginnt bei dem durch den Pointer des Files bezeichneten Element. Nach dem Lesen weist der Pointer auf das nächste Element des Datenfiles. Wird beim Lesen das Ende des Files überschritten, erfolgt eine Fehlermeldung. Ist jedoch der Parameter "EOF Zeilennr." angegeben, wird das Programm bei Erreichen des Fileendes bei der Zeile "Zeilennr." fortgesetzt, ohne dass eine Fehlermeldung erfolgt.

BEMERKUNGEN:

- In einem Randomfile kann vor der Anweisung READ: die Anweisung SETW: stehen (siehe Anweisung SETW:), um den Pointer an die Stelle innerhalb des Files zu setzen, ab welcher gelesen werden soll. Fehlt die Anweisung SETW: vor der ersten READ:-Anweisung, beginnt das Lesen beim ersten Wort des Files.
- Textfiles können mit READ: wie sequentielle Datenfiles gelesen werden. Jede Textzeile entspricht einem String, in dem auch die (vierstellige) Zeilennummer enthalten ist.
- Der Filedesignator muss grösser sein als Null und kleiner oder gleich der Anzahl Filenamen in der FILES-Anweisung.
- In einer READ:-Anweisung kann eine Variable als Index eines nachfolgenden Feldelementes derselben READ:-Anweisung verwendet werden.

BEISPIELE:

```
LIST
FILE     READ1


0010 REM*** READ: ***
0020 REM
0030 FILES SFILE
0040 DCL 6A$
0045 DCL S(A)
0050 RESTORE :1
0060 FOR I=1 TO 100 STEP 1
0070 READ :1,A,A$
0080 PRINT A$;A,
0090 NEXT I
0100 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
STRING 1        STRING 2        STRING 3        STRING 4        STRING 5
STRING 6        STRING 7        STRING 8        STRING 9        STRING 10
STRING 11       STRING 12       STRING 13       STRING 14       STRING 15
STRING 16       STRING 17       STRING 18       STRING 19       STRING 20
STRING 21       STRING 22       STRING 23       STRING 24       STRING 25
ERROR 84  IN LINE 70
```

```
LIST
FILE     READ2


0010 REM*** READ:   MIT EOF-ANWEISUNG***
0020 REM
0030 FILES SFILE
0040 DCL 6A$
0045 DCL S(A)
0050 RESTORE :1
0060 FOR I=1 TO 100 STEP 1
0070 READ :1,A,A$ EOF 95
0080 PRINT A$;A,
0090 NEXT I
0095 PRINT "FILE-ENDE NACH";I-1;" WERTEPAAREN ERREICHT"
0100 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
STRING 1        STRING 2        STRING 3        STRING 4        STRING 5
STRING 6        STRING 7        STRING 8        STRING 9        STRING 10
STRING 11       STRING 12       STRING 13       STRING 14       STRING 15
STRING 16       STRING 17       STRING 18       STRING 19       STRING 20
STRING 21       STRING 22       STRING 23       STRING 24       STRING 25
FILE-ENDE NACH 25  WERTEPAAREN ERREICHT
```

REM

ANWEISUNG: REM (remark)

FUNKTION: Einschub von erläuternden Texten in ein Programm.

FORMAT: REM Kommentar

"Kommentar" beliebiger Text.

WIRKUNG: REM ist eine nicht ausführbare Anweidung. Der Kommentar erscheint im Ausdruck (Listing), ist jedoch für die Programmausführung bedeutungslos.

Beim Testen eines Programms können Zeilen vorübergehend von der Programmausführung ausgeschlossen werden, indem hinter die Zeilennummer die Anweisung REM angefügt wird (im Command-Mode mit dem Systembefehl FETCH). Dadurch geht die Zeile nicht verloren und kann später wieder reaktiviert werden.

BEISPIEL:

```
LIST
FILE


0010 REM*** DIES IST EIN BEISPIEL ***
0020 REM***   FUER DIE ANWEISUNG  ***
0030 REM
0040 REM          REMARK
0050 REM
0060 REM
0070 END

END OF LISTING
```

RESTORE

ANWEISUNG: RESTORE

FUNKTION: Setzen des Pointers des internen Files auf das erste Element des Files (d.h. auf das erste Element der DATA-Anweisung mit der niedrigsten Zeilennummer).

FORMAT: RESTORE

WIRKUNG: Der Pointer des internen Files wird auf die erste Stelle gesetzt.

BEMERKUNG: Siehe Anweisungen DATA und READ.

BEISPIEL:

```
LIST
FILE


0010 REM*** RESTORE ***
0020 REM
0030 DATA 1,2,3,4,5,6,7,8,9
0040 FOR I=1 TO 5 STEP 1
0050 READ A
0060 PRINT A,
0070 NEXT I
0080 READ A,B,C,D
0090 PRINT D,C,B,A
0100 RESTORE
0110 FOR I=1 TO 9 STEP 1
0120 READ X
0130 PRINT X;
0140 NEXT I
0150 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
 1              2              3              4              5
 9              8              7              6
 1  2  3  4  5  6  7  8  9
```

RESTORE:

ANWEISUNG: RESTORE:

FUNKTION: Positionieren des Pointers eines externen Text- oder sequentiellen Datenfiles auf das erste Element im File und Setzen des Files in den Lese-Modus.

FORMAT: RESTORE: filedesignator

"filedesignator" arithmetischer Ausdruck.

WIRKUNG: Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl n gerundet.

Der Pointer des durch den Filedesignator n bestimmten Files wird auf die erste Stelle gesetzt.

BEMERKUNGEN:

- Der Filedesignator muss eine Zahl sein, die grösser als Null und kleiner oder gleich der Anzahl der Filenamen in der FILES-Anweisung ist.
- Ein sequentielles Daten- oder Textfile ist nach der Anweisung RESTORE: im Lese-Modus.

BEISPIEL:

```
FILE     RESTOR


0010 FILES SFILE
0020 RESTORE :1
0030 FOR I=1 TO 10 STEP 1
0040 READ :1,A
0050 PRINT A,
0060 NEXT I
0070 PRINT
0080 RESTORE :1
0090 FOR K=1 TO 5 STEP 1
0100 READ :1,B
0110 PRINT B,
0120 NEXT K
0130 END

END OF LISTING



RUN RESTOR

 10              20              30              40              50
 60              70              80              90              100

 10              20              30              40              50
```

RETURN

ANWEISUNG: RETURN

FUNKTION: RETURN bildet das logische Ende eines Unterprogrammes und bewirkt den Rücksprung in die unmittelbar auf das aufrufende GOSUB oder ON...GOSUB folgende Programmzeile.

FORMAT: RETURN

WIRKUNG: Siehe Anweisungen GOSUB und ON...GOSUB.

REVERSE

ANWEISUNG: REVERSE

FUNKTION: Umwandlung des Bildschirminhaltes von positiver auf negative Darstellung und umgekehrt.

FORMAT: REVERSE

WIRKUNG: Die Darstellung der Schrift auf dem Bildschirm wird mit der Farbe des Hintergrundes vertauscht.

ANWEISUNG: RKB (read keyboard)

FUNKTION: Eingabe beliebiger Zeichen (einschliesslich aller Sonderzeichen, wie Komma(,) und Anführungszeichen (")) über die Tastatur und ihre Zuweisung an eine Stringvariable.

FORMAT: RKB Stringvariable

"String-
variable" einfache oder indizierte Variable.

WIRKUNG: Der Programmablauf wird unterbrochen. Im Display erscheint ein Fragezeichen (?). Die eingegebene Zeichenfolge wird der Stringvariablen zugewiesen.

BEMERKUNG: Die Eingabe kann sowohl über die Tastatur als auch durch die Ausführung von Prozeduren (siehe Kapitel 6.4) erfolgen.

BEISPIEL:

```
FILE


0010 REM ** BEISPIEL RKB **
0020 REM
0030 INPUT A$,B$
0040 INPUT C$
0050 PRINT
0060 PRINT A$,B$,C$
0070 RKB A$
0080 RKB B$
0090 RKB C$
0100 PRINT A$,B$,C$
0110 END

END OF LISTING



RUN
?
LEERSTELLEN,SPEZ.ZEICHEN
?
KOMMA,KOMMA
TOO MUCH INPUT-EXCESS IGNORED

LEERSTELLEN     SPEZ.ZEICHEN    KOMMA
?
LEERSTELLEN
?
"SPEZ.ZEICHEN"
?
KOMMA,KOMMA
  LEERSTELLEN   "SPEZ.ZEICHEN"  KOMMA,KOMMA
```

SCRATCH:

ANWEISUNG: SCRATCH:

FUNKTION: Vorbereiten für das Schreiben auf ein externes sequentielles Datenfile ab dem ersten Element.

FORMAT: SCRATCH: filedesignator

"filedesignator" arithmetischer Ausdruck.

WIRKUNG: Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl n gerundet.

Der Pointer des Files mit dem Filedesignator n wird an den Anfang des Files gesetzt. Mit WRITE: können nun die Daten sequentiell auf das externe File geschrieben werden.

BEMERKUNGEN:
- Der Filedesignator muß größer als Null und kleiner oder gleich der Anzahl der Filenamen in der FILES-Anweisung sein.
- Die SCRATCH:-Anweisung darf nur bei sequentiellen Datenfiles angewendet werden.

BEISPIEL:

```
FILE


0010 FILES SFILE
0020 SCRATCH :1
0030 INPUT A
0040 WRITE :1,A
   .
   .
   .
   .
```

SETW:

ANWEISUNG: SETW: (set word)

FUNKTION: Setzen des Pointers eines Randomfiles auf ein beliebiges Wort im File.

FORMAT: SETW: filedesignator TO worddesignator

"filedesignator" arithmetischer Ausdruck.

"worddesignator" arithmetischer Ausdruck.

WIRKUNG: Die Ergebnisse der arithmetischen Ausdrücke werden auf die nächste ganze Zahl gerundet.

Der Pointer des Files mit dem Filedesignator n wird auf das m-te Wort (Worte zu 4 Bytes) im File positioniert.

BEMERKUNGEN:

- Das File mit dem Designator n muß vom Typ Random (R oder Z) sein. Nach der Anweisung SETW: können Daten in das File geschrieben (WRITE:) oder Daten vom File gelesen (READ:) werden.
- Der Filedesignator n muß eine ganze Zahl sein, die größer als Null und kleiner oder gleich der Anzahl der Filenamen in der FILES-Anweisung ist.
- Der Wortdesignator m muß größer als Null und (von 4 Bytes-Worten ausgehend) kleiner oder gleich der Anzahl maximal zu speichernder Worte sein (siehe Kapitel 5.2 "Erstellen von Datenfiles").

BEISPIEL:

```
FILE     SETW

0010 REM ** WORTE 1-25 MIT ZAHLEN 1-25 BESCHREIBEN **
0020 REM
0030 DCL SINGLE
0040 REM
0050 FILES RANDOM
0060 SETW :1 TO 1
0070 FOR I=1 TO 25 STEP 1
0080 WRITE :1,I
0090 NEXT I
0100 REM
0110 REM ** SUCHEN DURCH MANUELLE POINTERWAHL **
0120 REM
0130 INPUT P
0140 SETW :1 TO P
0150 READ :1,A
0160 PRINT "POINTER AUF";P;TAB(20);"WERT=";A
0170 GOTO 130
0180 END

END OF LISTING


RUN SETW

POINTER AUF 2       WERT= 2
POINTER AUF 3       WERT= 3
POINTER AUF 3.4     WERT= 3
POINTER AUF 3.5     WERT= 4
POINTER AUF 8       WERT= 8
POINTER AUF 11.99   WERT= 12
POINTER AUF 20      WERT= 20
POINTER AUF 25      WERT= 25
POINTER AUF 26      WERT= 0
-9.9
ERROR 96  IN LINE 140
```

ERROR 1:Wort nicht beschrieben
Unerlaubter Worddesignator
(negativ)

STOP

ANWEISUNG: STOP

FUNKTION: Unterbrechen der Programmausführung.

FORMAT: STOP

WIRKUNG: Der Programmablauf wird unterbrochen. Auf dem Display erscheint die Meldung "STOP IN LINE Zeilennummer", worin "Zeilennummer" die Nummer der die STOP-Anweisung enthaltenden Zeile angibt.

Das System geht in den Debugging-Mode (siehe Kapitel 11).

BEMERKUNGEN:

- STOP darf nicht in einer mehrzeiligen Funktionsdefinition vorkommen. Die Programmausführung kann mit CONTINUE oder "START Zeilennummer" oder durch Drücken der Taste STEP schrittweise fortgesetzt werden.
- Die Meldung "STOP IN LINE Zeilennummer" kann unterdrückt werden, wenn zuvor eine mit Strichpunkt (;) abgeschlossene DISP-Anweisung steht.

BEISPIEL:

```
LIST
FILE


0010 INPUT A,B
0020 STOP
0030 PRINT A,B
0040 END

END OF LISTING



RUN
?
2,5
STOP       IN LINE 20                    CONTINUE
 2                 5


FILE


0010 INPUT A,B
0015 DISP "PAPIER EINGELEGT?";
0020 STOP
0030 PRINT A,B
0040 END

END OF LISTING



RUN
?
2,5
PAPIER EINGELEGT?                        CONTINUE
 2                 5
```

TRACE OFF

ANWEISUNG: TRACE OFF

FUNKTION: Aufhebung der Wirkung von TRACE ON.

FORMAT: TRACE OFF

WIRKUNG: Die Nummern der ausgeführten Programmzeilen werden nicht mehr gedruckt. (Die Gültigkeit von TRACE ON wird aufgehoben.)

BEMERKUNG: Fehlt TRACE OFF, werden die Zeilennummern solange gedruckt, bis die END-Anweisung erreicht ist. Die Anweisung TRACE OFF hebt auch die Wirkung des Systembefehls TRC auf.

BEISPIEL: Siehe TRACE ON.

TRACE ON

ANWEISUNG: TRACE ON

FUNKTION: Ausdruck der Zeilennummern während des Programmablaufes.

FORMAT: TRACE ON

WIRKUNG: Die Zeilennummer der jeweils ausgeführten Anweisung wird ausgedruckt.

Ausnahmen: Nicht ausführbare Anweisungen (REM, DCL, DIM, DEF FN usw.) und die Anweisung TRACE ON.

Der Ausdruck der Zeilennummmern erfolgt zwischen den programmgesteuerten Ausgaben.

BEMERKUNG: Die Anweisung TRACE ON hat dieselbe Wirkung wie Systembefehl TRC. Die Wirkung wird durch die Anweisung TRACE OFF oder durch Eingabe des Systembefehls TRC in Debugging-Mode aufgehoben.

BEISPIEL:

```
FILE


0010 TRACE ON
0020 REM
0030 DCL SINGLE
0040 DIM A(25)
0050 REM
0060 DEF FNA(X)
0070 LET FN*=X*X
0080 FNEND
0090 PRINT TAB(10);
0100 FOR I=1 TO 3 STEP 1
0110 PRINT FNA(I);
0120 NEXT I
0130 PRINT
0140 GOSUB 160
0150 GOTO 240
0160 PRINT TAB(10);
0170 FOR K=10 TO 30 STEP 10
0180 PRINT FNA(K);
0190 NEXT K
0200 TRACE OFF
0210 PRINT
0220 PRINT
0230 RETURN
0240 END

END OF LISTING



RUN
#90
#100
#110
#120
#110
#120
#110
#120
#130
           1   4   9
#140
#160
#170
#180
#190
#180
#190
#180
#190
#200
           100   400   900
```

WHERE:

ANWEISUNG: WHERE:

FUNKTION: Abfrage der Position des Pointers, des Types des adressierten Datenelementes und gegebenenfalls der Länge des adressierten Strings.

FORMAT: WHERE: filedesignator, $n_1\left[,n_2\left[,n_3\right]\right]$

"filedesignator" arithmetischer Ausdruck.

"n_1", "n_2", "n_3" numerische Variable.

WIRKUNG: Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl gerundet.

Der Variablen "n1" wird die Position des Pointers zugewiesen.

Der Varibalen "n2" wird ein Wert zugewiesen, der den Datentyp des adressierten Datenelementes angibt.

"n2" kann folgende Werte annehmen:

Wert von n_2	Bedeutung
0	Der Pointer weist auf ein nicht identifizierbares Wort.
1	Der Pointer weist: - auf eine einfach genaue numerische Variable oder - auf das zweite Wort einer doppelt genauen numerischen Variablen oder - auf ein Wort innerhalb eines Strings (d.h. nicht auf das Kontrollwort).
2	Der Pointer weist auf das erste Wort einer doppelt genauen numerischen Variablen.
3	Der Pointer weist auf das Kontrollwort eines Strings.
4	Die End-of-File-Marke ist erreicht.
5	Der Pointer weist auf eine nicht initialisierte, einfach genaue numerische Variable.

Wert von n_2	Bedeutung
6	Der Pointer zeigt auf das erste Wort einer nicht initialisierten, doppelt genauen numerischen Variablen.
7	Der Pointer zeigt auf das Kontrollwort eines nicht initialisierten Strings.

Weist der Pointer auf den Anfang eines alphanumerischen Datenelementes (n2=3), wird der Variablen "n3" die Anzahl Zeichen des Strings zugewiesen. Andernfalls ist n3=0.

BEMERKUNGEN:

- Die Variablen "n2" und "n3" können für sequentielle Files nur dann spezifiziert werden, wenn sich das externe Datenfile im Lesemodus befindet.
- "n2" erhält auch dann den Wert 1, wenn nicht auf das erste Wort eines Datenelementes positioniert wurde, sondern an dieser Stelle ein gültiger Identifikator steht.

BEISPIEL: Ausdruck eines externen Datenfiles unbekannter Struktur (Randomfile).

```
FILE     FLP


0010 FILES *
0020 DCL S(X),1023(X$)
0030 DISP "FILENAME",
0040 INPUT F$
0050 FILE   1,F$
0060 PRINT "FILE ";F$
0070 PRINT
0080 PRINT "WORT-NR.","TYP/LAENGE","INHALT (ZWISCHEN > UND <)"
0090 PRINT
0100 PRINT
0110 LET P=1
0120 SETW :1 TO P
0130 WHERE : 1,N1,N2,N3
0140 REM **  N1=WORTNR.  N2=DATENTYP   N3=STRINGLAENGE
0150 IF N2>0 THEN 200
0160 REM **  N2=0 → NICHT ANFANG EINES DATENELEMENTES  **
0170 PRINT ,,"POINTER INNERHALB EINES DATENELEMENTES"
0180 LET P=P+1
0190 GOTO 130
0200 IF N2>1 THEN 260
0210 REM **  N2=1 → EINFACH GENAUE NUMERISCHE VARIABLE  **
0220 READ :1,X
0230 PRINT N1," S";TAB(32);X
0240 LET P=P+1
0250 GOTO 130
0260 IF N2>2 THEN 320
0270 REM **  N2=2 → DOPPELT GENAUE NUMERISCHE VARIABLE  **
0280 READ :1,Y
0290 PRINT N1," D";TAB(32);Y
0300 LET P=P+2
0310 GOTO 130
0320 IF N2>3 THEN 490
0330 REM **  N2=3 → STRING DER LAENGE N3  **
0340 READ :1,X$
0350 PRINT N1,N3;TAB(32);">";
0360 LET I=1
0370 IF LEN(EXT$(X$,(I-1)*40+1,LEN(X$)))<=40 THEN 430
0380 IF I=1 THEN 400
0390 PRINT ,,
0400 PRINT EXT$(X$,(I-1)*40+1,I*40)
0410 LET I=I+1
0420 GOTO 370
0430 IF I=1 THEN 450
0440 PRINT ,,
0450 PRINT EXT$(X$,(I-1)*40+1,LEN(X$));"<"
0470 LET P=P+INT((N3-1)/4+2)
0480 GOTO 130
0490 IF N2>4 THEN 550
0500 REM **  N2=4 → FILE-ENDE IST ERREICHT  **
0510 PRINT
0520 PRINT "FILE-ENDE ERREICHT"
0530 GOTO 580
0540 REM **  N2>4 → WORT NICHT BESCHRIEBEN  **
0550 PRINT N1,,"NICHT BESCHRIEBEN"
0560 LET P=P+1
0570 GOTO 120
0580 END

END OF LISTING
```

```
RUN FLP

FILE RANDOM

WORT-NR.        TYP/LAENGE      INHALT (ZWISCHEN > UND <)


 1                              NICHT BESCHRIEBEN
 2              D               6542.325
 4              S               555
 5              D               6.5896586E+08
 7                              NICHT BESCHRIEBEN
 8                              NICHT BESCHRIEBEN
 9              S              -1000
 10             15             >TEXTPROBE NR. 1<
 15                             NICHT BESCHRIEBEN
 16                             NICHT BESCHRIEBEN
 17             D               3652.01
 19             S               6000
 20                             NICHT BESCHRIEBEN
 21             45             >  TEXTPROBE NR.2 - asdfghjklpoiuytrewqzx
                                cvbnm<
 34                             NICHT BESCHRIEBEN
 35                             NICHT BESCHRIEBEN
 36             D               36
 38                             NICHT BESCHRIEBEN
 39             67             >TEXTPROBE NR.3 - MEHRZEILIGER TEXT - AUF
                                GEFUELLT MIT 'BLANKS'       <
 57                             NICHT BESCHRIEBEN
 58                             NICHT BESCHRIEBEN
 59                             NICHT BESCHRIEBEN
 60             D               6542111
 62                             NICHT BESCHRIEBEN
 63                             NICHT BESCHRIEBEN
 64             S               999999

FILE-ENDE ERREICHT
```

WRITE:

ANWEISUNG: WRITE:

FUNKTION: Schreiben von Daten auf ein externes File.

FORMAT: WRITE: filedesign,{num.Ausdr. / Stringausd}[,{num.Ausdr. / Stringausd}]..[EOF Zeilenn]

"filedesignator" arithmetischer Ausdruck.

WIRKUNG: Der Wert des arithmetischen Ausdruckes wird berechnet und auf die nächste ganze Zahl n gerundet. Diese Zahl n bezeichnet das für die Aufzeichnung vorgesehene File in der FILES-Anweisung. Der Schreibvorgang beginnt bei dem durch den Pointer des Files bezeichneten Element. Nach dem Schreiben weist der Pointer auf das nächste Element des Datenfiles. Wird das Ende eines Files überschritten, erscheint eine Fehlermeldung. Ist jedoch der Parameter "EOF Zeilennummer" angegeben, wird das Programm beim Erreichen des File-Endes ohne Fehlermeldung bei der Zeile "Zeilennr." fortgesetzt.

BEMERKUNGEN:

- Steht WRITE: bei einem Randomfile als erste Anweisung ohne vorherige SETW:-Anweisung oder folgt WRITE: auf eine FILE:- oder RESTORE:-Anweisung, wird mit dem Schreiben beim ersten Wort des Files begonnen.
- Ein numerischer Ausdruck wird in doppelter Genauigkeit geschrieben, auch wenn für die Variablen einfache Genauigkeit vereinbart wurde:

```
0010 DCL SINGLE
0100 WRITE: 1,2*X
```

Der Wert 2*X wird doppelt genau ins File geschrieben

```
0010 DCL SINGLE
0100 Y=2*X
0110 WRITE:1,Y
```

Der Wert Y=2*X wird einfach genau ins File geschrieben.

BEISPIEL:

```
FILE


0010 FILES SFILE
0020 SCRATCH :1
0030 FOR I=1 TO 1000 STEP 1
0040 WRITE :1,I
0050 NEXT I
0060 END

END OF LISTING



RUN
ERROR 34  IN LINE 40
```

Fileende ohne Kontrolle erreicht.

```
FILE


0010 FILES SFILE
0020 SCRATCH :1
0030 FOR I=1 TO 1000 STEP 1
0040 WRITE :1,I EOF 60
0050 NEXT I
0060 PRINT I."ELEMENTE GESPEICHERT"
0070 END

END OF LISTING



RUN
 513 ELEMENTE GESPEICHERT
```

Fileende mit EOF geprüft.

9.2 Standardfunktionen

Die Standardfunktionen liefern als Ergebnis einen numerischen oder alphanumerischen Wert. Ist das Ergebnis der Funktion numerisch, so wird von einer numerischen Funktion gesprochen. Entsprechend wird von einer alphanumerischen Funktion gesprochen, wenn das Ergebnis der Funktion alphanumerisch ist.

Numerische Funktionen können auch alphanumerische Parameter (z.B. LEN) und alphanumerische Funktionen auch numerische Parameter (z.B. CHR$) aufweisen.

Der Aufruf numerischer Standardfunktionen ist innerhalb numerischer Ausdrücke (bzw. anstelle von numerischen Ausdrücken) möglich; analog dazu ist der Aufruf alphanumerischer Funktionen innerhalb von Stringausdrücken erlaubt.

Das allgemeine Format für Funktionen lautet:

```
FKT (Arg1,[Arg2]...)
```

"FKT" Name der Funktion, der bei numerischen Funktionen einer Folge von 3 Buchstaben entspricht, welche die Bedeutung der Funktion beschreibt. Alphanumerische Funktionen haben Namen, deren viertes Zeichen ein Dollarzeichen ($) ist.

"Arg" Die Argumente (Parameter) der Funktion können numerische und/oder alphanumerische Ausdrücke sein.

9.2.1 Trigonometrische Funktionen

Die trigonometrischen Funktionen

SIN(num. Ausdr.)
COS(num. Ausdr.)
TAN(num. Ausdr.)
COT(num. Ausdr.)
ASN(num. Ausdr.)
ACS(num. Ausdr.)
ATN(num. Ausdr.)

liefern die Werte der entsprechenden Sinus-, Cosinus-, Tangens-, Cotangens-, Arcussinus-, Arcuscosinus- und Arcustangensfunktion. Das Argument muß im Bogenmaß vorliegen; bei der Arcussinus-, Arcuscosinus- und Arcustangensfunktion ist das Ergebnis ein Winkel im Bogenmaß.

Für die Umwandlung eines Argumentes von Altgrad in Bogenmaß bzw. von Bogenmaß in Altgrad dienen die Funktionen

RAD(num. Ausdr.)
DEG(num. Ausdr.)

Werden die Funktionen im Calculator- oder Debugging-Mode verwendet, so ist mit der Vorwahl

SDEG
SGRAD
SRAD

die Angabe der Winkel in Altgrad (SDEG), Neugrad (SGRAD) oder im Bogenmaß (SRAD) möglich. Werden keine Angaben gemacht, sind Winkel im Bogenmaß einzugeben.

```
FILE     WINKEL


0010 DEF FNA(X)=PI/180*X
0020 DISP "VON GRAD, BIS GRAD,SCHRITTWEITE";
0030 INPUT A,E,S
0040 IF S=0 THEN 160
0050 PRINT "VON";A;"GRAD BIS";E;"GRAD, SCHRITTWEITE";S;"GRAD"
0060 PRINT
0070 PRINT " GRAD         BOGEN       SINUS      COSINUS      TANGENS     COTANGENS"
0080 PRINT
0090 FOR I=FNA(A) TO FNA(E) STEP FNA(S)
0100 PRINT USING 120,DEG(I),I,SIN(I),COS(I),TAN(I),COT(I)
0110 NEXT I
0120 ###.####    ###.####     #.####    #.####    ####.####  ####.####
0130 PRINT
0140 PRINT
0150 GOTO 20
0160 END

END OF LISTING



RUN WINKEL

VON 10 GRAD BIS 45 GRAD, SCHRITTWEITE 5 GRAD

 GRAD         BOGEN       SINUS      COSINUS      TANGENS     COTANGENS

 10.0000      0.1745      0.1736     0.9848       0.1763      5.6713
 15.0000      0.2618      0.2588     0.9659       0.2679      3.7321
 20.0000      0.3491      0.3420     0.9397       0.3640      2.7475
 25.0000      0.4363      0.4226     0.9063       0.4663      2.1445
 30.0000      0.5236      0.5000     0.8660       0.5774      1.7321
 35.0000      0.6109      0.5736     0.8192       0.7002      1.4281
 40.0000      0.6981      0.6428     0.7660       0.8391      1.1918
 45.0000      0.7854      0.7071     0.7071       1.0000      1.0000
```

9.2.2 Mathematische Standardfunktionen

EXP(X)	Exponentialfunktion (e^x, e: Eulersche Zahl)
HSN(X)	Sinus hyperbolicus von X
HCS(X)	Cosinus hyperbolicus von X
HTN(X)	Tangens hyperbolicus von X
LOG(X)	Natürlicher Logarithmus von X
LGT(X)	Zehnerlogarithmus (Briggscher Logarithmus) ($\log_{10} x$)
SQR(X)	Quadratwurzel von X ($\sqrt{x}$)
ABS(X)	Absolutbetrag von X ($\|x\|$)
INT(X)	Nächste ganze Zahl, die kleiner oder gleich dem Wert des Argumentes ist
SGN(X)	Vorzeichen von X Die Funktion liefert bei positivem X als Ergebnis +1, bei negativem X -1 und 0, falls X gleich 0 ist.
X:	Arithmetischer Ausdruck

```
FILE


0010 REM *** BEISPIEL FUER 'ABS' / 'INT' / 'SGN' ***
0020 REM
0030 INPUT A
0040 PRINT USING 50,A,ABS(A),A,INT(A),A,SGN(A)
0050  ABS(###.##)=###.##, INT(###.##)=###.##, SGN(###.##)=##
0060 PRINT
0070 GOTO 30
0080 END

END OF LISTING


RUN FUNK1

ABS( -2.00)=  2.00, INT( -2.00)= -2.00, SGN( -2.00)=-1

ABS(  1.59)=  1.59, INT(  1.59)=  1.00, SGN(  1.59)= 1

ABS( -1.59)=  1.59, INT( -1.59)= -2.00, SGN( -1.59)=-1

ABS(  0.00)=  0.00, INT(  0.00)=  0.00, SGN(  0.00)= 0
```

9.2.3 Numerische Funktionen ohne Argument

FUNKTION: PI

Die Funktion PI liefert als Ergebnis die Zahl π=3,141592.... in doppelter Genauigkeit.

FUNKTION: RND

Die Funktion RND (random number) liefert eine im Intervall (0,1) liegende Zufallszahl. Bei wiederholtem Aufruf von RND wird eine Standardfolge von Zufallszahlen geliefert. Wurde vor dem Aufruf von RND die Anweisung RANDOMIZE ausgeführt, liefert die Funktion RND eine von der Standardfolge verschiedene Folge von Zufallszahlen.

BEISPIEL ZU RND:

```
FILE


0010 FOR I=1 TO 10 STEP 1
0020 PRINT RND,
0030 NEXT I
0040 END

END OF LISTING



RUN
 .63829321        .41839390        .97356004        .88649157        .21507853
 4.7279296E-02    .98707879        .95457924        .55713896        .81827105

FILE


0005 RANDOMIZE
0010 FOR I=1 TO 10 STEP 1
0020 PRINT RND,
0030 NEXT I
0040 END

END OF LISTING



RUN
 .46928183        .93340150        .73981330        .93170842        .66550775
 4.9468396E-02    .41167724        .16474361        4.9257438E-02    .45449761
```

FUNKTION: DET

Die Funktion DET liefert als Ergebnis den Wert der Determinante der zuletzt invertierten numerischen Matrix. Bei Aufruf der Funktion muß das System mit der Option MAT initialisiert sein.

BEMERKUNG: Kann eine Matrixinversion nicht ausgeführt werden, da sie (fast) singulär ist, so liefert die Funktion DET dennoch ein korrektes Ergebnis.

BEISPIEL:

```
FILE     DET

0010 DIM A(4,4),B(4,4)
0020 MAT INPUT A
0030 MAT B=INV(A)
0040 PRINT "MATRIX A:"
0050 PRINT
0060 MAT PRINT A
0070 PRINT
0080 PRINT "INVERSE VON A:"
0090 PRINT
0100 MAT PRINT B
0110 PRINT
0120 PRINT "DETERMINANTE VON A=";DET
0130 END

END OF LISTING


RUN DET

MATRIX A:

 2.1             5               6.3             -4.9
 8               9               6.7             -5
 0               4              -1                2.5
 3.05            6.81            5.1              7

INVERSE VON A:

-.22800456        .17731504      -.14760641        1.9766979E-02
 7.1911362E-02   -4.9343865E-03   .23347374       -3.6570088E-02
 .12798795       -7.1199091E-02  -.16770262        9.8628865E-02
-6.3862999E-02   -2.0584618E-02  -4.0639036E-02    9.7963686E-02

DETERMINANTE VON A= 1698.7927
```

9.2.4 Spezielle numerische Funktionen

Folgende spezielle numerische Funktionen stehen zur Verfügung:

IOC	liefert den Zustand der Status-Bits einer peripheren Einheit über ein Arbeitsregister.
LEN	liefert als Ergebnis die aktuelle Länge (Anzahl Zeichen) des als Argument angeführten Stringausdruckes.
SCN	ermöglicht das Aufsuchen der Position eines Teilstrings in einem String.
TAB	ermöglicht bei der Ausgabe von Werten im Standardformat das Drucken ab einer bestimmten Druckposition.

FUNKTION: IOC (input/output control)

FUNKTION: Abfragen des Zustandes einer peripheren Einheit über ein Arbeitsregister.

FORMAT: IOC (X)

"X" numerischer Ausdruck.

WIRKUNG: Der numerische Ausdruck wird berechnet und gegebenenfalls auf die nächste ganze Zahl gerundet. Durch die Funktion IOC (X) können die im Arbeitsregister enthaltenen Informationen über den Zustand einer peripheren Einheit abgefragt und im Programm verarbeitet werden.

Die durch IOC (X) gelieferte Information wird im Handbuch "Programmierung von peripheren Einheiten" beschrieben.

BEMERKUNGEN:

- Bevor der Zustand einer peripheren Einheit durch IOC abgefragt werden kann, muss der Inhalt des dazugehörigen Zustandsregisters dieser Peripherie durch TEST bzw. WAIT ins Arbeitsregister übertragen werden.
- Die Interpretation der IOC-Werte ist abhängig vom verwendeten I/O-Kanal.
- Die Funktion IOC nimmt im allgemeinen nur die Werte 0 oder 1 an.

