Diese Bedienungsanleitung führt Sie zunächst in die Handhabung der Kontron PSI-Computer-Systeme und der Systemsoftware Kontron KOS ein. Es folgen die Beschreibungen des Editorprogramms EDIT und des BASIC-Interpreters unter dem Betriebssystem KOS. Im Anhang sind Codetabellen, Tastaturbelegungen etc. enthalten.

Technische Änderungen bleiben vorbehalten.

Dieses Handbuch ist mit größter Sorgfalt erstellt worden. Es wird jedoch keine Gewähr für die Freiheit von Fehlern und Irrtümern gegeben.

Für alle Anfragen stehen Ihnen unsere Technischen Büros und Ihr Distributor zur Verfügung.

Mai 1984

DOK-PSI/KOS-D0584

Inhaltsverzeichnis der Bedienungsanleitung:

Literaturhinweise:

Weitere Informationen entnehmen Sie bitte den folgenden Abschnitten:

Technische Beschreibung:	Konzepte und Grundlagen Systemkommandos zum Betriebssystem KOS KOS-Betriebssystembeschreibung KOS-Utilities Assembler, Linker, Crossreference-Generator Debugging Module (KDM) Anhang: Tabellen
Hardware Beschreibung:	Kontron PSI 80/82-Baugruppen Kontron PSI 908/9C/98-Baugruppen Kontron PSI 980-Baugruppen

Weitere Unterlagen:

Installation:	siehe Installationshandbuch
Service Unterlagen:	siehe jeweilige Dokumentation
Optionale Software:	siehe jeweilige Dokumentation
Kontron ECB-Computer-baugruppen:	siehe jeweilige Dokumentation

Integrierte Schaltungen: siehe Dokumentation der jeweiligen Hersteller

STAND:

Version 4.33/5.44/5.54/6.05	Mai 1984

FRÜHERE STÄNDE:

Version 4.33/5.44/5.54/6.04	August 1983
Version 4.33/5.44/5.54/6.03	April 1983
Version 4.33/5.4/5.5/6.0	Dezember 1982
Version 4.3/5.3	November 1981
Version 3.2	April 1980

Übersicht

über

Kontron PSI Dokumentation

Stand: Mai 1984

Version: 4.33/5.44/5.54/6.05

Kontron PSI Systeme werden mit umfangreicher Dokumentation geliefert.

Der erste Abschnitt dieser Bedienungsanleitung gibt Ihnen die Übersicht, wo welche Themen erläutert sind.

Übersicht über die Kontron PSI-Dokumentation

Die Einsatzmöglichkeiten Ihres Kontron PSI-Systems sind vielfältig. Wir wollen Ihnen die Anwendung dieser Computersysteme als Werkzeug in Ihrer Aufgabenstellung durch die möglichst vollständige Beschreibung aller Funktionen erleichtern.

Um Ihnen das Auffinden der für Sie wichtigen Informationen leicht zu machen, haben wir die Dokumentation in folgende Abschnitte aufgeteilt:

Teil 1: Bedienungsanleitung für Kontron PSI-Systeme

Dieser Teil enthält die notwendige Grundinformation zum Umgang mit Dokumentation, Software und Hardware. Dieser Teil ist sowohl für den Erstbenutzer als auch für den erfahrenen Anwender von Computern gleichermaßen wichtig, denn er gibt Ihnen

- Übersichten über Dokumentation zu Kontron PSI-Systemen, Computerbaugruppen, Bauelementen und Software
- Hinweise zur ersten Inbetriebnahme
- Einführungen in die Handhabung der Systemsoftware
- Informationen über den Kontron Texteditor EDIT
- Informationen über den Kontron BASIC-Interpreter und
- Codetabellen, Tastaturbelegungen etc.

Teil 2: Technische Beschreibung zu Kontron PSI-Systemen

Dieser Teil ist notwendig für den Anwender, der die Möglichkeiten der Hardware in maschinen-naher Programmierung ausschöpfen, die Funktionen des transparenten Betriebssystems KOS in seinem Anwendungssystem benutzen, eigene Peripherietreiberprogramme entwickeln oder einfach alle Hilfen der Systemdienstprogramme für sich nutzen will.

In der Technischen Beschreibung finden Sie:

- Konzepte und Grundlagen der Systemsoftware KOS und der Dienstprogramme
- Beschreibung der Systemkommandos und des Betriebssystems KOS
- Treiber und andere Utilities
- Assemblerbedienung
- Programmtest (Debugging Monitor)

Teil 3: Hardware-Beschreibung der Kontron PSI-Systeme

Dieser Teil befaßt sich mit den Zentraleinheiten und E/A-Baugruppen Ihres Kontron PSI-Systems. Dieses Wissen ist notwendig für Sie, wenn Sie Ihr System um eigene oder Standard-Hardware an den dafür verfügbaren Schnittstellen erweitern, oder wenn Sie spezielle Eigenschaften der Hardware in Ihrer Anwendung einsetzen wollen.

Wo finden Sie Informationen über noch nicht genannte Gebiete Ihres Interesses?

Folgende Übersicht soll Ihnen Hinweise geben:

Integrierte Bausteine in den Kontron PSI-Einheiten:

Kurzinformationen enthält der Teil 3: Hardware-Beschreibung.

Weitergehende Informationen entnehmen Sie bitte den Datenblättern, die Sie bei den jeweiligen Herstellern und deren Distributoren anfordern können.

Installation:

Kurzinformationen enthält das Installations-Handbuch.

Hardware-Module, wie Floppy Disk-Laufwerke, Bildschirme etc.:

Die wesentlichen Informationen sind in Service-Handbüchern enthalten, die bei Kontron bestellt werden können.

Z80-Befehlssatz:

Bitte bestellen Sie die ausführliche Z80-Befehlssatz-Dokumenation, Bestellbezeichnung: **Z80-Assembler-Sprache Benutzerhandbuch**

Optionale Software und Computer-Sprachen:

Die volle Dokumentation über die von Kontron Mikrocomputer verfügbaren Software-Optionen sind im Lieferumfang der jeweiligen Software-Bestellung enthalten. Kurzübersichten sind verfügbar bzw. in Vorbereitung: Fragen Sie Ihr Technisches Büro oder Ihren Distributor.

Kontron ECB-Computerbaugruppen als Erweiterung zu Kontron PSI-Systemen:

Übersichten und detaillierte Beschreibungen aller Kontron ECB-Baugruppen können bei Kontron Mikrocomputer bestellt werden. Fragen Sie Ihr Technisches Büro oder Ihren Distributor.

Anwendungssoftware:

Für eine Reihe von Branchen und Anwendungen existieren fertige und erprobte Software-Lösungen. Fragen Sie Ihr Technisches Büro oder Ihren Distributor.

Peripheriegeräte:

Drucker, Plotter, Graphische Eingabegeräte etc. sind von Kontron Mikrocomputer verfügbar. Sprechen Sie mit Ihrem Technischen Büro oder Ihren Distributor.

Spezifikationen:

Auskünfte über den spezifizierten Leistungsumfang und die garantierten Technischen Daten enthalten die jeweiligen Spezifikationen aus Ihrem Angebot bzw. der aktuellen Preisliste. Beachten Sie bitte, daß ausschließlich diese Spezifikationen nach den Kontron Verkaufs- und Lieferbedingungen den Leistungsrahmen unserer Systeme definieren. Technische Änderungen, Weiterentwicklungen und Erweiterungen bleiben vorbehalten.

Einführung:

Last not least - wir haben für Sie einen Einführungskurs entwickelt, der Sie Schritt für Schritt in Computertechnik und -Anwendung voranbringt.

Für alle Fragen sind unsere Technischen Büros und Ihr Distributor für Sie da. Rufen Sie Ihren nächstgelegenen Partner an:

Kontron Deutschland - Technische Büros:

1000 Berlin 41
Albrechtstr. 34
Tel. 030/7923031-3
Telex 0185484

8057 Eching
Obere Hauptstr. 5
Tel. 089/31901-318
Telex 05213671

2000 Hamburg 70
Königsreihe 2
Tel. 040/68295-0
Telex 0211998

6000 Frankfurt 70
Kennedy-Allee 34
Tel. 0611/636061
Telex 0414881

3000 Hannover 81
Hermann-Guthe-Str. 3
Tel. 0511/839051-57
Telex 0923729

7000 Stuttgart 30
Maybachstr. 39 a
Tel. 0711/814621
Telex 0723061

4000 Düsseldorf 1
Ronsdorfer Str. 143
Tel. 0211/7361-0
Telex 08582675

8500 Nürnberg 20
Rennweg 60/62
Tel. 0911/533306
Telex 0626391

Kontron Internationale Distribution

AUSTRIA
Kontron GmbH & Co KG
Eisgrubengasse 2
A-2334 Vösendorf/b. Wien
Phone: 0043-222670631
Telex: 047-131699

BELGIUM
Heliagraph p.v.b.a
Gulden Sporenlaan 19
B-9220 Merelbeke
Phone: 0032-91-301314
Telex: 046-12104

DENMARK
SC METRIC A/S
Skodsborgvej 305
DK-2850 Naerum
Phone: 0045-2-804200
Telex: 055-37163

DENMARK
E BLICHFELD APS
Gl. Hovedgade 10 b/Box 249
DK-2970 Hoersholm
Phone: 0045-2-867900
Telex: 055-37548

FRANCE
Kontron S.A. France
6, rue des Freres Caudron
F-78140 Velizy-Villacoublay
Phone: 00333-9469722
Telex: 042-695672

ISRAEL
TABOR ELECTRONICS
Shemen Beach
Industrial Zone
P.O. Box 901
Haifa
Phone: 00972-4-644280
Telex: 0606-46591

ITALY
Eledra 3S S.p.A.
Viale Elvezia, 18
I-20154 Milano
Phone: 0039-2-349751
Telex: 043-314155

ITALY
KONTRON S.p.A.
Head Office:
Via Medici del Vascelo. 26
I-20138 Milano
Phone: 0039-250721
Telex: 043-312288

NETHERLANDS
Tekelec Airtronic
Storkstraat 7
Postbus 63
NL-2700 AB Zoetermeer
Phone: 0031-79-310100
Telex: 044-33332

NORWAY
Bergman Instrumentering A/S
P.O. Box 129
Veitvet
Sven Oftedals Vei 10
N-Oslo 5
Phone: 0047-2-162210
Telex: 056-17271

SOUTH AFRICA
BIWAVE Instruments (Pty) Ltd.
P.O.Box 31518
Braamfontein
2017 Republic of South Africa
Phone: 0027-11-339-1121/2/3
Telex: 095-4-30763

SPAIN
Kontron S.A.
Salvatierra, 4
E-Madrid 34
Phone: 0034-1-7291155
Telex: 052-23382

SWITZERLAND
Kontron Electronic AG
Abt. Microcomputer
Bernerstr. Süd 169
CH-8048 Zürich
Phone: 0041-1-4354111
Telex: 045-822195
045-822070 (Datasystems)

SWITZERLAND
KONTRON ELECTRONIC SA
10, chemin des Croisettes
CH-1066 Epalinges
Phone: 0041-21-331535
Telex: 045-26398

THAILAND
TRANE INTERNATIONAL Co., Ltd.
194/7 Ladprao Road. Bangken
BANGKOK/0900
Phone: 511-2587
Telex: 086-82189 ATTN TRANE

JAPAN
Kontron K.K.
Kowa Bldg. No. 25
8-7, Sanbancho
Chiyoda-Ku-Tokyo 102
Phone: 03-263-4801
Telex: 0720-2323190

UNITED KINGDOM
Kontron Electronics Ltd.
Campfield Road/PO Box 88
GB-St. Albans AL1 5JG
Phone: 0044-727-66222
Telex: 051-267-102

UNITED STATES
Kontron Electronic Inc.
630 Price Ave.
Redwood City, CA. 94063
Phone: 001-415-361-1012
Telex: 0230-910 378 5207

Zur Schulung:

Wenn Sie sich weiterbilden wollen in Computer-Hardware, Computer-Baugruppen, Programmierung und Systementwicklung: Kontron führt jährlich mehr als 100 Kurstage in Eching und in den Häusern unserer Kunden durch. Fragen Sie uns!

Und zum Service für Kontron PSI-Systeme und Peripheriegeräte:

Kontron bietet über die übliche Werksgarantie (6 Monate, Reparatur/Wartung in Eching) hinaus **'Service vor Ort'**: innerhalb von 48 bzw. 24 Stunden wird im Rahmen Ihres Wartungsvertrges Ihr System bei Ihnen im Haus (BRD) geprüft und in Stand gesetzt. Der Wartungsvertrag sichert Ihnen die Verfügbarkeit Ihrer Investition.

Fragen Sie Ihr Technisches Büro nach dem Kontron Wartungsvertrag!