BEISPIEL:

```
FILE     IOC

0010 BUFFER #9,132
0020 SEND #9,"OLIVETTI=" AND GO
0030 TEST #9
0040 IF IOC(6)=1 THEN 70
0050 GOTO 100
0060 REM
0070 PRINT "BITTE DRUCKER EINSCHALTEN"
0080 STOP
0090 GOTO 30
0100 REM
0110 END

END OF LISTING


RUN IOC

BITTE DRUCKER EINSCHALTEN
```

FUNKTION: LEN (length)

FUNKTION: Die Funktion LEN liefert als Ergebnis die aktuelle Länge (Anzahl Zeichen) des als Argument angeführten Stringausdruckes.

FORMAT: LEN (Stringausdruck)

BEISPIEL:

```
FILE


0010 RKB A$
0020 PRINT "STRING A$ =";A$
0030 PRINT "LEN(A$)=";LEN(A$)
0040 PRINT
0050 GOTO 10
0060 END

END OF LISTING



STRING A$ =OLIVETTI  P 6066
LEN(A$)= 16

STRING A$ =TEXT1
LEN(A$)= 5

STRING A$ =######
LEN(A$)= 6

STRING A$ =#
LEN(A$)= 1
```

FUNKTION: SCN (scan)

FUNKTION: Die Funktion SCN ermöglicht das Aufsuchen der Position eines Teilstrings in einem String.

FORMAT: SCN (Stringausdr.$_1$,Stringausdr.$_2$,num.Ausdruck$_1$,num.Ausdruck$_2$)

WIRKUNG: Zunächst werden die Ergebnisstrings der beiden Stringausdrücke gebildet ("String1"/"String2") sowie die Ergebnisse "p1" und "p2" der beiden numerischen Ausdrücke errechnet und gerundet.

"String2" ist ein Teilstring von "String1".

"p1" gibt an, das wievielte Aufteten von "String2" in "String1" ermittelt werden soll.

"p2" gibt die Stelle in "String1" an, ab der mit dem Suchen begonnen werden soll.

Als Ergebnis wird die Stelle des ersten Zeichens von "String1" geliefert, an der "String2" (gesucht ab der p2-ten Stelle) das p1-te Mal in "String1" auftritt.

BEMERKUNGEN:
- Kommt "String2" im angegebenen Teil von "String1" nicht oder weniger als "p1" mal vor, so liefert die Funktion SCN als Ergebnis den Wert 0.
- Ist "p1" negativ, so wird "String1" mit der Schrittweite der Länge von "String2" durchsucht.
- Ist "p2" negativ, so wird als Ergebnis die Stelle geliefert, an der in "String1" der erste von "String2" verschiedene Teilstring gefunden wird.

BEISPIEL: Calc.-Mode aktiviert.

```
SCN("OLIVETTI","OLIVE",1,1)
 1

SCN("OLIVETTI P6060","60",2,1)
 13

SCN("OLIVETTI P6060","60",1,2)
 11

SCN("OLIVETTI P6060","OLIVE",2,1)
 0
```

FUNKTION: TAB (tabulation)

FUNKTION: Die Funktion TAB ermöglicht bei der Ausgabe von Werten im Standardformat mit DISP oder PRINT die Tabulation an eine bestimmte Position.

FORMAT: TAB (X)

"X" numerischer Ausdruck.

BEISPIEL:

```
FILE


0010 INPUT A,B,C
0020 REM *** PRINT OHNE 'TAB' ***
0030 PRINT A;B;C
0040 REM *** PRINT MIT 'STANDARD-TAB' ***
0050 PRINT A,B,C
0060 REM *** PRINT MIT 'TAB' ***
0070 PRINT A;TAB(20);B,TAB(50);C
0080 END

END OF LISTING



RUN
 23.65  654.22  600
 23.65           654.22          600
 23.65              654.22                          600
```

9.2.5 Alphanumerische Funktionen

An alphanumerischen Funktionen stehen zur Verfügung:

CHR$	Umwandlung eines numerischen Ausdruckes in ein ISO-Zeichen.
EXT$	Einem Stringausdruck wird ein Teilstring entnommen.
BLN$	Die Zeichen zweier Strings werden nach den Gesetzen der Boole'schen Algebra Bit für Bit verknüpft.
REP$	In einem Stringausdruck wird ein Teilstring durch einen anderen ersetzt.

CHR $

FUNKTION: CHR$ (charakter)

FUNKTION: Die Funktion liefert als Ergebnis das dem Wert des Argumentes entsprechende Zeichen aus der ISO-Code-Tabelle.

FORMAT: CHR$ (num.Ausdruck)

WIRKUNG: Der Wert des numerischen Ausdruckes wird berechnet und gegebenenfalls gerundet. Sein Wert n muß zwischen 0 und 255 liegen. Als Ergebnis wird ein String der Länge 1 geliefert, dessen binärer Wert dem Argument n entspricht.

BEMERKUNGEN:

- Ist n kleiner als 128, so wird das entsprechende Zeichen der ISO-Code-Tabelle als Ergebnis geliefert.
- Ist n = 128, wird das Zeichen Φ, ist n > 128, wird das Zeichen ▮ als Ergebnis geliefert.
- Liegt der Wert des numerischen Ausdruckes nicht zwischen 0 und 255, so wird als Ergebnis der Nullstring geliefert und ERROR 2 gemeldet.

BEISPIEL:

```
FILE


0010 DCL 80A$
0020 LET A$=""
0030 PRINT "NR. DES ZEICHENS:";
0040 DISP "NR. DES ZEICHENS";
0050 INPUT I
0060 IF I<0 THEN 140
0070 IF I>99 THEN 110
0080 PRINT " ";
0090 IF I>9 THEN 110
0100 PRINT " ";
0110 PRINT I;
0120 LET A$=A$+"   "+CHR$(I)+" "
0130 GOTO 40
0140 PRINT
0150 PRINT "ISO-ZEICHEN:      ";A$
0160 PRINT
0170 PRINT
0180 GOTO 20
0190 END

END OF LISTING



RUN
NR. DES ZEICHENS:  79   76   73   86   69   84   84   73
ISO-ZEICHEN:        O    L    I    V    E    T    T    I


NR. DES ZEICHENS:  80   54   48   54   54   32   66   65   83   73   67
ISO-ZEICHEN:        P    6    0    6    6        B    A    S    I    C


NR. DES ZEICHENS:   1    2    3    4    5    6    7    8    9   10
ISO-ZEICHEN:        ┌    ┴    ┘    ≥    ■    ·    ○    ↖    →    ≡
```

EXT$

FUNKTION: EXT$ (extract)

FUNKTION: Einem Stringausdruck wird ein Teilstring entnommen.

FORMAT: EXT$ (Stringausdruck, num.Ausdruck$_1$, num.Ausdruck$_2$)

WIRKUNG: Der Ergebnisstring "String" des Stringausdruckes wird gebildet und die beiden numerischen Ausdrücke werden berechnet und gegebenenfalls gerundet. Die Ergebnisse lauten "p1" und "p2".

Als Ergebnis des Funktionsaufrufes wird der Teilstring vom p1-ten Zeichen bis zum p2-ten Zeichen des Ergebnisstrings "String" geliefert.

BEMERKUNGEN:
- Ist "p1" = "p2", wird als Ergebnis ein Zeichen geliefert.
- "p1" muß größer als 0 und kleiner oder gleich "p2" sein.
- "p2" muß kleiner oder gleich der aktuellen Länge des Ergebnisstrings sein.

BEISPIEL:

```
FILE


0010 DCL 80A$
0020 DISP "BASIS-STRING";
0030 RKB A$
0040 DISP "EXTRAKT VON P1 BIS P2";
0050 INPUT P1,P2
0060 PRINT "BASIS-STRING: ";A$
0070 PRINT "EXTRAKT VON POS.";P1;"BIS";P2;": ";EXT$(A$,P1,P2)
0080 GOTO 20
0090 END

END OF LISTING



RUN

BASIS-STRING: BEISPIEL DER FUNKTION 'EXT'
EXTRAKT VON POS. 1 BIS 8 : BEISPIEL

BASIS-STRING: BEISPIEL DER FUNKTION 'EXT'
EXTRAKT VON POS. 14 BIS 27 : FUNKTION 'EXT'
```

FUNKTION: BLN$ (boolean)

FUNKTION: Die Zeichen zweier Strings werden nach den Gesetzen der Boole'schen Algebra Bit für Bit verknüpft.

FORMAT: BLN$ (num.Ausdr., Stringausdr.$_1$,Stringausdr.$_2$)

WIRKUNG: Der numerische Ausdruck wird berechnet und gegebenenfalls auf die nächste ganze Zahl P gerundet.

P gibt an, welche Boole'sche Operation als Basis der Verknüpfung dient.

Die Ergebnisse der Stringausdrücke werden gebildet und ergeben S1 (String1) und S2 (String2). Weisen S1 und S2 verschiedene Längen auf, so wird der kürzere String mit Byte-Inhalt "NUL"-Zeichen bis zur Länge des längeren Strings ergänzt. Die einander entsprechenden Bits werden mit der durch P angegebenen Operation verknüpft und das entsprechende Bit des Ergebnisstrings S3 gebildet.

Der Ergebnisstring S3 hat die Länge 0, wenn P kleiner als 0 oder größer als 15 ist, andernfalls weist er die Länge des größeren der beiden zu verknüpfenden Strings auf.

Argument P	Ergebnisstring S3
0	alle Bytes mit dem ISO-Code 0 belegt.
1	S1 AND S2
2	S1 AND (NOT S2)
3	S1
4	(NOT S1) AND S2
5	S2
6	S1 ORE S2 (OR exclusiv)
7	S1 OR S2
8	S1 NOR S2
9	NOT (S1 ORE S2) (NOR exclusiv)
10	NOT S2
11	S1 OR (NOT S2)
12	NOT S1
13	(NOT S1) OR S2
14	S1 NAND S2
15	alle Bytes mit dem ISO-Code 255 belegt.

Tabelle der Boole'schen Operationen

		P														
S1	S2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			AND		S1		S2	ORE	OR	NOR	NO-RE	$\overline{S2}$		$\overline{S1}$		NA-ND	

BEISPIELE:

```
BLN$(6,"A","p")          liefert "1"
BLN$(6,"B","p")          liefert "2"
        .
        .
        .
BLN$(6,"I","p")          liefert "9"
BLN$(6,"1","p")          liefert "A"
BLN$(6,"2","p")          liefert "B"
        .
        .
        .
BLN$(6,"9","p")          liefert "I"
```

```
FILE     BLN

0010 REM *** UMWANDLUNG VON GROSS- IN KLEINBUCHSTABEN UND UMGEKEHRT ***
0020 REM
0030 DCL 40(A$,B$)
0040 LET B$=" "
0050 PAD B$,32
0060 DISP "STRING (NUR BUCHSTABEN)";
0070 INPUT A$
0080 PRINT A$
0090 LET B$=EXT$(B$,1,LEN(A$))
0100 REM
0110 PRINT BLN$(6,A$,B$)
0120 REM
0130 PRINT
0140 GOTO 40
0150 END

END OF LISTING



RUN
ASDFGHJasdfghj
asdfghjASDFGHJ
```

REP$

FUNKTION: REP$ (replace)

FUNKTION: In einem Stringausdruck wird ein Teilstring durch einen anderen String ersetzt.

FORMAT: REP$ (Stringausdruck$_1$,Stringausdruck$_2$,Stringausdruck$_3$, num.Ausdruck$_1$,num.Ausdruck$_2$)

WIRKUNG: Die Stringausdrücke werden ausgewertet und liefern die Ergebnisstrings "String1", "String2" und "String3". Die beiden numerischen Ausdrücke werden berechnet und gegebenenfalls gerundet. Sie liefern die Ergebnisse "p1" und "p2".

"String1"	ist der durch die Funktion zu modifizierende String.
"String2"	ist der durch die Funktion zu ersetzende Teilstring von "String1".
"String3"	ist der String, durch den "String2" in "String1" ersetzt wird.
"p1"	gibt an, wie oft "String2" durch "String3" ersetzt wird (p1-mal).
"p2"	gibt die Position des Zeichens in "String1" an, ab welcher mit der Suche nach "String2" begonnen werden soll.

BEMERKUNGEN:

- Ist "p1" = 0, hat die Funktion keine Wirkung.
- Ist "p1" kleiner als 0, wird "String2" bei jedem Auftreten durch "String3" ersetzt.
- Ist "p1" > 0 und "String2" ein Nullstring, so wird "String3" vor dem Zeichen "p2" in "String1" p1-mal eingefügt.
- Ist "p1" < 0 und "String2" ein Nullstring, so wird ein (behebbarer) Fehler gemeldet (ERROR 2).

BEISPIEL:

```
FILE


0010 DCL 80A$
0020 LET A$="TEXT TEXT TEXT TEXT"
0030 PRINT "GRUNDSTRING: ";A$
0040 PRINT
0050 REM *** ERSETZEN X DURCH S (ERSTE 3) ***
0060 PRINT "1) ";REP$(A$,"X","S",3,1)
0070 REM *** ERSETZEN X DURCH S (ALLE) ***
0080 PRINT "2) ";REP$(A$,"X","S",-1,1)
0090 REM *** ERSETZEN T DURCH 'NICHTS' (4 MAL AB POS.5) ***
0100 PRINT "3) ";REP$(A$,"T","",4,5)
0110 REM *** EINFUEGEN VON E ***
0120 PRINT "4) ";REP$(A$," ","E ",-1,1)
0130 END

END OF LISTING



RUN

GRUNDSTRING: TEXT TEXT TEXT TEXT

1) TEST TEST TEST TEXT
2) TEST TEST TEST TEST
3) TEXT EX EX TEXT
4) TEXTE TEXTE TEXTE TEXT
```

9.3 Anweisungen für Matrizenoperationen

Die Anweisungen erlauben die Verarbeitung aller Matrix-Operationen mit numerischen Matrizen und Ein-/Ausgabeoperationen mit zweidimensionalen alphanumerischen Feldern.

Man unterscheidet zwischen den deklarierten und den aktuellen Dimensionen einer Matrix. Die deklarierten Dimensionen werden in einer DIM-Anweisung festgelegt oder, falls diese fehlt, mit den Dimensionen 10 x 10 (10 Zeilen und 10 Spalten) für numerische Matrizen und 5 x 5 für alphanumerische Matrizen angenommen. Die aktuellen Dimensionen einer Matrix können bei bestimmten Operationen verändert werden, sie sind aber immer kleiner oder gleich den deklarierten Dimensionen.

Die Namen der Felder werden in den Anweisungen der Option MAT ohne Indizes bzw. Klammern geschrieben. Einige Anweisungen erlauben zur Festlegung der aktuellen Dimensionen einen Klammernausdruck nach dem Feldnamen.

MAT...=

ANWEISUNG: MAT...=(Matrix-Zuweisung)

FUNKTION: Die Elemente einer Matrix werden den Elementen einer anderen Matrix zugewiesen.

FORMAT: MAT Matrix = Matrix

WIRKUNG: Die Werte aller Elemente der Matrix auf der rechten Seite des Gleichheitszeichens werden den entsprechenden Elementen der Matrix auf der linken Seite zugewiesen.

BEMERKUNG: Die deklarierten Dimensionen der Matrix auf der linken Seite des Gleichheitszeichens müssen größer oder gleich den aktuellen Dimensionen der rechten Matrix sein.

BEISPIEL:

```
FILE


0010 DIM A(11,11),B(11,11)
0020 FOR Z=1 TO 11 STEP 1
0030 FOR K=1 TO 11 STEP 1
0040 LET A(Z,K)=INT(RND*89)+10
0050 NEXT K
0060 NEXT Z
0070 PRINT "MATRIX A:"
0080 MAT PRINT A;
0090 PRINT
0100 MAT B=A
0110 PRINT "MATRIX B:"
0120 MAT PRINT B;
0130 END

END OF LISTING



RUN

MATRIX A:
 66  47  96  88  29  14  97  94  59  82  14
 71  25  13  54  65  89  85  75  72  24  31
 41  48  94  53  17  27  50  88  95  47  34
 86  44  47  44  34  30  13  63  19  88  29
 10  80  77  82  77  84  45  97  92  45  27
 37  45  98  68  32  42  90  92  22  79  32
 43  36  44  16  81  29  39  95  64  72  34
 96  90  67  67  24  35  95  13  60  95  98
 98  67  26  91  72  70  38  83  17  90  76
 56  92  36  14  52  46  77  33  55  97  39
 36  75  84  21  97  12  90  76  47  40  25

MATRIX B:
 66  47  96  88  29  14  97  94  59  82  14
 71  25  13  54  65  89  85  75  72  24  31
 41  48  94  53  17  27  50  88  95  47  34
 86  44  47  44  34  30  13  63  19  88  29
 10  80  77  82  77  84  45  97  92  45  27
 37  45  98  68  32  42  90  92  22  79  32
 43  36  44  16  81  29  39  95  64  72  34
 96  90  67  67  24  35  95  13  60  95  98
 98  67  26  91  72  70  38  83  17  90  76
 56  92  36  14  52  46  77  33  55  97  39
 36  75  84  21  97  12  90  76  47  40  25
```

ANWEISUNG: MAT...±(Matrix-Addition/-Subtraktion)

FUNKTION: Addition bzw. Subtraktion der Elemente zweier Matrizen. Die Ergebnisse können einer dritten Matrix zugewiesen werden.

FORMAT: MAT Matrix = Matrix {±} Matrix

WIRKUNG: Die Werte der korrespondierenden Elemente der beiden Matrizen rechts vom Gleichheitszeichen werden addiert bzw. subtrahiert und die Ergebnisse dieser Operation der Matrix links vom Gleichheitszeichen zugewiesen. Die Matrix auf der linken Seite nimmt die aktuellen Dimensionen der beiden rechten Matrizen an.

BEMERKUNGEN:
- Die Matrizen auf der rechten Seite des Gleichheitszeichens müssen dieselben aktuellen Dimensionen haben. Die deklarierten Dimensionen der Matrix auf der linken Seite des Gleichheitszeichens müssen größer oder gleich den aktuellen Dimensionen der beiden rechten Matrizen sein.
- Die Ergebnismatrix kann auch als Operand auf der rechten Seite eines oder auch beider Operanden stehen.

BEISPIEL:

```
FILE


0010 DIM A(3,4),B(3,4),C(3,4),D(3,4)
0020 FOR I=1 TO 3 STEP 1
0030 FOR J=1 TO 4 STEP 1
0040 LET A(I,J)=INT(RND*10)
0050 LET B(I,J)=INT(RND*10)
0060 NEXT J
0070 NEXT I
0080 PRINT "MATRIX A:"
0090 MAT PRINT A;
0100 PRINT
0110 PRINT "MATRIX B:"
0120 MAT PRINT B;
0130 PRINT
0140 PRINT "MATRIC C = MATRIX A + MATRIX B:"
0150 MAT C=A+B
0160 MAT PRINT C;
0170 PRINT
0180 PRINT "MATRIX D = MATRIX A - MATRIX B:"
0190 MAT D=A-B
0200 MAT PRINT D;
0210 END

END OF LISTING



RUN

MATRIX A:
 6  9  2  9
 5  0  1  5
 8  7  1  3

MATRIX B:
 4  8  0  9
 8  6  0  6
 8  7  2  4

MATRIC C = MATRIX A + MATRIX B:
 10  17  2  18
 13  6  1  11
 16  14  3  7

MATRIX D = MATRIX A - MATRIX B:
 2  1  2  0
-3 -6  1 -1
 0  0 -1 -1
```

MAT...* (scalar multip)

ANWEISUNG: MAT...Skalar*(Skalare Multiplikation)

FUNKTION: Die Werte aller Elemente einer numerischen Matrix werden mit dem Ergebnis eines arithmetischen Ausdruckes multipliziert und die Ergebnisse den korrespondierenden Elementen der Matrix auf der linken Seite des Gleichheitszeichens zugewiesen.

FORMAT: MAT Matrix = (num.Ausdruck) * Matrix

BEMERKUNGEN:
- Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen größer oder gleich den aktuellen Dimensionen der Matrix rechts vom Gleichheitszeichen sein. Die angegebene Matrix auf der linken Seite des Gleichheitszeichens kann auch als Operand auf der rechten Seite auftreten.
- Die Matrix auf der linken Seite des Gleichheitszeichens nimmt die aktuellen Dimensionen der rechten Matrix an.

BEISPIEL:

```
0010 DIM A(3,4),B(5,5)
0020 FOR I=1 TO 3 STEP 1
0030 FOR J=1 TO 4 STEP 1
0040 LET A(I,J)=INT(RND*10)
0050 NEXT J
0060 NEXT I
0070 PRINT "MATRIX A:"
0080 MAT PRINT A;
0090 PRINT
0100 PRINT "MATRIX B = MATRIX A * (INT(RND*10+1) :"
0110 MAT B=(INT(RND*10+1))*A
0120 MAT PRINT B;
0130 END

END OF LISTING



RUN

MATRIX A:
 6  4  9  8
 2  0  9  9
 5  8  0  6

MATRIX B = MATRIX A * (INT(RND*10+1) :
 12  8  18  16
 4  0  18  18
 10  16  0  12
```

ANWEISUNG: MAT...*(Matrix-Multiplikation)

FUNKTION: Durchführung einer Matrizenmultiplikation (Zeilen x Spalten). Das Ergebnis wird einer dritten Matrix zugewiesen.

FORMAT: MAT Matrix = Matrix * Matrix

WIRKUNG: Die beiden numerischen Matrizen auf der rechten Seite des Gleichheitszeichens werden nach den Regeln der Matrizenmultiplikation multipliziert. Voraussetzung ist, daß die Anzahl der Spalten der linken Matrix gleich der Anzahl der Zeilen der rechten Matrix ist.

Wird eine Matrix A mit den aktuellen Dimensionen (p, m) mit einer Matrix B mit den Dimensionen (m, n) multipliziert, so ist das Resultat eine Ergebnismatrix C mit den Dimensionen (p, n), für die gilt:

$$i = 1,2,\ldots, p$$
$$j = 1,2,\ldots, n$$

$$c_{i,j} = \sum_{k=1}^{m} a_{i,k} * b_{k,j}$$

BEMERKUNGEN:
- Die Ergebnismatrix darf nicht als Operand auf der rechten Seite des Gleichheitszeichens stehen.
- Die deklarierten Dimensionen der Matrix auf der linken Seite des Gleichheitszeichens müssen größer oder gleich der Anzahl Zeilen der ersten Matrix auf der rechten Seite und der Anzahl Spalten der zweiten Matrix sein.

BEISPIEL:

```
FILE


0010 DIM A(4,3),B(3,3),C(4,4)
0020 FOR I=1 TO 3 STEP 1
0030 FOR J=1 TO 3 STEP 1
0040 LET A(I,J)=INT(RND*10+1)
0050 LET B(I,J)=INT(RND*10+1)
0060 NEXT J
0070 NEXT I
0080 FOR I=1 TO 3 STEP 1
0090 LET A(4,I)=INT(RND*10+1)
0100 NEXT I
0110 PRINT "MATRIX A:"
0120 MAT PRINT A;
0130 PRINT
0140 PRINT "MATRIX B:"
0150 MAT PRINT B;
0160 PRINT
0170 PRINT "MATRIX C = MATRIX A * MATRIX B:"
0180 MAT C=A*B
0190 MAT PRINT C;
0200 END

END OF LISTING


RUN
MATRIX A:
 7  10  3
 10  6  1
 2  6  9
 8  8  2

MATRIX B:
 5  9  1
 10  9  7
 1  7  9

MATRIX C = MATRIX A * MATRIX B:
 138  174  104
 111  151  61
 79  135  125
 122  158  82
```

MAT...CON

ANWEISUNG: MAT...CON(Einermatrix)

FUNKTION: Jedem Element einer Matrix wird der Wert 1 zugewiesen.

FORMAT: MAT Matrix = CON[(num.Ausdr., num.Ausdr.)]

WIRKUNG: Jedem Element der numerischen Matrix links vom Gleichheitszeichen wird der Wert 1 zugewiesen.

Falls Parameter angegeben sind, werden die beiden numerischen Ausdrücke berechnet und auf die nächste ganze Zahl gerundet. Diese beiden Zahlen bestimmen die aktuellen Dimensionen der Matrix.

BEMERKUNG: Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen größer oder gleich den aktuellen, sich aus den numerischen Ausdrücken ergebenden Dimensionen sein.

BEISPIEL:

```
FILE


0010 DIM A(8,8)
0020 PRINT "MATRIXDIMENSION 8 × 8 :"
0030 MAT A=CON
0040 MAT PRINT A;
0050 PRINT
0060 PRINT "MATRIXDIMENSION 4 × 5 :"
0070 MAT A=CON(4,5)
0080 MAT PRINT A;
0090 END

END OF LISTING



RUN

MATRIXDIMENSION 8 × 8 :
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1

MATRIXDIMENSION 4 × 5 :
 1  1  1  1  1
 1  1  1  1  1
 1  1  1  1  1
 1  1  1  1  1
```

ANWEISUNG: MAT...IDN(Einheitsmatrix)

FUNKTION: Eine quadratische, numerische Matrix erhält die Werte der Einheitsmatrix.

FORMAT: MAT Matrix=IDN[(num.Ausdruck,num.Ausdruck)]

WIRKUNG: Den Elementen der Hauptdiagonale wird der Wert 1 zugewiesen. Alle anderen Elemente der quadratischen Matrix werden mit Null belegt. Falls als Parameter angegeben, werden die numerischen Ausdrücke berechnet und die Ergebnisse gerundet. Diese ganzen Zahlen legen die aktuellen Dimensionen der quadratischen Matrix fest.

BEMERKUNG: Die Ergebnisse der beiden numerischen Ausdrücke müssen gleich und größer Null sein. Die deklarierten Dimensionen der quadratischen Matrix müssen größer oder gleich den sich aus den arithmetischen Ausdrücken ergebenden Zahlen sein.

BEISPIEL:

```
FILE


0010 DIM A(8,8)
0020 PRINT "MATRIXDIMENSION 5 x 5 :"
0030 MAT A=IDN(5,5)
0040 MAT PRINT A;
0050 PRINT
0060 PRINT "MATRIXDIMENSION 3 x 3 :"
0070 MAT A=IDN(3,3)
0080 MAT PRINT A;
0090 END

END OF LISTING



RUN

MATRIXDIMENSION 5 x 5 :
 1  0  0  0  0
 0  1  0  0  0
 0  0  1  0  0
 0  0  0  1  0
 0  0  0  0  1

MATRIXDIMENSION 3 x 3 :
 1  0  0
 0  1  0
 0  0  1
```

MAT INPUT

ANWEISUNG: MAT INPUT (Matrix Input)

FUNKTION: Zuweisung von Werten an die Elemente einer Matrix während der Ausführung eines Programmes.

FORMAT: MAT INPUT Feld[(num.Ausdr.,num.Ausdr.)]

WIRKUNG: Die Programmausführung wird unterbrochen und auf dem Display erscheint ein Fragezeichen (?). Die Elemente werden nun zeilenweise, durch Komma getrennt, eingegeben.

Am Ende der Eingabe aller Werte über die Tastatur muß die Taste EOL gedrückt werden. Wurden nicht alle Elemente der Matrix mit Werten belegt, erscheinen auf dem Display zwei Fragezeichen und das System wartet auf die restlichen Eingaben. Sind alle Elemente belegt, wird die Ausführung des Programmes fortgesetzt.

Sind die Parameter angegeben, werden die numerischen Ausdrücke berechnet und die Ergebnisse gerundet. Die ganzen Zahlen bestimmen die aktuellen Dimensionen der Matrix und legen somit die Anzahl der einzugebenden Werte fest.

BEMERKUNGEN:
- Die Eingabewerte müssen, entsprechend dem Typ des Feldes, numerisch oder alphanumerisch sein.
- Die Anzahl der Eingabewerte muß gleich der Anzahl Matrixelemente oder gleich dem sich aus den numerischen Ausdrücken ergebenden Produkt der ganzen Zahlen sein.
- Strings müssen in Anführungszeichen stehen, wenn sie ein Komma oder Leerzeichen am Anfang oder Ende enthalten.
- Die Zuweisung der Werte kann entweder durch Eingabe über die Tastatur oder durch Ausführung von Prozeduren mit automatischer Input-Übernahme erfolgen (siehe Kapitel 6.4).

BEISPIEL:

```
FILE


0010 DIM A(6,6)
0020 DISP "INPUT MATRIX A (3×4)";
0030 MAT INPUT A(3,4)
0040 MAT PRINT A;
0050 END

END OF LISTING



RUN

 3  5  6  2
 3  8  6  2
 6  3  5  3
```

MAT ... INV

ANWEISUNG: MAT...INV(Inverse Matrix)

FUNKTION: Bildung der inversen Matrix einer Matrix.

FORMAT: MAT Matrix = INV(Matrix)

WIRKUNG: Die Inverse der Matrix rechts vom Gleichheitszeichen wird gebildet. Die Werte der inversen Matrix werden in der entsprechenden Reihenfolge den Elementen der Matrix links vom Gleichheitszeichen zugewiesen.

BEMERKUNGEN:

- Die zu invertierende Matrix muß in ihren aktuellen Dimensionen quadratisch sein.
- Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen gleich oder größer als jene der Matrix rechts vom Gleichheitszeichen sein. Die Berechnung der inversen Matrix liefert auch den Wert der Determinante. Er wird mit der Standardfunktion DET abgerufen.
- Ist die zu invertierende Matrix (fast) singulär, kann sie nicht invertiert werden. Die Funktion DET liefert dennoch den korrekten Wert der Determinante. Ist eine Matrix singulär, kann es aufgrund von internen Umrechnungen dazu kommen, daß die Elemente der inversen Matrix von den eigentlichen Werten gering abweichen.

BEISPIEL:

```
FILE


0010 DIM A(5,5)
0020 MAT INPUT A
0030 PRINT "MATRIX A:"
0040 MAT PRINT A;
0050 PRINT
0060 MAT A=INV(A)
0070 PRINT "INVERSE VON MATRIX A:"
0080 MAT PRINT USING 90,A
0090 : ##.###  ##.###  ##.###  ##.###  ##.###
0100 PRINT
0110 PRINT "DETERMINANTE =";DET
0120 END

END OF LISTING



RUN

MATRIX A:
 3  2  5  0  1
 1  0  2  5  9
 5  1  5  6  7
 2  0  4  1  3
 6  8  7  4  1

INVERSE VON MATRIX A:
  1.492   0.062   0.542  -1.802  -0.441
  0.424   0.226  -0.119  -0.554   0.034
 -1.042  -0.189  -0.288   1.489   0.297
 -1.483  -0.291  -0.085   1.438   0.381
  0.890   0.308   0.051  -0.929  -0.229

DETERMINANTE = 354
```

MAT PRINT

ANWEISUNG: MAT PRINT (Matrix Print)

FUNKTION: Ausgabe der Werte der Elemente einer oder mehrerer Felder über den Drucker.

FORMAT: MAT PRINT Feldname [{;,}[Feldname{;,}...]]

"," und ";" Trennzeichen für die Ausgabe der Elemente des davorstehenden Feldes.

WIRKUNG: Die Werte der Feldelemente werden zeilenweise im Standardformat gedruckt. Die Position der Elemente in der Druckzeile wird mit den Zeichen "," und ";" bestimmt.

Positionskontrolle in der Druckzeile

Das erste Element jeder Zeile eines Feldes wird immer in der ersten Position einer neuen Druckzeile ausgegeben.

Enthält eine Anweisung MAT PRINT mehr als ein Feld, so müssen die Feldnamen durch "," oder ";" getrennt werden.

Folgt einem Feld ein Komma (,), so werden die Elemente jeder Zeile des Feldes ab Anfang einer der 5 Druckspalten einer Zeile gedruckt.

Folgt dem Feld ein Strichpunkt (;), so werden die Elemente jeder Zeile des Feldes unmittelbar an den bestehenden Ausdruck anschliessend ausgegeben.

Folgt dem letzten Feld einer MAT PRINT-Anweisung kein Trennzeichen, so wird das Komma (,) als Trennzeichen interpretiert.

Die Felder werden entsprechend ihren aktuellen Dimensionen gedruckt.

BEISPIEL:

```
FILE


0010 DIM A(3,4),A$(2,3)
0020 MAT INPUT A
0030 PRINT "MATRIX A (TRENNZEICHEN ','):"
0040 MAT PRINT A,
0050 PRINT
0060 PRINT "MATRIX A (TRENNZEICHEN ';'):"
0070 MAT PRINT A;
0080 PRINT
0090 MAT INPUT A$
0100 PRINT "MATRIX A$ (TRENNZEICHEN ','):"
0110 MAT PRINT A$,
0120 PRINT
0130 PRINT "MATRIX A$ (TRENNZEICHEN ';'):"
0140 MAT PRINT A$;
0150 END

END OF LISTING



RUN

MATRIX A (TRENNZEICHEN ','):
 5                3                4                2
 6                1                1                1
 0                6                5                3

MATRIX A (TRENNZEICHEN ';'):
 5  3  4  2
 6  1  1  1
 0  6  5  3

MATRIX A$ (TRENNZEICHEN ','):
BEISPIEL         FUER             STRING
AUSGABE          VON              TEXT

MATRIX A$ (TRENNZEICHEN ';'):
BEISPIELFUERSTRING
AUSGABEVONTEXT
```

MAT PRINT USING

ANWEISUNG: MAT PRINT USING

FUNKTION: Druck der Elemente eines oder mehrerer Felder in einem vom Benutzer spezifizierten Format.

FORMAT: MAT PRINT USING { Zeilennr. / Stringvar. } ,Feld[,Feld]...

"Zeilennummer" Nummer der Zeile mit der gewählten Formatspezifikation.

"Stringvariable" Formatspezifikation.