Und nun: Lernen Sie die Leistungsfähigkeit Ihres Kontron PSI-Systems für die Lösung Ihrer Aufgabenstellung kennen!

Kontron PSI Computer-Systeme

Hardware und Betriebssoftware KOS

Stand: Mai 1984

Version: 4.33/5.44/5.54/6.05

Kontron PSI-Systeme sind eine Serie von allgemeinen Computern für kommerzielle, technisch-wissenschafliche und industrielle Anwendungen.

Der folgende Abschnitt dokumentiert die Zusammenhänge zwischen Typenbezeichnung, wesentlichen Merkmalen, Hardware-Eigenschaften und Betriebssoftware-Versionen.

Kontron PSI-Computersysteme - Hardware und Betriebssoftware

Die **Typenbezeichnung** enthält die wesentlichen Informationen über die Hardware-Konfiguration:

a) Die Serie:

```
Kontron        PSI       80

   *            *        *
   *            *        ******   Reihe: 80.... Kompaktsysteme
   *            *                        82.... Industriecomputer
   *            *                        98.... Kompaktsysteme
   *            *                        9xx... 'Ergo Line' Computer
   *            *
   *            **************   Serie: PSI
   *
   ***************************   Hersteller: Kontron, Eching/München
```

b) die Ausstattung:

Kontron PSI80D/M2

```
              * * *
              * * *****  Konfiguration:  S2:  2 FD-Laufwerke
              * *                        M2:  2 FD-Laufwerke, erweiterbar
              * *                             durch Kontron ECB-Computer-
              * *                             Baugruppen
              * *                       W10:  Integrierte Festplatte,
              * *                             10 MByte (formatiert)
              * *                       W20:  Integrierte Festplatte,
              * *                             17.8 MByte (formatiert)
              * *                       W40:  Integrierte Festplatte,
              * *                             40 MByte (formatiert)
              * *                        TC:  Terminal Computer
              * *                         C:  Terminal Computer
              * *
              **********-:  FD-Kapazität 154kByte(nichtfür
              *               Neuentwicklungen) bzw. ohne FD
              *           D: FD-Kapazität 308 kByte
              *           Q: FD-Kapazität 616 kByte, grüner Sichtschirm
              *           H: FD-Kapazität 616 kByte, hochauflösender
              *              schwarz/weiß Sichtschirm (nur Kontron PSI980H)
              *
              ***********  Hauptspeicherkapazität und CPU:
                           80/82:           64 kByte/Z80A
                           980/908/9C/98:   256 KByte/Z80A
```

Die **Zentraleinheit-Baugruppen** sind:

Serie/Typ:	**Baugruppe:**	**Charakteristik:**
Kontron PSI80	KDT4	Zentraleinheit (Z80A, 64kB) und Ein-/Ausgabe, FD-Kapazität begrenzt auf 154 kByte (nicht für Neuentwicklung)
Kontron PSI80 **Kontron PSI82**	KDT5	Zentraleinheit (Z80A, 64kB) Ein-/Ausgabe, FD-Kapazität 308 bis 616 kByte, integrierte Festplatte möglich, Option Kontron KOBUS möglich
Kontron PSI908 **Kontron PSI9C** **Kontron PSI98**	KDT6	Zentraleinheit (Z80A, 256 kByte) und Ein-/Ausgabe, FD-Kapazität je 616 kByte, KOBUS-fähig
	IOC/9xx IOC/98	Schnittstellen
Kontron PSI980	TCB/Z80-1	Zentraleinheit (Z80A, 256 kByte), FD-Kapazität 616 kByte, integrierte Festplatte und Wechselplatte möglich, KOBUS-fähig
	TCB/IOV-1/2	Ein-/Ausgabe-Baugruppe (Video, IEEE488, serielle Schnittstellen, etc.)
	TCB/BUS	Bus-Baugruppe mit ECB/TCB-Erweiterungsmöglichkeiten

* *
* *
* *

* *
* *

Die zugehörigen **Betriebssoftware-Stände** sind:

Kontron PSI80-KDT4	KOS4.33
Kontron PSI80/82-KDT5	KOS5.44/5.54
Kontron PSI908/9C/98-KDT6	KOS6.05
Kontron PSI980-TCB/Z80-1	KOS6.05

Die Dokumentation der Systemsoftware gilt für KOS4, KOS5 und KOS6 gleichermaßen. Auf Erweiterungen wird im Einzelfall hingewiesen.

Die Hardware-Dokumentation für Ihr Kontron PSI-System ist im jeweiligen Band 'Hardware' enthalten.

Für optionale Software, Peripherie und für Hardware-Erweiterungen sind die zugehörigen Dokumentationen Teil des Lieferumfangs.

Und nun: Zur Inbetriebnahme Ihres Kontron PSI-Systems!

Inbetriebnahme der

Kontron PSI-Computersysteme

Einschaltanleitung, Handhabung von Disketten, Pflege und Reinigung des Computersystems, erste Hilfe bei technischen Störungen.

Version: 4.33/5.44/5.54/6.05

Mai 1984

Die folgende Beschreibung gibt Ihnen Hilfestellung beim ersten Einschalten Ihres Systems.

Unabhängig davon, welchen Wissenstand Sie bezüglich Computersystemen haben: die hier und im folgenden Kapitel 'Bedienung' vermittelten Informationen ersparen Ihnen später mit Sicherheit viel Zeit.

Bitte beachten Sie insbesondere die Hinweise zum Arbeiten mit Disketten.

Inbetriebnahme von Kontron PSI-Systemen

Bevor Sie Ihr Kontron PSI-System einschalten: Machen Sie sich mit den folgenden Regeln zum Arbeiten mit Floppy Disks vertraut:

Sie verwenden Disketten zur Programm- und Datensicherung. Wir möchten Sie daher zunächst über Grundlagen informieren, die für die Behandlung der Floppy-Disk-Laufwerke und der Datenträger (= Disketten) wichtig sind, da hiervon **Betriebssicherheit und Zuverlässigkeit** in hohem Maße abhängen.

Grundlegend sind zunächst die für die in Kontron PSI-Systemen verwendeten Laufwerke gegebenen Spezifikationen für das Diskettenmaterial:

- Double density: Aufzeichnungsdichte längs einer Spur
- Double sided: verwendet in "Q/H"-Systemen/Single sided: verwendet in "D"-Systemen
- 100 bzw. 96 tpi: Spurdichte von 100 bzw. 96 Spuren pro Zoll, wird auch definiert als 80 Spuren pro Diskettenseite.

Disketten, die diese Grundspezifikationen nicht erfüllen, sind nicht zuverlässig betreibbar.

Darüber hinaus wurden folgende, nicht direkt spezifizierte Einflußgrößen in umfangreichen Versuchen ermittelt:

- Durchbiegung von Disketten erschwert die Wiederholbarkeit der Justage beim Einlegen.
- Schlechte Gleitfähigkeit der Diskettenscheibe in der Diskettenhülle erschwert das Zentrieren bei aus der Mittellage gerutschten Disketten
- Verstärkungsringe verhindern die durch beim "gefühllosen" Schließen der Laufwerkstür mögliche Beschädigung des Innenlochs von Disketten.
- Glatte Oberfläche des Magnetmaterials erleichtert die Verschiebbarkeit und schont den Schreib-/Lesekopf des Laufwerks.

Wesentlich ist jedoch vor allem die Handhabung der Disketten:

- Staub, Magnetfelder, Fingerabdrücke auf dem Diskettenmaterial, Hitze, Sonnen- und Feuchtigkeitseinwirkung und normale Abnutzung beschädigen Disketten.
- Langsames Schließen der Laufwerkstür, möglichst bei drehendem Laufwerksmotor, erleichtert das Zentrieren der Disketten. Hier hilft die Eigenschaft der Kontron PSI-Systeme, daß Diskettenlaufwerke noch für einige Sekunden nachlaufen. Ist eine kritische Diskette beim ersten Einlegen nicht lesbar, dann genügt ein Öffnen der Laufwerkstür und sorgfältiges Wiedereinlegen der Diskette während der Motor noch läuft.

Was ist bei Diskettenproblemen zu beachten?

1) Bitte überprüfen Sie als erstes die Spezifikation der Disketten (Schreibdichte, ein-/zweiseitiger Betrieb, Spurdichte).

2) Überprüfen Sie die Zentrierbarkeit der Disketten: Exzentrisch verschobene Disketten wieder grob zentrieren und bei laufendem Motor einlegen.

3) Verwenden Sie Disketten mit Verstärkungsring, da ausführliche Tests gezeigt haben, daß mit diesen Disketten auch bei weniger gut ausgebildetem Bedienungspersonal das Einlegen von Disketten weniger Anlaß zu Fehlern bietet.

4) Bitte beachten Sie, daß bei der Herstellung von Disketten auch chargenabhängige Abweichungen durchaus auftreten können. Dies betrifft vor allem die Verschiebbarkeit der Disketten innerhalb der Diskettenhülle. **Kontron Disketten sind deswegen vom Hersteller speziell in diesem Punkt zusätzlich überprüft.**

Nachstehend Disketten, die geprüft und für geeignet befunden worden sind:

Memorex Mini Flexible Disc 2d-80 Part.No. 3201-3501
Verbatim Datalife 5 1/4"
BASF Flexy-Disk 5 1/4" 2/96 mit Verstärkungsring

Und wie steht es mit Reinigungsdisketten?

Während der letzten Jahre wurden sehr viel Reinigungsmaterialien verwendet, die sehr starke Schmirgelwirkung hatten und deswegen das Material der Schreib-/Leseköpfe stark beanspruchten. Andere Reinigungsmaterialien hinterließen auf den Schreib-/Leseköpfen Rückstände, die zusätzlich verschmutzend wirkten.

Inzwischen ist auch hier die Technik fortgeschritten: Wir können heute Reinigungsdisketten Typ BASF Cleaning Flexy-Disk 5 1/4" empfehlen,
- wenn sie höchstens einmal pro Woche verwendet werden,
- wenn sie bei jeder Benutzung nur für wenige Sekunden eingesetzt werden (den Zugriff auf/die Aktivierung der Diskettenstation erreichen Sie durch eine beliebige Kommandoeingabe)
- für doppelseitige Laufwerke (Kontron PSI80Q, PSI9xx...) wird die Reinigungsdiskette zweimal eingelegt, einmal für jede Seite. Genauere Gebrauchsanleitungen werden mit der Reinigungsdiskette geliefert.

Geeignete Disketten und Reinigungsdisketten können bei Kontron unter folgenden Bezeichnungen bestellt werden:

Disketten für alle Kontron PSI-Systeme: DISK
Reinigungsdisketten: DISK-CLEAN

Wegen der Wichtigkeit dieser Regeln zur Handhabung von Disketten fassen wir noch einmal zusammen:

1) Stecken Sie Disketten grundsätzlich **erst dann** in die Laufwerke ein, wenn Sie das Gerät ans Netz angeschlossen und eingeschaltet haben.

2) Nehmen Sie Disketten immer aus den Laufwerken heraus, **bevor** Sie das Gerät ausschalten oder den Rücksetzknopf betätigen; andernfalls können die Disketten beschädigt werden.

3) Das Einlegen von Disketten geschieht wie folgt:

   - Diskette einschieben bis zum Einrasten. Beschriftung zeigt nach links (Kontron PSI80/82) bzw. nach oben (Kontron PSI98/908/980) Spurabtastschlitz zeigt in das Laufwerk hinein.
   - Laufwerktür schließen durch **langsames** Bewegen des Hebels bis zum Einrasten. Wird die Laufwerkstür zu schnell geschlossen, kann sich die Diskette evtl. nicht zentrieren. Folge: Geräuschentwicklung, Zugriffsfehler

4) Das Herausnehmen von Disketten geschieht wie folgt:

   - Laufwerktür durch Drücken gewaltlos lösen
   - Bei Kontron PSI80 Auswerfen der Diskette durch leichtes Drücken des Auswerfhebels am Laufwerk nach links.

5) Bitte verwenden Sie grundsätzlich nur Arbeitskopien Ihrer Disketten und bewahren Sie das Original sorgfältig vor Staub, Feuchtigkeit, Hitze und Magnetfeldern geschützt in der Diskettenhülle auf. Verwenden Sie dieses Muster nur zum erneuten Kopieren für den Fall, daß Ihre Arbeitsdiskette ausfällt.

6) Beschriften Sie die Diskettenetiketten nur mit Filzstift, nicht mit Kugelschreiber oder Bleistift. Eindrücke von 'harten' Schreibern können Datenverlust zur Folge haben. Radieren Sie niemals auf dem Etikett. Staub auf der Diskette stellt die Betriebssicherheit des Gesamtsystems infrage!

7) Disketten sind Low-Cost-Datenträger, die aus konstruktiven Gründen einem gewissen Verschleiß unterworfen sind; der Ausfallzeitpunkt ist nicht vorhersehbar. Fertigen Sie daher von allen wichtigen Disketten Kopien an und benutzen Sie diese ausschließlich zu Archivierungszwecken, nicht aber zum täglichen Gebrauch.

8) Prüfen Sie unbeschriebene Disketten vor Gebrauch durch das FORMAT-Kommando (siehe dort).

9) Disketten-Schreibschutz:
   Disketten verfügen über einen Hardware-Schreibschutz.
   Am Rand der Diskettenhülle befindet sich hierfür eine rechteckige Aussparung. Durch Überkleben dieser Aussparung wird die Diskette schreibgeschützt. Eine so gesicherte Diskette kann weder überschrieben noch gelöscht, geändert oder formatiert werden. Dies ist für Masterdisketten unbedingt zu empfehlen.

Pflege und Reinigung des Computersystems

Ihr Computersystem dankt Ihnen pflegliche Behandlung durch ungestörten Betrieb. Regelmäßige vorbeugende Wartung ist bei normalen Einsatz im Bürobereich nicht notwendig.

Vermeiden Sie jedoch Beschädigungen und Verschmutzung von Disketten, Tastaturen, Computersystemen und Peripheriegeräten, insbesondere durch Getränke, wie z.B. Kaffee, Tee, Wasser oder Coca Cola oder durch Fremdkörper, wie Büroklammern, Radierstaub etc.

Zur Schonung Ihrer Augen wischen Sie am besten täglich den Staub vom Bildschirm, am besten mit weichem Gewebe oder Antistatic-Tüchern.

Zur Reinigung der Gehäuse verwenden Sie am besten ein Reinigungsmittel, das leicht fettlösend wirkt, jedoch den Lack nicht angreift; Abwischen mit einem weichen nur leicht angefeuchteten Tuch oder Gewebe genügt im allgemeinen.

Was tun bei technischen Störungen?

Bevor Sie Ihre Wartungsstelle anrufen (Kontron bietet Wartungsverträge an!), überprüfen Sie bitte genauso wie z.B. bei einem Haushaltsgerät alle einfachen Störungsmöglichkeiten:

- Gerät eingeschaltet?
  Leuchten die Anzeigen, zeigt der Bildschirm etwas an? Netzstecker? Sicherung? Übrigens: Nach Ziehen des Netzsteckers können Sie mit einem Schraubenzieher die Gerätesicherung aus einer "Schublade" direkt neben dem Netzsteckereingang entnehmen. Hier befindet sich auch eine Ersatzsicherung!

- Richtige Disketten korrekt eingelegt? Disketten in Ordnung (siehe Prüfmöglichkeiten bei "FORMAT")

- Papier im Drucker eingelegt, Drucker eingeschaltet? Alarmanzeige am Drucker?

- Kabelverbindungen festgeschraubt bzw. fest sitzend?

- Bei Verdacht auf Störungen in der Rechnerelektronik: Testprogramme siehe im Teil "Technische Beschreibung", Abschnitt "KOS-Utilities".

Und nun: die Inbetriebnahme

1. Stecken Sie das Tastatur-Anschlußkabel in die mit "Keyboard" gekennzeichnete Buchse auf der Rückseite des Gerätes ein.

2. Verbinden Sie das Gerät mit dem 220V-Netz mit dem zugehörigen Netzkabel. Bei Kontron PSI980 und Kontron PSI82 ist der Bildschirm separat an das Netz und an das Hauptsystem anzuschließen (Stecker TTL-Video)

3. Jetzt können Sie die Anlage über den rechts vorne befindlichen Netzschalterknopf bzw. den Schlüsselschalter einschalten. Bei separatem Monitor ist auch der Bildschirm einzuschalten.

4. Startdiskette/Systemdiskette in rechtes Laufwerk (Kontron PSI80/82/908) bzw. oberes Laufwerk (Kontron PSI98/980) einstecken (Aufkleber: "Start..." oder "Syskmd...").

5. Das Urladeprogramm sucht automatisch das Betriebssystem und lädt es in den Arbeitsspeicher des Computers.
   (Geräte, die sich nach dem Einschalten mit 'BOS' melden, erfordern das Drücken der Tasten 'K' und 'RETURN'.)

6. Danach wird eine Kommandofolge ausgeführt, die auf Kontron-Start-Disketten Ihr erstes Programm startet, das Ihnen das erste Arbeiten mit dem System leicht macht. Und wenn sich Ihr System mit "Login:" meldet: Kontron hat hier die Antwort "*", gefolgt von der RETURN-Taste, vordefiniert.

   Von da an steht Ihnen die ganze Leistungsfähigkeit des Kontron-PSI-Betriebssystems KOS und aller Dienstprogramme zur Verfügung.

Zur besonderen Beachtung:

Auf jeder Kontron-Diskette befinden sich Informationsdateien, die Ihnen Auskunft über die aktuelle Software-Version und über eventuelle Abweichungen gegenüber den Handbüchern geben. Diese Informationen sollten Sie unbedingt vor Ihrer Arbeit mit dem System ansehen.

Sie erreichen dies durch Eingabe von INFO<--- oder durch Anwahl aus dem Startprogramm. **Hier wie im folgenden symbolisiert '<---' das Betätigen der 'RETURN'-Taste.**

Das **Ausschalten** erfolgt, nach Entnahme der Disketten, durch den Netzschalterknopf bzw. den Schlüsselschalter, wenn alle Disketten- und Plattenaktivitäten abgeschlossen sind.
Bei KOBUS-Mehrplatzsystemen ist das Ein-/Ausschalten von Slave-Stationen jederzeit bei eingeschalteter und auf KOBUS-Betrieb aktivierter Masterstation möglich. Die Masterstation darf nur ein-/ausgeschaltet werden, wenn sämtliche Slaves ausgeschaltet sind bzw. nicht auf KOBUS zugreifen. Letzteres gilt auch für Slavestationen mit eigenem Massenspeicher (z.B. Floppy Disk), die autonom arbeiten können.

Noch einige Hinweise:

Bitte beachten Sie die Beschriftung der Kontron-Disketten:

KOPF: zeigt die Kontron PSI-Serie an: z.B. Kontron PSI80
Hinweis:
Disketten, die im Kontron PSI80/82-System beschrieben wurden, können im Kontron PSI9xx-System nicht gelesen werden und umgekehrt.

NAME: zeigt den Inhalt an: z.B. KOS, UTILITY, FORTRAN usw.

VERSION: zeigt den Status an: z.B. 5.44

SPRACHE: zeigt die Sprachversion an:
D = deutsch
E = englisch
* = mehrsprachig

AUFSCHRIEB: zeigt das Aufschriebsverfahren an:
DD = doppelte Schreibdichte (für Kontron PSI80D/80Q/82D/9xx-Systeme)
SD = einfache Schreibdichte (für Kontron PSI80-Systeme und für Kontron PSI80D/82D-Systeme mit Treiber $MISS)

LAUFWERK: zeigt den Laufwerktyp an:
SS = einseitig (Kontron PSI80/80D/82D)
DS = doppelseitig (Kontron PSI80Q/9xx)

So bedeutet z.B.:

```
KOS  5.44D/DD/SS
!    !   !  !  !----- single sided
!    !   !  !
!    !   !  !-------- double density
!    !   !
!    !   !------------ deutsche Version
!    !
!    !---------------- Versionsnummer
!
!--------------------- Betriebssystem
```

Im allgemeinen erhalten Sie zu Ihrem Kontron PSI-System folgende Disketten:

- START: Startprogramm. Enthält Betriebssysteme! (nur Kontron PSI80/82)
- SYSKMD...: Systemkommandos
- UTILITY..: Hilfsprogramme

Kopieren Sie sich diese Disketten mit Hilfe der Startdiskette und bewahren Sie die Originale sorgfältig auf. Verwenden Sie für Ihre Arbeiten grundsätzlich nur die Arbeitsdisketten!

Das Gleiche gilt für optionale Software ebenso wie für Ihre eigenen Programmdisketten.

Kontron haftet nicht für verlorene oder beschädigte Disketten!

Bitte beachten Sie diese Hinweise sorgfältig. Denn: wir wünschen uns, daß Sie viel Erfolg mit Ihrem Kontron PSI-System haben!

Einführung in

Betriebssystem KOS und Systemkommandos

der Kontron PSI-Computersysteme

Version: 4.33/5.44/5.54/6.05

Mai 1984

Dieser Teil des Handbuches führt Sie ein in die Handhabung des Betriebssystems KOS und der Systemdienstprogramme. Sie verwenden diese z.B. beim Erstellen von Disketten für Ihre Anwendung, beim Kopieren und Löschen von Dateien oder beim Vorbereiten des Computersystems für einen Programmablauf, also beim Bedienen Ihres Computersystems.

Es schließt sich die Einführung in die Text-Grundsoftware und die Handhabung der Hilfsprogramme der Utility-Diskette an.

Die ausführliche Beschreibung vom Betriebssystem KOS und den Systemkommandos finden Sie in dem Abschnitt 'Systemkommandos' und 'KOS-Beschreibung' des Technischen Handbuches.

Texteditor und BASIC-Interpreter sind in den folgenden Abschnitten der Bedienungsanleitung genauer beschrieben.

Bei der weitergehenden Einführung helfen Ihnen unser 'Einführungskurs' und die Schulungen aus unserem Seminarangebot.

Inhaltsverzeichnis

1. Übersicht über Systemkommandos

Solange Ihr Computersystem vom Betriebssystems KOS gesteuert wird (also als letzte Meldung des Computers "KOS:" erschienen ist), können Sie eine Anzahl allgemeiner Funktionen (wie Disketten kopieren, formatieren usw.) über "Systemkommandos" veranlassen.

Hierfür gibt es "KOS-interne" und "KOS-externe" Kommandos.

"Interne" Kommandos werden ohne Externspeicherzugriff, also schneller ausgeführt. Einzelheiten hierüber beschreibt der Abschnitt "System-Kommandos" im "Technischen Handbuch".

a) KOS-interne Kommandos (Auswahl)

- C	Umschaltung Groß-/Kleinschreibung
- I	Ausdruck des Inhaltsverzeichnis eines Mediums
- M	Ausdruck und/oder Definition des Mastermediums
- N	Neuinitialisierung von Medien- oder E/A-Treibern
- P	Vor- und Rückwärtsblättern der Sichtschirm-Anzeige
- X	Wiederholung eines Programms (ohne Neuladung)

b) KOS-externe Kommandos (Auswahl)

- BASIC	Aufruf des BASIC-Interpreters (siehe Bedienungshandbuch, Abschnitt F)
- COPY	Kopieren von Daten von beliebigen Quellen auf beliebige Ziele
- DEL	Löschen von Dateien oder Dateigruppen
- DO	Ausführung einer Kommandodatei
- EDIT	Aufruf des Texteditors (siehe Bedienungshandbuch, Abschnitt E)
- FORMAT	Formatieren einer Diskette
- IL	Ausdruck des Inhaltsverzeichnisses eines Mediums (Diskette oder Plattenspeicher)
- INFO	Bildschirm-orientierter Ausdruck einer Textdatei
- IODC	Verwaltung von Ein-/Ausgabe-Kanälen
- MOVE	Kopieren von Dateien oder Dateigruppen
- PRINT	Ausdruck einer Textdatei
- REN	Umbenennung von Dateien
- STATUS	Ausdruck der Belegung aller aktiven Medien

Ihre Anwendungsprogramme stehen gleichberechtigt neben den "externen" Kommandos.

2. Start des Betriebssystems und Kommandoeingabe

Laden des Betriebssystems

Nach dem Einschalten des Computers und dem Einlegen der Systemdiskette in eines der Laufwerke wird automatisch das Betriebssystem KOS geladen. Auch von den integrierten Festplattenlaufwerken kann geladen werden, wenn die Festplatte richtig vorbereitet ist. Wenn Ihr System sich mit 'Login:' meldet, dann geben Sie Ihre Benutzerkennung ein und schließen die Eingabe mit der Taste 'RETURN' ab. Von Kontron ist grundsätzlich '*' als Login-Eingabe vordefiniert. Als erste Aktion des Betriebssystems wird dann die Kommandodatei KOS.INI geladen und die darin enthaltenen Kommandos werden ausgeführt.

Durch KOS.INI ist ein direktes Eintreten in ein Programmsystem zur Bearbeitung einer spezifischen Anwendung möglich. Standardmäßig enthält die Kommandodatei KOS.INI der Kontron Startdiskette eine Kommandofolge, die Sie direkt in ein Programm zum Erstellen von Sicherungskopien hineinführt (nur Kontron PSI80/82, KOS5.44/5.54).