WIRKUNG: Die Werte der Elemente jeder Zeile eines Feldes werden in dem durch die Formatspezifikation oder durch den Inhalt der Stringvariablen festgelegten Format ausgegeben. Der Ausdruck erfolgt von links nach rechts in der entsprechenden Reihenfolge, wobei jedem Element jeder Zeile des Feldes ein Formatelement entspricht.

BEMERKUNGEN:

- Mit jeder Zeile des Feldes wird eine neue Druckzeile begonnen. Enthält ein Feld mehr Elemente in einer Zeile, als im Format vorgesehen, so werden die restlichen Werte in der nächsten Zeile in demselben Format ausgegeben.
- Sind dagegen weniger Elemente pro Zeile vorhanden als im Format vorgesehen, so werden die restlichen Formatfelder mit Blanks aufgefüllt.
- Die Werte der Feldelemente müssen im Typ den Formatspezifikationen entsprechen.
- Weitere Detailangaben siehe PRINT USING (Seite 9.99).

BEISPIEL:

```
FILE


0010 DIM A(7,8)
0020 FOR I=1 TO 7 STEP 1
0030 FOR J=1 TO 8 STEP 1
0040 LET A(I,J)=RND*10
0050 NEXT J
0060 NEXT I
0070 REM
0080 REM *** FORMAT FUER 'MAT PRINT USING' ***
0090 REM
0100 : ##.###  ##.###  ##.###  ##.###  ##.###  ##.###  ##.###  ##.###
0110 MAT PRINT USING 100,A
0120 END

END OF LISTING


RUN

  6.393   4.184   9.736   8.865   2.151   0.473   9.871   9.546
  5.571   8.183   0.477   6.868   1.761   0.358   5.030   6.266
  8.883   8.525   7.372   7.006   1.648   2.385   3.533   4.317
  9.532   4.930   0.892   1.993   4.531   8.768   9.619   4.164
  2.804   8.619   3.911   4.159   3.904   2.807   2.344   0.412
  6.057   1.092   8.868   2.171   0.014   7.896   7.603   8.189
  7.623   8.361   3.958   9.790   9.291   4.010   1.962   3.119
```

MAT READ

ANWEISUNG: MAT READ

FUNKTION: Zuweisung von Werten aus dem (mit DATA generierten) internen Datenfile an die Elemente eines Feldes. Das Feld kann sowohl numerisch als auch alphanumerisch sein.

FORMAT: MAT READ Feld[(num.Ausdr.,num.Ausdr.)][,Feld[(num.Ausdr., num.Ausdr.)]]...

WIRKUNG: Durch die Ausführung der Anweisung MAT READ werden die Werte aus dem internen File der Reihe nach, beginnend ab der durch den Pointer bezeichneten Position, den Feldelementen der in der MAT READ-Anweisung angegebenen Feldern zugewiesen.

Bei MAT READ (ohne Parameter) bestimmen die deklarierten Dimensionen die Grösse des Feldes.

Sind Parameter angegeben, werden die numerischen Ausdrücke berechnet und gegebenenfalls auf die nächste ganze Zahl gerundet. Diese Zahlen sind die aktuellen Dimensionen des Feldes.

Den Feldelementen werden die Werte aus dem File zeilenweise zugewiesen.

Nach jeder Wertzuweisung zeigt der Pointer auf das nächste Fileelement.

BEMERKUNGEN:

- Jedem Element der Matrix muss ein Wert in der internen Tabelle entsprechen.
- Die Werte des internen Files müssen dem Typ nach dem Feld entsprechen (numerisch oder alphanumerisch).
- Strings in einer DATA-Anweisung müssen in Anführungszeichen stehen, wenn sie ein Komma (,) oder Leerzeichen am Anfang oder Ende enthalten.

BEISPIEL:

```
FILE


0010 DIM A(3,2),B(2,3),A$(2,5)
0020 MAT READ A
0030 RESTORE
0040 MAT READ B,A$
0050 PRINT "MATRIX A:"
0060 MAT PRINT A;
0070 PRINT
0080 PRINT "MATRIX B:"
0090 MAT PRINT B;
0100 PRINT
0110 PRINT "MATRIX A$:"
0120 MAT PRINT A$;
0130 PRINT
0140 DATA 10,20,30,40,50,60
0150 DATA "HANS ","UND ","LIESE ","GEHEN ",SPAZIEREN.,"PAUL ","UND ","LOTTE "
0160 DATA SCHWIM,MEN IM SEE.
0170 END

END OF LISTING



RUN

MATRIX A:
 10  20
 30  40
 50  60

MATRIX B:
 10  20  30
 40  50  60

MATRIX A$:
HANS UND LIESE GEHEN SPAZIEREN.
PAUL UND LOTTE SCHWIMMEN IM SEE.
```

MAT READ:

ANWEISUNG: MAT READ:

FUNKTION: Zuweisung von Werten aus externen Datenfiles an ein oder mehrere Felder.

FORMAT: MAT READ: filedesignator,Feld[(n.A.,n.A.)][,Feld[(n.A.,n.A)]]
...[EOF Zeilennr.]

"file-designator"	arithmetischer Ausdruck.
"n.A."	numerischer Ausdruck.

WIRKUNG: Es gelten die gleichen Regeln wie für die Anweisung READ:. Die aus dem Datenfile gelesenen Daten werden den Elementen der angegebenen Matrix zeilenweise zugewiesen. Sind nach dem Namen der Matrix numerische Ausdrücke angegeben, so erhält die Matrix diese Werte als aktuelle Dimensionen.

BEMERKUNG: Das Produkt der Dimensionen der Matrix darf die Anzahl der im Datenfile vorhandenen Daten nicht überschreiten.

BEISPIEL:

```
FILE


0010 FILES SFILE
0020 DIM A(3,4),B(5,6)
0030 RESTORE :1
0040 PRINT "MATRIX A (3x4):"
0050 MAT READ :1,A
0060 MAT PRINT A;
0070 PRINT
0080 RESTORE :1
0090 PRINT "MATRIX B (5x6):"
0100 MAT READ :1,B EOF 130
0110 MAT PRINT B;
0120 GOTO 160
0130 PRINT "(MATRIX UNVOLLSTAENDIG)"
0140 PRINT
0150 GOTO 110
0160 END

END OF LISTING



RUN
MATRIX A (3x4):
 61  43  87  80
 27  13  88  86
 54  75  13  64

MATRIX B (5x6):
(MATRIX UNVOLLSTAENDIG)

 61  43  87  80  27  13
 88  86  54  75  13  64
 24  12  50  60  81  78
 68  66  0  0  0  0
 0  0  0  0  0  0
```

MAT...TRN

ANWEISUNG: MAT...TRN(Transponierte Matrix)

FUNKTION: Berechnung der Transponierten einer Matrix (Vertauschen von Zeilen und Spalten).

FORMAT: MAT Matrix = TRN(Matrix)

WIRKUNG: Die Elemente jeder Zeile der Matrix auf der rechten Seite werden spaltenweise der Matrix links vom Gleichheitszeichen zugewiesen.

Die Werte der Spalte x der Matrix rechts vom Gleichheitszeichen stimmen mit den Werten der Zeile x der linken Matrix überein; ebenso korrespondieren die Werte der Zeile y der rechten Matrix mit den Werten der Spalte y der linken Matrix.

BEMERKUNG: Die deklarierten Dimensionen der Matrix links vom Gleichheitszeichen müssen größer oder gleich den aktuellen Dimensionen der rechten Matrix sein. Die beiden Matrizen dürfen nicht den gleichen Namen haben.

BEISPIEL:

```
FILE


0010 DIM A(3,4),B(4,3)
0020 FOR I=1 TO 3 STEP 1
0030 FOR K=1 TO 4 STEP 1
0040 LET A(I,K)=I*10+K
0050 NEXT K
0060 NEXT I
0070 PRINT "MATRIX A:"
0080 MAT PRINT A;
0090 PRINT
0100 MAT B=TRN(A)
0110 PRINT "MATRIX B =TRANSPONIERTE VON A:"
0120 MAT PRINT B;
0130 END

END OF LISTING



RUN

MATRIX A:
 11  12  13  14
 21  22  23  24
 31  32  33  34

MATRIX B =TRANSPONIERTE VON A:
 11  21  31
 12  22  32
 13  23  33
 14  24  34
```

MAT WRITE:

ANWEISUNG: MAT WRITE:

FUNKTION: Schreiben der Elemente der angeführten Matrizen auf das durch den Filedesignator bestimmte externe Datenfile.

FORMAT: MAT WRITE: filedesignator,Feld[,Feld]...[EOF Zeilennr.]

"file-
designator" arithmetischer Ausdruck.

WIRKUNG: Es gelten die gleichen Regeln wie für die Anweisung WRITE:.

Die Elemente der angegebenen Matrizen werden zeilenweise ins Datenfile geschrieben.

BEISPIEL:

```
FILE

0010 FILES SFILE
0015 SCRATCH :1
0020 DIM A(3,4)
0030 FOR I=1 TO 3 STEP 1
0040 FOR K=1 TO 4 STEP 1
0050 LET A(I,K)=10*I+K
0060 NEXT K
0070 NEXT I
0080 PRINT "MATRIX A  VOLLSTAENDIG AUFS FILE GESCHRIEBEN:"
0090 MAT PRINT A;
0100 PRINT
0110 REM
0120 MAT WRITE :1,A
0130 REM
0140 END

END OF LISTING


RUN

MATRIX A  VOLLSTAENDIG AUFS FILE GESCHRIEBEN:
 11  12  13  14
 21  22  23  24
 31  32  33  34
```

ANWEISUNG: MAT...ZER(Null Matrix)

FUNKTION: Setzen aller Elemente einer numerischen Matrix auf Null.

FORMAT: MAT Matrix = ZER[(num.Ausdruck,num.Ausdruck)]

WIRKUNG: Jedem Element der angegebenen Matrix wird der Wert Null zugewiesen. Bei Angabe der Parameter werden die numerischen Ausdrücke berechnet und die Ergebnisse auf die nächste ganze Zahl gerundet. Sie bilden die aktuellen Dimensionen der Matrix.

BEMERKUNG: Die deklarierten Dimensionen der Matrix müssen größer oder gleich den angegebenen aktuellen Dimensionen sein.

BEISPIEL:

```
FILE


0010 DIM A(4,5)
0020 FOR I=1 TO 4 STEP 1
0030 FOR J=1 TO 5 STEP 1
0040 LET A(I,J)=10*I+J
0050 NEXT J
0060 NEXT I
0070 PRINT
0080 PRINT "MATRIX A (4x5):"
0090 MAT PRINT A;
0100 PRINT
0110 MAT A=ZER(3,3)
0120 PRINT "REDUZIERTE MATRIX A=NULL (3x3):"
0130 MAT PRINT A;
0140 END

END OF LISTING



RUN

MATRIX A (4x5):
 11  12  13  14  15
 21  22  23  24  25
 31  32  33  34  35
 41  42  43  44  45

REDUZIERTE MATRIX A=NULL (3x3):
 0  0  0
 0  0  0
 0  0  0
```

9.4 I/O-Kanal TASTATUR

Wie in den vorangehenden Kapiteln beschrieben, kann die Tastatur des Systems während eines Programmablaufes zur Dateneingabe (INPUT- oder RKB-Anweisung), aber auch zur Abfrage von Variablenwerten und für Wertzuweisungen an Variable (Debugging-Mode) verwendet werden.

Daneben kann die Tastatur von einem BASIC-Programm wie eine externe Peripherie-Einheit angesprochen werden, was folgendes ermöglicht:

- Die Dateneingabe kann während des Programmablaufes überlappend erfolgen, das heißt, die Programmausführung wird nicht, wie bei einer INPUT- oder RKB-Anweisung, unterbrochen.
- Das BASIC-Programm kann Strings an den Tastaturbuffer senden. Diese Strings können modifiziert und anschließend wie eine normale Eingabe ans Programm zurückgegeben werden.

9.4.1 BASIC-Anweisungen für den Kanal TASTATUR

Folgende BASIC-Anweisungen können für den Tastatur-Kanal verwendet werden:

BUFFER # Peripherie-Einheit, Buffergröße

CMD # Peripherie-Einheit, Steuerbefehl [AND GO]

RECEIVE # Peripherie-Einheit, Stringvariable [AND GO]

SEND # Peripherie-Einheit, Stringausdruck [AND GO]

TEST # Peripherie-Einheit

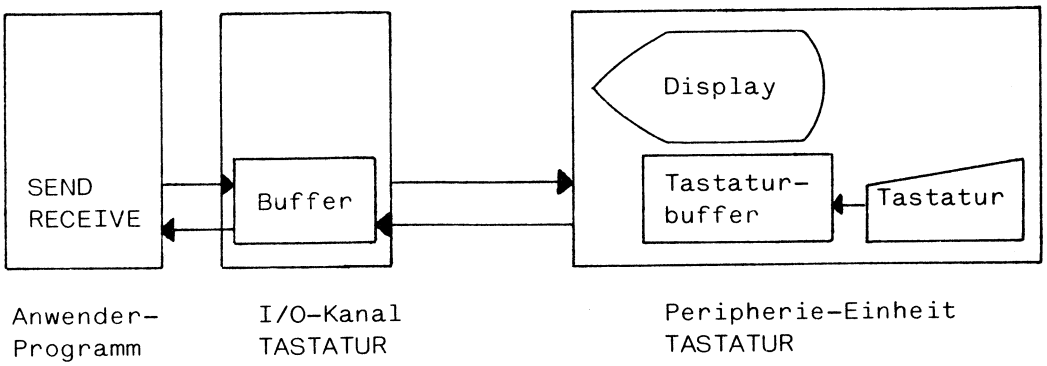
WAIT # Peripherie-Einheit

Ferner kann die Funktion "IOC(num.Ausdruck)" verwendet werden.

Bei Ausführung einer RECEIVE-Anweisung gibt das System die Tastatur für die Dateneingabe frei. Die eingetasteten Zeichen werden im Tastaturbuffer gespeichert und im Display angezeigt. Nach Drücken der Taste END OF LINE wird der eingegebene String vom Tastaturbuffer in den Buffer des Tastaturkanales übertragen und von dort aus der in der RECEIVE-Anweisung aufgeführten Stringvariablen zugewiesen.

Bei Ausführung einer SEND-Anweisung wird der darin angeführte String in den durch die BUFFER-Anweisung definierten Buffer des Tastaturkanales geladen und von dort in den Tastaturbuffer übertragen. Der String kann durch eine CMD-Anweisung im Display sichtbar gemacht werden.

Der String kann modifiziert und über eine INPUT-, RKB- oder RECEIVE-Anweisung ans Programm zurückgegeben werden.



Tastatur des Systems als Peripherie-Einheit

Der I/O-Kanal TASTATUR verlangt folgende Werte als Adresse "Peripherie-Einheit":

INPUT	OUTPUT
32-39	40-47

Unter diesen Adressen kann eine beliebige gewählt werden, welche jedoch anschließend konsequent beibehalten werden sollte. Wird beispielsweise in einer RECEIVE-/AND GO-Anweisung die Adresse 32 bestimmt, muß eine nachfolgende TEST- oder WAIT-Anweisung dieselbe Adresse aufweisen, da für jede der Adressen (von 32-47) ein eigenes Zustandsregister eingerichtet wird. Statusmeldungen werden nur in dem der Adresse der Anweisung entsprechenden Register abgelegt. Steht also in der TEST-Anweisung eine andere Adresse, wird ein anderes Zustandsregister getestet, so daß die erhaltene Information irrelevant ist.

Für die einzelnen BASIC-Anweisungen gilt bei Verwendung der Tastatur als periphere Einheit folgendes:

BUFFER — Als Adresse "Peripherie-Einheit" kann eine beliebige Zahl zwischen 32 und 47 gewählt werden. Die Buffergröße sollte 80 Bytes nicht überschreiten, da auch der Tastaturbuffer nur 80 Zeichen aufnehmen kann.

CMD

Folgende Steuerbefehle sind gültig:

Steuerbefehl	Wirkung
0	Ohne Wirkung.
1	Inhalt des Tastaturbuffers wird im Display angezeigt.
16	Desaktiviert die Tastatur als periphere Einheit.
2-15 17-31	Diese Codes werden nicht benützt.

Die zugehörige Adresse "Peripherie-Einheit" liegt zwischen 40 und 47.

RECEIVE

Fehlt "AND GO", wird die Programmausführung wie bei einer INPUT- oder RKB-Anweisung unterbrochen und das System wartet auf die Eingabe eines Strings durch den Benutzer. Die Konsollampe ON LINE leuchtet. Sie erlischt, wenn der Benutzer die Taste END OF LINE drückt. Der nun im Tastaturbuffer vorhandene String wird in den Buffer des I/O-Kanales übertragen und von dort aus der Stringvariablen zugewiesen. Die Programmausführung wird fortgesetzt.

Ist "AND GO" angeführt, wird die Tastatur für die Eingabe aktiviert und die Konsollampe ON LINE leuchtet auf. Die Programmausführung wird jedoch nicht unterbrochen, sondern sofort das nächste Statement verarbeitet. Nach Drücken der Taste END OF LINE erlischt die Konsollampe ON LINE und der String wird in den Buffer des I/O-Kanales übertragen. Er wird nicht sofort der in der Anweisung angegebenen Stringvariablen zugewiesen, sondern erst, wenn eine weitere, sich auf den Kanal TASTATUR beziehende I/O-Anweisung auszuführen ist, zum Beispiel, wenn eine Anweisung CMD mit dem Steuerbefehl 0 folgt.

Für die Adresse "Peripherie-Einheit" sind Werte zwischen 32 und 39 zulässig.

Es kann auch ein Leerstring eingegeben werden (Drücken der Taste END OF LINE, ohne vorher ein Zeichen einzugeben).

Werden über die Tastatur mehr Zeichen eingegeben, als der Parameter "Buffergröße" in der Anweisung BUFFER zuläßt, wird ein Leerstring in den Buffer des I/O-Kanales geschrieben, das heißt, die eingegebenen Zeichen sind verloren.

SEND — Der Wert des Stringausdruckes "Stringausdr." (im folgenden kurz "String" genannt) wird in den durch die Anweisung BUFFER definierten Buffer geschrieben. Von dort wird er in den Tastaturbuffer übertragen. Unterscheidet sich der String vom Leerstring, werden dadurch sich im Tastaturbuffer befindliche Zeichen überschrieben. Der String wird im Display nicht angezeigt, außer wenn der SEND-Anweisung eine CMD-Anweisung mit dem Steuerbefehl 1 folgt.

Ist dies der Fall, wird der String im Display visualisiert. Der Pointer befindet sich hinter dem letzten Zeichen des Strings, der nun mittels der Editing-Tasten (← → CHAR DELETE) der Tastatur modifiziert werden kann. Ferner können weitere Zeichen bis zu einer Gesamtlänge von 80 Zeichen ein- oder angefügt werden.

Um den so erhaltenen String wieder einer Stringvariablen zuzuweisen, muß eine RECEIVE-, INPUT-, MAT INPUT- oder RKB-Anweisung folgen.

Ist die Länge des Wertes des Stringausdruckes in der SEND-Anweisung größer als 80 Zeichen, werden nur die ersten 80 Zeichen in den Tastaturbuffer geladen.

Für die Adresse "Peripherie-Einheit" sind Werte zwischen 40 und 47 zulässig.

Ist der Wert des Stringausdruckes "Stringausdr." ein Leerstring, bleibt der Inhalt des Tastaturbuffers erhalten.

TEST — Der Inhalt des Zustandsregisters wird ins Arbeitsregister übertragen, ohne das Ende einer laufenden I/O-Operation abzuwarten.

WAIT — Der Inhalt des Zustandsregisters wird nach Abschluß einer laufenden I/O-Operation ins Arbeitsregister übertragen.

Für die IOC-Funktion sind nur die Argumente 7 und 8 relevant. Dabei gilt:

IOC(8)=1
IOC(7)=1 } Bei Ausführung einer RECEIVE AND GO-Anweisung ist die Eingabe noch nicht durch END OF LINE abgeschlossen.

Sonst gilt: IOC(8)=IOC(7)=0.

Da bei Ausführung einer RECEIVE-Anweisung ohne AND GO die Programmausführung unterbrochen wird, tritt keine Konfliktsituation im Zusammenhang mit Fehlermeldungen durch das System oder mit INPUT- bzw. RKB-Anweisungen auf.

Bei Ausführung einer RECEIVE AND GO-Anweisung, die überlappend erfolgt, also die Programmausführung fortgesetzt wird, kann jedoch der Fall eintreten, daß INPUT- bzw. RKB-Anweisungen auszuführen sind oder das System in den Debugging-Mode geht, bevor die Eingabe durch END OF LINE abgeschlossen ist. Für diese Fälle gilt folgende Regelung:

- Übergang in den Debugging-Mode:

  Die Signallampe READY leuchtet, ohne zu blinken. Am Bildschirm erscheint die Meldung "Step in Line...". Der Inhalt des Tastaturbuffers bleibt unverändert. Sollen Variablenwerte abgefragt oder Wertzuweisungen an Variable durchgeführt werden, muß der Tastaturbuffer durch Drücken der Tasten SHIFT und CLEAR geleert werden. Dadurch werden aber auch die bereits eingegebenen Zeichen gelöscht.

  Wird der Debugging-Mode verlassen und die Programmausführung fortgesetzt (durch Drücken von SHIFT EXIT), beginnt die Konsoltaste RUNNING wieder zu blinken und die Eingabe kann fortgesetzt werden. Wurde der Inhalt des Tastaturbuffers gelöscht, muß der String vollständig neu eingegeben werden.

- INPUT- bzw. RKB-Anweisung:

  Die Lampe READY leuchtet, ohne zu blinken.

  Die INPUT-, MAT INPUT- und RKB-Anweisungen haben Vorrang gegenüber der RECEIVE-Anweisung, das heißt, daß gegebenenfalls der Tastaturbuffer gelöscht und die Werte für die Variablen in der INPUT-, MAT INPUT- bzw. RKB-Anweisung eingegeben werden müssen.

  Sind diese Eingaben durchgeführt, beginnt die READY-Lampe wieder zu blinken. Nun können die Zeichen für die in der RECEIVE-Anweisung aufgeführten Stringvariablen erneut eingegeben werden.

In der Tabelle sind die Statusanzeigen zusammenfassend dargestellt:

Konsollampe	Status
RUNNING	
konstant	INPUT, MAT INPUT oder RKB ohne RECEIVE
konstant	Debugging-Mode ohne RECEIVE
blinkt	RECEIVE mit AND GO
konstant	RECEIVE ohne AND GO oder INPUT, MAT INPUT, RKB und RECEIVE mit AND GO
konstant	Debugging-Mode und RECEIVE mit AND GO

9.4.2 Bemerkungen

- Soll parallel zur Anweisung RECEIVE eine Display-Anzeige als Inputanforderung gegeben werden, darf sie nicht mit Strichpunkt abgeschlossen sein, da RECEIVE kein Fragezeichen erzeugt.
- Die Abfrage, ob eine Eingabe bereits durchgeführt wurde, kann nach RECEIVE...AND GO mit der Anweisung TEST und nachfolgender Abfrage auf IOC(7)=0 erfolgen. Ist IOC(7)=1, wurde END OF LINE noch nicht gedrückt.

  Ist die Eingabe für die Weiterverarbeitung bereits erforderlich, kann mit WAIT auf den Abschluß der Eingabe gewartet werden.
- Soll eine nur während eines Teiles des Programmes zulässige Eingabemöglichkeit (z.B. Unterbrechungsmöglichkeit) wieder aufgehoben werden, so ist CMD 40,16 auszuführen. Die Taste END OF LINE wird nicht akzeptiert.
- Ist bei END oder CHAIN die Eingabe für eine RECEIVE...AND GO-Anweisung nicht abgeschlossen, wird die Vereinbarung des Tastaturkanales gelöscht und das Programm beendet. Der Inhalt des Tastaturbuffers bleibt jedoch erhalten.
- Alternativ zu den Möglichkeiten der Abfrage auf vollständige Eingabe mit TEST und WAIT kann der externe Interrupt durch INTERRUPT ENABLE behandelt werden. Die Beschreibung befindet sich in Kapitel 9.3.4.

9.4.3 BASIC-Anweisungen

Im folgenden Kapitel sind die BASIC-Anweisungen für die Programmierung von peripheren Einheiten, in alphabetischer Reihenfolge geordnet, für den Tastaturkanal beschrieben. Die ausführliche Beschreibung ist im Handbuch "Peripherie und Datenfernübertragung" enthalten.

BUFFER#

ANWEISUNG: BUFFER #

FUNKTION: Reservierung eines Buffers (temporärer Speicher) im Arbeitsspeicher für den Datenaustausch mit einer peripheren Einheit über einen I/O-Kanal.

FORMAT: BUFFER # per.Einh., Buffergröße

"per.Einh." ganze Zahl zwischen 0 und 255; für den Tastaturkanal: 32 - 47.

"Buffergröße" ganze Zahl zwischen 1 und 32767 (Bytes); für den Tastaturkanal: max. 80.

WIRKUNG: Für den I/O-Kanal, an den die periphere Einheit mit der Adresse "per.Einh." angeschlossen ist, wird im Arbeitsspeicher ein Buffer für den Datenaustausch eingerichtet. Die Größe (in Bytes) wird durch den Operanden "Buffergröße" festgelegt.

BEMERKUNGEN:
- Die Anweisung BUFFER ist eine nicht ausführbare Anweisung. Sie kann an einer beliebigen Stelle des Programmes stehen (analog einer DIM- oder DCL-Anweisung).
- Enthält ein Programm mehrere, sich auf denselben Kanal beziehende BUFFER-Anweisungen, wird der Buffer durch die höchste in den Anweisungen angegebene Größe festgelegt.
- Soll eine RECEIVE-Anweisung ausgeführt werden, darf die deklarierte Länge der den Bufferinhalt aufnehmenden String-Variablen nicht größer sein als die Kapazität des dem entsprechenden I/O-Kanal zugewiesenen Buffers.

BEISPIEL: Im folgenden Beispiel wird dem Tastaturkanal ein Buffer von 80 Bytes zugewiesen.

```
0010 BUFFER#32,80
```

ANWEISUNG: CMD # (command)

FUNKTION: Senden eines oder mehrerer Steuer- und/oder Prüfbefehle an eine periphere Einheit.

FORMAT: CMD # per.Einh.,Steuerbef.[,Steuerbef.]...[AND GO]

"per.Einh."	Adresse der Peripherie. Der (gerundete) Wert muß ein numerischer Ausdruck mit einem Wert zwischen 0 und 255 (einschließlich) sein. Beim Tastaturkanal liegt "per.Einh." zwischen 40 und 47.
"Steuerbef."	numerischer Ausdruck, der für einen Steuerbefehl für die periphere Einheit steht.
"AND GO"	legt fest, daß der Steuerbefehl (wenn mehrere Steuerbefehle gesendet werden, der letzte in der Reihe) überlappend mit anderen Operationen der CPU ausgeführt wird.

WIRKUNG:

- Fehlt "AND GO", werden die durch die Operanden "Steuerbef." spezifizierten Steuerbefehle ausgeführt.

  Ins Zustandsregister der Peripherie "per.Einh." werden die den Status der Peripherie angebenden Informationen übertragen. Treten während der Übertragung der Steuerbefehle Fehler auf, werden diese vom System angezeigt.

- Ist "AND GO" angegeben, wartet das System das Ende der Übertragung und die Rückmeldung der Peripherie nicht ab, sondern setzt die Programmausführung sofort mit der nächsten Anweisung fort.

  Eventuelle Fehlermeldungen werden erst angezeigt, wenn dieselbe Peripherie erneut angesprochen wird.

RECEIVE

ANWEISUNG: RECEIVE #

FUNKTION: Abruf von Daten durch die Zentraleinheit von einer peripheren Einheit.

FORMAT: RECEIVE # per.Einh.,Stringvar.[AND GO]

"per.Einh."	numerischer Ausdruck, Adresse der Peripherie; zulässig für Tastatur: 32-39.
"Stringvar."	Stringvariable.
"AND GO"	legt fest, daß die Operation "RECEIVE" überlappend mit anderen Operationen der Zentraleinheit ausgeführt wird.

WIRKUNG:

- Fehlt "AND GO", wird von der peripheren Einheit mit der Adresse "per.Einh." ein Datensatz (String) in den Buffer des von ihr besetzten I/O-Kanales übertragen. Der Inhalt des Buffers wird dann der Stringvariablen "Stringvar." zugewiesen. Eventuell auftretende Fehler bei der Übertragung werden vom System angezeigt.
- Ist "AND GO" angeführt, wird die Übertragung eines Strings in den I/O-Buffer initialisiert. Das Ende der Übertragung wird nicht abgewartet.

  Der im Buffer stehende String wird der Stringvariablen "Stringvar." erst zugewiesen, wenn die nächste, sich auf denselben I/O-Kanal beziehende I/O-Operation ausgeführt wird. Eventuell auftretende Fehler werden vom System erst angezeigt, wenn eine sich auf dieselbe periphere Einheit beziehende I/O-Operation ausgeführt wird.

BEMERKUNGEN:

- Ist "per.Einh." keine ganze Zahl, wird der Wert gerundet.
- Die deklarierte Länge (in Bytes) der Stringvariablen "Stringvar." darf nicht größer sein als der Buffer des I/O-Kanales, an den die periphere Einheit "per.Einh." angeschlossen ist.

SEND#

ANWEISUNG: SEND #

FUNKTION: Senden eines Strings aus dem Arbeitsspeicher an eine periphere Einheit.

FORMAT: SEND # per.Einh.,Stringausdr.[AND GO]

"per.Einh."	numerischer Ausdruck, Adresse der Peripherie; zulässig für den Tastaturkanal: 40-47.
"Stringausdr."	Stringausdruck.
"AND GO"	legt fest, daß die Übertragung überlappend mit anderen Operationen der Zentraleinheit ausgeführt wird.

WIRKUNG:

- Fehlt "AND GO", wird der Resultatstring von "Stringausdr." in den Buffer des entsprechenden I/O-Kanales geladen und an die periphere Einheit namens "per.Einh." übertragen. Eventuell bei der Datenübertragung auftretende Fehler werden vom System angezeigt.
- Ist "AND GO" in der Anweisung angeführt, wird die Übertragung des Strings in den I/O-Buffer initialisiert und die Programmausführung mit der nächsten Anweisung fortgesetzt.

  Eine weitere, sich auf denselben I/O-Kanal beziehende I/O-Operation wird erst ausgeführt, wenn die Peripherie wieder frei ist.

  Eventuell bei der Uebertragung auftretende Fehler werden vom System angezeigt, sobald eine I/O-Operation mit derselben peripheren Einheit "per.Einh." auszuführen ist.

BEMERKUNGEN:

- Ist "per.Einh." keine ganze Zahl, wird der Wert gerundet.
- Die Anzahl Zeichen des Strings (aktuelle Länge), aber auch das Ergebnis des Stringausdruckes, darf nicht größer sein als die Kapazität des Buffers des I/O-Kanales, an den die periphere Einheit "per.Einh." angeschlossen ist.

ANWEISUNG: TEST #

FUNKTION: Der Inhalt des Zustandsregisters der peripheren Einheit wird ins Arbeitsregister übertragen, ohne daß das Ende einer eventuell laufenden I/O-Operation mit dieser Peripherie abgewartet wird.

FORMAT: TEST # per.Einh.

"per.Einh." numerischer Ausdruck, Adresse der Peripherie; für den Tastaturkanal: 32-39 oder 40-47.

WIRKUNG: Die Information im Zustandsregister der peripheren Einheit "per.Einh." wird ins Arbeitsregister übertragen.

BEMERKUNGEN:
- Hat "per.Einh." keinen ganzzahligen Wert, wird dieser auf die nächste ganze Zahl gerundet.
- Der Inhalt des Arbeitsregisters kann durch die Funktion IOC (num.Ausdr.) abgefragt werden.

WAIT

ANWEISUNG: WAIT #

FUNKTION: Warten, bis eine laufende I/O-Operation mit der angesprochenen Peripherie beendet ist. Der Inhalt des zur peripheren Einheit gehörenden Zustandsregisters wird vom I/O-Kanal ins Arbeitsregister übertragen.

FORMAT: WAIT # per.Einh.

"per.Einh." numerischer Ausdruck, Adresse der Peripherie; für den Tastaturkanal: 32-39 oder 40-47.

WIRKUNG: Die im Zustandsregister der peripheren Einheit "per.Einh." stehende Information wird ins Arbeitsregister übertragen.

BEMERKUNGEN:
- Ist "per.Einh." keine ganze Zahl, wird auf die nächste ganze Zahl gerundet.
- Das Arbeitsregister kann durch die Funktion IOC (num.Ausdr.) abgefragt werden.