Eingabebereitschaft

Als Hinweis, daß KOS bereit ist, Eingaben zu akzeptieren, erscheint die Meldung "KOS:" am Zeilenanfang. Daraufhin können Kommandos eingegeben werden. Alle Programme des Typs "COM" (siehe IL-Kommando) sind durch die Eingabe ihres Namens als Kommando aufrufbar.

Eine **Kommandoeingabe** ist mit der Taste 'RETURN', im folgenden symbolisiert durch "<---", abzuschließen.

Eingabefehler können jederzeit vor Abschluß einer Kommandozeile korrigiert werden. Hierzu stehen zwei Tasten zur Verfügung:

- RUBOUT löscht die gesamte bisher eingegebene Zeile
- CURSOR LEFT löscht das zuletzt eingegebene Zeichen

Beispiel für eine Kommandoeingabe:
Ein für eine erste Überprüfung des Diskettеninhaltes nützliches Systemkommando ist (Diskette im rechten bzw. oberen Laufwerk):

```
IL P=*<---
```

Daraufhin wird bei einer Systemdiskette 'SYSKMD' die Liste der "externen Systemkommandos" am Bildschirm erscheinen.

Eine **Kommandozeile** kann beliebig viele Kommandos enthalten, solange die Zahl der eingegebenen Zeichen 256 nicht überschreitet.

Beispiel:

```
M 1;IL P=*;DEL *.XXX<---
```

Trennzeichen zwischen den einzelnen Kommandos ist der Strichpunkt (;).

Mit einem Kommandoaufruf können Parameter für das aufgerufene Programm übergeben werden, die sonst von der Tastatur her einzugeben wären. Diese sind in Anführungszeichen (") zu setzen. Die Eingabe von "RETURN" wird durch "<" ersetzt.

Beispiel:

```
BASIC "LOAD MAGIC<RUN"
```

Damit können Sie schon im Aufruf die von dem aufgerufenen Programm erwarteten Eingaben mit angeben. Dies ist besonders wichtig für Kommandodateien (s.u.). Soweit CP/M-Programme schon beim Aufruf mit Parametern versorgt werden können, so gelten beim Aufruf noch KOS-Mediennummern (0,1...), während während des Laufes dieser Programme CP/M-Laufwerksbezeichnungen (A, B....) einzugeben sind.

Eingaben zur Ausgabesteuerung

Zunächst noch einige Hinweise über das Verhalten von KOS-Systemprogrammen bei Eingaben während ihres Ablaufs:

Alle Systemprogramme, die Ausgaben auf den Sichtschirm des Kontron PSI-Computers veranlassen, reagieren bei Eingaben während der Ausgabe mit nachstehender Tasteninterpretation:

ESC :	Programmabbruch (ESCAPE)
CTRL-S:	Geschwindigkeitsumschaltung (= 'Speed') der Bildschirmausgabe (Taste "CTRL" und Taste "S" gleichzeitig)
CTRL-P:	Ausgabeunterbrechung mit der Möglichkeit mit den Cursortasten auf den zurückliegenden Text (bis zu 8 Seiten) zurückzuschalten, bis wieder eine beliebige Taste gedrückt wird (= 'Page-Mode'). Es gilt dabei: CURSOR UP : eine Zeile vorwärts CURSOR DOWN : eine Zeile rückwärts CURSOR RIGHT: eine Seite vorwärts CURSOR LEFT : eine Seite rückwärts
übrige Tasten:	Programmunterbrechung, bis wieder eine beliebige Taste gedrückt wird.

HELP-Funktion

Alle KOS-Dienstprogramme, die Parameter benötigen, geben beim Aufruf ohne jeglichen Parameter Hinweise über die erforderliche Syntax eines korrekten Kommandoaufrufs aus. Derartige Ausgaben erfolgen grundsätzlich in Klartext und enthalten im einfachsten Fall zwei bis drei Textzeilen. Bei komplexeren Kommandos (z.B. COPY) werden detaillierte Hinweise und Beispiele ausgegeben.

Beispiel:

```
IODC<---  führt zum Ausdruck der Syntax
          des IODC-Kommandos.
```

3. Medien (Dateiorientierte Datenträger)

Jeder Dateiadresse kann eine **Mediennummer 'mn'** (Ziffer 0 bis 9), gefolgt von einem Doppelpunkt, vorangestellt werden, wenn ein Datenträger gezielt angesprochen werden soll, z.B. IL 1:TEST<---

Das Betriebssystem KOS sucht Kommandos bzw. Datendateien auf dem "Mastermedium". Dies ist zunächst das Floppy Disk-Laufwerk 0 (Kontron PSI80/82/908/98: rechtes Laufwerk, Kontron PSI980: oberes Laufwerk oder, wenn das System von dort geladen wurde, die integrierte Festplatte.

Ein Medium ist ein Datei-orientierter Datenträger, also etwa ein Disketten-Laufwerk oder ein Plattenlaufwerk. Medien werden über einen Medientreiber angesprochen. Medien besitzen eine Indexdatei, in der die auf dem Medium geführten Dateien mit Name, Typ, Eigenschaften, Länge und physikalischer Adresse auf dem Medium verwaltet werden. Diese Datei hat den Typ 'DIR'. Ihr Name ist der Medienname.

Auskünfte über Medienname und Belegungszustand gibt das Systemkommando STATUS, über den Inhalt die Systemkommandos I und IL. Standardmäßig ist dem rechten Floppy Disk-Laufwerk die Mediennummer 0, dem linken die Mediennummer 1 zugeordnet. Bei Kontron PSI80/82-Systemen mit integrierter Festplatte sind die Mediennummern 0 (FD) und 2 (HD) beim Laden von KOS von Diskette, bzw. 2 (FD) und 0 (HD) beim Laden von der Festplatte zugewiesen. Dies gilt jeweils nach Aktivieren des anderen Mediums durch das IODC-Kommando.

Bei Kontron PSI980/908/98-Systemen bestimmt die werksseitige oder benutzerspezifische Einstellung der Betriebssystemmodule die Medienzuordnung, siehe KOSGEN-Kommando.

4. Dateinamen, Dateitypen und Dateieigenschaften

In KOS-Systemkommandos werden in der Regel eine oder mehrere Dateiadressen als Parameter angesprochen. Ihr Aufbau wird im folgenden erläutert.

Dateiadressen sind symbolische Adressen (Namen) für Informationsaufzeichnungen auf Floppy Disks oder anderen externen Speichermedien (Platte etc.). Sie geben die Zuordnung zwischen symbolischer und physikalischer Adresse (Name <---> Spur/Sektor) durch die Dateiverwaltung an, sorgen also dafür, daß Texte, Daten und Programme auf den Massenspeichern wieder auffindbar sind.

Dateiadressen bestehen aus der Mediennummer, dem **Dateinamen** und dem **Dateityp.** Name und Typ sind durch einen Punkt getrennt; die maximale Zeichenanzahl beträgt 8 für den Namen bzw. 3 für den Typ. Der zulässige Zeichenvorrat umfaßt alle zum Aufbau von Kommandos (A...Z, a...z, 0...9) erlaubten Zeichen. Zwischen Namen und Typ dürfen neben dem Punkt (.) keine weiteren Trennzeichen stehen. Die Typbezeichnung ist frei wählbar. Auf Kontron-Disketten sind diese **Vorzugstypen** standardmäßig verwendet:

.SRC	Assemblerquellcode	.BAS	BASIC-Programme
.OBJ	Programmoduln	.BSC	BASIC-Programme, editierbar
.COM	Maschinencodeprogramme	.DAT	BASIC-Datendateien
.XXX	Editorgrundtyp	.INF	Informationstexte (INFO)
.CMD	Kommandodateien	.PRN	Listing- oder Druckdateien
.INI	nur für Systemstart	.SEL	Menüdateien (RLOAD SELECT)
.DIR	Directory-Datei (Inhalt)	.SYS	Betriebssystemdateien
.FOR	FORTRAN-Quellcode	.MAS	Maskendateien
.LST	Listen	.IOD	E/A-Treiber(gelinkt)
.COB	COBOL-Quellcode	.BAK	WordStar-Backup

Die **Mediennummer** ist beim Aufsuchen von Dateien optional, da KOS Dateien automatisch auf allen aktiven Medien sucht, wenn keine Mediennummer angegeben ist.

Dateigruppen

Wichtige Kommandos, wie z.B. IL oder DEL, können eine beliebige Anzahl von Dateien mit einer gemeinsamen Eigenschaft gleichzeitig ansprechen. Eine solche gemeinsame Eigenschaft kann beispielsweise sein:

- ein bestimmter Dateityp
- ein bestimmter Buchstabe im Dateinamen
- ein bestimmter Dateiname
- eine bestimmte Dateieigenschaft
- ein bestimmtes Medium

Die Auswahl nach Namen oder Typ geschieht durch zwei Sonderzeichen:

- das Zeichen '*' wirkt als Universalbezeichner. Es kann für eine beliebige Anzahl von Zeichen eines der beiden Teile einer vollständigen Dateiadresse stehen.
- Das Zeichen '?' ersetzt ein beliebiges im jeweiligen Namensteil zugelassenes Zeichen einer Dateiadresse.

Beispiele:

*.ASM	alle Dateien vom Typ ASM
ABC.*	alle Dateien des Namens ABC mit beliebigem Typ
ABC*.BAS	alle Dateien, die einen mit ABC beginnenden Namen und den Typ BAS haben
X*.?	alle Dateien, deren Name mit X beginnt und deren Typkennung aus einem Zeichen besteht
1:*	alle Dateien auf dem Medium 1

Dateieigenschaften

kennzeichnen zusätzlich zur Dateiadresse die Art der Datei. Eine Datei kann als geheim, löschgeschützt, schreibgeschützt, etc. gekennzeichnet werden und mit einem Benutzeridentifikationswort (Schlüssel) versehen sein, s. DEFP-Kommando.

Diese Leistung von KOS hilft Ihnen bei der Bildung von Dateigruppen zur übersichtlicheren Handhabung und sie erhöht die Sicherheit gegen unbeabsichtigtes oder beabsichtigtes Verändern und Löschen von Dateien.

Working Directories

Unter KOS5.54/6.0 können Dateien in logische Gruppen zusammengefaßt werden. Eine solche Gruppe wird durch ihr Working Directory wdir gekennzeichnet. Alle Kontron-Programme werden als "Public" oder im Working Directory "*" geliefert. Dieses Working Directory ist auch mit dem Login "*" vordefiniert.

Die volle Dateiadresse lautet somit: /wdir/mn:name.typ

Beispiel:

```
IL /*/0:ABC???.DAT<---
```

5. Ein-/Ausgabe

Die Ein-/Ausgabe ist über aktivierbare Treiber gelöst, die 'logischen Kanälen' zugewiesen werden (IODC-Kommando).

Anwendungsprogramme kommunizieren nur mit logischen Kanälen, sind also im Grundsatz unabhängig von z.B. der Art des daran angeschlossenen Druckers oder Peripheriegerätes, sofern natürlich der notwendige Funktionsumfang gegeben ist.

Die Utility-Diskette enthält Treiberprogramme für die Kontron-Schnittstellen und Standardperipherie. Diese Programme werden im Quellcode geliefert und können somit als Muster für anwendungsspezifische Treiber verwendet werden.

6. KOS-interne Systemkommandos *

KOS-interne Kommandos werden beim Start des Betriebssystems mitgeladen. Sie stehen jederzeit zur Verfügung, da sie nicht neu von Diskette geladen werden müssen. Interne Kommandos bestehen grundsätzlich aus nur einem Kennbuchstaben, der sowohl groß, als auch kleingeschrieben werden kann. Klammern im Aufrufformat kennzeichnen optionale Parameter.

C - Kommando (Change Case)

Aufruf: C<---

Umschaltung zwischen Groß- und Kleinschreibung. Nach dem Laden des Betriebssystems ist dieses im Modus 'Großschreibung'. Das C-Kommando schaltet den jeweiligen Modus um. Als Hinweis auf den gerade aktiven Modus gibt das Betriebssystem die Promptzeichen 'KOS:' klein- oder großgeschrieben aus (nicht bei KOS6).

I - Kommando (Index)

Aufruf: I (mn):(dateiname.typ)<---
I (mn):(dateigruppe)<---

Auflistung aller im Parameterfeld des Kommandos spezifizierten Dateien. Die Angabe aller Parameter ist optional. Fehlt die Mediennummer, so wird nur das Mastermedium adressiert. Ist das Parameterfeld nicht besetzt, so werden alle Dateinamen des derzeitigen Mastermediums ausgegeben. Es werden nur die nicht geheimen Dateien aufgelistet, s. DEFP-Kommando.

Beispiele:

I ABC.*<---	listet alle Dateien namens ABC des Masterlaufwerks
I ???.BAS<---	alle vom Typ BAS mit 3-stelligem Namen
I 2:<---	alle (nicht geheimen) Dateien von Medium 2
I<---	alle (nicht geheimen) Dateien des Mastermediums

* Mehr Information über diese und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".

M - Kommando (Master medium)

Aufruf: M (mn)<---

Definition und/oder Ausdruck des Mastermediums. Ist kein Parameter spezifiziert, wird die Nummer des derzeitigen Mastermediums ausgedruckt:
---> KOS: Mastermedium = 0

Das Mastermedium ist dasjenige Medium, auf dem bei Dateireferenzen mit fehlender Mediennummer zuerst gesucht wird.

N - Kommando (New)

Aufruf: N $iodn<---
N mn<---
N<---

Neuinitialisierung der Dateiverwaltung oder eines E/A-Treibers. Die Funktion wird automatisch beim Wechsel von Disketten durchgeführt, sofern die Diskettennamen sich voneinander unterscheiden. **Wenn zwei Disketten den gleichen Namen haben - was man möglichst vermeiden sollte - muß dieses Kommando beim Wechsel gegeben werden.**

Das Kommando 'N $iodn' führt zur Neuinitialisierung eines E/A-Treibers, z.B. um neue Steuerparameter einzugeben.

P - Kommando (Page Mode)

Aufruf: P<---

Vor- und Zurückblättern der bisherigen Sichtschirmausgaben. Der Bildwiederholspeicher des Kontron PSI-Computers speichert ständig die letzten 8 Textseiten. Nach dem P-Kommando kann mit den Cursortasten ein beliebiger Ausschnitt der letzten 8 Textseiten auf dem Sichtschirm dargestellt werden. Durch dieses Kommando können z.B. auf einfachste Weise Bedienfehler rekonstruiert oder Ausgaben angesehen werden, die vom Bediener 'verpaßt' wurden, ohne daß dabei die gesamte Ausgabeaktivität wiederholt werden muß.

Tastenbedeutung:

CURSOR UP :	eine Zeile vorwärts schalten
CURSOR DOWN :	eine Zeile rückwärts schalten
CURSOR RIGHT:	eine Seite vorwärts schalten
CURSOR LEFT :	eine Seite rückwärts schalten

Alle anderen Tasten bringen den ursprünglich vorhandenen Bildausschnitt wieder zurück und schließen das P-Kommando ab.

*** Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".**

X - Kommando (Execute)

Aufruf: X (p1 p2 ... pN)<---

Mit diesem Kommando können Sie ein bereits geladenes Programm erneut starten. Wenn das Programm zusätzliche Informationen (p1...pN) zum Ablauf fordert, dann werden diese nach "X" in der gleichen Art angegeben wie beim ersten Aufruf des Programms.

Wenn Sie ein Programm nur laden wollen, ohne es sofort auszuführen, dann ist der Aufrufname (ohne weitere Parameter) mit ',' abzuschließen: z.B. "INFO,<---". Dann können z.B. Disketten gewechselt werden oder andere interne KOS-Kommandos (z.B.: I, N, M, etc.) vor dem Start des geladenen Programms ausgeführt werden. Das geladene Programm belegt jedoch den Speicherplatz solange, bis es wenigstens einmal ausgeführt wird, solange kann kein anderes Programm in den gleichen Speicherbereich geladen werden.

* Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".

7. Diskresidente KOS-Systemkommandos *

COPY - Kommando (allgemeiner Datentransfer)

Aufruf: COPY quelle ziel<---

Das Programm COPY dient zum Transfer von Daten von einer beliebigen Quelle an ein beliebiges Ziel. Als Datenquelle und/oder Datenziel sind möglich:

- eindeutig spezifizierte Dateien auf einem Medium (z.B. Diskette)
- E/A-Treiber (gekennzeichnet durch ein $-Zeichen)

Ist die Zieldatei auf dem Zielmedium bereits vorhanden, so antwortet COPY mit der Rückmeldung:

----> COPY: Zieldatei bereits vorhanden - Datei überschreiben? (J)

Bei den Eingaben 'J' und 'Y' wird die Datei dann überschrieben.

Beispiele:

```
COPY 0:PSI1 1:PSI2<---
COPY PSI1.COM<---
COPY TEST.PRN $SIOA<---
COPY $KEY $SIOA<---
COPY $PSIA TEST.X<---
```

COPYM2 - Kommando (Diskettenkopieren in 2-Laufwerk-Systemen)

Aufruf: COPYM2<---

Kopiert den gesamten Inhalt einer Diskette im Medium 0 auf eine bereits formatierte (s. "FORMAT") Diskette im Medium 1. Zur Verhinderung von unbeabsichtigtem Löschen von Disketten stellt COPYM2 zunächst die Rückfrage:

```
Original von Medium M-0
Kopie auf Medium M-1
Medien bereit ? (J/N)
```

Wird die Rückfrage mit "J" beantwortet, dann beginnt der Kopiervorgang. Satz für Satz wird übertragen. Das Programm kann jederzeit durch die Taste 'ESC' abgebrochen werden.

Beispiele:

```
COPYM2<---   Erzeugt im Laufwerk1 die Kopie der Diskette in Lw.0.
```

*** Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".**

DEFP - Kommando Dateieigenschaften (Properties definieren)

Aufruf: DEFP (mn):name.typ<---

Anzeigen und Ändern der Dateieigenschaften

Dateieigenschaften kennzeichnen eine Datei ebenso wie ein Dateiname. Sie sind nützlich, um z.B. Gruppen von Dateien zu kennzeichnen. Alle KOS-Kommandos sind in Dateien mit der Kennung "K=Public" gespeichert.

Die Kennung "S" (secret) sorgt dafür, daß die so markierten Dateien beim I-Kommando nicht mit aufgelistet werden, sodaß nur die Arbeitsdateien eines Benutzersystems sichtbar werden.

Zudem können Dateien schreibgeschützt ("W") und löschgeschützt ("E") sein, und auch eine Benutzer-Identifikation tragen: dies erhöht die Sicherheit gegen unbeabsichtigtes Ändern oder Löschen von wichtigen Dateien.

Werden die Kommandos DEL, IL, MOVE und REN ohne den Parameter P=* oder P=S angewendet, dann bleiben geheime Dateien unberücksichtigt.

DEL - Kommando (Datei löschen)

Aufruf: DEL (mn):name.(typ) (P=*)<---

Mit diesem Kommando können Sie einzelne Dateien und Dateigruppen löschen. Das Programm DEL bringt den Namen der ersten zu löschenden Datei auf den Bildschirm und wartet dann auf eine Eingabe.

Folgende Eingaben sind daraufhin möglich:

- J - Die Datei wird gelöscht
- N - Die Datei wird nicht gelöscht
- A - Alle Dateien der aufgerufenen Dateigruppe werden gelöscht, Rückfragen werden nicht mehr gestellt
- K - Abbruch des Kommandos (KOS-Rücksprung)

Bei Dateien mit Datei-Eigenschaften muß der Parameter P=* angegeben werden (siehe 'DEFP'-Kommando).

Beispiele:

DEL DATEI.ABC<---	Datei "DATEI.ABC" kann gelöscht werden
DEL *.PRN<---	Alle Dateien des Typs "PRN" sind zu löschen
DEL *.COM P=S<---	Alle geheimen Dateien des Typs "COM" sind zu löschen.

* Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".

DO - Kommando (Ausführung einer Kommandodatei)

Aufruf DO (mn):name.typ p1 p2 ... p9<---

Ausführung von Kommandos aus einer Kommandodatei. Eine Kommandodatei ist eine mit dem Editor EDIT erstellte Datei, die eine beliebige Anzahl von Kommandos enthält. Zur Trennung einzelner Kommandos innerhalb der Datei dienen die Zeichen Strichpunkt (;) sowie RETURN (Zeilenende).

Bis zu 9 Parameter können beim Aufruf des DO-Kommandos übergeben werden.

Beispiel:
DO PRIVAT BILD1<--- führt die Kommandodatei "PRIVAT" aus

Die Datei PRIVAT.CMD hat folgenden Inhalt:
BASIC "LOAD #1<RUN"

Dadurch wird erst der BASIC-Interpreter geladen, der nun das Programm BILD1 lädt und startet.

IODC - Kommando (Ein-/Ausgabetreiber-Aktivierung)

Aufrufe:

(1)	IODC $(mn:)iodn=ACTIVE<---	Aktivierung
(2)	IODC $iodn=DEACTIVE<---	Deaktivierung
(3)	IODC X-n=$iodn<---	Kanalzuordnung (X=I,O,M)
(4)	IODC LIST<---	Auflistung der aktiven Treiber
(5)	IODC SYNTAX<---	Hilfe-Funktion für IODC

Damit können Treiber aktiviert (1), deaktiviert (2), logischen Kanälen für Eingabe "I", für Ausgabe "O" oder einem Medienkanal "M" der Nummer n (3) zugeordnet werden. Mit einem Aufruf können bis zu 9 Aktivitäten ausgelöst werden.

Beispiele:

IODC $SIOA=ACTIVE O-2=$SIOA LIST<---
Aktivierung und Kanalzuweisung für Treiber $SIOA, Auflisten.

IODC $SIOA=DEACTIVE $UPA=ACTIVE O-2=$UPA<---
Umlenken der Ausgaben auf Kanal 2 vom Treiber $SIOA auf $UPA.

FORMAT - Kommando (Diskette formatieren)

Aufruf: FORMAT P V<---

Mit diesem Kommando wird eine Diskette zum Gebrauch auf Ihrem Kontron PSI-System vorbereitet. FORMAT stellt Rückfragen nach dem Laufwerk, in dem die Diskette formatiert werden soll, nach dem Namen, der diese Diskette kennzeichnen soll (gleiche Namen vermeiden!), und beginnt den Formatiervorgang nach der Bestätigung durch den Benutzer.

*** Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".**

IL - Kommando (Inhaltsausgabe im Langformat)

Aufruf: IL (mn):name(.typ) (J) (P=*)<---

Ausdruck des Inhaltsverzeichnisses eines Mediums mit Angabe der Dateilänge und weiterer Informationen. Wenn als "geheim" gekennzeichnete Dateien mit angezeigt werden sollen, so muß der Parameter P=* oder P=S mit dem Aufruf angegeben werden.

Beispiele:

IL TEST.*<---	Alle Dateien des Namens "TEST" auf dem Masterlaufwerk werden angezeigt.
IL 1: P=* O=$SIOA<---	Alle Dateinamen von Laufwerk 1, auch die geheimen, werden auf ein Gerät übertragen, das an den Serienkanal A des Kontron PSI-Systems angeschlossen ist.

INFO-Kommando (Ausgabe einer ASCII-Datei)

Aufruf: INFO (mn):name.typ<---

Dieses Kommando bringt Textdateien auf den Sichtschirm und ermöglicht über die Eingabe von "Leertaste" bzw. "RETURN" das Vor- und Zurückblättern im Text.

Beispiele:

INFO KOS<---	Bringt die Textdatei KOS.INF zur Anzeige
INFO<---	Sucht die erste Datei des Typs "INF" des Masterlaufwerks und bringt ihren Inhalt auf den Sichtschirm.

*** Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".**

MOVE - Kommando (Kopieren von Dateien)

Aufruf: MOVE (quellmedium:)name(.typ) (zielmedium) (J) (P=*)<---

Mit diesem Kommando können - im Gegensatz zum COPY-Kommando - auch Gruppen von Dateien von einem Medium auf ein anderes Medium (z.B. von Diskette auf Platte) transportiert werden.

Der Dateiname bleibt hierbei erhalten. Bereits vorhandene Dateien des angegebenen Namens werden überschrieben, sofern sie nicht die Eigenschaft "schreibgeschützt" haben.
Die Angabe von 'quellmedium:','.typ', 'zielmedium' und 'param' ist optional, die Reihenfolge von 'zielmedium' und 'param' ist beliebig.

Ist für 'param' 'J' angegeben, so erfolgt vor dem Start jedes Kopiervorgangs eine Rückfrage an den Benutzer.

Beispiele:

MOVE * J<---	Alle nicht geheimen Datein werden von Medium 0 auf Medium 1 übertragen, sofern die Rückfrage mit "J" beantwortet wird.
MOVE 0:TEST.* 2: P=*<---	Alle Dateien werden von Medium 0 auf Medium 2 übertragen.

PRINT - Kommando (Ausgabe einer ASCII-Datei)

Aufruf: PRINT (mn):name.typ (P=*) (O=$iodn)<---

Ausgabe einer ASCII-Datei auf den Sichtschirm des Kontron PSI Computers oder auf ein beliebiges anderes Peripheriegerät. Die Angabe von 'mn:' ist optional. Der Ausgabekanal kann explizit angegeben werden.