9.4.4 Beispiele

Beispiel 1:

Im ersten Programmbeispiel wird eine einfache Routine zur überlappten Eingabe von Datensätzen vorgestellt. Der Unterschied gegenüber einer Eingabe über eine INPUT- oder RKB-Anweisung besteht darin, daß das System kein "?" als Aufforderung zur Eingabe ausgibt und dadurch die Eingabe "kontinuierlich" erfolgen kann.

```
FILE

0010 REM *** DIE TASTATUR ALS PERIPHERE EINHEIT ***
0020 REM
0030 BUFFER @32,80
0040 DCL 80T$
0050 FILES FILE1
0060 SCRATCH :1
0070 RECEIVE @32,T$ AND GO
0080 FOR I=1 TO 10 STEP 1
0090 RECEIVE @32,T$ AND GO
0100 WRITE :1,T$
0110 NEXT I
0120 END

END OF LISTING
```

Beispiel 2:

Im zweiten Beispiel wird eine Routine zur Korrektur gespeicherter Datensätze beschrieben. Die zu ändernden Datensätze müssen dabei nicht neu eingegeben werden, denn die Korrekturen sind mittels des Pointers im Display durchführbar.

```
FILE     SEND

0010 REM *** DIE TASTATUR ALS PERIPHERE EINHEIT ***
0020 REM
0030 REM *** KORREKTUR VON DATEN, DIE IM EINEM FILE GESPEICHERT SIND ***
0040 REM
0050 BUFFER #32,80
0060 FILES FILE2
0070 DCL 32(A$,B$)
0080 PRINT "ALTER NAME:",,"NEUER NAME:"
0090 PRINT
0100 FOR I=1 TO 10 STEP 1
0110 SETW :1 TO (I-1)*9+1
0120 READ :1,A$
0130 SEND #40,A$
0140 CMD #40,1
0150 PRINT A$;
0160 RECEIVE #32,B$
0170 SETW :1 TO (I-1)*9+1
0180 WRITE :1,B$
0190 PRINT ,,B$
0200 NEXT I
0210 END

END OF LISTING


RUN SEND

ALTER NAME:                    NEUER NAME:

PAUL MAYER                     PAUL MEYER
ESTHER LECHNER                 ESTHER LECHNER
ANNA RAPPOLD                   ANNA RAPPOLDI
ERNST STEINBACH                ERNST STEINBACH
CARL LAUER                     KARL LAUER
ELSA FISCHER                   ELSA FISCHBACHER
BILL BRAUN                     BILL BROWN
ALPHONS LATTMAN                ALPHONS LATTMAN
PETER WOECKEL                  PETER WOECKEL
PAUL KRIPS                     PAUL KNIRPS
```

9.5 Graphisches Arbeiten durch Option 'Plot' und 'Graphik-Display' (OPT PLO oder OPT GDI)

Für das graphische Arbeiten mit dem System gibt es zwei Optionen: Option 'Plot' und Option 'Graphik-Display'. Die normale Arbeitsweise ist sicher die mit dem graphischen Schirm (Option 'Graphik-Display'). Dabei wird eine Graphik am Bildschirm dargestellt. Nachdem die Zeichnung vollständig ist, kann per Programmbefehl oder Systembefehl eine Hardcopy des Bildschirminhaltes auf dem angeschlossenen Thermodrucker gemacht werden. Die Zeichnung auf dem Thermopapier hat eine Größe von ca. 20 x 14 cm.

Durch Option PLO besteht die Möglichkeit, längere Zeichnungen als 14 cm auszuplotten. Bei dieser Arbeitsweise ist jedoch eine Darstellung der Zeichnung auf den Bildschirm nicht möglich (es kann entweder die Option GDI oder die Option PLO geladen werden).

Bei der Arbeitsweise mit OPT GDI können zu den graphischen Darstellungen auf dem Schirm gleichzeitig alphanumerische Zeichen (Texte) dargestellt werden. Diese Zeichen werden über den BASIC-Befehl DISP 0 gesteuert, der in Abschnitt 9.5.5 beschrieben ist. Dabei können in 39 Zeilen jeweils bis zu 80 Zeichen ausgegeben werden. Man beachte, daß es über den alphanumerischen Controler möglich ist, über DISP 0 (Abschnitt 9.1) 23 Zeilen à 80 Zeichen anzusprechen.

Zu den beiden Bildschirmformaten siehe BASIC-Anweisung DISP 0 (Kapitel 9.1).

9.5.1 Grundlagen des Plottens

In diesem Kapitel wird anhand eines einfachen BASIC-Programmes die Erstellung einer Zeichnung erklärt. Die für die graphische Darstellung notwendigen Anweisungen sind mit einer Nummer versehen und werden im folgenden genauer beschrieben.

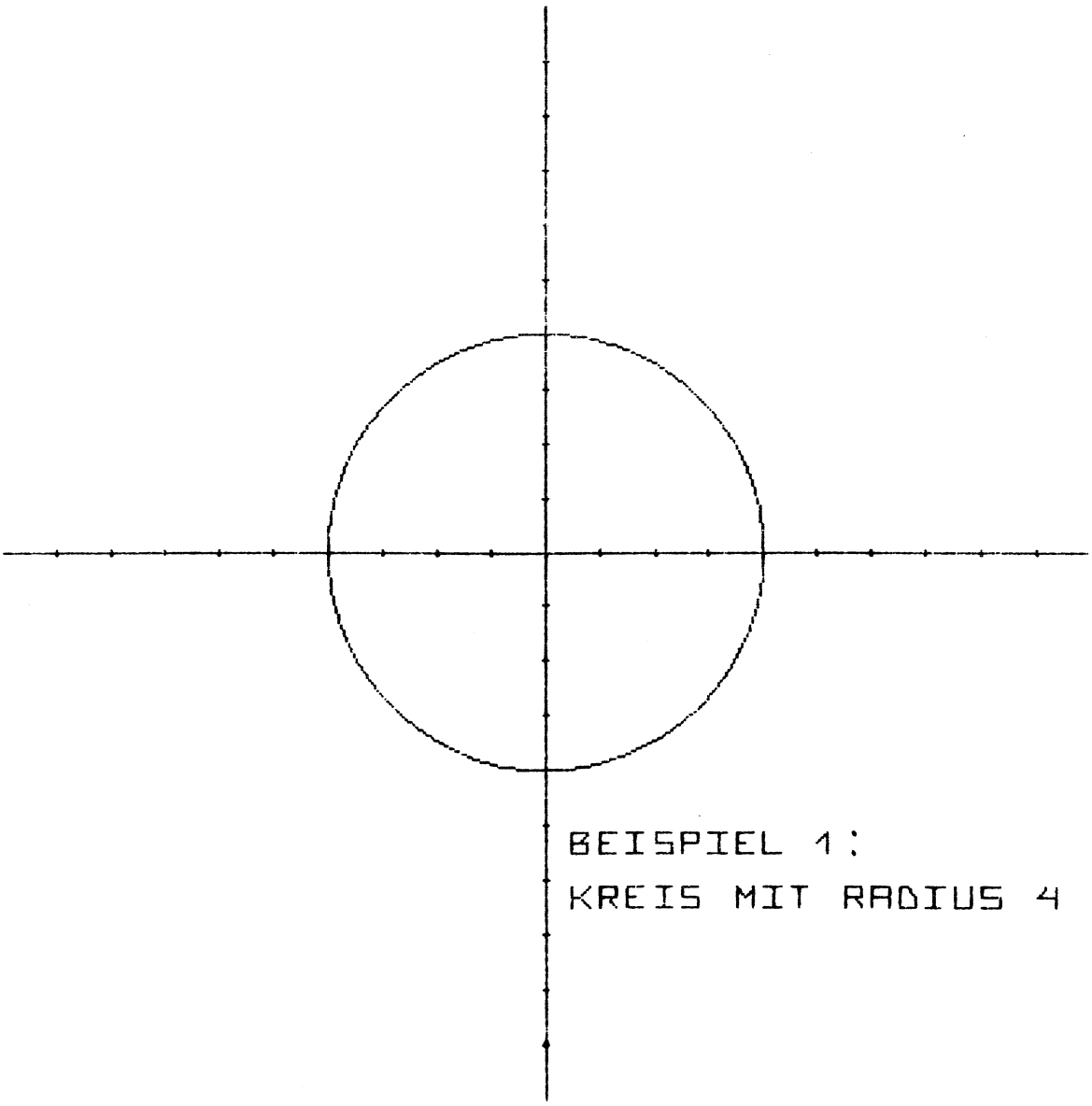
Das Beispielprogramm plottet einen Kreis, die x- und y-Achse des Koordinatensystems sowie eine Beschriftung der Zeichnung. Es ist in dieser Form sowohl für den Thermodrucker als auch für den Bildschirm verwendbar.

Abweichungen in der Wirkung bei einigen Plotanweisungen für Thermodrucker bzw. Bildschirm sind in Abschnitt 9.5.5 angeführt.

```
FILE    *KREIS

0010 INIMAGE BEISP,2
0020 FRAME 6,6
0030 SCALE -10,10,-10,10
0040 XAXIS 0,1
0050 YAXIS 0,1
0060 MOVE 2,0
0070 FOR I=0 TO 2*PI STEP PI/30
0080 PLOT 4*COS(I),4*SIN(I)
0090 NEXT I
0100 MOVE .5,-5.5
0110 CSIZE .15,.18,0
0120 CPLOT "BEISPIEL 1:"
0130 MOVE .5,-6.5
0140 CPLOT "KREIS MIT RADIUS 4"
0150 DRAW
0160 END

END OF LISTING
```



(1) Die Anweisung INIMAGE bewirkt die Vorbereitung für eine neue Zeichnung.

Beim Plotten mit dem Thermodrucker reserviert die Anweisung INIMAGE im Arbeitsspeicher einen Buffer mit vorgegebener Größe und bestimmt ein (sequentielles) Datenfile in einer Bibliothek zur Speicherung der Zeichnung. Dies ist durch die Arbeitsweise des Thermodruckers bedingt:

Durch die vorgegebene Vorschubrichtung des Papiers ist eine unmittelbare Ausgabe der Elemente einer Zeichnung nicht möglich. Es wird daher mit Hilfe eines Buffers und des Datenfiles das Bild zuerst komplett generiert und abgespeichert und nach seiner internen Fertigstellung in einem Zuge ausgegeben. Im Gegensatz dazu werden beim Plotten mit dem Bildschirm die einzelnen Anweisungen unmittelbar sichtbar ausgeführt.

Im vorliegenden Beispiel soll das Bild im Datenfile BEISP (muß mit dem Befehl CREATE zuvor in einer Bibliothek angelegt worden sein) abgespeichert werden. Für den Buffer werden 2 KB reserviert.

(2) Mit der Anweisung FRAME wird die verwendete Zeichenfläche in Zoll definiert (hier: 6 x 6 Zoll).

(3) Die Anweisung SCALE gibt die kleinsten und größten Koordinatenwerte an, die noch innerhalb der Zeichenfläche liegen. Im Beispiel liegen in Richtung der x- und der y-Achse Punkte im Wertebereich von -10 bis 10 innerhalb der Zeichenfläche.

(4) Durch die Anweisung XAXIS wird eine x-Achse geplottet. Der erste Parameter, 0, gibt die Lage der Achse in Bezug auf die y-Koordinate an. Der zweite Parameter gibt den Abstand der Markierungen auf der x-Achse an.

(5) Die Anweisung YAXIS bewirkt analog zu XAXIS, das Zeichnen einer y-Achse.

(6) Die Darstellung von Zeichnungen erfolgt durch Zeichnen von Strecken zwischen dem Standpunkt einer "imaginären" Feder und angegebenen Zielkoordinaten.

Mit MOVE wird ein Punkt im Koordinatensystem angesprochen, von dem ab geplottet werden soll. Im Beispiel soll vom Punkt mit der x-Koordinate 4 und der y-Koordinate 0 das Plotten des Kreises begonnen werden.

(7) Mit FOR.... und NEXT wird eine Programmschleife definiert, in der das Plotten durchgeführt wird. Die Laufvariable I dient als Winkel in der Parameterdarstellung des Kreises; sie startet bei 0 und durchläuft in Schritten von II/30 die Werte 2 II, wodurch die Darstellung des Kreises durch Annäherung als 60-Eck erreicht wird.

Durch Verkleinerung der Schrittweite erreicht man im allgemeinen genauere Zeichnungen, es ist jedoch zu beachten, daß bei Erreichen der Grenze des Auflösungsvermögens die Genauigkeit nicht mehr verbessert werden kann, die Folge sind dann nur unnötig lange Rechenzeiten.

(8) Die Anweisung PLOT bewirkt das Zeichnen einer Geraden vom zuletzt angesprochenen Punkt zu dem Punkt, dessen Koordinaten in der Anweisung PLOT angegeben werden. Der in diesem Befehl geplottete Kreis setzt sich also aus einzelnen Geradenabschnitten zusammen, die umso kleiner sind, je niedriger die Schrittweite in Zeile 70 gewählt wurde.

(9) Nach dem Plotten des Kreises soll eine Beschriftung angefügt werden. Mit MOVE wird die Position des ersten Zeichens, das geplottet werden soll, bestimmt.

(10) Zur Vorbereitung des Plottens von Zeichen (Texten) dient die Anweisung CSIZE, in der Breite und Höhe (in Zoll) eines Zeichens festgelegt werden und die Schriftrichtung durch den Winkel (im Bogenmaß) zwischen der Schrift und der Richtung der positiven x-Achse definiert wird. Im Beispiel wird eine Zeichengröße von 0,15 Zoll für die Breite und 0,18 Zoll für die Höhe sowie eine horizontale Schriftrichtung (Winkel = 0) festgelegt.

(11) Wenn Anfangspunkt und Zeichengröße feststehen, kann der Text mit CPLOT ausgegeben werden. Hier wird zunächst der Text "BEISPIEL 1:" geplottet.

Da noch ein zweiter Text in einer neuen Zeile geplottet werden soll, wird mit MOVE der Startpunkt festgelegt. Das erste Zeichen der neuen Zeile wird ausgehend vom Punkt mit den Koordinaten x = 0,5, y = -6,5 geplottet, d.h. eine Einheit unterhalb der ersten Textzeile. Der auszugebende Test ist "KREIS MIT RADIUS 4", Zeichengröße und Schreibrichtung sind durch Zeile 110 festgelegt. (Jedes CSIZE bleibt solange maßgebend, bis ein neues CSIZE im Programm die alten Werte ersetzt).

(12) Wenn die Erstellung des Bildes im Programm abgeschlossen ist, kann mit der Anweisung DRAW die Ausgabe über den Thermodrucker bewirkt werden. Die nötigen Informationen werden aus dem File BEISP, in dem die Zeichnung während der Generierung abgespeichert wurde, geholt und die Plot-Zeichnung wird zeilenweise erstellt. Der Inhalt von BEISP ist auch nach dem Ende des Programms noch vorhanden, wodurch in anderen Programmen die Zeichnung weiter verwendet bzw. mit DRAW erneut ausgegeben werden kann.

Das hier angegebene Beispielprogramm soll nur einen ersten Eindruck vom Plotten mit dem System vermitteln. In den folgenden Abschnitten werden alle Möglichkeiten des Plottens im einzelnen erläutert.

9.5.2 Plotten über den Thermodrucker PR 2400 mit OPT PLO (ohne Darstellung auf Bildschirm)

Der Thermodrucker PR 2400 kann als digitaler Plotter betrachtet werden, d.h. auf der Zeichenfläche steht ein festes Punktraster zur Verfügung, in dem der Abstand zwischen 2 Rasterpunkten als Grund-Einheit definiert ist. Die Plotterzeichnung setzt sich nur aus Punkten dieses Rasters zusammen. Dazwischenliegende Koordinatenwerte können nur durch Rasterpunkte angenähert werden. Eine Grund-Einheit entspricht 1/70 Zoll, das entspricht 560 darstellbaren Punkten für die maximal mögliche Breite einer Zeichnung.

9.5.2.1 Darstellung von Punkten, Linien und Zeichen

Eine Plot-Zeichnung ist die Gesamtheit der während des Plottens markierten Punkte des Grundrasters. Als Elemente stehen in der Programmierung einzelne Punkte, Gerade und Zeichen zur Verfügung.

Jeder zu zeichnende Punkt ist durch seine Koordination (x-Wert und y-Wert) definiert und wird durch Markierung des nächstliegenden Rasterpunktes dargestellt. Der durch die erforderliche Rundung maximal mögliche Fehler in der Darstellung eines Punktes beträgt somit ca. 1/100 Zoll.

Gerade werden durch Markierung aller sich durch Rundung ergebenden Rasterpunkte entlang der rechnerisch ermittelten Geraden zwischen dem Ausgangspunkt und den angegebenen Koordinaten des Zielpunktes dargestellt.

Für die Ausgabe von Zeichen steht ein Zeichensatz (siehe Anweisung CPLOT) zur Verfügung, in dem jedes Zeichen durch 9 signifikante Punkte definiert ist. Bei der Ausgabe von Zeichenfolgen wird die Zuordnung zu den sich ergebenden Rasterpunkten vom System durchgeführt.

9.5.2.2 Ausführung von Plotprogrammen

Während der Ausführung eines Plotprogramms für den integrierten Printer wird das Bild komplett entwickelt, abgespeichert und erst dann geplottet. Das hat folgende Gründe:

- der Druckkopf kann nur Bewegungen von links nach rechts ausführen, das Papier wird nur in eine Richtung weitertransportiert,

- der Drucker könnte evtl. für die Ausführung von PRINT-Anweisung während des Ablaufs des Plotprogramms benötigt werden.

Für das Speichern des Bildes müssen ein externes Datenfile und ein Buffer im Arbeitsspeicher vorhanden sein. Während des Programmlaufs werden markierte Punkte zunächst im Buffer abgespeichert. Sobald dieser voll ist, wird sein Inhalt als Teilbild in das externe File übertragen und mit der Speicherung eines neuen Teilbildes fortgesetzt. Bei der Ausführung von DRAW, CHAIN, oder END wird der letzte Teil des Bildes aus dem Buffer übertragen. Bei der Ausführung von DRAW wird der Buffer zum Zusammenfügen der Teilbilder benutzt. Das daraus resultierende Bild wird vom Thermodrucker ausgegeben. Das Daten-File muß als sequentielles File (mit dem Systembefehl (CREATE) vor der Programmausführung erstellt werden. Es muß groß genug sein, um alle Teilbilder aus dem Buffer aufnehmen zu können. Der Name des Files und die Größe des Buffers müssen in einer INIMAGE- oder LDIMAGE- als erste Plot-Anweisung des Programms definiert werden. Ein zum Plotten verwendetes File kann in der Folge nicht mit den Anweisungen READ: oder WRITE: angesprochen werden. Bezüglich der günstigen Buffer- und Filegröße wird auf Anhang B verwiesen.

9.5.3 Graphische Darstellung am Bildschirm mit OPT GDI

Zum graphischen Arbeiten mit dem Bildschirm ist der Systembefehl OPT(ion) mit dem Parameter GDI erforderlich. Auch hier ist der Bildschirm in zwei Bereiche geteilt: den oberen (graphischen) Bereich mit einer Fläche von 280 x 19,6 mm (392 Zeilen mit je 560 Punkten) und dem unteren mit zwei alphanumerischen Zeilen von je 80 Zeichen. Die Bereiche sind durch einen Abstand getrennt.

Der Pointer

Neben der Ausgabe von Zeichnungen ist durch Verwendung eines graphischen Pointers, der durch zwei verschiebbare Achsen dargestellt wird, die Bestimmung von Koordinaten beliebiger Punkte der Zeichenfläche möglich.

Nach dem Erreichen des gewünschten Punktes werden nach dem Drücken der Tasten SHIFT EXIT die Koordinaten des gewählten Punkts in die angegebenen Variablen des Programms übernommen. Die vertikale Achse des Pointers wird mit den Tasten → und ← gesteuert, die horizontale mit den Tasten ↑ und ↓ . Dabei kann die REPEAT-Taste wie üblich verwendet werden. SHIFT hat für die Tasten → und ↑ die Funktion einer Feinsteuerung des Pointers. Teile einer Zeichnung, die von den Achsen des Pointers überdeckt werden, erscheinen heller und sind dadurch besonders gekennzeichnet. Der Pointer erscheint auf dem Bildschirm durch Ausführung der BASIC-Anweisung POINTER (siehe Kap. 5).

Beim graphischen Arbeiten kann der Bildschirm wie ein digitaler Plotter verwendet werden. Jede Zeichnung setzt sich aus einzelnen Punkten des Bildschirmes zusammen. Der graphische Bereich des Bildschirms läßt sich als rechteckige Punktmatrix ansehen, in der der Abstand zwischen zwei nebeneinanderliegenden Punkten einer Grund-Einheit entspricht. Die Abmessungen werden in Zoll angegeben; eine Grund-Einheit entspricht etwa 0,02 Zoll. (Das Verhältnis der Punkt-Abstände von Thermoprinter und Bildschirm zueinander ist etwa 1:1.38).

Eine Zeichnung auf dem Bildschirm wird durch ein BASIC-Programm unter Verwendung der für das Plotten erforderlichen Anweisungen erzeugt. Diese Anweisungen ermöglichen das Zeichnen von Punkten, Linien und Zeichen. Die Lage der einzelnen Bildpunkte wird durch Angabe von kartesischen Koordinaten x und y bestimmt.

Im Gegensatz zur Ausführung am Thermodrucker ist bei der Ausgabe am Bildschirm die Wirkung der Plot-Anweisungen unmittelbar sichtbar. Daher wird für das Plotten mit Bildschirm kein zusätzlicher Buffer und kein Datenfile benötigt, die Befehle INIMAGE, LDIMAGE und DRAW haben dadurch eine etwas andere Wirkung als beim Plotten mit Drucker.

Werden während des Plottens PRINT-Anweisungen ausgeführt, so erscheinen die auszugebenden Daten nur am Systemdrucker. Wenn NO PRINT aktiviert ist, werden diese Daten unterdrückt.

Soll die Ausgabe von PRINT-Zeilen am Bildschirm wieder ermöglicht werden, so ist das durch Ausführung einer speziellen DISP-Anweisung oder durch Programmwechsel möglich.

Generell sind Programme zum Plotten über Thermodrucker auch für den Bildschirm verwendbar, es müssen jedoch folgende Punkte beachtet werden:

- Die Zeichnung hat generell das Format, das FRAME 8.5.6 entspricht. Um eine Verzerrung zu vermeiden, muß der Wertebereich in SCALE in diesem Verhältnis vereinbart werden.

- Abspeicherung einer Zeichnung

  Um eine auf dem Bildschirm erstellte Zeichnung abzuspeichern, muß ein STIMAGE-Befehl eingegeben werden. (Beim Plotten mit Thermodrucker wird das Bild automatisch abgespeichert).

- Programme, die für das Plotten auf dem Bildschirm geschrieben wurden, können Anweisungen enthalten, die beim Plotten über den Thermodrucker nicht erlaubt sind. Ein Beispiel dafür ist die Anweisung ERASE, die den Bildschirminhalt löscht.
  Diese Anweisungen sind in Kap. 6 erläutert.

9.5.4 Plotten mit einem externen Plotter

Über eine der verfügbaren Schnittstellen können geeignete Plotter von Fremdherstellern an das System angeschlossen werden. Da die Intelligenz und Ausstattung dieser Plotter unterschiedlich ist, muß eine Zwischenstufe eingeschoben werden, die die Umsetzung der BASIC-Anweisungen für die entsprechende Plottertype durchführt.

Bemerkungen zur Programmierung

1. Das Programm muß die Anweisung EXTERNAL PLOTTER enthalten. Diese Anweisung kann an jeder Stelle des Programmes stehen und bewirkt, daß alle Plotanweisungen nicht unmittelbar ausgeführt sondern als Aufruf einer Funktion interpretiert werden.

2. Das Programm muß eine mehrzeilige Funktion enthalten, durch die die für die Ausführung erforderlichen Plotterbefehle des externen Plotters generiert werden.

   Besonderheiten der Funktion:

   - sie muß mit FNP bezeichnet werden

   - sie muß in der DEF-Anweisung sechs formale Parameter enthalten, wobei der fünfte eine String-Variable sein muß

- innerhalb der Funktion muß ein Wert an FN* zugewiesen werden, der aber keine weitere Bedeutung für ihre Wirkung hat.

Beispiel:

```
1000 DEF FNP (A, B, C, D, E$, F)
   .
   .
   .
   .
2100 FN* = 0
2110 FN END
```

Die ersten vier Parameter der Funktion übernehmen numerische Werte von Plotanweisungen (z.B. SCALE -10, 10, -20, 20: die erste Variable erhält den Wert -10. die dritte -20 und die vierte 20.). Falls die Anweisung weniger als vier Parameter erfordert, werden die nicht benötigten Variablen auf 9.999999E99 gesetzt. Die fünfte Variable (Stringvariable) wird für die Anweisung CPLOT benötigt. Ihr wird der Stringausdruck zugewiesen, der geplottet werden soll. Daher muß die deklarierte Länge dieser Variablen mindestens so groß sein, wie der größte zu plottende String im Programm. (Bei allen übrigen Plotanweisungen wird der fünften Variablen der "Leerstring" zugewiesen.)

Bei CPLOT wird zusätzlich die vierte Variable benutzt: sie erhält ebenfalls den Wert 1, wenn % im String vorkommt, sonst den Wert 0. Diese vierte Variable wird ebenfalls bei der Anweisung ERASE verwendet. Sie erhält den Wert 0, wenn kein Parameter angegeben wurde, den Wert 1, wenn ON angegeben wurde und Wert 2, wenn OFF angegeben wurde. Die sechste Variable erhält einen Wert, der die auszuführenden Plotanweisungen bestimmt. Die den einzelnen Anweisungen zugeordneten Werte sind:

SCALE	10
CSIZE	11
OFFSET	12
CTAB	13
XAXIS	14
YAXIX	15
PLOT	16
IPLOT	17
CPLOT	18
IDOT	19
DOT	20
MOVE	21
FRAME	22
POINTER	23
REVERSE	24
ERASE	25

Die oben beschriebene Funktion ermöglicht die Benutzung der Plotprogramme für verschiedene externe Plotter; sie setzt die BASIC-Plotanweisungen in den Befehlsvorrat des Plotters um.

Da durch den Aufruf der Funktion FNP die Wirkung der Plotanweisungen durch die Funktion bestimmt ist, haben alle Anweisungen, die zur Steuerung der Ausgabe am Bildschirm oder Thermodrucker dienen, keine Wirkung. Es sind dies die Anweisungen INIMAGE, LDIMAGE, STIMAGE, DRAW.

Bemerkung: Zur Zeit existieren Plotter-Driver (FNP) für Plotter der Firmen Calcomp, Benson und Tewidata (Servoger).

9.5.5 Basic-Anweisungen der Module OPTION GDI/PLO

Kurzbeschreibung

CPLOT	Ausgabe einer Liste von Strings
CSIZE	Definition der Zeichengröße und Schreibrichtung für einen String
CTAB	Tabulation bei der Ausgabe
DISP	Darstellung von Text auf dem Bildschirm
DOT	Markieren eines Punktes
DRAW	Ausgabe des Bildes über den integrierten Drucker
ERASE	Löschen des Bildschirminhalts (in GDI und EXD enthalten
EXTERNAL PLOTTER	Vorbereiten des Systems zur Ausgabe auf externen Plotter
FRAME	Festlegung der Größe der Zeichenfläche
IDOT	Zeichen eines Punktes mit den Inkrementen dx und dy
INIMAGE	Vorbereiten des Systems zum Plotten
IPLOT	Zeichnen einer Geraden zu dem Punkt, der durch Inkremente dx und dy erreicht wird
LDIAGE	Laden eines gespeicherten Bildes
MOVE	Positionieren auf einen Punkt
OFSET	Definiert einen neuen Ursprung für das Kartesische Koordinatensystem
PLOT	Zeichnen einer Geraden zu dem durch Koordinaten definierten Punkt
POINTER	Darstellen des graphischen Pointers auf dem Bildschirm
REVERSE	Umkehren des Bildes auf dem Bildschirm von positiv auf negativ oder umgekehrt (in GDI enthalten)
SCALE	Festelegen von Lage und Maßstab des Koordinaten-systems
XAXIS	Zeichnen einer Parallelen zur X-Achse
YAXIS	Zeichnen einer Parallelen zur Y-Achse

ANWEISUNG: CPLOT (Characterplot)

FUNKTION: Ausgabe einer Liste von Strings mittels Plot-Zeichengenerator.

FORMAT:

CPLOT{String-Ausdr. [,String-Ausdr.] ... [,%] / ,%}

mit: "String-Ausdr." für alphanumerische Ausdrücke,
Beschränkung auf maximal 8 Stringausdrücke.

WIRKUNG: Das Zeichen % bewirkt eine Zeilenschaltung, Stringausdrücke werden an die Stelle geschrieben, die vorher positioniert wurde. Die Größe der Zeichen und die Schreibrichtung richtet sich nach der Anweisung CSIZE.

Wird als letztes Zeichen in der Anweisung das Trennzeichen ";" geschrieben, wird auf den Ausgangspunkt für das direkt nachfolgende Zeichen positioniert. Wird "%" am Ende angegeben, dann wird als neuer Ausgangspunkt der Stringanfang eine Zeile tiefer gesetzt. Dies ermöglicht eine einfache spaltenmäßige Stringausgabe.

BEMERKUNG: Die Anweisung CPLOT ist für die Option PLOT und GDI verwendbar. Der Zeichengenerator erzeugt folgende Zeichen:

- die Großbuchstaben

  A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,R,S,T,U,V,W,X,Y,Z

- die Ziffern:

  0,1,2,3,4,5,6,7,8,9

- die Sonderzeichen:

  : # $ % & / ( ) * + , - : . < = > ? @ [ ] ↑ _ \

Enhält ein Sting andere Zeichen, so werden diese durch das Symbol "|||" dargestellt.

Es wird nur der Teil des Strings geschrieben, der in die definierte Zeichenebene hineinpaßt. Wird über den Rand hinausgeschrieben, wird als Hinweis nach der Ausgabe des Bildes das Maximum oder Minimum in X und (oder) Y ausgegeben.

ANWEISUNG: CSIZE (Charactersize)

FUNKTION: Definition der Zeichengröße von Strings und der Schreibrichtung, bezüglich der X-Achse.

FORMAT: CSIZEb, h, w

mit: numerischen Werten für die Parameter b (Breite), h (Höhe) und den Winkel w im Bogenmaß.

b, h $\geq$ 0.1

WIRKUNG: Die numerischen Werte für die Breite und Höhe stehen für das Punktraster, in dem die Zeichen generiert werden. Die Dimension für Breite und Höhe ist das Maß Zoll. Der Winkel, der im Bogenmaß angegeben wird, hat als Drehrichtung die mathematisch positive Definition (entgegen dem Uhrzeigersinn).

BEMERKUNG: Die Anweisung CSIZE ist für die Optionen PLOT und GDI verwendbar. Sie gilt so lange für alle nachfolgenden CPLOT-Anweisungen, bis eine erneute CSIZE-Anweisung erfolgt.

Wenn die Anweisung CSIZE fehlt, werden für Breite und Höhe auf dem Thermodrucker 0.14 Zoll und auf dem Bildschirm 0.19 Zoll angenommen und der Winkel auf Ø gesetzt.

Von den festgelegten Werten für Breite und Höhe werden 6/10 für das Zeichen selbst und 4/10 für den Zwischenraum zum nächsten Zeichen nach rechts bzw. oben genommen.

ANWEISUNG: CTAB (Charactertab)

FUNKTION: Tabulation auf einen Punkt, dessen Koordination durch eine Verschiebung um n Zeichen und m Zeilen gegeben sind (bezüglich der letzten ausgeführten CPLOT-Anweisung)

FORMAT: CTAB n, m
mit: den Parametern n und m als numerische Ausdrücke

WIRKUNG: Die nächste String-Ausgabe mittels CPLOT erfolgt um n Zeichenbreiten und m Zeilen (gemäß CSIZE) versetzt. Die Anweisung CTAB bezieht sich immer auf die letzte CPLOT-Anweisung.

BEMERKUNGEN: Die Anweisung CTAB ist für die Optionen PLOT und GDI verwendbar. CTAB ist vergleichbar mit der Anweisung MOVE, die eine Koordinatenbewegung zu einem durch Koordinaten definierten Punkt ausführt und als Vorbereitung für PLOT, DOT, ... verwendet wird. Der Vorteil der Anweisung CTAB besteht darin, daß man nicht auf Koordinaten umrechnen und auch die Schreibrichtung für den String nicht berücksichtigen muß.

Das Positionieren sollte so ausgeführt werden, daß innerhalb der Zeichenfläche geblieben wird. Reicht der Rahmen für das Gesamtbild nicht aus, so wird am Ende das Maximum oder Minimum der Koordinatenwerte als Hinweis ausgegeben.

ANWEISUNG: DISP(Display)

FUNKTION: Darstellung von Text auf dem Bildschirm im Graphik-Mode.

FORMAT: DISP"Θ Code[a b]"[Stringausdr.]

mit:

Θ	Zeichen Nr. 27 der ISO-Code-Tabelle (wird erzeugt durch gleichzeitiges Drücken der Tasten CONTROL und)
Code	einer der Buchstaben Y, X, W, Z, K, J, B, A oder S
a	Zeichen der ISO-Code-Tabelle (zwischen 32 und 70), das die Zeilennumer bestimmt (siehe Bemerkungen)
b	Zeichen der ISO-Code-Tabelle (zwischen 32 und 111), das die Position innerhalb der Zeile bestimmt (siehe Bemerkungen)
Stringausdr.	ist eine beliebige Folge von Zeichen, die auf dem Bildschirm an der durch a und b festgelegten Stelle erscheinen soll.

BEMERKUNG: Genauere Erläuterungen zur Darstellung von Text im Graphik-Mode finden sich ab Seite 9.48 (Zusätzliche Eigenschaften der DISP-Anweisung)

ANWEISUNG: DOT

FUNKTION: Markieren eines Punktes, der durch Koordinaten definiert ist.

FORMAT: DOT X,Y

mit: X und Y sind numerische Ausdrücke, die die Abszisse und die Ordinate des Punktes angeben.

WIRKUNG: Die numerischen Werte X und Y bestimmen Abszisse und Ordinate des Punktes, der markiert werden soll. Die Koordinatenbewegung zu dem angegebenen Punkt erfolgt, ohne daß eine Linie gezogen wird.

BEMERKUNG: Die Anweisung DOT ist für die Optionen PLOT und GDI verwendbar. Liegt der Punkt außerhalb der Bildfläche, wird er nicht markiert. In diesem Falle wird der Wert der Abszisse und (oder) Ordinate als Außenpunkt registriert. Als Hinweis wird nach der Ausgabe des Bildes, das Maximum und (oder) Minimum der Außenpunkte ausgegeben.

ANWEISUNG: DRAW

FUNKTION: Ausgabeanweisung für das Bild über den Drucker (Hardcopy)

FORMAT: DRAW[d]

mit: d numerischen Ausdruck

(0.1<=d<8) für die Verschiebung des Koordinatenursprungs in X-Richtung (in Zoll)

WIRKUNG:

bei Option PLOT:

Das erzeugte Bild wird über den Thermodrucker ausgegeben. Es wird aus den Teilbildern des Datenfiles, festgelegt durch INIMAGE oder LDIMAGE, zusammengesetzt. Ist der Parameter d (in Zoll) angegeben, dann wird das gesamte Bild um den entsprechenden Betrag nach rechts verschoben.

bei Option GDI:

Das auf dem Bildschirm dargestellte Bild wird über den Thermodrucker geplottet. Dabei wird die Kopie im Verhältnis 1:1.38 verkleinert.

Die Angabe des Parameters d ist ohne Bedeutung.

Bei externem Plotter:

Ist die Anweisung EXTERNAL PLOTTER im Programm enthalten, dann ist die Anweisung DRAW wirkungslos.

BEMERKUNGEN: Die Anweisung DRAW veranlaßt zunächst die Aufzeichnung des Buffers im Arbeitsspeicher auf das genannte Datenfile der Diskette. Der Inhalt des Datenfiles bleibt so lange erhalten, bis mit den Anweisungen INIMAGE oder LDIMAGE neue Plotoperationen mit dem gleichen File ausgeführt werden.

Nach dem Schreiben aus dem Buffer steht im Arbeitsspeicher die Information wie sich das Bild aus gespeicherten Teilbildern zusammensetzt: Es sind maximal n = (Buffergröße - 256)/128 (alle Angaben in Byte) Teilbilder möglich, wobei jedes Teilbild durch eine Abspeicherung des Bufferinhalts auf die Diskette erzeugt wird.

Ist in den Anweisungen INIMAGE oder LDIMAGE als Parameter OFF aufgeführt, dann werden mit DRAW nur die Extremwerte, Maximum und Minimum der Abszissen und Ordinaten des Bildes, ausgegeben. Diese Werte, gemessen in Zoll, zeigen an, welcher Bereich für das Bild vorzusehen ist, und können dann in der Anweisung SCALE für die Festlegung der Zeichenfläche angegeben werden.

Die Ausführung der Anweisung DRAW kann durch Drücken der Tasten EXIT oder CTRL und EXIT unterbrochen werden. EXIT führt in den Debugging-Mode, die Ausführung des Programms kann mit SHIFT und EXIT ab dem auf DRAW folgenden Befehl fortgesetzt werden: das Bild wird jedoch nicht weiter geplottet. CTRL und EXIT bedeutet vorzeitiges Programmende. In jedem Fall werden jedoch die Extremwerte ausgedruckt.

Falls DRAW in einer mehrzeiligen Funktion vorkommt, kann der Wert der Funktion im Debugging-Mode nicht abgefragt werden.

ERASE

ANWEISUNG: ERASE

FUNKTION: Löscht den Bildschirm

FORMAT: ERASE [{ON / OFF}]

WIRKUNG: Für Option GDI:

Ist kein Parameter angegeben, wird der Bildschirm gelöscht.

Falls ON abgegeben ist, wird die Wirkung aller folgenden Plot-Befehle umgekehrt, z.B. bewirkt dann die Anweisung DOT 3, 4, daß der Punkt mit den Koordinaten 3 und 4 gelöscht wird.

Falls OFF angegeben ist, wird der ursprüngliche Zustand wiederhergestellt und das Plotten ist wieder möglich.

Für Option PLOT:

Der Befehl wird nicht ausgeführt, es erscheint ERROR 75.

BEMERKUNG: Kommt im Programm die Anweidung EXTERNAL PLOTTER vor, dann hat ERASE keine Wirkung.

EXTERNAL PLOTTER

ANWEISUNG: EXTERNAL PLOTTER

FUNKTION: Vorwahl für einen externen Plotter

FORMAT: EXTERNAL PLOTTER

WIRKUNG: Mit dieser Anweisung wird angegeben, daß das Bild nicht in einem Datenfile erzeugt wird und auch nicht auf dem Bildschirm erscheint, sondern direkt über den angeschlossenen externen Plotter gezeichnet wird. Die eigentliche Plottergrundsoftware, die bei verschiedenen Plottern unterschiedlich ist, muß in der Funktion FNP zur Verfügung stehen. Diese Funktion enthält auf BASIC-Ebene alle spezifischen Anweisungen für den externen Plotter, sie muß im Programm vorhanden sein.

BEMERKUNG: Die Anweisung EXTERNAL PLOTTER ist eine nicht ausführbare Anweisung, kann also an jeder Stelle des Programms stehen (z.B. in FNP).

Die Anweisungen INIMAGE, LDIMAGE, STIMAGE und DRAW gelten nur für den integrierten Printer bzw. den Bildschirm. Steht im gleichen Programm die Anweisung EXTERNAL PLOTTER, dann werden diese Anweisungen überlesen.

FRAME

ANWEISUNG: FRAME

FUNKTION: Festlegung der Größe der Zeichenfläche

FORMAT: FRAME b, h

mit: b (Breite) und h (Höhe) als numerische Ausdrücke

$0.1 \leq b \leq 8$ (ca. 20 cm)
$0.1 \leq h \leq 936$ (ca. 23.4 m)

WIRKUNG: Für Option PLOT:

Die numerischen Werte für "Breite" und "Höhe" definieren die Größe der Zeichenfläche, in der das Bild generiert wird. Die Dimension der Parameter ist Zoll.

Für Option GDI:

Die Anweisung hat keine Wirkung, es wird generell eine Größe von 8 x 5.6 Zoll angenommen.

BEMERKUNGEN: Die obere Grenze der "Breite" wird durch die Breite des Thermopapiers bestimmt und entspricht ca. DIN A 4 mit Hochformat. Die Höhe ist nur theoretisch limitiert (23 Meter!) und hängt, genau wie die Gesamtfläche des Bildes, von der Größe des Buffers im Arbeitsspeicher und derjenigen des externen Datenfiles ab.

Im Programm muß die Anweisung FRAME nach INIMAGE, aber vor allen anderen Plotanweisungen stehen. Wird die Anweisung FRAME nicht ausgeführt, dann wird für das Bild eine Größe von 8 x 8 Zoll angenommen.

Enthält das Programm die Anweisung EXTERNAL PLOTTER, so hat die Anweisung FRAME keine Wirkung, die Größe des zu plottenden Bereichs muß in der Funktion FNP definiert werden.

ANWEISUNG: IDOT (Incrementdot)

FUNKTION: Zeichnen eines Punktes durch Angabe von Inkrementen dx und dy

FORMAT: IDOT dx, dy

mit: dx und dy als numerische Ausdrücke

WIRKUNG: Die numerischen Werte für dx und dy heißen Inkremente und definieren einen Punkt, der bezüglich des zuletzt angesprochenen Punktes X/Y die Koordinaten dx/dy hat (Koordinaten bezüglich des Ursprungs: X + dx/Y + dy).

BEMERKUNGEN: Die Anweisung IDOT ist für die Optionen PLOT und GDI verwendbar. Wenn IDOT auf keinen Bezugspunkt zurückgreifen kann, wird der Ursprung des Koordinatensystems als Ausgangspunkt genommen.

Wenn der zu zeichnende Punkt außerhalb des Rahmens fällt, werden das Maximum und (oder) Minimum von Abszisse und Ordinate gespeichert, damit ein Hinweis auf zu kleine Zeichenfläche gegeben werden kann.

ANWEISUNG: INIMAGE

FUNKTION: Vorbereiten des Systems zum Plotten

FORMAT: INIMAGE [OFF / FILENAME] [,n]

mit: Filename = Name des sequentiellen Datenfiles

n = Größe des Buffers (2<= n<48 Kbyte)

BEMERKUNGEN: Für Option PLOT:

Erster Parameter:

- OFF:

  Das gesamte Plot-Programm wird ausgeführt. Anstelle der Markierung von Punkten und der Speicherung des Bildes wird Maximum und Minimum der Abszissen und Ordinaten ausgegeben, wenn es außerhalb des Zeichenbereiches liegt, sonst erscheint die Meldung "NO OUTER POINTS"

- Filename:

  Das File mit dem Namen "Filename" ist für die Speicherung des Bildes vorgesehen. Das sequentielle File darf nicht gesichert sein und kann der Package-, Common- oder User-Bibliothek angehören. Mit der Anweisung INIMAGE wird das File zur Speicherung des Bildes vorbereitet und ein eventuell bereits vorhandenes Bild gelöscht. Damit ist das File kein gewöhnliches Datenfile.

- Wenn weder OFF noch ein Filename als Parameter eingesetzt ist, wird als Filename SYSPLO gesetzt. Existiert das genannte File SYSPLO nicht, dann wird das Bild im Buffer des Arbeitsspeichers erzeugt, und, soweit möglich, geplottet. Nach dem Speichern eines Bildes ist das in INIMAGE angegebene File geschützt und nur noch als Plotfile verwendbar. Als Datenfile kann es erst nach PURGE und erneutem CREATE verwendet werden.

Zweiter Parameter:

- Im Arbeitsspeicher wird ein Puffer von n Kbyte eingerichtet, der alle Informationen über die gezeichneten Punkte aufnehmen soll. Nach dem Aufbau des Puffers sind keine Punkte darin gespeichert. Ist der Puffer durch fortlaufende Plotanweisungen gefüllt, wird der Pufferinhalt auf das genannte File als Teilbild gespeichert und danach neu aufgebaut. Auf diese Weise wird das Bild aus Teilbildern zusammengesetzt.
- Ist "Filename" aufgeführt, aber fehlt "n", dann wird n = 3 gesetzt.
- Ist "OFF" aufgeführt, hat "n" keine Wirkung.
- Die Anweisung INIMAGE initialisiert die Standard-Parameter für die im Programm folgenden Plot-Anweisungen im Arbeitsspeicher.

  Diese sind:

  - Die maximale Dimension des Bildes: 8 x 8 Zoll

  - Die Maßeinheiten für die Achsen, Minimum und Maximum von Abszissen und Ordinaten für die Zeichenfläche:

    1 Maßeinheit = 1 Punkt = 1/70 Zoll

    Abszisse: Minimum = -279.5; Maximum = 279.5
    Ordinate: Minimum = -279.5; Maximum = 279.5

    Dies bedeutet, daß der Ursprung des Koordinatensystems in der Mitte der Zeichenfläche liegt.

  - Die Dimension der Zeichen im Zeichengenerator:

    Breite des Punktrasters eines Zeichens =
    1/7 Zoll (10 Punkte)

    Höhe des Punkterasters eines Zeichens =
    1/7 Zoll (10 Punkte)

    Winkel = 0 (Bogenmaß)

  Die Werte für die Plot-Parameter können durch die Anweisungen FRAME, SCALE, OFFSET und CSIZE modifiziert und damit dem Problem angepaßt werden.

Für Option GDI:

Die Angabe von Parametern hat keine Wirkung. INIMAGE versetzt den Bildschirm in den Zustand ERASE OFF (siehe Anweisung ERASE) und initialisiert die Standard-Parameter für im Programm folgende Plot-Anweisungen. Diese sind:

- Die maximale Dimension des Bildes:
  560 x 392 Punkte (d.h. ca. 11.04 x 7.73 Zoll)
- Die Maßeinheiten für die Achsen, Minimum und Maximum von Abszissen und Ordinaten für die Zeichenfläche:

  1 Maßeinheit = 1 Punktabstand = 0.02 Zoll

  Abszisse: Minimum = -279.5, Maximum = 279.5
  Ordinate: Minimum = -279.5, Maximum = 279.5

  Dies bedeutet, daß der Ursprung des Koordinatensystems in der Mitte des Bildschirms liegt.
- Die Dimension der Zeichen im "Zeichengenerator":

  Breite des Punkterasters eines Zeichens =
  0.2 Zoll (10 Punkte)

  Höhe des Punkterasters eines Zeichens =
  0.2 Zoll (10 Punkte)

  Winkel = 0 (Bogenmaß)

  Die Werte für die Plot-Parameter können durch Anweisungen SCALE, OFFSET und CSIZE verändert werden.

  INIMAGE bewirkt, daß alle im Programm folgenden PRINT-Anweisungen nicht ausgeführt werden. Mit dem Code A in DISP kann diese Auswirkung rückgängig gemacht werden.

Für beide Optionen:

Die Anweisung INIMAGE (oder alternativ LDIMAGE) muß als erste Plot-Anweisung geschrieben sein. Alle Plot-Anweisungen greifen in der Ausführung auf Parameter zurück, die durch INIMAGE definiert werden. Die Ausführung eines neuen INIMAGE kann sich nicht nur auf den Inhalt des Buffers sondern auch des Datenfiles auswirken und bedeutet immer die Vorbereitungen für ein neues Bild und damit das Löschen des gespeicherten Bildes.

Enthält eine mehrzeilige Funktion die Anweisung INIMAGE, dann kann ihr Wert nicht im Debugging Mode abgefragt werden.

ANWEISUNG: IPLOT (Incrementplot)

FUNKTION: Zeichnen einer Verbindungslinie zu dem Punkt, der durch Inkremente dx und dy erreicht wird.

FORMAT: IPLOT dx, dy

mit: dx und dy als numerische Ausdrücke

WIRKUNG: Die numerischen Werte dx und dy bezeichnen nicht Koordinaten sondern Inkremente, d.h. die Koordinaten dx/dy bezüglich der Koordinaten X/Y des zuletzt angesprochenen Punktes (Koordinaten bezüglich des Ursprungs: X + dx/Y +dy).

Bis zu diesem so definierten Punkt wird linear interpoliert und eine gerade gezeichnet.

BEMERKUNGEN: Die Anweisung IPLOT ist für die Optionen PLOT und GDI verwendbar. Wenn noch kein Ausgangspunkt beim ersten Aufruf vorliegt, wird der Ursprung des Koordinatensystems als Ausgangspunkt angenommen.

Liegen die Elemente der Verbindungslinie ganz oder teilweise außerhalb der Zeichenfläche, dann wird der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und Minimum festgestellt und als Hinweis nach Ausgabe des Bildes geschrieben.

ANWEISUNG: LDIMAGE (Loadimage)

FUNKTION: Übernahme des letzten gespeicherten Bildes mit den Parametern aus einem Plotfile und Aufbau des Arbeitsspeichers für Plot-Anweisungen.

FORMAT: LDIMAGE[Filename]

mit: Filename als Name des Plotfiles.

WIRKUNG: Für Option PLOT:

Das in dem File "Filename" gespeicherte Plot-Bild wird in den Arbeitsspeicher geladen, kann anschließend ergänzt werden und wird dann zurückgespeichert. Das File muß als sequentielles Datenfile in einer der Bibliotheken kreiert sein und ist nach dem ersten Speichervorgang nur noch für Plot-Operationen offen.

In den Arbeitsspeicher werden auch die Parameter für die Plot-Anweisungen geladen. Die Parameter sind die gleichen, wie sie bei INIMAGE beschrieben sind. Es gelten die Werte, die zum Zeitpunkt des Abspeicherns für das Bild definiert waren. Der Arbeitsspeicher enthält auch Informationen, wie z.B. eine Veränderung des Koordinatenursprungs durch OFFSET und den zuletzt markierten Punkt. Wurde ein Buffer im Arbeitsspeicher durch INIMAGE dimensioniert, gilt dieser auch für LDIMAGE.

Wird "Filename" weggelassen, dann wird der Standardname SYSPLO angenommen.

Für Option GDI:

Ist "Filename" angegeben, dann wird das in diesem File enthaltene Bild auf dem Bildschirm dargestellt, zusätzlich zu den eventuell schon vorhandenen Informationen.

Der Arbeitsspeicher wird mit den für die Erstellung des Bildes gültigen Plot-Parametern initialisiert.

Wird "Filename" weggelassen, dann wird der Standardname SYSPLO angenommen.

Das automatische Weiterschieben von Textzeilen wird unterdrückt.

Der Bildschirm wird in den Zustand "ERASE OFF" versetzt (siehe Anweisung ERASE).

BEMERKUNGEN: Die Anweisung LDIMAGE (oder alternativ INIMAGE) muß die erste ausgeführte Plotoperation im Programm sein. Alle Plotanweisungen benötigen Informationen, die durch LDIMAGE bereitgestellt sind. Nach der Ausführung der LDIMAGE-Anweisung kann das geladene Bild ergänzt werden. Es können neue Punkte dazu gezeichnet werden, aber bereits vorhandene Punkte können nicht gelöscht werden. Das Laden eines vorhandenen Bildes mit LDIMAGE ist schneller als das erneute Erstellen mit INIMAGE.

Enthält das Programm die Anweisung EXTERNAL PLOTTER, dann hat LDIMAGE keine Wirkung.

Wenn eine mehrzeilige Funktion die Anweisung LDIMAGE enthält, kann ihr Wert im Debugging Mode nicht abgefragt werden.

Wenn in ein File ein Bild gespeichert wurde, ist es geschützt und kann nicht wie ein gewöhnliches sequentielles Datenfile benutzt werden.

Wurde ein Programm für den Thermodrucker geschrieben und die in FRAME definierte Größe weicht von der für den Bildschirm erforderlichen Größe (8 x 5.6) ab, dann beziehen sich weitere Plot-Anweisungen auf die Standard-Werte für SCALE und CSIZE. Überschreitet die Höhe des Bildes den Wert 5.6, dann wird dieser Teil abgeschnitten.

Die Ausführung von LDIMAGE kann durch Drücken von EXIT oder CTRL und EXIT abgebrochen werden. Nach einer Unterbrechung mit EXIT kann durch Drücken der Taste SHIFT und EXIT der Programmablauf mit dem ersten auf LDIMAGE folgenden Befehl fortgesetzt werden.

ANWEISUNG: MOVE

FUNKTION: Positionieren auf einen durch Koordinaten definierten Punkt

FORMAT: MOVE X, Y

mit: "X" und "Y" als numerische Ausdrücke, die Abszisse und Ordinate eines Punktes bezeichnen.

WIRKUNG: Die numerischen Werte für "X" und "Y" stehen für die Abszisse und Ordinate eines Punktes. Der Punkt wird nicht markiert. Er wird als Ausgangspunkt der nachfolgenden Plot-Operationen gespeichert.

BEMERKUNGEN: Die Anweisung MOVE ist für die Optionen PLOT und GDI verwendbar.

Ein Positionieren kann aus der Zeichenfläche herausführen. In diesem Falle werden das Maximum oder Minimum der Abszissen und Ordinaten der außenliegenden Punkte als Hinweis nach dem Plot ausgegeben.

OFFSET

ANWEISUNG: OFFSET

FUNKTION: Verschiebt den Koordinatenursprung im Koordinatensystem

FORMAT: OFFSET X, Y

mit: X und Y numerische Ausdrücke.

WIRKUNG: Die numerischen Werte "X" und "Y" sind die neuen Koordinaten für eine Verschiebung des Koordinatenursprungs und legen damit ein neues rechtwinkliges Koordinatensystem fest.

Die Koordinaten aller Punkte nach der OFFSET-Anweisung beziehen sich auf dieses neue Koordinatensystem. Der Ursprung des Koordinatensystems darf auch außerhalb der Zeichenfläche liegen.

BEMERKUNGEN: Die Anweisung OFFSET ist für die Optionen PLOT und GDI verwendbar.

Der neue Ursprung, der durch OFFSET definiert wird, bezieht sich immer auf den Koordinatenursprung, der aus der letzten SCALE-Anweisung resultiert, auch wenn bereits zuvor eine andere OFFSET-Anweisung ausgeführt wurde.

Der Koordinatenursprung kann durch die Anweisung OFFSET oder ein neues SCALE geändert werden.

ANWEISUNG: PLOT

FUNKTION: Zeichnen einer Geraden zu dem durch Koordinaten definierten Punkt.

FORMAT: PLOT X, Y

mit: X und Y als numerische Ausdrücke

WIRKUNG: Die numerischen Werte X und Y stehen für die Abszisse und Ordinate eines Punktes. Es wird eine Gerade gezeichnet, die von dem zuletzt gespeicherten Punkt zu dem in den Parametern definierten Punkt führt. Danach gilt diese Koordinate als Ausgangspunkt für die nächste Plot-Operation.

BEMERKUNGEN: Die Anweisung PLOT ist für die Optionen PLOT und GDI verwendbar.

Ist noch kein Ausgangspunkt vorhanden, wird als Ausgangswert der Ursprung des Koordinatensystems verwendet. Liegen die Elemente der Geraden ganz oder teilweise außerhalb der Zeichenfläche, dann wird nur der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und (oder) Minimum festgestellt und als Hinweis nach der Erstellung des Bildes ausgegeben.

POINTER

ANWEISUNG: POINTER

FUNKTION: Der graphische Pointer erscheint auf dem Bildschirm; man kann die Koordinaten jedes gewünschten Punktes damit erhalten.

FORMAT: POINTER var 1, var 2

mit: var 1 und var 2 als beliebigen numerischen Variablen

WIRKUNG: Für Option GDI:

Der graphische Pointer erscheint an dem durch die Variablen var 1 und var 2 definierten Punkt. Mit Hilfe der Steuertasten (siehe Kap. 3) läßt sich der Pointer beliebig verschieben. Steht der Pointer an der gewünschten Stelle, dann wird durch Drücken der Taste SHIFT und EXIT der Programmablauf fortgesetzt, der Pointer verschwindet und die Variablen var 1 und var 2 enthalten die Koordinaten des zuletzt bezeichneten Punktes.

Für Option PLOT:

Die Anweisung wird nicht ausgeführt, es erscheint ERROR 75.

BEMERKUNGEN: Wird mit der POINTER-Anweisung ein Punkt außerhalb des Zeichenbereichs angesprochen, erscheint die Meldung

POINTER OUT OF FRAME

und der Pointer wird in die linke untere Ecke des Bildschirms gesetzt. Durch Drücken der Taste EXIT geht das System in den DEBUGGING MODE; dadurch können die Werte der Variablen var 1 und var 2 abgefragt und geändert werden. Nach dem Drücken von SHIFT und EXIT wird die Anweisung POINTER erneut durchgeführt.

REVERSE

ANWEISUNG: REVERSE

FUNKTION: Texte oder Zeichnungen auf dem Bildschirm werden statt positiv negativ dargestellt oder umgekehrt.

FORMAT: REVERSE

WIRKUNG: Für Option GDI:

Die Farbe der Schrift oder die Zeichnungen auf dem Bildschirm wird mit der Farbe des Hintergrunds vertauscht.

Für Option PLOT:

Die Anweisung wird nicht ausgeführt; es erscheint ERROR 75.

BEMERKUNGEN: REVERSE hat für das Erstellen von Kopien über Thermodrucker oder das Abspeichern von PLOT-Zeichnungen keine Bedeutung.

ANWEISUNG: SCALE

FUNKTION: Festlegen der Einheiten des Koordinatensystems durch Minimum und Maximum der beiden Achsen; implizit werden dadurch Maßstabsfaktoren und der Koordinatenursprung definiert.

FORMAT: SCALE X-min, X-max, Y-min; Y-max

mit: X-min, X-max, Y-min, Y-max als numerische Ausdrücke unter Berücksichtigung der Bedingungen X-min X-max und Y-min Y-max

WIRKUNG: Die numerischen Werte für X-min, X-max, Y-min und Y-max definieren die Zeichenfläche als X- und Y-Intervall. Die Breite der Zeichenfläche in Zoll, in der Anweisung FRAME festgelegt oder als Standardparameter angenommen, entspricht dem Intervall (X-min - X-max). In diesem Fall gilt als Maßeinheit für die Abszisse:

(Breite/(X-max - X-min))Zoll

Analog gilt für die Ordinate die Maßeinheit:

(Höhe/(Y-max - Y-min))Zoll

Implizit bestimmen die Werte X-min, X-max, Y-min und Y-max die Lage des Nullpunktes (Koordinatenursprung).

BEMERKUNGEN: Die Anweisung SCALE ist für die Optionen PLOT und GDI verwendbar.

Der Koordinatenursprung darf auch außerhalb der Zeichenfläche liegen.

Die Maßeinheit für beide Achsen kann unterschiedlich sein und durch eine neue SCALE-Anweisung geändert werden.

Fehlt SCALE, wird als Maßeinheit eine Punkteinheit für beide Achsen genommen. Als Koordinatenursprung wird die Bildmitte mit den Parametern

X-min = -(Breite/2) *70
X-max = (Breite/2) *70
Y-min = -(Höhe/2) *70
Y-max = (Höhe/2) *70

definiert.

STIMAGE

ANWEISUNG: STIMAGE

FUNKTION: Abspeichern der Plot-Zeichnung auf dem Bildschirm in ein Datenfile

FORMAT: STIMAGE[FILENAME][,n]

mit: Filename als Name des Files und n als der Puffergröße im Arbeitsspeicher (2 =n 48)

WIRKUNG: Für Option GDI:

Die Plot-Zeichnung auf dem Bildschirm wird in das mit "Filename" bezeichnete Datenfile abgespeichert. Ist "Filename" nicht angegeben, wird als Name SYSPLO angenommen.

Die Zahl n gibt die Größe des Buffers im Arbeitsspeicher (in KByte) an, über den das Bild zwischengespeichert wird. Wird n nicht angegeben, nimmt das System als Standardwert n = 3 an.

Für Option PLOT:

Die Anweisung wird nicht ausgeführt, es erscheint ERROR 75.

BEMERKUNGEN: Das File muß als sequentielles Datenfile angelegt sein.

Informationen über die Dimensionierung von File und Buffer sind in Anhang B zusammengefaßt. Eine Anweisung EXTERNAL PLOTTER im Programm bewirkt, das STIMAGE bedeutungslos wird.

XAXIS

ANWEISUNG: XAXIS

FUNKTION: Zeichnen einer Parallelen zur X-Achse

FORMAT: XAXIS Y[,t][,X1,X2]

mit: Y, t, X1 und X2 als numerischen Variablen oder Ausdrücken und $t \neq 0$

WIRKUNG: Es wird eine Parallele zur X-Achse mit dem Abstand Y gezeichnet. Mit X1 als Startpunkt und X2 als Endpunkt wird das Intervall für die X-Achse definiert. Fehlt diese Information, dann wird die Parallele zur Achse über die gesamte Zeichenfläche gezeichnet. Ist t angegeben, wird die Achse vom Startpunkt beginnend mit Markierungen im Abstand t versehen.

BEMERKUNGEN: Die Anweisung XAXIS ist für die Optionen PLOT und GDI verwendbar.

Ist X1<X2 und t negativ oder auch X1>X2 und t positiv oder |t|>|X2-X1|, dann wird die Achse nicht markiert.

Wird die Achse ganz oder teilweise außerhalb der Zeichenfläche definiert, dann wird nur der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und (oder) Minimum festgestellt und als Hinweis nach der Ausgabe des Bildes geschrieben.

YAXIS

ANWEISUNG: YAXIS

FUNKTION: Zeichnen einer Parallelen zur Y-Achse.

FORMAT: YAXIS X[,t][,Y1,Y2]

mit: X, t, Y1 und Y2 als numerischen Variablen oder Ausdrücken und $t \neq 0$

WIRKUNG: Es wird eine Parallele zur Y-Achse mit dem Abstand X gezeichnet. Mit Y1 als Startpunkt und Y2 als Endpunkt wird das Intervall für die Y-Achse definiert. Fehlt diese Information, dann wird die Parallele zur Achse über die gesamte Zeichenfläche gezeichnet. Ist t angegeben, wird die Achse von Y1 beginnend mit Markierungen im Abstand t versehen.

BEMERKUNGEN: Die Anweisung YAXIS ist für die Optionen PLOT und GDI verwendbar:

Ist Y1<Y2 und t negativ (Y1>Y2 und t positiv) oder $|t|>|Y2-Y1|$, dann wird die Achse nicht markiert.

Wird die Achse ganz oder teilweise außerhalb der Zeichenfläche definiert, dann wird nur der Teil gezeichnet, der ins Bild paßt. Für außerhalb liegende Punkte wird das Maximum und (oder) Minimum festgestellt und als Hinweis nach der Ausgabe des Bildes geschrieben.

9.5.6 Systembefehle für den Bildschirm

Es besteht die Möglichkeit, den Bildschirm außer durch Programm- befehle auch direkt durch Systembefehle zu steuern. Diese können im Command- oder im Debugging-Mode eingegeben werden.

Ein Systembefehl besteht aus dem Befehlswort und eventuell aus einem oder mehreren Parametern, die spezielle Informationen über die Art der Ausführung des Befehls liefern. Parameter müssen durch ein Leerzeichen (blank) vom Befehlwort getrennt sein. Bei der Angabe des Befehlswortes genügen die ersten drei Buchstaben. Nach Drücken der Taste END OF LINE wird der Befehl ausgeführt oder es erscheint eine Fehlermeldung.

Liste der Systembefehle für den Bildschirm:

DRAW	Erzeugt einen Ausdruck der gesamten Information auf dem Bildschirm über Thermodrucker.
ERASE	Löscht den Bildschirm vollständig.
LDIMAGE	Eine auf einem Datenfile abgespeicherte Plotzeichnung wird auf den Bildschirm übertragen.
REVERSE	Stellt den Bildschirminhalt statt positiv negativ dar oder umgekehrt.
STIMAGE	Speichert eine auf dem Bildschirm befindliche Plot-Zeichnung auf ein Datenfile.

Nur für LDIMAGE und STIMAGE ist Option GDI erforderlich. Für DRAW, ERASE und REVERSE siehe Kapitel 6, Systembefehle.

LDIMAGE

SYSTEMBEFEHL: LDIMAGE

FUNKTION: Eine auf einem Datenfile abgespeicherte Plot-Zeichnung wird auf den Bildschirm übertragen.

FORMAT: LDI[MAGE][Filename][lib.-ref.]

mit: "Filename" als Name des Files, das die Zeichnung enthält. Ist kein Name angegeben, wird der Name SYSPLO angenommen.

"lib.-ref." kann in einer der folgenden Arten angegeben werden:

1. (Bibl.-Name, Einheit)
2. (Bibl.-Name)
3. (,Einheit)

wobei "Bibl.-Name" den Namen der (offenen) Bibliothek und "Einheit" den symbolischen Namen der Magnetplatten- oder Diskettenein-heit bedeutet, in der auf das angegebene File zurückgegriffen werden kann.

WIRKUNG: Das im Datenfile enthaltene Bild erscheint auf dem Bildschirm, wobei dort bereits vorhandene Texte oder Zeichnungen nicht gelöscht werden.

BEMERKUNGEN: Ist nur "Bibl.-Name" angegeben, wird das File auf der ersten offenen Bibliothek mit diesem Namen gesucht. Ist (Bibl.-Name, Einheit) angegeben, muß das File in der genannten Bibliothek der entsprechenden Einheit vorhanden sein.

Ist (,Einheit) angegeben, wird das File in der ersten offenen Bibliothek der entsprechenden Einheit gesucht.

Ist der Parameter "Bibliothek" nicht angegeben, wird das File zunächst auf der ersten offenen Bibliothek, später wenn nötig auf weiteren Bibliotheken gesucht, und zwar in der Reihenfolge, in der sie geöffnet wurden.

Nach LDIMAGE ist das System nicht automatisch zum Plotten initialisiert.

Die Ausführung von LDIMAGE kann durch EXIT oder CTRL und EXIT abgebrochen werden.

Bei Eingabe von LDIMAGE im Debugging Mode befindet sich das System nach der Ausführung weiterhin im Debugging Mode.

STIMAGE

SYSTEMBEFEHL: STIMAGE

FUNKTION: Speichert eine auf dem Bildschirm befindliche Plot-Zeichnung auf ein Datenfile.

FORMAT: STI[MAGE][Filename][,libr.-ref.][, n]

mit: "Filename" als Name des Files, auf dem die Zeichnung abgespeichert werden soll. Ist kein Name angegeben, wird der Name SYSPLO angenommen. "libr-ref." kann in einer der drei folgenden Arten angegeben werden:

1. (Bibl.-Name, Einheit)
2. (Bibl.-Name)
3. (, Einheit)

wobei "Bibl.-Name" den Namen der (offenen) Bibliothek und "Einheit" den symbolischen Namen der Magnetplatten oder Disketteneinheit bedeutet, in der auf das angegebene File zugegriffen werden kann.

n gibt in KByte die Größe des Puffers im Arbeitsspeicher an, in dem das Bild während der Ausführung von STIMAGE zwischengespeichert wird.

WIRKUNG: Die auf dem Bildschirm befindliche Plot-Zeichnung wird in dem angegebenen File abgespeichert. Dabei werden Informationen über die Erstellung der Zeichnung, wie z.B. die Parameter von INIMAGE und SCALE mit abgespeichert.

BEMERKUNGEN: Das angegebene File muß als sequentielles Datenfile kreiert sein. Ist als zweiter Parameter von STIMAGE nur ein Bibliotheksname angegeben, dann wird das angegebene File in der zuerst geöffneten Bibliothek mit diesem Namen gesucht. Ist nur die Einheit angegeben, dann wird es in der zuerst geöffneten Bibliothek dieser Einheit gesucht.
Ist der zweite Parameter weggelassen, dann wird es in der generell zuerst geöffneten Bibliothek gesucht.