Beispiele:

PRINT 1:TEST.SRC<---	Die Textdatei TEST.SRC von Medium 1 wird angezeigt.
PRINT TEST.PRN O=$SIOA<---	Die Textdatei TEST.PRN wird über das Treiberprogramm $SIOA am Serienkanal A ausgegeben.

*** Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".**

REN - Kommando (RENAME = Umbenennung einer Datei)

Aufruf: REN (mn):namealt.typ nameneu.typ<---

Umbenennung der Datei 'namealt.typ' in 'nameneu.typ'. Bei der Datei 'namealt' kann der Typ durch den Universalbezeichner '*' ersetzt werden, dann werden alle Dateien des Namens 'namealt' umbenannt. Die Dateieigenschaften müssen angegeben werden (siehe 'DEFP'-Kommando).

Beispiele:

REN BASIC.COM BNEU<---	die Datei wird umbenannt in BNEU.COM
REN TEST.* TEST1<---	alle Dateien namens TEST werden zu TEST1
REN DEL DELETE P=*<---	das DEL-Kommando wird in DELETE umbenannt. Bei allen Beispielen wird die Datei auf dem Mastermedium gesucht.

STATUS - Kommando (Information über Medienbelegung)

Aufruf: STATUS<---

Ausgabe der Medienbelegung: Treiber, Medienname, gefüllter Bereich und freier Bereich werden angezeigt. **Wichtiges Kommando zur Überprüfung, ob ein Medium noch logisch korrekt (consistent) belegt ist.**

Hinweis: "Consistent" bedeutet, daß die Verwaltung der Dateien auf der Platte oder Diskette korrekt abläuft, d.h.: neue Dateien werden in freie Bereiche abgelegt, ohne die bisherigen Dateien zu stören. Disketten können "inconsistent" werden, wenn Disketten **gleichen Namens** während des Betriebes gewechselt werden, ohne daß der Dateiverwaltung dies durch z.B. ein N-Kommando mitgeteilt wird. Die Dateiverwaltung merkt sich nämlich freie Bereiche auf einer Diskette und schreibt neue Daten dort hinein. Bei Namensgleichheit der Diskette erkennt die Dateiverwaltung nicht immer einen Diskettenwechsel und überschreibt dann möglicherweise Teile von Dateien. Sollte dies eingetreten sein, dann hilft Ihnen das Programm FSCHECK, siehe "Technische Beschreibung", Abschnitt "Systemkommandos".

*** Mehr Information über dieses und weitere Kommandos im Abschnitt "Systemkommandos" in der "KOS Technischen Beschreibung".**

8. Einführung in den Editor EDIT

Grundsätzlich dient er zu zwei wichtigen Aufgaben:

a) zur computerunterstützten Erstellung von Dokumentation, Briefen Texten, Formularen usw. So macht EDIT ihren Kontron PSI-Computer zu einem leistungsfähigen Textverarbeitungssystem.

b) zur Eingabe, Pflege und Archivierung von Programmen, die in ASSEMBLER, FORTRAN, PASCAL oder anderen Sprachen geschrieben sind (die Erstellung von BASIC-Programmen kann direkt durch den BASIC-Interpreter erfolgen).

Der Editor unterscheidet zwei Betriebsarten:

1) Zeichen-orientierte Betriebsart:
In der Zeile am unteren Bildrand, der Kommandozeile, erwartet der Editor Kommandos zur Bearbeitung der darüberliegenden Zeile und/oder nachfolgender Zeilen.

2) Cursor-orientierte Betriebsart:
Die Cursor-Kommandos wirken auf die jeweilige Cursorposition. Die Positionierung erfolgt durch die Cursortasten (Pfeil: links, rechts, oben, unten, Cursor = blinkendes Zeichen).

Neueingabe von Programmen oder Texten

Hierzu dient das EDITOR-Programmpaket, das vom Betriebssystem aus geladen werden muß. Den von Ihnen gewählten, bisher nicht existierenden Dateinamen bezeichnen wir mit 'name.typ'. Der Aufruf des Editors geschieht mit

```
EDIT name.typ<---
```

Als Antwort erhalten Sie nach dem Laden des Editorprogramms die Aufforderung 'Eingabe', wenn eine Datei dieser Dateiadresse noch nicht existiert.

Falls es sich nicht wie in unserem Beispiel um eine neue, sondern um eine bereits existierende Datei handelt, erscheint in der untersten Bildschirmzeile (= 'Kommandozeile') die Aufforderung 'KMD' (= Kommando), die Sie darauf hinweist, daß der Editor nun bereit ist, Kommandos zu verarbeiten. Soll in diesem Zustand ein neuer Text oder ein neues Programm eingegeben werden, ist hier die Eingabe

```
E<---
```

erforderlich. Wie bereits beschrieben, wird bei neuen Dateien dieser Eingabezustand automatisch eingestellt.

Wollen Sie nach der Eingabe dem Editorprogramm wieder Anweisungen ('Kommandos') geben, müssen Sie zweimal unmittelbar hintereinander die RETURN-Taste drücken. Auf der Kommandozeile am unteren Bildschirmrand erscheint dann

KMD:

und der EDITOR ist wieder bereit, Ihre Anweisungen zur Text-Bearbeitung entgegenzunehmen.

Eine erste Übersicht über die Fähigkeiten des Editors gibt Ihnen das Kommando

?<---

das Ihnen jederzeit von der Kommandobetriebsart aus Hilfe beim Editieren bietet.
Zurück ins Betriebssystem gelangt man durch Eingabe von

KOS<---

Damit wird auch der eben eingegebene Text auf die Diskette abgespeichert.

Die Eingabe von

KOS-<---

führt ohne Abspeicherung zurück ins Betriebssystem KOS, d.h., die Datei auf dem Medium wird nicht verändert, bzw. eine neue Datei wird nicht abgespeichert.

Änderung von bereits vorhandenen Programmtexten mit dem EDITOR und Abspeichern auf Diskette

Soll ein Programm oder eine beliebige Datei modifiziert werden, ist ebenfalls der Editor aufzurufen. Dies geschieht wieder mit dem Kommando

EDIT mn:name.typ<---

wobei 'name.typ' der Name des bereits auf einem Medium gespeicherten Textes ist. Danach erwartet das System Ihre Anweisungen.

Die genaue Beschreibung der Editor-Kommandos finden Sie im Kapitel "EDITOR-Beschreibung" in diesem Handbuch.

9. Peripheriegeräte

Zum Anschluß eines Peripheriegerätes an Kontron PSI-Systeme werden 2 Komponenten benötigt:

1. Hardware: das Gerät, das serielle oder parallele Verbindungskabel und die serielle oder parallele Schnittstelle am Kontron PSI-System

2. Software: ein Ein-/Ausgabe-Programm (= Treiber), welches die Funktionen des Peripheriegerätes und der Schnittstelle steuert.

Hardware:

Die Buchsenbelegung an der Geräterückseite ist durch Namen gekennzeichnet, zum Beispiel:

- Tastatur : Keyboard
- Serienkanal 1 : SIO-A
- Serienkanal 2 : SIO-B
- Parallelschnittstelle: Centronics
- weitere Schnittstellen

Serielle Geräte sind i.a. an Serienkanal 1 oder 2 anzuschließen, parallele an die Parallelschnittstelle. Die Belegung des Verbindungskabels ersehen Sie bei Standardperipherie aus der Info-Datei auf der Treiber- oder Utility-Diskette bzw. aus dem Abschnitt "KOS-Utilities" der "Technischen Beschreibung".

Software:

Jedes Peripheriegerät benötigt ein Treiberprogramm, das die "Hardware" mit der "Betriebssoftware" verbindet. Bei den von Kontron Mikrocomputer GmbH angebotenen Geräten wird ein Basistreiber auf Diskette mitgeliefert. Um das Peripheriegerät ansprechen zu können (z.B. Drucker), muß der Treiber aktiviert sein; bei Ansprache aus Sprachen muß auch ein Ein- oder/und Ausgabekanal zugewiesen werden. Dies geschieht mit dem IODC-Kommando:

```
IODC $iodn=ACTIVE<---     ($iodn = Ein-/Ausgabetreibername)
IODC x-n=$iodn<---        (x=I,O, n=1...9)
```

Näheres hierzu unter 'Diskresidente KOS Systemkommandos' in diesem Abschnitt und in "Technischer Beschreibung", Abschnitt "Systemkommandos" und "KOS-Utilities".

Die von Kontron gelieferten Universal-Treiber haben im allgemeinen folgende Einstellmöglichkeiten:

serielle/parallele Betriebsart
deutsche/englische Meldungen

Weitgehend sind alle Möglichkeiten bereits fertig vorbereitet und durch Editieren und Assemblieren der jeweiligen Quellcodedatei anzuwählen, näheres siehe "Technische Beschreibung", Abschnitt "KOS-Utilities".

Wollen Sie ein beliebiges nicht standardmäßig unterstütztes Peripheriegerät ansprechen, stehen Ihnen dazu die allgemeinen Treiber PSIA, PSIB, SIOA, SIOB, UPA, UPB, UPP und PIO zur Verfügung, die die seriellen Schnittstellen A oder B oder die parallele Schnittstelle bedienen. Dazu näheres in der "Technischen Beschreibung", Abschnitt "KOS-Utilities".

Die Abstimmung der Fähigkeiten von Peripheriegerät auf ein Programmpaket (z.B.: Microjustification aus WordStar heraus etc.) ist im allgemeinen kundenseitig vorzunehmen. Kontron übernimmt keine Verantwortlichkeit für die Funktionalität von beliebigen Peripheriegeräten mit beliebigen Programmen. Die zur Verfügung gestellten Basis-Treiber erlauben jedoch den Anschluß an serielle (RS232C) oder parallele (Centronics) Schnittstellen und die grundsätzliche Bedienung von Geräten mit diesen Schnittstellen.

10. Erstellung von Arbeitsdisketten

Auf den Kontron gelieferten Disketten sind eine Reihe von Programmen enthalten, die für die tägliche Arbeit bei einer Anwendung immer gebraucht werden, z.B. IODC, DO, Treiber etc. Diese müssen im Zugriff des Systems sein. Andere Programme sind nur zu seltener anfallenden Arbeiten notwendig oder sind nur für den Programmentwickler von Interesse, Beispiele sind FORMAT, COPYM2, etc.
Es spart Platz und Zugriffszeit und bringt bessere Übersicht, wenn nur die benötigten Programme auf einer 'Arbeitsdiskette' oder der Festplatte zusammengefaßt sind. Übrigens kann dieser Satz von Programmen jederzeit mit Hilfe des Dienstprogrammes 'MOVE' erweitert werden.

Im allgemeinen brauchen Sie für Ihre tägliche Arbeit folgende Programme bzw. Dateien:

KOS.SYS	Betriebssystem Kontron PSI80/82
KOSx.SYS	Betriebssystem Kontron PSI9xx (x=0..8, x=9 für KOBUS-Master, x=A für externe Platte WINS, x=B für integrierten CP/M-Translator)
BOOT2.SYS	Urlader für das Betriebssystem
DO.COM	Ausführen einer Kommandodatei
IODC.COM	Treiberaktivierung (bei KOS6 zusätzlich IODC.MAS)
KOS.INI	Kommandodatei zum automatischen Systemstart
Treiber	für Drucker, Festplatte/Diskettenlaufwerk etc.

Dazu kommen Ihre Anwendungsprogramme und Datendateien.

Für die Diskettenpflege gehören dazu:

COPY.COM	Kopieren/Umbenennen von Dateien
MOVE.COM	Kopieren von Dateien/Dateigruppen
COPYM2.COM	Kopieren von Disketten bei 2-FD-Laufwerken
COPYWFD.COM	Kopieren von Disketten bei Festplattensystemen
FORMAT.COM	Formatieren/Prüfen von Disketten/FD-Laufwerken
MAKEFS.COM	Logisches Formatieren von Disketten/Festplatten
IL.COM	Auflisten von Inhaltsverzeichnissen
REN.COM	Umbenennen von Dateien
STATUS.COM	Anzeige der Diskettenbelegung
CPFILE.COM	Vergleich von Dateien
DATE.COM	Einstellen des Diskettendatums
WDIR.COM	Einstellen von Working Directories (meist bei Plattensystemen, mit WDIR.MAS und WDIR.LST)

Für die Text- und Programmerstellung benötigen Sie:

EDIT.COM	Texteditor
BASIC.COM	BASIC-Interpreter
PRINT.COM	Ausdruck von Textdateien
DEFP.COM	Einstellen von Dateikennungen

Bei der Systemsteuerung helfen Ihnen:

RLOAD.COM	Relokativer Lader
SELECT.COM	Menü-Programm (mit MENU.MAS u. *.SEL)
SPOOL.COM	Druckerspooler
PTASK.COM	Drucker-Spooltask
TASK.COM	Task-Anzeige/Verwaltung

Weitere Programme, z.B. für die Programmierung in COBOL, FORTRAN, PASCAL, BASIC80-Compiler/Interpreter etc., siehe dort.