9.5.7 Beispiele

1. Für Bildschirm

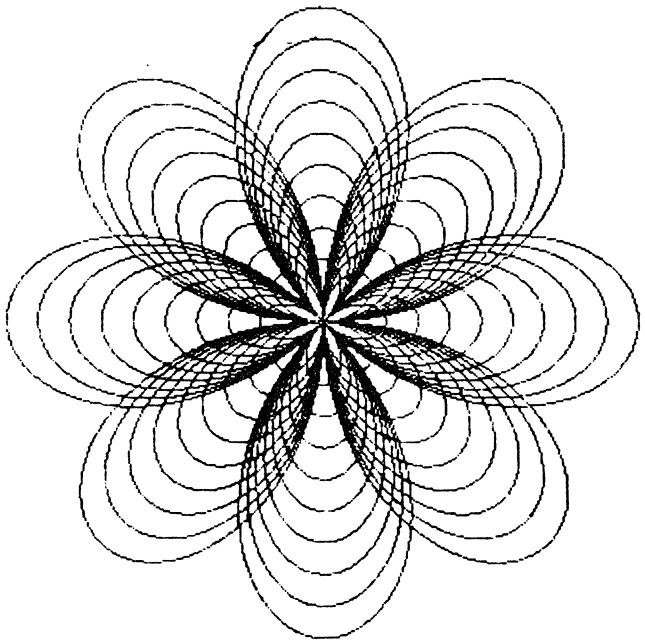
   (Die Ausgabe der Zeichnungen durch den Thermodrucker wird mit dem Systembefehl DRAW bewirkt)

   Programm-Listing und Zeichnung

```
FILE     BLUME

0100 INIMAGE FLOWER
0110 FRAME 8,5.6
0120 SCALE -4.3,4.3,-3,3
0130 FOR A=2.5 TO 0.5 STEP -0.25
0140 MOVE 0,0
0150 FOR T=0 TO 2*PI STEP PI/100
0155 DISP A,T
0160 LET B=SIN(2*T)
0170 PLOT A*B*SIN(T),A*B*COS(T)
0180 NEXT T
0190 NEXT A
0200 FOR A=2.5 TO 0.5 STEP -0.25
0210 MOVE A,0
0220 FOR T=0 TO 2*PI STEP PI/100
0225 DISP A,T
0230 LET B=COS(2*T)
0240 PLOT A*B*COS(T),A*B*SIN(T)
0250 NEXT T
0260 NEXT A
0270 DRAW
0280 END

END OF LISTING
```



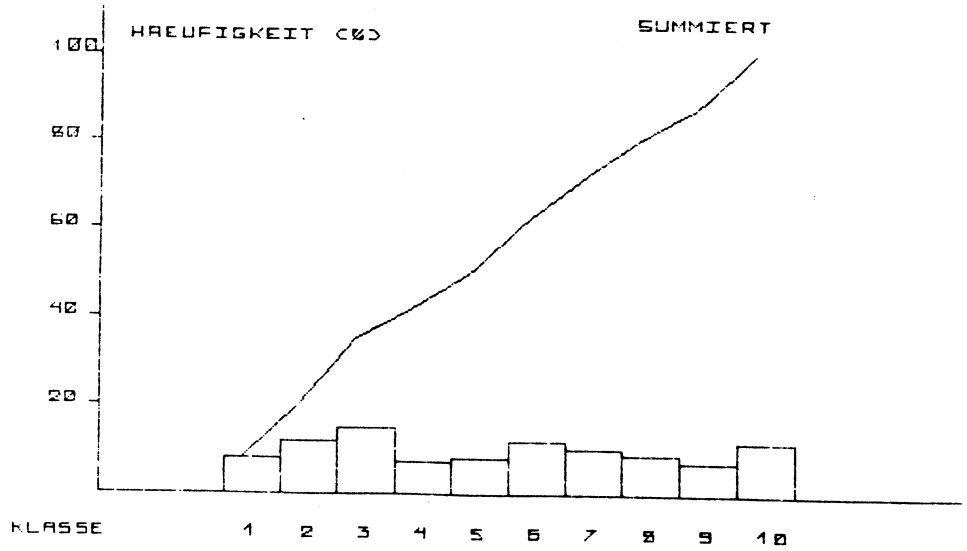
2. Histogramm (graphische Darstellung einer Häufigkeitsverteilung) für Bildschirm

```
FILE    *HIST

0010 REM  **************   HISTOGRAMM   **************
0020 .##.###↑↑↑↑
0030 DIM C(35),A(35)
0040 DISP "ANZAHL MESSWERTE";
0050 INPUT M
0060 DISP "UNTERE GRENZE DER MESSWERTE";
0070 INPUT X0
0072 DISP "OBERE GRENZE DER MESSWERTE";
0075 INPUT X1
0080 DISP "ANZAHL KLASSEN";
0090 INPUT L0
0095 LET D1=(X1-X0)/L0
0100 FOR I=1 TO L0 STEP 1
0110 DISP "Anzahl Messwerte in Klasse";I;
0120 INPUT C(I)
0130 NEXT I
0140 ININAGE PFILE,2
0150 LET L1=8
0160 LET L2=5.6
0170 FRAME L1,L2
0180 IF L0>15 THEN 220
0190 LET Q9=17
0200 LET Q1=-1.5
0210 GOTO 280
0220 IF L0>25 THEN 260
0230 LET Q9=27
0240 LET Q1=-2.5
0250 GOTO 280
0260 LET Q9=37
0270 LET Q1=-3.5
0280 SCALE Q1,Q9,-57,110
0290 LET S1=(Q9-Q1)/L1
0300 LET S2=167/L2
0310 MOVE Q9,0
0320 PLOT 0,0
0330 PLOT 0,110
0340 CSIZE .13,.13,0
0350 MOVE .5,103
0360 CPLOT "HAEUFIGKEIT (%)"
0370 MOVE L0+2-.11*10*S1,106
0380 CPLOT "SUMMIERT"
0390 LET B=0
0400 CSIZE .12,.12,0
0410 FOR I=1 TO 5 STEP 1
0420 LET B=B+20
0430 MOVE 0,B
0440 PLOT -.08*S1,B
0450 MOVE -.5*S1,B
0460 BUILD B$,B
0470 CPLOT B$
0480 NEXT I
0490 LET L=-.11*S2
0500 LET L7=.11*S1
0510 MOVE Q1,3*L
0520 CPLOT "KLASSE"
0530 FOR I=1 TO L0 STEP 1
0540 MOVE 1.3+I,3*L
0550 BUILD E$,I
0560 CPLOT E$
0570 NEXT I
```

```
0580 MOVE 1,10*L
0590 BUILD A$,D1
0600 CPLOT "KLASSENBREITE =";A$
0610 MOVE 35*L7,10*L
0620 BUILD A$,X0
0630 CPLOT "UNTERE GRENZE =";A$
0640 MOVE 1,13*L
0650 BUILD A$,L0
0660 CPLOT "ANZAHL KLASSEN =";A$
0670 MOVE 35*L7,13*L
0680 BUILD A$,X1
0690 CPLOT "OBERE GRENZE =";A$
0700 FOR I=1 TO L0 STEP 1
0710 LET A(I)=C(I)*100/M
0720 MOVE I+1.2,0
0730 PLOT I+1.2,A(I)
0740 PLOT 2.2+I,A(I)
0750 PLOT 2.2+I,0
0760 NEXT I
0770 FOR I=2 TO L0 STEP 1
0780 LET A(I)=A(I)+A(I-1)
0790 NEXT I
0800 MOVE 2.5,A(1)
0810 FOR I=2 TO L0 STEP 1
0820 PLOT I+1.5,A(I)
0830 NEXT I
0840 DRAW
0850 END

END OF LISTING
```



KLASSENBREITE = 1

ANZAHL KLASSEN = 10

UNTERE GRENZE = 0

OBERE GRENZE = 10

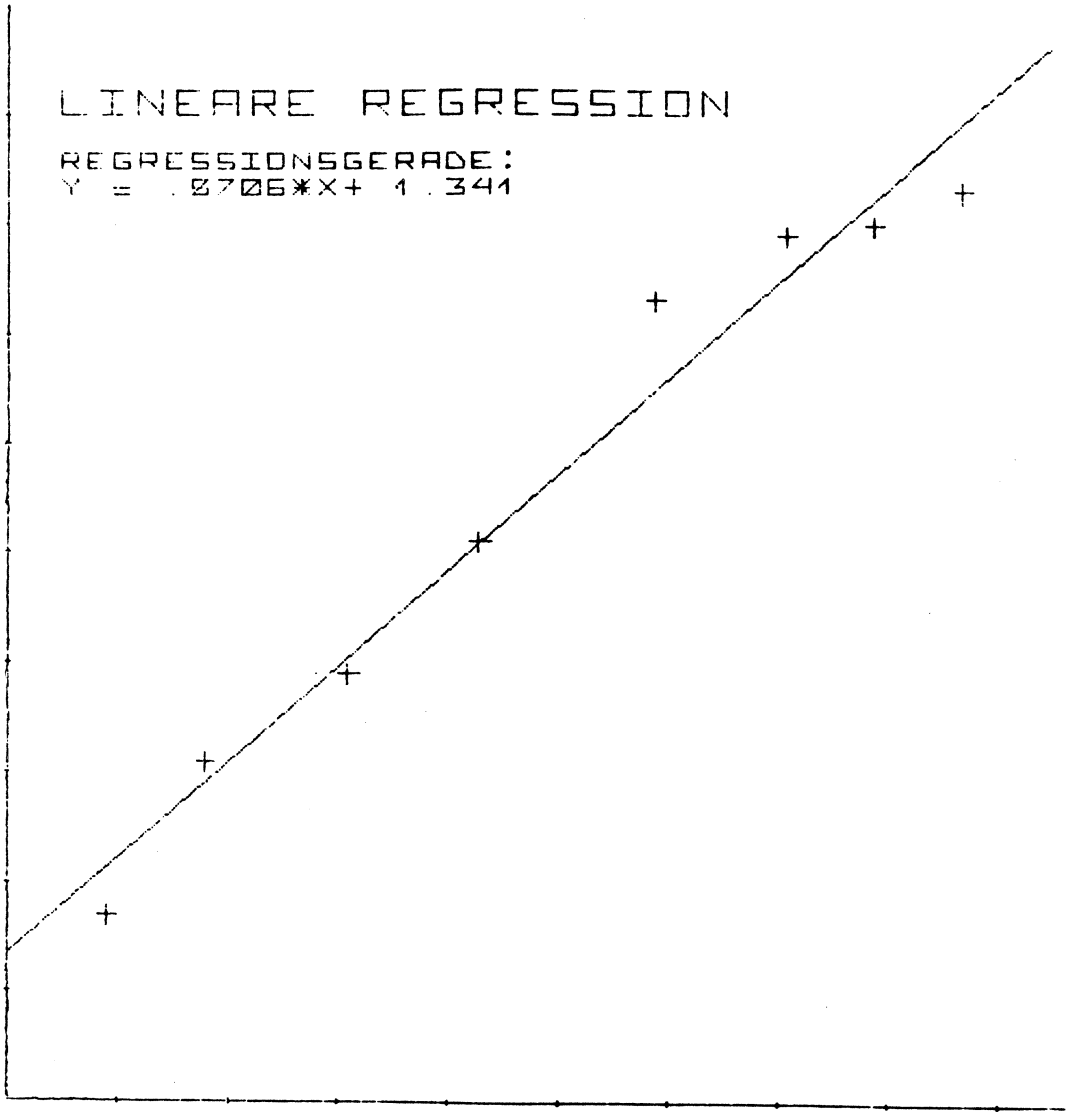
3. Lineare Regression (OPT PLO, Thermodrucker)

```
FILE    *LREG

0010 ININAGE PFILE,4
0020 DIM X(20),Y(20)
0030 FRAME 8,8
0040 SCALE -1,10,-1,10
0050 DISP "ANZAHL MESSWERTE";
0060 INPUT N2
0070 LET X0=Y0=X1=X2=0
0080 FOR I=1 TO N2 STEP 1
0090 DISP "KOORDINATEN X(";I;"), Y(";I;")";
0100 INPUT X(I),Y(I)
0110 PRINT ,"X(";I;") =";X(I),"Y(";I;") =";Y(I)
0120 LET X0=X0+X(I)
0130 LET Y0=Y0+Y(I)
0140 LET X1=X1+X(I)*Y(I)
0150 LET X2=X2+X(I)↑2
0160 NEXT I
0170 LET X0=X0/N2
0180 LET Y0=Y0/N2
0190 LET M=(X1-N2*X0*Y0)/(X2-N2*X0↑2)
0200 LET B=Y0-M*X0
0210 MOVE .5,9
0220 CSIZE .25,.25,0
0230 CPLOT "LINEARE REGRESSION"
0240 XAXIS 0,1,0,10
0250 YAXIS 0,1,0,10
0260 CSIZE .17,.17,0
0270 BUILD M$,M
0280 BUILD B$,B
0290 MOVE .5,8.5
0300 CPLOT "REGRESSIONSGERADE:",%
0310 CPLOT "Y ="+EXT$(M$,1,6)+"*X+"+EXT$(B$,1,6)
0320 FOR I=1 TO N2 STEP 1
0330 LET K9=FNC(X(I),Y(I))
0340 NEXT I
0350 MOVE 0,B
0360 PLOT 9.5,M*9.5+B
0370 DRAW
0380 DEF FNC(X,Y)
0390 MOVE X-.1,Y
0400 PLOT X+.1,Y
0410 MOVE X,Y+.1
0420 PLOT X,Y-.1
0430 LET FN*=1
0440 FNEND
0450 END

END OF LISTING
```

```
X( 1 ) = .9      Y( 1 ) = 1.7
X( 2 ) = 1.8     Y( 2 ) = 3.1
X( 3 ) = 3.1     Y( 3 ) = 3.9
X( 4 ) = 4.3     Y( 4 ) = 5.1
X( 5 ) = 5.9     Y( 5 ) = 7.3
X( 6 ) = 7.1     Y( 6 ) = 7.9
X( 7 ) = 7.9     Y( 7 ) = 8
X( 8 ) = 8.7     Y( 8 ) = 8.3
```



BEMERKUNGEN: Zunächst werden die Koordinatenwerte ausgedruckt. Die Markierungen der zugehörigen Punkte werden anschließend mit Hilfe der Funktion FNC in Zeile 380 bis 440 erzeugt.

Der erläuternde Text links oben in der Zeichnung wird mit CSIZE definiert (Zeilen 220 und 260). Die Zahlenwerte werden vor der Ausgabe mit CPLOT (310) in String umgewandelt (270 und 280).

10.	EINGABE UND EDITING EINES PROGRAMM- ODER TEXTFILES	10.1
10.1	Struktur eines Programmes	10.1
10.2	Struktur eines Textfiles	10.2
10.3	Eingabe eines Programmes	10.4
10.3.1	Vorbereitung des Hauptspeichers	10.4
10.3.2	Zeilennumerierung	10.4
10.3.3	Programmeingabe und Syntaxkontrolle	10.4
10.4	Eingabe eines Textfiles	10.5
10.5	Speichern von Programmen oder Textfiles	10.5
10.6	Korrektur (Editing) eines Programmes oder Textes	10.5
10.6.1	Hilfstasten und Befehle für das Editing	10.5
10.6.2	Editing am Beispiel eines Programmes	10.6
10.6.3	Editing im Display	10.9
10.7	Zusammenfassung der Möglichkeiten beim Erstellen und Editing von Programmen und Texten	10.10

10. EINGABE UND EDITING EINES PROGRAMM- ODER TEXTFILES

Dieses Kapitel vermittelt die notwendigen Kenntnisse, um ein Programm- oder Textfile zu erzeugen oder auszugeben.

10.1 Struktur eines Programmes

Ein BASIC-Programm besteht aus einer Folge von Zeilen (Anweisungen genannt). Jede Programmzeile muß eine Zeilennummer und mindestens ein BASIC-Schlüsselwort enthalten.

Mögliche Zusammensetzung einer BASIC-Programmzeile:

- Zeilennummer
- ein oder mehrere BASIC-Schlüsselwörter
- ein oder mehrere Operatoren
- ein oder mehrere Operanden

Die Zeilennummer:

identifiziert jede Programmzeile. Sie setzt sich aus vier Ziffern zusammen. Nummern, die kleiner sind als 1000, werden mit führenden Nullen ausgegeben.

Das BASIC-Schlüsselwort:

gibt die Funktion der Anweisung an.

Die Operatoren:

sind Symbole, die Operationen und Relationen zwischen den Operanden kennzeichnen.

Die Operanden:

können Konstante, Variable oder Ausdrücke sein.

In der letzten Zeile eines BASIC-Programmes muß die Anweisung END stehen.

Beispiel eines BASIC-Programmes:

```
FILE     BASIC

0010 REM *** BEISPIEL FUER EIN BASIC-PROGRAMM ***
0020 REM
0030 DCL SINGLE
0040 FOR I=1 TO 10 STEP 1
0050 DISP "EINGABE WERT";I;
0060 INPUT A
0070 PRINT "WERT";I;"=";A
0080 NEXT I
0090 PRINT
0100 END

END OF LISTING
```

10.2 Struktur eines Textfiles

Ein Textfile besteht aus einer Folge von Zeilen. Jede Zeile besteht aus einer Zeilennummer, gefolgt von Zeichen aus dem Zeichenvorrat des Systems (siehe Anhang). Jede Zeile umfasst maximal 80 Zeichen (einschließlich Zeilennummer).

Beispiel eines Textfiles:

```
TEXT
AUTO#
10 Beispiel fuer ein Textfile:
20
30 Ein beliebiger Text kann zeilenweise (mit Zeilennummer)
40 eingegeben werden.
50 Alle Editiermoeglichkeiten der Programme sind gueltig.
60
70 Textausgabe:
80 Wird der Befehl 'LIST' mit dem letzten Parameter 'X'
90 verwendet, erfolgt die Textausgabe ohne Zeilennummer.
100
110 Textspeicherung:
120 Texte koennen wie Programme mit 'SAVE' auf Disketten
130 gespeichert und mit 'OLD' wieder abgerufen werden.
140 Textfiles koennen wie sequentielle Datenfiles
150 mit 'READ' gelesen werden.
160
```

Textfiles bieten eine Reihe von Anwendungsmöglichkeiten:

1. Erstellen von Prozeduren:

   Commandfiles werden am einfachsten als Textfiles erstellt. Jede Zeile des Textfiles enthält eine ausführbare Prozedurzeile.

   Um aus diesem Textfile ein ausführbares Commandfile zu erhalten, muß es mit TRANSCODE als sequentielles Datenfile gespeichert werden.

   Für Korrekturen kann ein ausführbares Commandfile mit TRANSCODE (Option T) in ein Textfile umgewandelt werden.

2. Erstellen von Assembler-Programmen:

   Das Betriebssystem "P6066 - Assembler Environment" sieht die Erstellung der Assembler-Programme als Textfile vor. Bei der Assemblierung werden diese Textfiles in ausführbare Objektcodes übersetzt und als Objektfile gespeichert. Die Erstellung und Assemblierung solcher Assembler-Programme ist nur an der P6066 möglich. Sie können aber mit dem Systemen M30/M40 abgearbeitet werden.

3. Eingabe für Programme:

   Einige Anwenderprogramme und Standard-Packages von Olivetti sehen die Eingabe der Daten in ein Programm mit Hilfe von Textfiles vor. Es handelt sich dabei vor allem um Programmpakete, die eine Festlegung der Eingaben in Form einer eigenen Programmiersprache erlauben.

4. Time-Sharing-Betrieb:

   Zur Batch-Übertragung von Daten oder Programmen in einer anderen Programmiersprache an einen anderen Rechner.

5. Zur einfachen Verwaltung von Datenbeständen:

   Aufgrund der Korrekturmöglichkeiten und der einfachen Handhabung empfiehlt sich in vielen Anwendungen die Erstellung und Wertung von Datenbeständen über Textfiles, da diese mit Hilfe von einfachen Programmen in anders organisierte Datenfiles umgewandelt werden können.

10.3 Eingabe eines Programmes

(Siehe auch Kapitel 7.12, Struktur eines BASIC-Programmes.)

10.3.1 Vorbereitung des Hauptspeichers

Nach dem Laden des Betriebssystems ist das System für die Eingabe eines neuen Programmes vorbereitet. Wurden bereits andere Operationen ausgeführt, so ist vor der Erstellung eines neuen Programmes der Befehl NEW einzugeben.

10.3.2 Zeilennumerierung

Die Programmeingabe erfolgt zeilenweise. Die Zeilennummern (1-9999) sind vorteilhafterweise in aufsteigender Reihenfolge einzugeben, jedoch kann mit jeder beliebigen Nummer begonnen und in beliebigen Abständen numeriert werden.

Lücken in der Zeilennumerierung erleichtern das spätere Einfügen von Zeilen.

Mit der Anweisung AUTO # [1. Zeilennr.][,Abstand] können die Zeilennummern automatisch vorgegeben werden. Standardwerte: 10,10.

Eine nachträgliche Änderung dieser Numerierung kann über die Systembefehle RESEQUENCE und SHIFT erfolgen.

10.3.3 Programmeingabe und Syntaxkontrolle

Jede Zeile wird durch Drücken der Taste EOL abgeschlossen und sofort vom System analysiert. Ist die Zeile syntaktisch korrekt (siehe Kapitel 8), so wird sie übersetzt und in den Arbeitsspeicher übertragen.

Bei fehlerhafter Syntax wird die Zeile vom System nicht akzeptiert. Im Display erscheint eine entsprechende Fehlermeldung (siehe Anhang).

Die fehlerhafte Zeile bleibt aber im Tastaturbuffer gespeichert. Sie kann mit der Taste RECALL zur Korrektur ins Display zurückgeholt werden. Dabei zeigt der Pointer auf eine der möglichen Fehlerstellen. Nach der Korrektur wird die Zeile durch Drücken der Taste EOL in den Arbeitsspeicher übernommen.

Tritt also bei Eingabe einer Programmzeile eine Fehlermeldung aufgrund eines syntaktischen Fehlers auf (z.B. ERROR 113), so ist die Taste RECALL zu drücken und die Zeile zu korrigieren. Nach der Korrektur kann sie sofort mit EOL abgeschlossen werden.

Durch solche Korrekturen wird die automatische Zeilennumerierung nicht unterbrochen.

10.4 Eingabe eines Textfiles

Ein Textfile kann nur nach dem Befehl TEXT (EOL) eingegeben werden. Auch ein Textfile wird zeilenweise erstellt. Die Zeilennumerierung erfolgt wie unter 10.3.2 beschrieben. Jede Zeile wird dem Arbeitsspeicher durch Drücken der Taste EOL übergeben. Es erfolgt keine syntaktische Kontrolle, da in einer Textzeile alle ISO-Zeichen gemäß Anhang erlaubt sind.

10.5 Speichern von Programmen oder Textfiles

Durch das Ausschalten des Systems geht der Inhalt des Arbeitsspeichers verloren. Deshalb ist es wichtig, Programme und Textfiles nach deren Eingabe in einer Bibliothek zu speichern. Sie erhalten zu diesem Zweck einen Namen.

Die Anweisung "SAVE name" (Format siehe Kapitel 6) bewirkt die gewünschte Speicherung in einer Bibliothek. Der Inhalt des Arbeitsspeichers wird dadurch nicht verändert.

Beispiel:

SAVE TEST1, BIBL1	Speicherung des Inhaltes des Arbeitsspeichers unter dem Namen TEST1 in der Bibliothek BIBL1

10.6 Korrektur (Editing) eines Programmes oder Textes

10.6.1 Hilfstasten und Befehle für das Editing

Die Funktion der für das Editing verwendbaren Tasten und Befehle wurde in Kapitel 1 bzw. Kapitel 6 beschrieben.

10.6.2 Editing am Beispiel eines Programmes

Zum besseren Verständnis der Technik des Editing diene das folgende Programm zur Lösung von quadratischen Gleichungen.

Die wesentlichsten Punkte sind im Beispiel am linken Rand durch Nummern gekennzeichnet.

1. NEW
2. AUTO#
3. 10 DIZP "GIB DIE KOEFFIZIENTEN EIN";
ERROR 102
10 DISP "GIB DIE KOEFFIZIENTEN EIN";
20 INPUT A,B,C
30 DISP
40 A1=0
50 A2=0
4. AUTO# 30
30 PRINT
40 PRINT
50 PRINT "DIE GLEICHUNG:";A;"X↑2 +";B;"X +";C;"= 0 HAT";
60 IF A=0 THEN 280
70 R=-B/(2*A)
80 D=R*R-C/A
90 X1=X2=R
100 IF D>0 THEN 210
110 IF D<0 THEN 160
120 PRINT "GLEICHE LOESUNGEN"
130 PRINT
140 PRINT "X1=X2=";X1
150 GOTO390
160 D=SQR(-D)
170 PRINT "KOMPLEXE LOESUNGEN"
180 PRINT
190 PRINT "X1 - i*I1=";X1;"-i";D;" X2 + i*I2=";X2;"+ i";D
200 GOTO 390
210 D=SQR(D)
220 X1=X1-D
230 X2=X2+D
240 PRINT "ZWEI LOESUNGEN"
250 PRINT
260 PRINT "X1=";X1;" X2=";X2
270 GOTO 390
280 IF B=0 THEN 340
290 X1=-C/B
300 PRINT "EINE LOESUNG"
310 PRINT
320 PRINT "X1=";X1
330 GOTO 390
340 IF C=0 THEN 380
350 PRINT "KEINE LOESUNG"
360 PRINT
370 GOTO 390
380 PRINT "EINE UNBESTIMMTE LOESUNG"
390 DISP "EINE ANDERE GLEICHUNG";
400 RKB A$
410 IF PRINT$<>"NEIN" THEN 10
ERROR 113
5. 410 IF A$<>"NEIN" THEN 10
420 END

1. Da sich im Hauptspeicher bereits ein Programm befindet, ist der Befehl NEW einzugeben.

2. Die Eingabe des Befehls AUTO bewirkt die automatische Zeilennumerierung mit einem Abstand von 10, beginnend bei 10. Im Display wird 10 angezeigt.

3. Das System meldet in der Display-Zeile einen syntaktischen Fehler (siehe Anhang). Das Schlüsselwort DISP wurde nicht korrekt eingegeben. Nach dem Drücken der Taste RECALL erscheint im Display:

   10 DI$_{o}$ZP "GIB DIE KOEFFIZIENTEN EIN";

   Wird die Taste → gedrückt, so erscheint im Display:

   10 DIZ$_{o}$P "GIB DIE KOEFFIZIENTEN EIN";

   Nach CHAR DELETE erscheint im Display:

   10 DI$_{o}$P "GIB DIE KOEFFIZIENTEN EIN";

   Nach Drücken von S erscheint im Display:

   10 DIS$_{o}$P "GIB DIE KOEFFIZIENTEN EIN";

   Mit EOL wird die Korrektur der Zeile beendet.

4. Die Zeile wurde akzeptiert und in den Arbeitsspeicher übertragen. Jetzt kann die Eingabe mit der nächsten Anweisung fortgesetzt werden. In Zeile 30 hat der Operator DISP und in den Zeilen 40 und 50 A1=0 bzw. A2=0 eingetastet. Die beiden letzten Zeilen sollen überschrieben werden. Dazu müssen die Tasten CLEAR/RECALL und SHIFT gedrückt werden, was bewirkt, daß die Zeile im Tastaturbuffer gelöscht und der Pointer auf deren erste Position gesetzt wird. Die automatische Zeilennumerierung wurde dadurch unterbrochen. Durch Drücken von AUTO# 30 und EOL erscheint im Display die Zahl 30 und die automatische Zeilennumerierung ist wieder eingesetzt. Nun können die neuen Zeilen eingegeben und dadurch die bisherigen Zeilen 30 bis 50 überschrieben werden.

5. Die Anweisung END wurde eingegeben.

   Durch Drücken von CLEAR/RECALL und SHIFT wird die automatische Zeilennumerierung unterbrochen. Das System befindet sich im Command-Mode. Nun kann jeder beliebige Befehl eingeben werden. Das Programm liegt in ausführbarer Form vor und befindet sich im Arbeitsspeicher; es kann ausgeführt, gespeichert oder geändert werden.

6. Nachträglich soll noch die Anweisung für eine Zeilenschaltung eingefügt werden. Die Eingabe lautet:

   385 PRINT EOL

   Diese Zeile wird im Arbeitsspeicher zwischen den Zeilen 380 und 390 eingefügt. Durch Drücken von RES (Resequence) werden die Zeilen des Programmes nochmals durchnumeriert, mit einem Abstand von 10, beginnend mit 10.

Wie aus folgendem Listing ersichtlich, werden auch alle Sprungziele automatisch neu numeriert.

Listen des Programmes: Tasten LIST und EOL drücken.

6.

```
385 PRINT
RES


LIST
FILE


0010 DISP "GIB DIE KOEFFIZIENTEN EIN";
0020 INPUT A,B,C
0030 PRINT
0040 PRINT
0050 PRINT "DIE GLEICHUNG:";A;"X↑2 +";B;"X +";C;"= 0 HAT";
0060 IF A=0 THEN 280
0070 LET R=-B/(2*A)
0080 LET D=R*R-C/A
0090 LET X1=X2=R
0100 IF D>0 THEN 210
0110 IF D<0 THEN 160
0120 PRINT "GLEICHE LOESUNGEN"
0130 PRINT
0140 PRINT "X1=X2=";X1
0150 GOTO 400
0160 LET D=SQR(-D)
0170 PRINT "KOMPLEXE LOESUNGEN"
0180 PRINT
0190 PRINT "X1 - i*I1=";X1;"-i";D;" X2 + i*I2=";X2;"+ i";D
0200 GOTO 400
0210 LET D=SQR(D)
0220 LET X1=X1-D
0230 LET X2=X2+D
0240 PRINT "ZWEI LOESUNGEN"
0250 PRINT
0260 PRINT "X1=";X1;" X2=";X2
0270 GOTO 400
0280 IF B=0 THEN 340
0290 LET X1=-C/B
0300 PRINT "EINE LOESUNG"
0310 PRINT
0320 PRINT "X1=";X1
0330 GOTO 400
0340 IF C=0 THEN 380
0350 PRINT "KEINE LOESUNG"
0360 PRINT
0370 GOTO 400
0380 PRINT "EINE UNBESTIMMTE LOESUNG"
0390 PRINT
0400 DISP "EINE ANDERE GLEICHUNG";
0410 RKB A$
0420 IF A$<>"NEIN" THEN 10
0430 END

END OF LISTING
```

Ausführung des Programms:

```
RUN
GIB DIE KOEFFIZIENTEN EIN?
0,3,3


DIE GLEICHUNG: 0 X↑2 + 3 X + 9 = 0 HAT EINE LOESUNG

X1=-3
EINE ANDERE GLEICHUNG?
JA
GIB DIE KOEFFIZIENTEN EIN?
9,3,0


DIE GLEICHUNG: 9 X↑2 + 3 X + 0 = 0 HAT ZWEI LOESUNGEN

X1=-.33333333  X2=-1.0000000E-13
EINE ANDERE GLEICHUNG?
JA
GIB DIE KOEFFIZIENTEN EIN?
0,0,0


DIE GLEICHUNG: 0 X↑2 + 0 X + 0 = 0 HAT EINE UNBESTIMMTE LOESUNG

EINE ANDERE GLEICHUNG?
JA
GIB DIE KOEFFIZIENTEN EIN?
1,2,3


DIE GLEICHUNG: 1 X↑2 + 2 X + 3 = 0 HAT KOMPLEXE LOESUNGEN

X1 - i*I1=-1 -i 1.4142136  X2 + i*I2=-1 + i 1.4142136
EINE ANDERE GLEICHUNG?
NEIN
```

10.6.3 Editing im Display

Das Editing gilt auch für Programmzeilen im Display, unter Zuhilfenahme der Tasten ↓ und ↑ sowie des Befehles FETCH.

Überschreitet eine Zeile im Display durch das Editing 76 Zeichen, so wird das Ausgabeformat durch Unterdrückung nicht signifikanter Leerstellen beim Listen verändert, da die Zeilennummer immer mit führenden Nullen auf 4 Stellen aufgefüllt wird.

Beispiel zum Editing im Display:

Nach FETCH 120 EOL erscheint im Display:

```
o120 DISP
```

Diese Anweisung soll in PRINT abgeändert werden:

Man drückt SHIFT und →. Im Display erscheint:

```
120 DISP o
```

das heißt, der Pointer wurde ans Ende der Anweisung verschoben.

Durch viermaliges Drücken von CHAR DELETE wird DISP gelöscht.

Nach dem Drücken von SHIFT [PRINT A] erscheint im Display:

```
120 PRINT o
```

Durch die Eingabe von EOL wird die im Programm bestehende Zeile durch die soeben eingegebene ersetzt.

Eine schnellere Korrektur ist in diesem Falle auch durch Eingabe einer neuen Zeile 120 PRINT gegeben. Durch Drücken von EOL wird die alte Zeile 120 im Arbeitsspeicher durch die neue überschrieben.

Nachträgliches Löschen einer Zeile im Programm:

[DELETE LINE] 280 EOL

Zeile 280 im Arbeitsspeicher ist gelöscht.

10.7 Zusammenfassung der Möglichkeiten beim Erstellen und Editing von Programmen und Texten

Neueingabe		
- bei Programmen		NEW
- bei Texten		TEXT
Abruf eines		
- gespeicherten Programmes		OLD
- gespeicherten Textes		OLD
- als Datenfile gespeicherten Textes		TRANSCODE T
Speichern eines		
- Programmes	erstmals	SAVE
	nach Änderungen	REPLACE
- Textes	erstmals	SAVE
	nach Änderungen	REPLACE
	als Datenfile	TRANSCODE D

Zeilennumerierung für	
- Eingabe	AUTO#
- Neunumerierung	RESEQUENCE
- um in der Numerierung Platz zu schaffen	SHIFT
Umwandlung von	
- Programm in Text	DECOMPILE
- Text in Programm	COMPILE
- Text in Datenfile	TRANSCODE D
- Datenfile in Text	TRANSCODE T
Zusammenfügen	
- Textfile in Programm	LINK
- Textfile in Text	MERGE
Editing eines Programmes oder Textes:	
Abruf	
- einer Zeile	FETCH
- der vorangehenden Zeile	↑
- der nachfolgenden Zeile	↓
Löschen von Zeilen	DELETE
Drucken	LIST
Editing einer Zeile:	
- Pointer verschieben	
links/rechts	← →
fortwährend	REPEAT+←,→
an Anfang/Ende	SHIFT+←,→
Zeichen löschen	CHAR DELETE
Tastaturbuffer löschen	CLEAR
Zeichen ersetzen	CHAR DELETE +neues Zeichen
Zeichen einfügen	Pointer an richtige Stelle+neues Zeichen
Wiederholen von Zeichen	REPEAT + Zeichen

Bemerkung:

Alle in diesem Abschnitt angeführten Tasten sind in Kapitel 1 des "Benutzerhandbuchs", alle Befehle in Kapitel 6 beschrieben.

11.	CALCULATOR- UND DEBUGGING-MODE	11.1
11.1	Calculator-Mode	11.1
11.2	Einsetzen des Calculator-Modes	11.1
11.3	Festlegen der Art des Argumentes trigonometrischer Funktionen	11.2
11.4	Befehl zur Belegung der Funktionstasten	11.3
11.5	Lokale Variable und numerische Ausdrücke	11.4
11.5.1	Zahlendarstellung	11.5
11.5.2	Standardfunktionen	11.5
11.6	Verarbeitung der Anweisungen	11.5
11.7	Ausgabe und Ausgabeformat der Ergebnisse	11.6
11.8	Debugging-Mode	11.7
11.8.1	Erreichen und Verlassen des Debugging-Modes	11.7
11.8.2	Operationen im Debugging-Mode	11.8
	Behebbare Fehler	11.8
	Abfrage von Variablenwerten	11.10
	Wertzuweisung an Variable	11.11
11.8.3	Rechnen im Debugging-Mode	11.12
11.8.4	START-, STOP-Befehle im Debugging-Mode	11.13

11. CALCULATOR- UND DEBUGGING-MODE

11.1 Calculator-Mode

In der Betriebsart Calculator-Mode kann das System als Tischrechner verwendet werden, ohne daß die sich im Arbeitsspeicher befindlichen Programme und Textfiles dadurch beeinflußt werden. Im einzelnen lassen sich numerische Ausdrücke berechnen, die aus numerischen Konstanten, Standardfunktionen und/oder gespeicherten Werten zuvor durchgeführter Rechenoperationen bestehen.

Als Standardfunktionen stehen alle auch in einem BASIC-Programm verwendbaren Funktionen zur Verfügung. Für die trigonometrischen Funktionen kann zusätzlich festgelegt werden, ob mit Bogenmaß, Altgrad oder Neugrad gerechnet werden soll.

Das Ergebnis einer Operation läßt sich in einer von 4 speziellen Variablen abspeichern. Es wird am Bildschirm angezeigt, kann aber auch gedruckt werden. Dazu ist zuvor die Taste PRINT ALL zu betätigen. Die Funktionstasten lassen sich im Calculator-Mode wie durch ein BASIC-Programm belegen. Die Standardbelegung ist danach durch den Befehl LDKEYS wieder einsetzbar.

11.2 Einsetzen des Calculator-Modes

Der Calculator-Mode wird durch Eingabe des Befehls CAL eingesetzt und kann durch erneute Eingabe dieses oder eines beliebigen anderen Systembefehles wieder verlassen werden.

Zusätzlich kann beim Aufruf ein anderes Format als das Standardformat gewählt werden. Siehe dazu Abschnitt 8.7.

Im einzelnen sind folgende Eingaben zulässig:

- Eingabe des Befehles CAL: der Calculator-Mode wird verlassen.
- Eingabe eines System-Befehles: der Calculator-Mode wird verlassen und der Befehl ausgeführt.
- Eingabe einer Calculator-Mode-Zeile:
  - Befehl, daß bei der Berechnung trigonometrischer Funktionen das Argument als Bogenmaß, Alt- oder Neugrad interpretiert werden soll.
  - Befehl zur Belegung der Funktionstasten.
  - Anweisung zur Berechnung eines numerischen Ausdruckes.

Nicht akzeptiert werden:

- BASIC-Statements;
- Zeilennummern vor einem der oben angeführten Befehle;
- Variablennamen, die von Φ, Φ0, Φ1 oder Φ2 verschieden sind.

Alle Berechnungen im Calculator-Mode werden durch Drücken von EOL abgeschlossen.

11.3 Festlegen der Art des Argumentes trigonometrischer Funktionen

Die Festlegung der Arbeitsweise im Bogenmaß, bzw. in Altgrad oder Neugrad erfolgt durch einen der folgenden Befehle:

- SRAD	Berechnung im Bogenmaß
- SDEG	Berechnung in Altgrad
- SGRAD	Berechnung in Neugrad

Nach dem Einsetzen des Calculator-Modes wird "SRAD" angenommen, das heißt, die Argumente trigonometrischer Funktionen werden als Bogenmaß interpretiert.

Nach dem Befehl "SDEG" wird in Altgrad gerechnet. Dabei sind die Winkel in dezimaler Form (z.B. 25.7 Grad), nicht in der Form "Grad, Minute, Sekunde", einzugeben.

Nach dem Befehl "SGRAD" wird in Neugrad gerechnet. Auch hier müssen die Winkel in dezimaler Form eingegeben werden.

Die Verwendung von Alt- oder Neugrad im Calculator-Mode hat keinen Einfluß auf die Interpretation der Argumente trigonometrischer Funktionen in einem BASIC-Programm, dort wird nur im Bogenmaß gerechnet.

Für den Zusammenhang von Altgrad, Neugrad und Bogenmaß gilt:

100 Neugrad $\hat{=}$ 90 Altgrad $\hat{=}$ π/2

π = 3,1415.....

Beispiel:

```
CAL
COS (PI)
-1.0000000000000

SDEG
COS (180)
-1.0000000000000

SGRAD
COS (200)
-1.0000000000000

SRAD
COS (PI)
-1.0000000000000
```

11.4 Befehl zur Belegung der Funktionstasten

FORMAT: FKEY n, String-Konstante

"n" positive ganze Zahl, $1 \leq n \leq 16$.

"String-Konstante" beliebige Folge von ISO-Zeichen.

Wie in der BASIC-Anweisung "FKEY" kann ein EOL durch einen Doppelpunkt am Ende des Strings codiert werden, so daß das Drücken der EOL-Taste entfällt, wenn die Funktionstaste gedrückt ist.

Bemerkung:

Dem FKEY-Befehl darf keine Zeilennummer vorangehen. Die ursprüngliche Standardbelegung kann durch Eingabe des Befehls LDKEYS erreicht werden. Dabei wird der Calculator-Mode verlassen und der Command-Mode eingesetzt.

Beispiel:

Standardbelegung der Funktionstasten:

```
F1: STANDARD
F2: BELEGUNG
CALC-MODE                          Calculator-Mode einschalten.
FKEY  1, CALCULATOR-MODE-          Funktionstaste 1 belegen.

F1 F2                              Tasten F1 und F2 drücken.
CALCULATOR-MODE-BELEGUNG           Die modifizierte Belegung
                                   wird angezeigt.

FKEY  1, 3*5:                      Nach Drücken der Funktionstaste
F1                                 wird der String 3*5 als numeri-
                                   scher Ausdruck interpretiert,
                                   berechnet und

15.0000000000000                   das Ergebnis angezeigt.

LDKEYS
F1 F2
STANDARDBELEGUNG                   Die Standardbelegung wird
                                   wieder hergestellt und ange-
                                   zeigt. Der Calculator-Mode
                                   wird verlassen.
```

Bemerkung:

Die Summe der Länge der Strings, mit welchen die Funktionstasten belegt sind, darf 238 nicht überschreiten.

11.5 Lokale Variable und numerische Ausdrücke

Eine Anweisung zur Berechnung numerischer Ausdrücke hat folgende Form:

num.Ausdruck

Φ[n] = num.Ausdruck

"Φ" Dieses Symbol ist der Taste RESULT zugeordnet. Die Taste Result ist die zweite Taste links von der Taste RES bzw. AUTO.

"num.Ausdruck" numerischer Ausdruck.

"n" eine der Ziffern 0, 1 oder 2.

Φ, Φ0, Φ1, Φ2 sind die Namen lokaler Variablen. Diesen Variablen können die Resultate der im Calculator-Mode durchgeführten Berechnungen als Wert zugewiesen werden.

Eine Anweisung im Calculator-Mode kann durch die Taste EOL oder SUM abgeschlossen werden. Dadurch ergeben sich vier mögliche Formate:

- num.expr. EOL
  Der Wert des Ausdruckes "num.expr." wird berechnet und der lokalen Variablen Φ zugewiesen.

- RESULT n = num.expr. EOL
  Der Wert des numerischen Ausdruckes wird berechnet und der Variablen Φ n zugewiesen.

Die Variablen werden durch eine der folgenden Anweisungen neu initialisiert, das heißt, es wird ihnen der Wert 0 zugewiesen:

- 0 EOL — es wird Φ = 0 gesetzt.
- RESULT n = 0 EOL — es wird die Variable Φ n = 0 gesetzt.

Numerische Ausdrücke im Calculator-Mode können enthalten:

- numerische Konstante
- lokale Variable
- numerische Standardfunktionen (SIN, ...)
- Operationszeichen (+, *, -, /, ↑)

Reihenfolge und Priorität der Berechnung sind die gleichen wie in einem BASIC-Programm, sie können durch Setzen von Klammern beeinflußt werden.

11.5.1 Zahlendarstellung

Numerische Werte werden, wie in einem BASIC-Programm, auch im Calculator-Mode intern im Gleitkomma-Format (halblogarithmische Darstellung) dargestellt. Die Berechnung und Darstellung erfolgt in doppelter Genauigkeit. Jeder Zahl Z wird eine Mantisse m und ein Exponent n zugeordnet:

$$Z = m.10^n$$

"m" 13-stellige Mantisse der Form

X.YYYYYYYYYYY
$1 \leq X \leq 9,\ 0 \leq Y \leq 9$

"n" ganze Zahl, $-99 < n < 99$

Eine Zahl kann als ganze Zahl oder als Dezimalzahl eingegeben werden; sie darf bis zu 13 Ziffern (ohne Dezimalpunkt und Vorzeichen) enthalten:

248 -5 1234.567890123

Eine weitere Form der Eingabe ist das Gleitkomma-Format:

nEa

"n" Dezimalzahl mit maximal 13 Ziffern

"a" ganze Zahl zwischen -99 und +99

5,698E2 -678.3456E4 2E-27

Da alle eingegebenen Zahlen durch das System in die halblogarithmische Darstellung übersetzt werden, ist die gewählte Eingabeform für die Berechnung unerheblich.

11.5.2 Standardfunktionen

Die im Calculator-Mode verwendbaren numerischen Standardfunktionen sind die gleichen wie in der Programmierung mit BASIC. Die Liste der Funktionen ist in Kapitel 9 enthalten.

11.6 Verarbeitung der Anweisungen

Erfolgt der Abschluß der Eingabe einer Anweisung durch EOL, wird die Anweisung syntaktisch überprüft. Ein eventueller Syntax-Fehler wird angezeigt und die Tastatur mit Ausnahme der Tasten RECALL und CLEAR gesperrt.

Durch Drücken der Taste RECALL wird die fehlerhafte Anweisung im Display sichtbar. Die Stellung des Pointers gibt eine der möglichen Fehlerpositionen an. Die Fehler können mit den Editing-Routinen korrigiert und die korrigierte Anweisung danach durch EOL wieder eingegeben werden. Durch Drücken von CLEAR wird die fehlerhafte Anweisung gelöscht.

11.7 Ausgabe und Ausgabeformat der Ergebnisse

Im Display sind sichtbar:

- alle Eingaben
- alle Fehlermeldungen
- die Ergebnisse

Wird RESULT gedrückt, erscheint im Display das Zeichen "Φ" als Symbol der lokalen Variablen Φ.

Die Werte der lokalen Variablen können durch folgende Anweisungen abgefragt werden:

- RESULT EOL:	der aktuelle Wert der Variablen Φ wird ausgegeben, bleibt aber unverändert.
- RESULT n EOL	der aktuelle Wert der Variablen Φ n wird ausgegeben und der Wert von Φ durch diesen überschrieben; der Wert von Φ n bleibt unverändert.
- RESULT n = RESULT n EOL	der Wert von Φ n wird ausgegeben; der Wert aller Variablen (inkl. Φ) bleibt unverändert.

Üblicherweise erfolgt die Ausgabe von Ergebnissen im Standardformat. Es kann jedoch beim Aufruf des Calculator-Modes in ein anderes Ausgabeformat angegeben werden.

Erfolgt der Aufruf mit

CAL n

mit $0 \leqslant n \leqslant 13$, so erfolgt die Ausgabe der Ergebnisse grundsätzlich mit n Nachkommastellen.

Erfolgt der Aufruf mit

CAL FL

so werden alle Ergebnisse in Gleitkommadarstellung mit allen intern dargestellten Ziffern angezeigt.

11.8 Debugging-Mode

Der Debugging-Mode dient zum Testen und Prüfen von Programmen und Programmteilen. Treten während der Ausführung eines Programmes nicht behebbare Fehler auf, so ermöglicht der Übergang in den Debugging-Mode die Abfrage von Variablenwerten sowie Berechnungen zur Ermittlung der Fehlerursache. Der Debugging-Mode kann in diesem Falle nur durch Drücken der Taste BREAK und somit durch Übergang in den Command-Mode verlassen werden. In allen anderen Fällen sind folgende Operationen verfügbar:

- Korrektur behebbarer Fehler
- Abfrage von Variablenwerten
- Zuweisung neuer Werte an Variable
- Zeilenweise Verarbeitung eines Programmes
- Neubelegung der Funktionstasten
- Fortsetzung des Programmlaufes in einer beliebigen Zeile
- Anweisung zum Stoppen des Programmlaufes in einer beliebigen Zeile

Außerdem besteht die Möglichkeit, mit vom Benutzer definierten, ein- oder mehrzeiligen Funktionen in gleicher Weise wie mit Standardfunktionen zu arbeiten.

11.8.1 Erreichen und Verlassen des Debugging-Modes

Befindet sich das System im Command- oder Calculator-Mode, so wird der Debugging-Mode durch Ausführung des Befehles PREPARE erreicht.

Befindet sich das System im Running-Mode, wird der Debugging-Mode erreicht durch:

- Ausführung eines (programmierten) STOP-Statements.
- Ausführen der Funktione STEP (Taste EXIT).
- Auffinden eines behebbaren Fehlers durch das System während eines Programmlaufes.
- Ausführung eines zuvor eingegebenen "STOP Zeilennummer"-Befehles.
- Auftreten eines nicht behebbaren Fehlers.

Erwartet das System eine Tastatureingabe, wird der Debugging-Mode durch Ausführen der Funktion STEP (Taste EXIT) erreicht.

Der Debugging-Mode wird verlassen durch:

- Ausführen der Funktion BREAK (Ctrl. EXIT). In diesem Falle wird der Command-Mode erreicht.
- Ausführen der Funktion CONTINUE (EXIT). In diesem Falle wird die Ausführung des Programmes fortgesetzt.
- Eingabe des Befehles "START Zeilennummer". In diesem Falle wird die Ausführung des Programmes bei der angegebenen Zeilennummer fortgesetzt. Es ist vom Anwender sicherzustellen, daß eine Ausführung des Programmes ab der angegebenen Zeilennummer sinnvoll ist.
- Ausführen der Funktion STEP. Es wird nur ein Statement ausgeführt und dann wieder in den Debugging-Mode zurückgekehrt. Durch dieses Vorgehen wird die Verarbeitung des Programmes in Einzelschritten ermöglicht.
- Ausführen der Funktion BREAK nach Auftreten eines nicht behebbaren Fehlers.

11.8.2 Operationen im Debugging-Mode

Behebbare Fehler

Behebbare Fehler sind solche, für die im Betriebssystem eine Standardlösung vorgesehen ist.

Beispiele solcher Fehler sind unter anderem:

- Eine Variable soll verarbeitet werden, ohne daß ihr zuvor ein Wert zugewiesen wurde.
- Die Quadratwurzel aus einer negativen Zahl soll gezogen werden.
- Eine singuläre Matrix soll invertiert werden (d.h. Determinante = Null).
- Die angesprochene periphere Einheit ist nicht eingeschaltet.
- Der für die Erzeugung eines Bildes beim Plotten vereinbarte Buffer kann keine weiteren Punkte aufnehmen.

Der Benutzer hat die Möglichkeit, die Fehler durch explizite Wertzuweisungen oder Änderungen zu beheben oder nur die Taste EXIT zu drücken. Im letzteren Falle werden die Werte vom System zugewiesen.

Ist die Fehlerbehebung im Programm durch die Kontrolle von internen Interrupts vorgesehen, so erfolgt keine Fehlermeldung und der Debugging-Mode wird nicht erreicht.

Die behebbaren Fehler sind in der Liste der Fehlercodes gesondert aufgeführt. Zusätzlich sind dort Lösungen angegeben, die vom System als Standardwerte angenommen werden.

Beispiel:

```
NEW                                  Eingabe eines neuen Programmes.
10 X = Y↑2
20 PRINT
30 PRINT X
40 END

RUN
ERROR 1 IN LINE 10                   Im Display erscheint die
                                     Meldung, daß ein behebbarer
                                     Fehler aufgetreten ist. Die
                                     Variable Y ist nicht defi-
                                     niert.
```

Möglichkeiten:

```
1. CONTINUE
   0                                 Der Variablen Y wird der
                                     Wert 0 zugewiesen und 0↑2 = 0
                                     ausgedruckt.

2. Wertzuweisung durch den
   Benutzer
   Y = 2                             Direkte Wertzuweisung an Y.
   START 10                          Fortsetzung des Programmlaufes.
   4                                 Der Wert 2↑2 = 4 wird ausge-
                                     geben.
```

Bemerkung:

Wird ein behebbarer Fehler gemeldet, so ist die den Fehler auslösende Anweisung bereits ausgeführt. Wird eine von der Standardlösung verschiedene Lösung gewünscht, so sind entweder alle von dieser Anweisung betroffenen Ergebnisse zu korrigieren oder die Programmausführung muß mit START sinnvoll fortgesetzt werden.

Abfrage von Variablenwerten

Werte von Variablen können durch Eingabe des Namens der Variablen und anschließendem Drücken der Taste EOL abgefragt werden.

Beispiel:

```
NEW                                  Eingabe eines neuen Programmes.
10 A = 3
20 B = A↑2
30 C = B↑2
40 STOP
50 END

RUN                                  Start des Programmlaufes.
STOP IN LINE 40                      Durch das STOP-Statement
                                     wird das System in den Debug-
                                     ging-Mode versetzt.
A  EOL                               Der Wert von A wird abgefragt.
3                                    Der Wert wird ausgegeben.
C  EOL                               Der Wert der Variablen C
                                     wird abgefragt
81                                   und ausgegeben.
```

Wie im Calculator-Mode erfolgt die Ausgabe der Werte über das Display. Ein Ausdruck erfolgt nur, wenn die Taste PRINT ALL aktiviert ist.

Die Ausgabe erfolgt im Standardformat.

Die Werte alphanumerischer Variablen können ebenfalls abgefragt werden.

Beispiel:

```
NEW                                  Eingabe eines neuen Programmes.
10 A$ = "TEST"
20 B$ = "AUSGABE"
30 C$ = A$+B$
40 STOP
50 END

RUN                                  Start des Programmlaufes.
STOP IN LINE 40                      STOP-Anweisung in Zeile 40.
                                     Das System geht in den Debug-
                                     ging-Mode.
A$ EOL                               Die Variablenwerte werden
                                     abgefragt
TEST                                 und ausgegeben.
C$ EOL
TESTAUSGABE
```

Zu beachten ist jedoch: Ein Programm wird zeilenweise verarbeitet. Daher können nur Variable abgefragt werden, die zu diesem Zeitpunkt bereits einen definierten Wert aufweisen. Andernfalls erfolgt eine Fehlermeldung. Ein Beispiel soll diesen Sachverhalt erläutern:

```
NEW                          Eingabe eines neuen Programmes.
10 STOP
20 A = 5
30 END

RUN
STOP IN LINE 10              Ausführung des STOP-Statements.
                             Übergang in den Debugging-Mode.
A EOL                        Abfrage der Variablen A:
0                            Auf dem Display erscheint
ERROR 1                      kurz die Zahl 0 und anschlies-
                             send die Fehlermeldung ERROR 1,
                             da die Variable A zur Zeit
                             der Ausführung des STOP-State-
                             ments noch keinen Wert zuge-
                             wiesen bekommen hat. Bevor
                             die Wertzuweisung erfolgen
                             oder die Taste CONTINUE ge-
                             drückt werden kann, ist die
                             Fehlermeldung ERROR 1 durch
                             Drücken der Tasten SHIFT
                             und CLEAR zu löschen.
```

Nach Abfrage der Variablenwerte wird der Programmlauf durch Drücken der Taste CONTINUE fortgesetzt usw.

Wertzuweisung an Variable

Die Wertzuweisung an Variable geschieht durch folgende Anweisung:

```
numerische Variable = num.Ausdruck EOL
String-Variable     = Stringausdruck EOL
```

Wertzuweisungen können an numerische und alphanumerische Variable erfolgen. Diese Variablen können globale, im Programm verwendete Variable oder die lokalen Variablen Φ, Φ0 - Φ2 des Calculator-Modes sein. Im Debugging-Mode an globale Variable zugewiesene Werte bleiben nach Wiederaufnahme des Programmlaufes durch CONTINUE usw. erhalten. Jede im Programm bereits definierte Variable sowie alle Funktionen können verwendet werden.

Beispiel:

```
NEW                                   Eingabe eines neuen Programmes.
10 A = 3
20 B$ = "TEST"
30 PRINT A; B$
40 STOP
50 PRINT A; B$
60 END

RUN
3 TEST
STOP IN LINE 40
A = 100 EOL
B$ = "PROGRAMM" EOL
CONTINUE
100 PROGRAMM
```

11.8.3 Rechnen im Debugging-Mode

Der Debugging-Mode umfaßt alle auch im Calculator-Mode zur Verfügung stehenden Möglichkeiten. Auch das Eingabeformat einer Anweisung ist das gleiche (siehe Kapitel 12).

Insbesondere gilt:

- Es kann in Neugrad, Altgrad oder im Bogenmaß gerechnet werden.
- Die Funktionstasten können belegt werden.
- Die Variablen Φ, Φ0, Φ1 und Φ2 stehen zur Verfügung.

Die entsprechenden Befehle sind die gleichen wie im Calculator-Mode, es ist aber zu beachten, daß bei Fortsetzung des Programmlaufes nur noch im Bogenmaß gerechnet wird! Zusätzlich zu den Möglichkeiten für das manuelle Rechnen sind folgende Operationen ausführbar:

- Verwendung aller im BASIC-Programm angeführten Variablen und nicht nur der Variablen Φ, Φ0, Φ1 und Φ2. Beide Arten können gemischt verwendet werden.
- Verwendung von im Programm definierten Variablenwerten.
- Verwendung von im Programm durch DEFFN oder DEFFN/FNEND definierten Funktionen.

Beispiel 1: Bestimmung des Maximums zweier Zahlen

```
NEW                                  Eingabe eines neuen Programmes.
10 DEFFNA (X,Y)
20 IF X > Y THEN 50
30 FN* = Y
40 GOTO 60
50 FN* = X
60 FNEND
70 END

PRE                                  Die Preexecution wird durchge-
                                     führt. Das System befindet
                                     sich im Debugging-Mode.
FNA(50,20) EOL                       Die Funktion FNA wird als
50                                   Standardfunktion verwendet.
100 EOL                              Der lokalen Variablen Φ wird
                                     der Wert 100,
Φ0 = 200 EOL                         der lokalen Variablen Φ0
                                     der Wert 200 zugewiesen.
FNA(Φ, Φ0) EOL                       Der Wert der Funktion FNA
                                     wird ermittelt.
200
```

11.8.4 START-, STOP-Befehle im Debugging-Mode

Der Befehl "STOP ln EOL", worin "ln" die Zeilennummer ist, bewirkt, daß der Programmlauf unmittelbar vor Ausführung des Statements mit der Zeilennummer "ln" unterbrochen wird. Durch diesen Befehl wird der Debugging-Mode aber nicht verlassen. Dazu muß die Taste "SHIFT EXIT" oder "EXIT" wieder gedrückt oder ein Befehl "START ln" eingegeben werden.

Bemerkung:

Der Befehl "STOP ln" bleibt gültig, bis entweder ein anderer STOP-Befehl eingegeben oder das Programmende erreicht wird.

Der Debugging-Befehl "STOP ln" hat die gleiche Wirkung wie das BASIC-Statement "ln STOP".

Durch die Kombination von START- und STOP-Befehlen ist es möglich, einzelne Programmteile isoliert auszuführen:

```
STOP ln 1 EOL
START ln 2 EOL
```

Das Programmstück zwischen den Zeilennummern ln 2 (inklusive) und ln 1 (exklusive) wird ausgeführt.

A N H A N G

Meldungen des Betriebssystems	A.1
Bedienungshinweise	A.1
Informationsmeldungen	A.2
Fehlermeldungen	A.3
Fehlercodes	A.5
Liste der Fehlercodes	A.7
1. Behebbare Fehler während der Ausführung von BASIC-Programmen	A.7
2. Fehler während der Preexecution eines Programmes	A.8
3. Nicht behebbare Fehler während der Ausführung eines BASIC-Programmes	A.9
4. Fehlermeldungen des BASIC-Compilers	A.10
5. Fehlermeldungen im Zusammenhang mit den Diskstationen	A.11
6. Nicht behebbare Fehler im Zusammenhang mit peripheren Einheiten	A.12
7. Fehler der Eingabe oder Ausführung von Systembefehlen	A.13
8. Fehler beim Aufruf oder Ausführung von Dienstprogrammen	A.16
9. Sich auf PLOT-OPERATIONEN beziehende Fehlermeldungen	A.16
10. Fehlermeldungen für die Interrupt-Behandlung	A.17
11. Fehlermeldungen für einen nicht normalen Systemzustand	A.18
ISO-Code-Tabelle	A.19
Liste der darstellbaren Zeichen und ihre entsprechenden dezimalen Werte	A.21
Darstellung der ISO-Zeichen der Spalten 0 und 1 der ISO-Code-Tabelle	A.21

A N H A N G

MELDUNGEN DES BETRIEBSSYSTEMS

Während der Programmausführung gibt das Betriebssystem des Systems Meldungen über den Bildschirm aus, die dazu dienen, einerseits den Anwender über den aktuellen Betriebszustand der Maschine zu informieren, und andererseits sowohl auf Programm- als auch auf Bedienungsfehler aufmerksam zu machen.

Drei Arten von Systemmeldungen können unterschieden werden:

1. Bedienungshinweise
2. Informationsmeldungen
3. Fehlermeldungen bei der Ausführung von
   - BASIC-Anweisungen
   - Systembefehlen
   - Dienstprogrammen

Nachstehend folgt eine kurze Beschreibung jeder Meldungsart.

Die Bedienungshinweise und Informationsmeldungen sind selbsterklärend. Die Fehlermeldungen sind in einer kompletten Liste der möglichen Codes enthalten, mit einer allgemeinen Anleitung zur Behebung dieser Fehler.

Bedienungshinweise

Die Meldungen signalisieren dem Anwender, dass eine Eingabe nicht korrekt erfolgte.

Werden zum Beispiel als Antwort auf eine Inputanforderung einer INPUT-Anweisung zu viele Daten eingegeben, so erscheint die Meldung

TOO MUCH INPUT - EXCESS IGNORED

In der Folge werden nur so viele Daten an Variable zugewiesen, wie angefordert wurden.

Es kann jedoch vorkommen, dass das System, aufgrund falscher Eingaben, Eingabezeilen nicht in der vom Anwender beabsichtigten Form interpretiert (z.B. Komma statt Dezimalpunkt bei der Eingabe von Dezimalzahlen).

Ist ein solcher Fehler Ursache der Meldung, so kann im allgemeinen eine Korrektur der fehlerhaften Eingabe über die im Programm vorhandenen Korrekturroutinen erfolgen.

Bei der Ausführung einer INPUT-Anweisung kann die Meldung

INCORRECT FORMAT - RETYPE LINE

auftreten. In diesem Falle wurde anstelle einer Zahl ein String eingegeben. Das System aber wartet auf die Eingabe der korrekten Zeile.

Bemerkung:

Wird anstelle eines Strings eine Zahl eingegeben, so kann dies vom System nicht als Fehler erkannt werden, da Strings auch aus einer Folge von Ziffern bestehen können.

Informations meldungen

Die Informationsmeldungen wird dem Anwender Auskunft über den Betriebszustand des Systems.

Die Meldung

READY

bedeutet, dass das System zur Ausführung von Systembefehlen bereit ist, das heisst, sich im Command-Mode befindet.

Die Meldung

PROGRAM name READY TO RUN

zeigt bei aktivierter Taste PRINT ALL an, dass die Preexecution infolge eines PREPARE-Befehles erfolgreich beendet wurde.

Die Ausführung des Programmes kann durch Drücken der Konsoltaste CONTINUE gestartet werden. Soll das Programm nicht ausgeführt werden, so ist BREAK zu drücken.

Die Meldung

PROGRAM name RUNNING

zeigt an, dass das Programm namens "name" ausgeführt wird. Diese Meldung kann durch Displaymeldungen des Anwenderprogrammes ersetzt sein.

Bei der Ausführung von Befehlen und Dienstprogrammen wird der Anwender über die erforderlichen Bedienungshinweise informiert. Die Bedeutung dieser Meldungen kann der Beschreibung des gewählten Befehles oder Dienstprogrammes entnommen werden.

Diese Meldungen bestehen oft aus zwei Teilen, wobei als letztes Zeichen ">" im Display steht. Der zweite Teil der Meldung kann dann durch Drücken der Tasten SHIFT und → sichtbar gemacht werden.

Soll während der Ausführung eines Dienstprogrammes oder DCH eine Diskette oder die Wechselplatte der DCU ausgetauscht werden, so treten Meldungen der Form

LOAD DISK ON...
RESTORE DISK

auf. Der Diskwechsel kann danach ohne weitere Eingabe durchgeführt werden. Anschliessend ist die Arbeit mit CONTINUE fortzusetzen.

Als Warnmeldungen können auftreten:

ACTION ON UNIT ...
LISTED LIBRARIES WILL BE DELETED

Diese Meldungen zeigen an, dass ein bestehender Inhalt einer Disk oder Diskette zerstört wird. Soll die Operation durchgeführt werden, so ist der Ablauf mit CONTINUE fortzusetzen. Wird die Konsoltaste BREAK gedrückt, so kann die Operation ohne Zerstörung des bestehenden Inhaltes abgebrochen werden.

Bemerkung:

Wurde CONTINUE nicht gedrückt, so werden Eingaben über die Tastatur mit ERROR 190 beantwortet. Anschliessend sind die Tasten RECALL, CONTINUE und END OF LINE zu drücken, damit die Eingabe akzeptiert wird.

Fehlermeldungen

Diese Meldungen bezeichnen Fehler, die bei Anwendung von

- Systembefehlen
- BASIC-Anweisungen
- Dienstprogrammen

auftreten können.

Die den Fehler durch eine entsprechende Nummer genau spezifizierenden Meldungen können in vier Gruppen eingeteilt werden:

1. Warnmeldungen
   Treten beim Initialisieren des Systems nicht den gespeicherten Konfigurationen entsprechende Konditionen auf, so werden diesbezügliche Fehlermeldungen gedruckt. Sie können sich auf nicht vorhandene Disketten, Bibliotheken oder konfigurierte Peripherien beziehen. Eine sofortige Behebung durch den Anwender ist nicht erforderlich.

2. Syntaktische Fehler

   Sie treten bei fehlerhafter Eingabe von Systembefehlen oder BASIC-Anweisungen auf.

3. Preexecution-Fehler

Diese Meldungen weisen vor der Ausführung eines Programmes auf Fehler in dessen Struktur hin.

4. Ausführungsfehler

Diese Fehlermeldungen treten auf, wenn während der Ausführung eines Programmes, Systembefehles oder Dienstprogrammes unzulässige oder die Ausführung verhindernde Konditionen auftreten.

Syntaktische Fehler werden vom System bereits bei der Eingabe erkannt. Wird nach der Fehlermeldung die Taste RECALL gedrückt, so kann die Eingabe sofort korrigiert werden. Der Pointer befindet sich in diesem Falle an der Stelle der Eingabezeile, an welcher das System den Fehler erkannt hat. Nach der Korrektur kann die Eingabe des Befehles durch Drücken von END OF LINE abgeschlossen werden.

Während der Preexecution erkannte Fehler werden bei der Ausführung des Systembefehles PREPARE oder RUN gemeldet und beziehen sich auf die logische Struktur des Programmes. Diese Fehlermeldungen werden ausgedruckt und das System geht in den Command-Mode, so dass alle notwendigen Korrekturen auf einfache Weise durchgeführt werden können.

Ausführungsfehler können während der Ausführung eines Programmes entweder aufgrund von Programmfehlern oder aufgrund unzulässiger Datenkombinationen auftreten. Sie können entweder behebbar oder nicht behebbar sein.

Behebbare Fehler können während der Ausführung des Programmes korrigiert werden und erfordern keinen Abbruch des Programmlaufes.

Wird ein solcher Fehler erkannt, so erscheint eine entsprechende Fehlermeldung und das System geht in den Debugging-Mode.

Die behebbaren Fehler treten meistens infolge von unzulässigen Variablenwerten auf. Das System sieht für diese Fälle eine Standardlösung vor. Wird sie akzeptiert, so kann die Ausführung des Programmes durch Drücken von CONTINUE wieder aufgenommen werden.

Ist die Standardlösung für den speziellen Fall nicht günstig, so können die Werte der Variablen mit den Möglichkeiten des Debugging-Modes korrigiert und das Programm anschliessend wieder aufgenommen werden. Dabei ist jedoch zu beachten, dass die Ausführung mit der nächsten Anweisung fortgesetzt wird.

Tritt ein behebbarer Fehler aufgrund einer Variablen auf, der noch kein Wert zugewiesen wurde (ERROR 1), so wird der Wert Null bzw. Nullstring für die Ausführung dieses Statements angenommen. Die Variable bleibt jedoch weiterhin nicht initialisiert.

Die Fortsetzung der Programmausführung kann nach einem behebbaren Fehler mit allen Möglichkeiten des Debugging-Modes wieder aufgenommen werden. Es kann also mit CONTINUE die Ausführung mit der nächsten Zeile fortgesetzt, mit STEP das Programm schrittweise verarbeitet oder mit START ab einer beliebigen Zeile ausgeführt werden.