EDITOR der Kontron

PSI-Systeme

Version 4.33/5.44/5.54/6.05

Mai 1984

Dieser Teil des Kontron PSI-Bedienungshandbuchs beschreibt die Text- und Programmeditierung. EDIT erlaubt Zeilen- und Cursor-orientiertes Arbeiten. Einprägsame Kommandowörter und die Bedienerführung in der Kommandozeile machen diesen Editor benutzerfreundlich.

Inhalt

Kontron - EDITOR

Das Kontron-Editorprogramm EDIT verbindet zeilen- und cursor-orientierte Funktionen in idealer Weise.

Besonderes Merkmal von EDIT ist, daß es alle wichtigen Funktionen mit einer sorgfältig durchdachten Auswahl sehr weniger **Kommandowörter** erfüllt; das macht ihn einfach zu erlernen und unanfällig für Bedienungsfehler.

24 Zeilen des Textes werden im Zusammenhang am Bildschirm dargestellt. Dies macht die Textbearbeitung übersichtlich.

In der **Zeilen-orientierten Betriebsart** bearbeitet EDIT die jeweils unterste der 24 Zeilen und die folgenden Zeilen.

In der **Cursor-orientierten Betriebsart** wirken die Kommandos auf die jeweilige Cursorposition. Die Positionierung erfolgt durch die Cursortasten.

Control-Zeichen können in den Text eingefügt werden. Sie werden in inverser Darstellung protokolliert.

Die **Kommunikation** mit anderen Dateien wird unterstützt durch Einfüge- und Abspeicher-Kommandos.

Die **Zeilennumerierung** wird stets auf aktuellem Stand gehalten: das Einfügen oder Löschen von Zeilen bewirkt die Umnumerierung aller folgenden Zeilen.

Die **Zeilenlänge** ist auf 71 Zeichen begrenzt, bei der Bearbeitung längerer Zeilen werden die überstehenden Zeichen abgeschnitten.

1. Aufruf des Editors

Das Editorprogramm ist in einer Datei des Namens "EDIT", des Typs ".COM", der Eigenschaften "KWES" (s. Systemprogramm DEFP) und der Länge 10kByte auf der Systemdiskette gespeichert. Der Editor wird geladen durch den Aufruf

EDIT /wdir/mn:**dateiname**.typ<---

Die Angabe des Working Directory /wdir/ ist optional. Vorbesetzung ist das aktuelle wdir.
Die Angabe der Mediennummer 'mn:' ist optional. Das Dateisystem sucht alle Medien (Laufwerke) ab, ob eine Datei dieses Namens und Typs existiert. Wird keine gefunden, dann wird eine neue Datei dieses Namens auf dem aktuellen Mastermedium oder dem durch 'mn:' spezifizierten Medium angelegt.
Der Name 'dateiname' muß angegeben werden und er muß den KOS-Bedingungen entsprechen: ein Dateiname darf maximal 8 Zeichen lang sein und kann aus den Buchstaben A..Z, a..z und den Ziffern 0..9 bestehen.
Der Dateityp '.typ' darf maximal 3 Zeichen lang sein (A..Z, a..z, 0..9). Seine Angabe kann entfallen, dann erzeugt der Editor eine Datei des Typs '.XXX'.

Es wird für neue Dateien zunächst kein Disketten-Speicherraum zugewiesen. Die Zuweisung findet erst statt, wenn der editierte Text abgespeichert werden soll. Es ist auch möglich, während des Editiervorgangs Disketten zu wechseln, wenn z.B. das Abspeichern auf einer bereits vollen Diskette nicht möglich ist. Die Disketten müssen sich in ihrem Mediennamen unterscheiden, **sonst ist mit der logischen Zerstörung der Disketten zu rechnen.**

Die Dateilänge ist begrenzt durch den zur Verfügung stehenden Speicherraum im Hauptspeicher, also maximal 36.000 Zeichen bei Kontron PSI80/82-Systemen, bis zu 46.000 Zeichen bei Kontron PSI9xx-Systemen.

Alle vom Editor verwendeten Dateien enthalten nur druckbare Zeichen (Hex-Code <80). Steuerzeichen (Hex-Code <20) werden ausgeführt oder in inverser Darstellung protokolliert.

Beispiele zum Editor-Aufruf:

EDIT 1:TEXT.TEX<---
eröffnet die Datei mit Namen "TEXT" und Typ "TEX" auf Medium 1.

EDIT NAME<---
sucht die Datei mit Namen "NAME" und Typ "XXX" und eröffnet sie.

Wenn dies eine neue Datei ist, dann wird sie auf dem im Aufruf genannten Medium 'mn:' oder, wenn kein Medium angegeben wurde, auf dem Mastermedium eröffnet. EDIT stellt dann automatisch den Eingabezustand her, d.h.: die folgenden Eingaben werden als Text interpretiert.
Wenn diese Datei bereits existiert, dann wird sie wieder eröffnet. EDIT ist dann im Kommando-Modus, d.h.: die folgenden Eingaben werden als Kommandos interpretiert.

Hinweis: Zu bearbeitende Dateien dürfen nicht schreib- oder löschgeschützt sein. Sie dürfen kein Benutzerkennzeichen tragen (siehe DEFP-Kommando)

2. EDIT-Kommandos

Bei der Editierung werden maximal 24 Textzeilen dargestellt.

Die **25. Bildschirmzeile** bringt Statusanzeigen und Meldungen.

Im **Eingabezustand** (Statusanzeige "Eingabe") werden alle Eingaben als Text betrachtet. Im **Kommandozustand** (Statusanzeige "Kommando") werden alle Eingaben als Steuerkommandos interpretiert. Meldungen des Editors erfolgen im Klartext.

Die **Kommandos** werden im Kommando-Status durch Eingabe eines Buchstabens aufgerufen. Die Eingabe ist formatfrei, Leertasten oder Tabulatoren zwischen den Kommandobuchstaben und ihren Parametern sind nicht notwendig und beeinflussen die Kommandoausführung nicht.

Kommandos wirken auf die unterste (24.) Textzeile und eventuell auf die folgenden Textzeilen.

Parameter spezifizieren das Kommando genauer. Sie sind im allgemeinen optional. Innerhalb der Kommandos sind es entweder Zahlen- oder Zeichenfolgen. Zahlen werden dezimal aufgefaßt. Zeichenfolgen müssen mit einem beliebigen Begrenzer (="Delimiter") beginnen, der nicht innerhalb der Zeichenfolge auftauchen darf; lediglich Ziffern sind als Begrenzer nicht zulässig. In Kommandos, die mit externen Dateien kommunizieren, wird ein KOS-kompatibler Dateiname angegeben.

Fehlende Parameter werden durch den Wert "1" ersetzt. Der Stern "*" steht für "größtmögliche Zahl" oder "alle".

Die Ausführung von Kommandos wird durch Betätigen der Return-Taste veranlaßt.

Das "A"-Kommando wiederholt das zuletzt eingegebene Kommando.

In der cursororientierten Betriebsart sind keine Parameterangaben möglich, jedes Kommando ist ein einzelner Buchstabe. Die Kommandos werden sofort wirksam.

Auskunft über die einzelnen Kommandos während der Arbeit im Editor ist mit der Auskunftsfunktion

```
?<---
```

möglich; sie gibt die auf der folgenden Seite dargestellte Tabelle aus, ohne den Dateiinhalt zu beeinflussen.

Alle EDIT-Kommandos können gleichbedeutend in Groß- oder Kleinschreibung geschrieben eingegeben werden.

2.1 EDIT Kommando-Übersicht

Zeilenmodus:	Parameter für Zeilennummer/Anzahl	Zeichenfolge	Cursormodus:
E nter	E<---	E/text	E
C hange case	C<---	C<---	C
G o to	Gn<---	G/text<---	-
N ext line(s)	Nn<---	N/text<---	N
P rint line (s)	Pn<---	P/text<---	-
U p	Un<---	U/text<---	U
L ocate label	-	L/text<---	-
M odify	M/te1/te2/n m	M/te1/te2<---	-
R eplace	-	R/te2<---	R
DE lete	DEn<---	DE/text<---	D
A gain	An<---	A<---	-
S tore	Sn dat<---	S/text/datei	-
F etch	Fdat<---	-	-
T ext find	-	T/text	-
K os-Rücksprung mit Text speichern	KOS<---	KOS<---	-
K os-Rücksprung ohne Text speichern	KOS-<---	KOS-<---	-
EDITOR-Auskunft	?<---	?<---	-

Mit:

n	Dezimalzahl, Anzahl Zeilen bzw. Zeilennr.
m	Dezimalzahl, max. Anzahl der Änderungen in einer Zeile
<---	Symbol für "RETURN"
/	Begrenzerzeichen (keine Ziffer), darf nicht in text, te1 oder te2 vorkommen
-	Kombination nicht möglich
dat	Dateiname: /wdir/mn:name.typ
text	Zeichenfolge, Suchbegriff
te1	Zeichenfolge, zu ersetzender Text
te2	Zeichenfolge, neuer Text

2.2 Beschreibung der Kommandos

Es werden zunächst die Kommandos für die Zeilen-orientierte Betriebsart beschrieben, anschließend wird ihre Funktion im Cursor-Modus dargestellt.

Die Kommandos sind der leichteren Auffindbarkeit wegen nach Funktionsgruppen sortiert.

2.2.1 Texteingabe und Textlöschung

E - Eingabe **Eingabe von Textzeilen**

Formate:		
	E<---	Eingabezustand herstellen
	E -Alpha 123	Eingabe einer Zeile
	E,-extra-extra-extra<---	Eingabe einer Zeile, in der das Zeichen "-" vorkommt

Dieses Kommando bewirkt in seiner Grundform die Umschaltung vom Kommando-Status in den **Eingabe-Status:**

E<---

Alle folgenden Eingaben werden als Text betrachtet.

"RUBOUT" löscht die gerade erfolgende Eingabe. "BACKSPACE" (CTRL-H oder CURSOR LEFT) löscht das zuletzt eingegebene Zeichen. Mit "<---" (=RETURN) werden neue Zeilen abgeschlossen.

Die Eingabe von "RETURN" in eine leere Zeile schaltet zurück in den Kommando-Status.

Das **Einschieben einer einzigen Zeile** erfolgt durch das Kommando

E /Alpha 123<---

Dieses Kommando schiebt eine Zeile mit dem Inhalt "Alpha 123" nach der untersten sichtbaren Textzeile in den Text ein. Bei diesem Kommando verbleibt der EDIT im Kommando-Status. Das Zeichen "/" im Beispiel wirkt als Begrenzer. Als Begrenzer sind alle Zeichen außer den Ziffern zulässig, sofern die Zeichen nicht im nachfolgenden Text der aktuellen Zeile vorkommen.

Hinweise:
Auch in den folgenden Beispielen ist als Begrenzer das Zeichen "/" verwendet.
Die Leerzeichen zwischen Kommandobuchstabe (wie hier "E" für "Eingabe") und Parametern (wie hier "/Alpha 123") sind optional, können also im Interesse einer vereinfachten Bedienung weggelassen werden. In der Beschreibung dienen sie lediglich der Übersichtlichkeit.

In der **Cursor-Betriebsart** bewirkt das Kommando "E" das Umschalten auf Eingabe an der aktuellen Cursorposition, sofern es eine gültige Position ist. Ungültige Positionen sind die Kommandozeile, die Zone der Zeilennumerierung und Positionen, die durch Tabulationen besetzt sind.

"RETURN" und die Cursortasten beenden den Eingabezustand.

DElete **Löschen von Zeilen**

Formate:	DE n<---	löscht die nächsten n Zeilen einschließlich der aktuellen Textzeile
	DE *<---	löscht den Rest des Textes einschließlich der aktuellen Textzeile
	DE /abc<---	löscht alle Zeilen bis **einschließlich der** Zeile, in der die Zeichenfolge "abc" auftaucht und einschließlich der aktuellen Textzeile
	DE<---	löscht die unterste am Bildschirm sichtbare Textzeile

Das Kommando "DE" löscht eine Anzahl n von Zeilen inklusiv der aktuellen Zeile bis zum Ende des Textes oder bis zum Auftreten einer Zeichenfolge, die als Parameter angegeben wurde.

In der **Cursor-Betriebsart** löscht "D" das Zeichen an der Cursorposition.