Fehlercodes

Fehlermeldungen werden aufgrund eines numerischen Fehlercodes identifiziert. Bei Fehlermeldungen der Preexecution und bei Ausführungsfehlern wird, wenn möglich, zusätzlich die Zeilennummer der fehlerhaften Programmzeile angegeben (Beispiel: ERROR 6 IN LINE 155).

Im folgenden Abschnitt sind die möglichen Fehlercodes mit einer Beschreibung aufgelistet, unterteilt in die oben beschriebenen Gruppen.

- Fehler mit den Nummern 1 bis 16 beziehen sich auf behebbare Fehler bei der Ausführung eines BASIC-Programmes.
- Fehler mit den Codes 40 bis 57 können bei der Preexecution eines Programmes auftreten.
- Die Fehlercodes 63 bis 98 bezeichnen nicht behebbare Ausführungsfehler.
- Codes zwischen 100 und 128 können bei fehlerhafter Eingabe von BASIC-Zeilen oder bei der Compilierung eines Textfiles auftreten.
- Die Fehlercodes mit den Nummern 131 bis 160 beziehen sich auf die Systeminstallation und auf Fehler, die beim Zugriff auf eine Disk- oder Diskettenstation auftreten.
- Nicht behebbare Fehler in Verbindung mit peripheren Einheiten werden durch die Codes 162 bis 172 gemeldet.
- Fehler, die bei der Eingabe oder Ausführung von Systembefehlen auftreten, tragen Nummern zwischen 173 und 227.
- Die Codes 228 bis 235 zeigen Fehler bei der Ausführung von Dienstprogrammen an.
- Behebbare und nicht behebbare Fehler der Option PLOT werden in den Codes 236 bis 254 beschrieben.
- Bei fehlerhaften Konditionen bei der Behandlung von Interrupts treten Fehler mit den Nummern 280 bis 284 auf.
- Abschliessend werden noch die einen Systemabbruch anzeigenden Fehler und die Meldungen für nicht betriebsbereite Einheiten angeführt.

LISTE DER FEHLERCODES

1. Behebbare Fehler während der Ausführung von BASIC - Programmen

Fehlercode	Bedeutung
1	Eine numerische oder alphanumerische Variable wurde nicht initialisiert. Es wird für das auszuführende Statement der Wert 0 bzw. der Leerstring angenommen, die Variable gilt jedoch weiterhin als nicht initialisiert.
2	Fehlerhaftes Argument in einer Stringfunktion
3	Overflow (z.B. bei Division durch Ø). Als Ergebnis der Operation wird die größte darstellbare Zahl mit dem richtigen Vorzeichen angenommen.
4	Underflow. Als Ergebnis der Operation wird Null angenommen.
6	Entweder ist das Argument einer Funktion außerhalb der zulässigen Grenzen, oder es soll die Wurzel einer negativen Zahl berechnet werden. Es wird die Wurzel des Absolutbetrages berechnet.
7	Bei einer Stringoperation entsteht ein String mit einer Länge von mehr als 1023 Zeichen. Der String wird nach der 1023. Stelle abgeschnitten.
8	Der String, der an eine Stringvariable zugewiesen werden soll, ist länger als die deklarierte Länge der Stringvariablen. Der String wird entsprechend der deklarierten Länge der Stringvariablen abgeschnitten.
9	Es soll der Logarithmus einer negativen Zahl berechnet werden. Es wird der Logarithmus des Absolutbetrages berechnet.
10	Es wird versucht, den Logarithmus von Null zu berechnen. Als Ergebnis wird -9.999999999999E+99 gesetzt.
11	Eine negative Zahl hat einen nicht ganzzahligen Exponenten. Es wird mit dem Absolutbetrag der Basis gerechnet.
12	Der Wert Null hat einen negativen Exponenten. Als Ergebnis wird die größte darstellbare Zahl (+9.999999999999E+99) gesetzt.
13	Die zu invertierende Matrix ist singulär (d.h. die Determinante ist gleich Null). die Ergebnismatrix enthält nicht definierte Werte, eine nachfolgende Verwendung der Funktion DET liefert jedoch ein konkretes Ergebnis.
14	Die adressierte Peripherie ist nicht angeschlossen oder eingeschaltet, oder es tritt während des Betriebes eine abnormale Kondition auf. Die Übertragung wird nicht ausgeführt.
15	Übertragungsfehler während eines Datenaustausches mit einer Peripherie. Die Datenübertragung wird nicht ausgeführt.
16	Beim letzten überlappten Input/Output für die gleiche Peripherie ist ein Übertragungsfehler aufgetreten.

2. Fehler während der Preexecution eines Programmes

Fehlercodes	Bedeutung
40	Unerlaubter Sprung: a) Sprung zu einer nicht existierenden Zeilennummer b) Sprung aus einer mehrzeiligen Funktion oder von außen in den Vereinbarungsteil einer mehrzeiligen Funktion c) Sprung von außen in eine FOR/NEXT-Schleife
41	NEXT ohne zugehörendes FOR oder Überschneidung von FOR/NEXT-Schleifen.
42	Innerhalb der Definition einer mehrzeiligen Funktion wird eine andere mehrzeilige Funktion definiert.
43	Eine aufzurufende Funktion wurde nicht definiert. (Fehler kann möglicherweise nur durch **DEC** und **COM** behoben werden.)
44	Es sind mehr als 15 FOR/NEXT-Schleifen geschachtelt.
45	a) FN*, FN*$, FNEND treten außerhalb der Definition einer mehrzeiligen Funktion auf b) In einer Definition einer mehrzeiligen Stringfunktion wird die Pseudovariable FN* oder in der Definition einer mehrzeiligen numerischen Funktion wird die Pseudovariable FN*$ verwendet.
46	Zwei- oder mehrere ineinander geschachtelte FOR/NEXT-Schleifen haben dieselbe Laufvariable.
47	FOR ohne ein nachfolgendes NEXT mit der gleichen Laufbariablen. Dieser Fehler kann weitere Fehlermeldungen während der Preexecution zur Folge haben.
48	In der Definition einer mehrzeiligen Funktion fehlt die **FNEND**-Anweisung.
49	Eine einfach indizierte und eine doppelt indizierte Variable haben denselben Namen.
50	**END** hat nicht die höchste Zeilennummer im Programm.
51	Das Programm enthält keine **END**-Anweisung.
52	Das auszuführende Programm enthält nicht compilierte Zeilen. Der Fehler tritt auf, falls die bei der Compilierung eines Textfiles gemeldeten Fehler nicht korrigiert wurden.
53	In der Definition einer mehrzeiligen Funktion erfolgt keine Wertzuweisung an die Pseudovariable FN* bzw. FN*$
54	Für eine **PRINT USING**-Anweisdung (**DISP USING**, ...) fehlt die zugehörende Formatvereinbarung.
55	Innerhalb der Definition einer mehrzeiligen Funktion befindet sich eine **STOP**-Anweisung.
56	Der für den **COMMON**-Bereich reservierte Platz ist für die Speicherzuweisung der als **COMMON** deklarierten Variablen zu klein.
57	Das File im Arbeitsspeicher hat keinen Namen.

3. Nicht behebbare Fehler während der Ausführung eines BASIC-Programmes

Fehlercodes	Bedeutung
63	Das System ist so konfiguriert, daß nur Programme mit bereits durchgeführter Preexekution ausgeführt werden können (Betriebssystem R0). Das Programm muß mit einem vollständigen (programmierbaren) System preexekutiert und anschließend auf die betreffende Disk gebracht werden.
65	Während der Programmausführung wird der zur Verfügung stehende Bereich des Anwenderspeichers überschritten (z.B. bei zu tiefer Schachtelung von Unterprogrammen). Das System geht direkt in den Command-Mode über.
66	Der Index einer indizierten Variablen hat einen nicht zulässigen Wert (d.h. kleiner als 1 oder größer als die maximale Dimension dieser Variablen).
67	Die Anweisung verlangt die Vereinbarung von Dimensionen einer Matrix, die größer sind als die maximalen Dimensionen.
68	Beim Aufruf der Programmausführung beginnend mit einer angeführten Zeilennummer (nach **RUN**-Zeilennummer, oder **START** Zeilennummer) tritt ein **NEXT** ohne zugehörendes **FOR** auf.
69	Die Parameter im Aufruf einer Funktion stimmen in ihrem Typ nicht mit den Parametern der Definition dieser Funktion überein.
70	Es soll ein **RETURN**-Statement ohne vorangehendes **GOSUB** ausgeführt werden.
71	Die Summe der Längen der Strings, die den Funktionstasten zugewiesen werden sollen, ist größer als die maximal zulässige Länge (238 Zeichen). Die Belegungen von **FKEY**-Tasten können durch **FKEY n**, gelöscht werden.
72	Die Anzahl der Parameter in einem Funktionsaufruf stimmt nicht überein mit der Anzahl der Parameter in der Definition der Funktion.
73	Die aktuellen Dimensionen einer Matrix erlauben nicht die Ausführung der gewünschten Operationen (z.B. nicht quadratische Matrix bei einer Inversion).
74	Innerhalb einer definierten Funktion wurde die maximal zulässige Anzahl von Aufrufen anderer definierter Funktionen (max. 255) überschritten.
75	Die zur Ausführung des Programmes benötigte Option wurde nicht in den Arbeitsspeicher geladen. Für die Behebung des Fehlers ist der Systembefehl **OPT...** unter Angabe der benötigten Option auszuführen.
76	Es wird versucht, ein File zu öffnen, das bei der vorherigen Ausführung eines Programms nicht physisch geschlossen wurde. Physisch offen gebliebene Files sind im CATALOG mit **OPEN** gekennzeichnet. Ohne genaueste Kenntnis des internen Programmablaufs muß auf der letzten Datensicherungskopie aufgesetzt werden. Alle seit diesem Stand angefallenen Arbeiten sind zu wiederholen!!
77	Es fehlt entweder das **FILES**-Statement oder der Filedesignator ist kleiner als 1 oder größer als die Anzahl der Files in der FILES-Anweisung.
78	Ein File ist nicht entsprechend dem Typ der auszuführenden Operationen geöffnet.
80	Der Wert des Wortdesignators einer **SETW**-Anweisung ist größer als es der Länge des entsprechenden Files entspricht.
81	Innerhalb eines Programmes wird versicht, ein bereits offenes File zu öffnen.

Fehlercodes	Bedeutung
82	Der verfügbare Platz in einem externen File reicht nicht für die Ausführung der verlangten Operation.
83	Das angesprochene File ist nicht geöffnet.
84	Es wurde bei einer Lese- oder Schreiboperation das Fileende ohne **EOF**-Angabe erreicht.
85	Das Argument einer **TAB**-Funktion in einer **PRINT** oder **DISP**-Anweisung ist kleiner als 1.
86	Einer numerischen Variablen soll ein String zugewiesen werden.
87	Die deklarierte Länge der Stringvariablen in einer **BBUILD**-Anweisung reicht nicht für die Ausführung der Anweisung.
88	Zu wenig Daten im internen File bei der Ausführung einer **READ**-Anweisung oder im Stringausdruck bei der Ausführung von **ASSIGN**.
89	die Formatfelder stimmen in ihrem Typ nicht mit den Daten der aufrufenden **PRINT USNING** oder **DISP USING**-Anweisung überein.
90	Ein Wert kleiner als Ø oder größer als 255 soll in ein **ISO**-Zeichen umgewandelt werden.
91	Der Ausdruck, der die Anzahl der in einer **CONVERT**-Anweisung umzuwandelnden Vectorelemente festlegt, ergibt einen negativen Wert.
92	Der in einer **CHAIN**-Anweisung angegebene Filename hat ein ungültiges Format.
93	Die Daten eines externen Files stimmen in ihrem Typ nicht mit den Variablen der entsprechenden **READ**-Anweisung überein. Derselbe Fehler tritt auch bei der Ausführung einer **BASSIGN**-Anweisung auf, falls der Typ der Daten und der entsprechenden Variablen nicht übereinstimmt.
96	Der Wortdesignator einer **SETW**:-Anweisung hat einen Wert kleiner 1.
97	Eine **SCRATCH**:- oder **APPEND**:-Anweisung bezieht sich auf ein Randomfile.
98	Die Anzahl der angegebenen Parameter in einer **WHERE**:-Anweisung verlangt, daß das sequentielle File für das Lesen geöffnet ist.

4. Fehlermeldungen des BASIC-Compilers

Fehlercodes	Bedeutung
100	Eine Programmzeile besteht nur aus der Zeilennummer.
101	a) Unzulässige Zeilennummer b) Bei Ausführung des Befehls **LINK** würden sich Zeilennummern überschneiden.
102	Unzulässiges Schlüsselwort
103	Unzulässiger Parameter
104	Unzulässiger Ausdruck
105	Der Operator ist für den Typ der Operanden nicht zulässig.
106	In einem Aufruf einer Funktion stimmt die Anzahl oder Typ der Paramter nicht mit den verlangten Parametern überein.
107	Ungültiges Format für einen Filenamen
109	Nicht interpretierbarer Syntaxfehler
110	Der Name der Funktion in der **DEF**-Anweisung kommt bereits in einer DEF-Anweisung mit einer anderen Zeilennummer vor.

Fehlercodes	Bedeutung
111	Es wurden bereits zuviele verschiedene Zeilennummern als Sprungziele definiert. Es sind maximal 255 Sprungziele zulässig. Jeder Funktionsaufruf gilt als eine Verzweigung. Nicht mehr verwendete Sprungziele können nur mit **DEC** und **COM** gelöscht werden.
112	Die Anzahl der verwendeten Variablen übersteigt die maximal zulässige Anzahl (max. 123 numerische bzw. 255 Stringvariable).
113	Unzulässiges Zeichen (z.B. bei verschiedener Anzahl von öffnenden und schließenden Klammern in einem numerischen Ausdruck).
114	Rekursiver Aufruf in der Definition einer einzeiligen Funktion.
115	Unzulässiger Bezug auf eine Variable oder eine Funktion.
116	Es wurden mehr als 26 numerische Felder vereinbart.
117	Speicherüberschreitung bei der Eingabe oder beim Laden eines Programmes oder Textes. Ggf. sind unnötige Options zu löschen.
118	Es wurde bereits eine **FILES**-Anweisung mit einer anderen Zeilennummer eingegeben.
119	Es wurde bereits die max. zulässige Anzahl von Funktionen definiert.
120	Die Zeilennummer, auf die Bezug genommen wird, existiert nicht.
128	Unzureichender Platz für die Compilierung der eingegebenen Zeile.

5. Fehlermeldungen im Zusammenhang mit den Diskstationen

Fehlercodes	Bedeutung
131	Ungültiger Operand zur Angabe des Typs einer Diskstation
132	Ungültiger interner Code für die Bezeichnung einer Peripherie.
133	Die angegebene Diskstation hat bereits einen Namen.
134	Der angegebene interne Code paßt nicht zum angegebenen Typ der Diskstation.
135	Syntaktischer Fehler im symbolischen Namen
136	Der symbolische Name wurde bereits für eine andere Station vergeben.
137	Ungültiges Zeichen im symbolischen Namen einer Diskstation.
138	Es wurde keine Floppy-Disk-Einheit konfiguriert.
146	In einer Assembler-Routine erfolgt ein unzulässiger Bezug auf einen externen Modul.
147	Das angegebene Assembler-Objekt-File ist nicht zusammenhängend (Extent 1) gespeichert.
151	Fehler auf der linken Floppy-Disk-Station
152	Fehler auf der rechten Floppy-Disk-Station

Fehlercodes	Bedeutung
154	Eine konfigurierte Einheit enthält keine Disk oder Floppy-Disk oder es wird eine im System nicht konfigurierte Einheit angesprochen.
155	Die Disk in der angegebenen Einheit ist zerstört
156	Die Konfiguration hat keine System-Disk
157	Die Disk mit dem internen Code AØ ist entweder beschädigt oder nicht betriebsbereit
158	Die Disk mit dem internen Code A1 ist entweder beschädigt oder nicht betriebsbereit
159	Die Disk mit dem internen Code A2 ist entweder beschädigt oder nicht betriebsbereit
160	Die Disk mit dem internen Code A3 ist entweder beschädigt oder nicht betriebsbereit

A N M E R K U N G : Zu den Fehlern 151, 152, 158 und 160:
Tritt der Fehler wiederholt und auch mit anderen Disks auf, so liegt ein Hardwarefehler vor. Bezieht sich der Fehler auf eine Disk, so kann mit den Dienstprogrammen **LIBCOPY** bzw. **FLCOPY** versucht werden, den bestehenden Inhalt zu kopieren.

6. Nicht behebbare Fehler im Zusammenhang mit peripheren Einheiten

162	**SEND** für Input-Einheit oder **RECEIVE** für Output-Einheit (falsche Peripherie-Adresse)
163	Der mit **SEND** zu übertragende String ist größer als der Puffer des entsprechenden Kanals
164	Ein File ist zerstört und sein Inhalt verloren. Das File wird im Katalog mit Parameter F durch **DEAD** gekennzeichnet (siehe CATALOG)
165	Das Statement bezieht sich auf einen peripheren Kanal, der in der Konfiguration nicht vorhanden ist
166	Für den angesprochenen Kanal wurde kein Puffer definiert
167	Es wurde eine negative Peripherieadresse oder eine negative Command-Nummer angegeben
168	Im **OPTION**-Befehl wurde gleichzeitig **PLO** und **GDI** angegeben. **OPT PLO** und **OPT GDI** sind unverträglich.
169	Eine Peripherieadresse oder eine Command-Nummer sind größer als 255
170	Die in einem **RECEIVE**-Statement angegebene Stringvariable ist länger als der Puffer des entsprechenden Kanals
171	Der mit **CONFIGURE** angegebene externe Drucker ist nicht betriebsbereit a) es kann ohne externen Drucker weitergearbeitet werden b) ist der Drucker wieder betriebsbereit, so muß neu konfiguriert oder initialisiert werden

7. Fehler der Eingabe oder Ausführung von Systembefehlen

Fehlercodes	Bedeutung
172	Der für **OPT GDI** notwendige Grafik-Controller ist im System nicht vorhanden.
173	Es ist keine Bibliothek offen. Für das Arbeiten mit Files muß mindestens eine Bibliothek offen sein.
174	Es sind bereits sechst Bibliotheken offen.
175	Im Befehl **CONFIGURE** wird mehr Speicherkapazität als im System vorhanden ist, oder weniger als 16K verlangt.
177	Die angegebene Diskstation existiert nicht oder es wurde ein falscher Name angegeben.
178	Die angegebene Bibliothek ist auf der durch Namen oder implizit festgelegten Einheit nicht offen. (Siehe Systembefehle **LBOPEN** und **LVTOC**.)
179	Auf der angegebenen Station ist keine Bibliothek geöffnet (Siehe Systembefehle **LBOPEN** bzw. **LVTOC**).
180	Die angegebene Bibliothek ist nicht offen. (Siehe Systembefehl **LBOPEN**)
181	Der zur Verfügung stehende Speicher reicht nicht für die Ausführung des Befehls.
182	Beim Befehl **TRANSCODE** wurde bei der Umwandlung eines Datenfiles in ein Textfile der Parameter angegeben, es sind jedoch im Datenfile keine Zeilennummern enthalten.
183	Die angesprochene Teil-Bibliothek ist in der Bibliothek nicht vorhanden, oder es wird die zulässige maximale File-Anzahl für eine Teil-Bibliothek überschritten.
186	In der Teil-Bibliothek existiert bereits ein File mit gleichem Namen
187	Das File mit dem angegebenen Namen wird nicht gefunden (z.B. falscher Filename, Bibliothek nicht offen, ...)
188	Auf der Disk ist in einer Bibliothek zu wenig Platz für die Ausführung des Befehls BEMERKUNG: Die Ausführung der Befehle **RUN** und **PRE** ohne Angaben benötigen einen sogenannten listfähigen Platz in der ersten offenen Bibliothek

Fehlercodes	Bedeutung
189	Ein Datenfile soll so verkleinert werden, daß möglicherweise signifikante Informationen gelöscht würden.
190	Der Befehl wird im gegenwärtigen Status des Systems nicht akzeptiert oder ist nicht bekannt (z.B. falscher Mode, fehlerhafte Eingabe, im konfigurierten System nicht enthalten. **SHIFT+EXIT** wurde vergessen.
191	Es wurde kein Filename angegeben oder das File im Arbeitsspeicher hat keinen Namen.
192	Unzulässiges Zeichen im Befehl
193	Es fehlen Parameter im Befehl
194	Aufruf einer nicht existierenden Zeilennummer
195	Im Debugging-Mode wird der Befehl "**START Zeilennummer**" nach einem Befehl "**RUN Filename**" nicht akzeptiert, da das auszuführende Programm ohne vorhergehende Preexecution auf der Disk abgespeichert ist. Das gilt auch für den Befehl "**PREPARE Filename**".
196	Unzulässiger Parameter oder fehlerhaftes Format bei der Eingabe eines Befehls.
197	Der Befehl bezieht sich auf eine Zeilennummer, die innerhalb der Definition einer mehrzeiligen Funktion liegt.
198	Es wird mehr Speicherplatz verlangt, als in der Bibliothek vorhanden ist.
199	Die verlangte Operation ist für ein geschütztes File (secured) nicht zulässig.
200	Unerlaubte Operationen, da die Teil-Bibliothek geschützt ist.
201	Der Befehl ist nicht ausführbar, da beide Diskstationen Disks enthalten müssen.
202	In der Ausführung eines Befehls müßte ein File in mehr als 4 Teile aufgeteilt werden.
203	In einem **LIST**- oder **DELETE**-Befehl ist die erste Zeilennummer größer als die zweite.
204	Die Ausführung einer Prozedur wird abgebrochen, da bei der Ausführung eines Programmes mehr Eingabezeilen auszuführen sind, als im Command-File IN-Zeilen existieren oder eine IN-Zeile enthält Strings, die numerischen Variablen zugewiesen werden sollen.
205	Unerlaubte Operation, da die Zeile geschützt ist.
206	Das File im Arbeitsspeicher ist kein Programm.

Fehlercodes	Bedeutung
207	Die Operation ist für den angesprochenen Filetyp nicht ausführbar.
208	Die im **OPTION**-Befehl angeführte Option ist im System nicht vorhanden.
209	Es wurde eine Zeilennummer größer als 9999 erzeugt.
210	In einem **LIST**-Befehl wurde für den Ausdruck eines Programmes der Paramter X angegeben.
211	Leerer Arbeitsspeicher
212	Die zu druckende(n) Zeile(n) existiert(existieren) nicht.
213	Bei der Decompilierung einer Zeile tritt ein Line-Overflow auf.
214	Im Unterprogramm, das als mehrzeilige Funktion mit **LINK** eingefügt werden soll, ist die erste Zeile kein **DEF**-Statement.
215	Die angegebene Bibliothek ist bereits offen.
216	Es wird eine Einheit angesprochen, die bei Installation des Systems nicht konfiguriert wurde.
217	Ungültiges Password für eine Bibliothek.
218	Die angegebene Bibliothek wurde nicht gefunden.
219	Die gewünschte Operation ist für die Disk, die das System enthält, nicht zulässig.
220	Das Programm kann nicht in den Arbeitsspeicher geladen werden, da der Platz für den vereinbarten **COMMON**-Bereich fehlt.
221	Der angegebene Name bezieht sich nicht auf eine Floppy-Disk-Einheit.
222	Die angegebenen Diskeinheiten haben einen verschiedenen Typ.
223	Der Volume-Label stimmt nicht mit dem angegebenen überein.
224	Die Operation verlangt mehr Platz auf einer Disk, als im Moment frei ist.
225	Die verlangte Operation ist nur für Single-Drive-Disk-Stationen vorgesehen.
226	Es wurde eine ungültige Disk eingelegt.
227	Es wurde keine Bibliothek angegeben.

8. Fehler beim Aufruf oder der Ausführung von Dienstprogrammen

Fehlercodes	Bedeutung
228	Es wurde entweder versucht, auf einem Datenträger mehr als 35 Bibliotheken einzurichten oder es ist zu wenig Platz auf der Diskstation.
229	Es existiert bereits eine Bibliothek mit dem gleichen Namen auf derselben Diskstation.
230	Die angegebene Disk kann nicht initialisiert werden, da sie entweder beschädigt ist, oder mehr Alternativspuren angelegt werden mußten als möglich ist.
232	In einem Aufruf von **LBCREATE** ist die Anzahl der Sektoren, die für den Katalog reserviert werden sollen, größer als die Anzahl der Sektoren in der Bibliothek.
234	Es fehlt die Angabe des Namens des Dienstprogrammes.
235	Das aufgerufene Dienstprogramm existiert nicht (z.B. falscher Name oder nicht im konfigurierten System enthalten).

9. Fehlermeldungen, die sich auf PLOT-OPERATIONEN beziehen

Fehlercodes	Bedeutung
236	Behebbar: das in **INIMAGE** genannte File existiert nicht, das Bild wird nur im Arbeitsspeicher generiert.
237	Behebbar: a) Das genannte File ist nicht sequentiell oder zu klein. b) Nach der Preexecution bleibt nur eine Anwenderkapazität kleiner als 1280 Bytes. Das Bild wird nur im Arbeitsspeicher generiert.
238	Für den externen Plotter fehlt die Funktion **FNP** (nicht behebbar).
239	In der Anweisung **DRAW** wird das Bild außerhalb des Zeichenbereiches verschoben (nicht behebbar).
240	Nicht behebbar: Zur Ausführung von **DRAW** fehlt der Thermodrucker.
241	Nicht behebbar: Es wurde weder **INIMAGE** noch **LDIMAGE** ausgeführt.
242	Behebbar: **FRAME** folgt nicht unmittelbar auf **INIMAGE**.
243	Behebbar: Als Abstand für die Markierung der Achsen wurde 0 angegeben. Der Operand wird ignoriert.
244	Behebbar: Nach der Preexecution ist der verfügbare Speicherplatz für den Anwender kleiner als 1280 Bytes, die Ausführung kann dadurch langsamer werden.
245	Nicht behebbar: in **FRAME** ist der Wert für die Breite nicht zulässig.
246	Nicht behebbar: in **FRAME** ist der Wert für die Höhe nicht zulässig.
247	Nicht behebbar: In **SCALE** ist X-Min ≥ X-Max oder Y-Min ≥ Y-Max.
248	Nicht behebbar: Der in **INIMAGE** vereinbarte Puffer ist für die in **FRAME** angegebene Größe des Bildes zu klein.
249	Behebbar: Die für das File vereinbarte Größe ist kleiner als der Puffer im Arbeitsspeicher. Das Bild wird nur im Arbeitsspeicher erzeugt.

Fehlercodes	Bedeutung
250	Behebbar: a) Die Größe des Puffers oder des externen Files ist zu klein, so daß keine weiteren Punkte gespeichert werden können. Alle folgenden **PLOT**-Anweisungen mit Ausnahme von **INIMAGE**, **LDIMAGE** und **DRAW** werden ignoriert. b) In einem Programm, für das kein externes File zur Speicherung des Bildes existiert, soll ein **DRAW**-Statement zum zweiten Mal ausgeführt werden. Alle folgenden **PLOT**-Anweisungen mit Ausnahme von **INIMAGE** und **LDIMAGE** werden ignoriert.
251	Nicht behebbar: In der Anweisung **CSIZE** ergibt die Angabe von Breite und Höhe einen Wert kleiner 0.1
252	Nicht behebbar: Das File in der Anweisung **LDIMAGE** enthält kein Bild.
253	Behebbar: Die im Debugging-Mode aufgerufene Funktion enthält ein **DRAW, INIMAGE** oder **LDIMAGE**-Statement.
254	Behebbar: Bei der Ausführung der Zeichnung wurde die maximal möglich Anzahl von Schreibvorgängen erreicht. Alle nachfolgenden **PLOT**-Anweisungen mit Ausnahme von **DRAW** werden ignoriert.

10. Fehlermeldungen für die Interrupt-Behandlung

Fehlercodes	Bedeutung
280	Der Funktionsname in einem **INTERRUPT ENABLE**-Statement ist länger als 1 Zeichen.
281	Die Länge der Maske für die Kanäle ist ungleich 16.
282	Während der Behandlung eines internen Interrupts tritt ein weiterer Interrupt auf.
283	Ungültiger Wert für die Priorität.
284	Die Funktion, die von **INTERRUPT ENABLE** aufgerufen werden soll, existiert nicht.

11. Fehlermeldungen für einen nicht normalen Systemzustand

Fehlercodes	Bedeutung
m*A	Falls m eine andere Zahl als 12 oder 16 ist, und der Fehler auch bei anderen als den eingelegten Disks auftritt, so liegt ein Hardwarefehler vor.
12*A	Eine Fehlerhafte Kondition im Betriebssystem ist aufgetreten.
16*A	Tritt der Fehler auch nach einem erneuten Initialisieren des Systems auf, so liegt ein Hardwarefehler vor.
A N M E R K U N G :	Fehlermeldungen, die nach dem numerischen Code *A haben, bewirken ein Löschen des Arbeitsspeichers. Nach Drücken der Tasten **SHIFT+EXIT** wird das System neu geladen.
ABN unit name	Die angegebene Einheit arbeitet nicht korrekt, es ist zu überprüfen, ob der Einschub geschlossen und eine Disk eingelegt wurde.
ABN PRT	Der Drucker arbeitet nicht korrekt.
ABN unit-name DCH OMITTED	Bei einer Diskstation wurde ohne vorherige Eingabe des **DCHANGE**-Befehls ein Einschub geöffnet oder eine Disk gewechselt. Dies kann behoben werden durch Drücken von **CLEAR** und anschließend der **SHIFT+EXIT**-Tasten. Ist danach kein Arbeiten mit dem System möglich, muß es neu initialisiert werden.
ABEND	Meldungen der Form **ABEND ...** werden von aufgerufenen Assembler-Routinen verursacht.
A N M E R K U N G :	Statusmeldungen **ABN** können nach Beheben der Fehlerursache mit **CLEAR** gelöscht und mit **SHIFT+EXIT** der Ablauf fortgesetzt werden.

ISO-Code-Tabelle

Hexa-dezimal		0	1	2	3	4	5	6	7
	Dual	0000	0001	0010	0011	0100	0101	0110	0111
0	0000	0 NUL	16 DLE	32 SP	48 0	64 @	80 P	96	112 p
1	0001	1 SOH	17 DC1	33 !	49 1	65 A	81 Q	97 a	113 q
2	0010	2 STX	18 DC2	34 "	50 2	66 B	82 R	98 b	114 r
3	0011	3 EXT	19 DC3	35 #	51 3	67 C	83 S	99 c	115 s
4	0100	4 EOT	20 DC4	36 $	52 4	68 D	84 T	100 d	116 t
5	0101	5 ENQ	21 NAK	37 %	53 5	69 E	85 U	101 e	117 u
6	0110	6 ACK	22 SYN	38 &	54 6	70 F	86 V	102 f	118 v
7	0111	7 BEL	23 ETB	39	55 7	71 G	87 W	103 g	119 w
8	1000	8 BS	24 CAN	40 (	56 8	72 H	88 X	104 h	120 x
9	1001	9 HT	25 EM	41)	57 9	73 I	89 Y	105 i	121 y
A	1010	10 LF	26 SUB	42 *	58 :	74 J	90 Z	106 j	122 z
B	1011	11 VT	27 ESC	43 +	59 ;	75 K	91 [	107 k	123 {
C	1100	12 FF	28 FS	44 ,	60 <	76 L	92 \	108 l	124 |
D	1101	13 CR	29 GS	45 -	61 =	77 M	93]	109 m	125 }
E	1110	14 SO	30 RS	46 .	62 >	78 N	94 ↑	110 n	126 ~
F	1111	15 SI	31 US	47 /	63 ?	79 O	95 ←	111 o	127 DEL

Liste der darstellbaren Zeichen und ihre entsprechenden dezimalen Werte

Zeichen	dezim. Wert
	0
	1
	2
	3
	4
	5
	6
	7
	8
	9
	10
	11
	12
	13
	14
	15
	16
	17
	18
	19
	20
	21
	22
	23
	24
	25
	26
	27
	28
	29
	30
	31
	32

Zeichen	dezim. Wert
!	33
"	34
#	35
$	36
%	37
&	38
'	39
(	40
)	41
*	42
+	43
,	44
-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
:	58
;	59
<	60
=	61
>	62
?	63
@	64
A	65

Zeichen	dezim. Wert
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
\	92
]	93
↑	94
_	95
`	96
a	97
b	98

Zeichen	dezim. Wert
c	99
d	100
e	101
f	102
g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
}	125
~	126
※	127
‡	128
▮	129
▮	

Darstellung der ISO-Zeichen der Spalten 0 und 1 der ISO-Code-Tabelle

Taste (CONTROL)	Darst. des Zeich.	ISO-Zeichen	dezim. Wert
@		NUL	0
A		SOH	1
B		STX	2
C		EXT	3
D		EOT	4
E		ENQ	5
F		ACK	6
G		BEL	7
H		BS	8
I		HT	9
J		LF	10

Taste (CONTROL)	Darst. des Zeich.	ISO-Zeichen	dezim. Wert
K		VT	11
L		FF	12
M		CR	13
N		SO	14
O		SI	15
P		DLE	16
Q		DC1	17
R		DC2	18
S		DC3	19
T		DC4	20
U		NAK	21

Taste (CONTROL)	Darst. des Zeich.	ISO-Zeichen	dezim. Wert
V		SYN	22
W		ETB	23
X		CAN	24
Y		EM	25
Z		SUB	26
[		ESC	27
\		FS	28
]		GS	29
↑		RS	30
_		US	31

Das Handbuch dient der Information, sein Inhalt ist ohne ausdrückliche schriftliche Vereinbarung nicht Vertragsgegenstand. Technische Änderungen behalten wir uns vor. Die angegebenen Daten sind lediglich Nominalwerte.

DEUTSCHE OLIVETTI DTS GMBH · FRANKFURT-NIEDERRAD · LYONER STRASSE 34