Replace **Ersetzen einer Zeile**

Formate:	R /abc<---	ersetzt die unterste am Bildschirm sichtbare Textzeile durch eine Zeile des Inhalts "abc".
	R<---	löscht die aktuelle Zeile

Die letzte sichtbare Zeile des dargestellten Textes wird ersetzt durch die neu eingegebene Zeile.

In der **Cursor-Betriebsart** werden die nachfolgend eingegebenen Zeichen an die jeweilige Cursor-Position geschrieben, bis "RETURN" oder eine Cursor-Steuertaste bedient wird.

2.2.2 Positionieren und Drucken in der Textdatei

Up, Next, Print	Aufwärts/Abwärts verschieben, Drucken
Formate: U<---	Verschieben des Bildes eine Zeile aufwärts
N<---	Verschieben des Bildes eine Zeile abwärts
U n<---	Verschieben um n Zeilen aufwärts
N n<---	Verschieben um n Zeilen abwärts
P n<---	Drucken der nächsten n Zeilen
U /abc<---	Verschieben nach oben bis zum nächsten Auftreten der Zeichenfolge "abc"
N /abc<---	Verschieben bis zum nächsten Auftreten der Zeichenfolge "abc"
P /abc<---	Drucken der nächsten Zeilen bis "abc"

Diese Kommandos bewegen den dargestellten Teil der Textdatei vorwärts und rückwärts. Sie können benutzt werden mit Zahlen oder Zeichenfolgen als Parametern. Eine Zahl als Parameter bedeutet Vorwärts- oder Rückwärtsbewegen des sichtbaren Ausschnittes um 'n' Zeilen.

"*" gilt wieder als "um alle" Zeilen: das Bewegen an den Anfang oder ans Ende der Textdatei.

Mit einer Zeichenfolge als Parameter, abgetrennt vom Kommandobuchstaben durch einen beliebigen, nicht in der Zeichenfolge selbst enthaltenen Begrenzer (außer Ziffern), wird die Suche zum nächsten Erscheinen dieser Zeichenfolge durchgeführt.

Wird keine solche Zeichenfolge gefunden, so ändert sich das Bild nicht.

"U", "N" und "P" wirken von der aktuellen Position der Datei aus. Sie arbeiten relativ.

Im Interesse einer raschen Bedienung kann statt des Kommandos

N<---

einfach durch Betätigen der "CURSOR DOWN"-Taste auf die nächste Textzeile geschaltet werden.

Das P-Kommando setzt voraus, daß auf dem Ausgabekanal 0-4 ein Treiber aktiviert ist (KOS-Kommando "IODC $iod=ACTIVE 0-4=$iod").

Hinweis: Auf der KOS-Ebene erfolgt das Drucken einer Editor-Datei durch das Kommando "COPY dateiname $iod".

In der **Cursor-orientierten Betriebsart** arbeiten "U" und "N" ohne Parameter und bewegen den Bildausschnitt jeweils um eine Zeile vorwärts oder rückwärts.

Go **Gehe nach**

Formate:		
	G<---	Gehe zur ersten Textzeile
	G *<---	Gehe ans Textende
	G 0<---	Gehe an den Textanfang **vor** die erste Textzeile
	G n<---	Gehe nach Zeile 'n'
	G /abc<---	Gehe zu der Zeile, die als erste (vom Dateianfang aus gerechnet) die Zeichenfolge "abc" enthält.

Das G-Kommando arbeitet entsprechend dem "N"- oder "U"-Kommando, es wirkt jedoch immer vom Anfang der Datei an.

"G" ist nur in der zeilenorientierten Betriebsart zu verwenden.

Locate Label **Lokalisiere Marke**

Format: L /abc<---

Das L-Kommando benötigt eine Zeichenfolge als Parameter. Nach dieser Zeichenfolge wird die Datei durchsucht, von der aktuellen Zeile an bis zum Ende der Datei. Es werden jeweils die ersten Zeichen einer Zeile auf Gleichheit überprüft. Ein Anwendungsfall ist die Suche nach dem Auftreten einer Marke (Label), die gewöhnlich am Zeilenanfang steht. "L" ist nur in der zeilenorientierten Betriebsart zu verwenden.

Text **Text finden**

Format: T /abc<---

Das T-Kommando durchsucht den Text von der aktuellen Zeile an nach der durch Trennzeichen abgegrenzten Zeichenfolge. Bei Erfolg werden die 24 Zeilen einschließlich der Zielzeile angezeigt. Das T-Kommando bringt nur diese Zeilen zur Anzeige, im Gegensatz dazu läuft beim N-Kommando der gesamte Text durch. Deswegen läuft das T-Kommando schneller ab.

2.2.3 Textänderungen

Modify **Modifiziere**

Formate: M /textalt/textneu<---	tauscht die Zeichenfolge 'textalt' in die Zeichenfolge 'textneu' in der aktuellen Zeile um.
M /textalt//<---	löscht die Zeichenfolge 'textalt' in der aktuellen Zeile.
M /textalt/textneu/n m<---	ändert in den nächsten 'n' Zeilen die Zeichenfolge 'textalt' in die Zeichenfolge 'textneu' um, in den ersten 'm' Erscheinungen von 'textalt' innerhalb jeder Zeile.

In diesen Beispielen steht das Zeichen "/" wieder für ein beliebiges Begrenzungszeichen, das jedoch keine Ziffer sein darf, und nicht in textalt bzw. textneu auftreten darf.

"M" wirkt auf die unterste, am Bildschirm sichtbare Zeile und bei Angabe eines gültigen Zeilenzahl-Parameters 'n' auf diese Anzahl von Zeilen. Bei Angabe von "*" für 'n' wirkt sie auf alle folgenden Zeilen der gesamten Datei.

Die Änderung betrifft jeweils das erste Erscheinen der zu ändernden Zeichenfolge in einer Zeile.

Ein zweiter Parameter 'm' spezifiziert die maximale Anzahl von Änderungen pro Zeile. Auch hier gilt "*" wieder für 'alle'.

Am Bildschirm werden nur die geänderten Zeilen dargestellt. Die Rückkehr zur Darstellung des Textinhaltes an der dann aktuellen Position erfolgt nach Eingabe eines beliebigen Zeichens.

"M" ist nur in der zeilenorientierten Betriebsart zu verwenden.

2.2.4 Kommunikation mit externen Dateien

S-Kommando

Formate:

S<---	Speichern der derzeit untersten Textzeile auf eine Hilfsdatei
S n<---	Speichern der nächsten 'n' Zeilen inklusiv der untersten Zeile in eine Hilfsdatei
S *<---	Alle folgenden Zeilen in Hilfsdatei speichern
S /abc<---	Speichern aller Zeilen bis zum Erscheinen der Zeichenfolge 'abc' in eine Hilfsdatei
S n /wdir/mn:datei.typ	Speichern der nächsten 'n' Zeilen
S -abc- /wdir/mn:datei.typ	bzw. aller bis einschließlich der Zeile, in der "abc" auftritt, in eine neue Datei mit dem Namen 'datei.typ' auf Laufwerk 'mn'.

Wird die Typenangabe ".typ" weggelassen, setzt der EDIT ".XXX" als Typ ein bzw. nimmt diesen Typ an.

Die Angabe des Ziel-Working-Directories /wdir/ ist optional. Wird sie weggelassen, dann bleibt die neue Datei im aktuellen wdir.

Die Angabe des Mediums ist optional. Ohne Angabe wird die externe Datei auf das Medium abgelegt, das auch die zu editierende Datei enthält.

Wird beim S- oder beim F-Kommando eine Mediennummer spezifiziert, so erfolgen alle folgenden Dateizugriffe auf dieses Medium, bis erneut eine Mediennummer spezifiziert wird.
Anwendungsbeispiel: Anlegen von Hilfsdateien auf RAM-Floppy ($VMED).

Zur besonderen Beachtung:
Die Hilfsdatei, die das S-Kommando aufbaut, wenn kein Dateiname angegeben ist, **wird beim Verlassen des EDITORs gelöscht**, steht also für einen späteren Editiervorgang nicht zur Verfügung.

F-Kommando

Formate: F<---	Einfügen der Hilfsdatei im Anschluß an die unterste Textzeile
F /wdir/mn:dateiname.typ<---	Einfügen des Inhalts der Datei 'dateiname.typ' im Anschluß an die unterste Textzeile.

Bei Angabe eines externen Dateinamens können beliebige ASCII-Dateien eingebunden werden. Der Dateityp '.typ' ist optional, Vorbesetzung ist der Typ ".XXX".

Die Angabe des Mediums und des Quell-Working-Directories sind optional.

Weglassen des Dateinamens bewirkt ein Einfügen der **Hilfsdatei** an der aktuellen Position (funktional nur während der aktuellen Editier-Sitzung, siehe S-Kommando).

Der Inhalt der Hilfs-Datei kann beliebig oft eingebunden werden. Er bleibt erhalten, solange er nicht durch ein neues "S"-Kommando geändert wird und solange der Editiervorgang nicht beendet wird.

Wird beim S- oder beim F-Kommando eine Mediennummer spezifiziert, so erfolgen alle folgenden Dateizugriffe auf dieses Medium, bis erneut eine Mediennummer spezifiziert wird.
Anwendungsbeispiel: Anlegen von Hilfsdateien auf RAM-Floppy ($VMED).

2.2.5 Wiederholung, Groß-/Kleinschreibung, Hilfe

A-Kommando

Die Wiederholung des letzten Kommandos wird durch das "A"-Kommando ermöglicht. Als Parameter kann angegeben werden, wie oft das vorige Kommando wiederholt werden soll:

```
E -abc<---
A 25<---
```

bewirkt das 1+25=26malige Einschieben einer Zeile mit dem Inhalt "abc".

Das A-Kommando wirkt nur im Zeilenmodus.

C-Kommando

Das "C"-Kommando schaltet die Tastatur zwischen der Voreinstellung Großschreibung und Kleinschreibung um.

Vom Editoraufruf her ist die Großschreibung eingestellt. Das Verlassen des Editors schaltet die Anlage zurück in diese Einstellung.

Format: C<---

Das C-Kommando ist auch im Cursormodus wirksam.

?-Kommando, Hilfefunktion

EDIT beinhaltet eine Auskunftsfunktion, die eine Übersicht gibt über die verfügbaren EDIT-Kommandos. Sie wird mit

?<---

aufgerufen.

Die Rückkehr zum zeilenorientierten Kommando-Status erfolgt durch Betätigen der Leertaste oder irgendeiner anderen Taste. Das vorherige Bild wird wiederhergestellt.

Das ?-Kommando wirkt nur im Zeilenmodus.

2.2.6 Cursororientierte Betriebsart

Die Verwendung der cursororientierten Betriebsart empfiehlt sich immer dann, wenn in einem bestehenden Text einzelne (also nicht systematische) Änderungen vorzunehmen sind. In diesem Fall ist das cursorgestützte Verfahren das schnellste und sicherste.

In den übrigen Fällen (z.B. beim Einfügen/Löschen ganzer Zeilen, systematischen Änderungen durch den ganzen Text oder FD-unterstützter Dateimanipulation) ist die zeilenorientierte Betriebsart vorzuziehen.

An dieser Stelle werden die bereits im vorangegangenen Text an der jeweiligen Stelle angesprochenen Cursor-Funktionen und ihre Anwendung nochmals zusammengefaßt.

Es stehen auf der Kontron PSI-Tastatur 5 Cursor-Steuertasten zur Verfügung:

CURSOR LEFT	(Pfeil nach links) bewegt den Cursor nach links (auch zur Korrektur im Normalbetrieb verwendbar).
CURSOR RIGHT	(Pfeil nach rechts) bewegt den Cursor nach rechts (auch zum Übergang von zeilenorientierter zu cursororientierter Betriebsart zu verwenden).
CURSOR UP	(Pfeil nach oben) bewegt den Cursor nach oben (auch zum Übergang von zeilenorientierter zu cursororientierter Betriebsart zu verwenden).
CURSOR DOWN	(Pfeil nach unten) bewegt den Cursor nach unten (auch anstelle des Kommandos N<--- in der zeilenorientierten Editier-Betriebsart zu verwenden).
HOME	(Aufschrift "HOME") bewegt den Cursor zurück in seine Ausgangsposition (wird zum Übergang von cursororientierter zu zeilenorientierter Betriebsart verwendet).

Der Wechsel von zeilenorientierter in cursororientierte Betriebsart erfolgt durch Betätigen der "CURSOR UP" oder der "CURSOR RIGHT"-Taste.

Die cursororientierte Betriebsart wird durch ein #-Zeichen in der Kommandozeile auf dem Bildschirm symbolisiert.

Die Rückkehr zur zeilenorientierten Betriebsart erfolgt durch Eingabe von HOME im Kommandozustand oder durch positionieren des Cursors in seine Ausgangsposition mit Hilfe der Cursor-Steuertasten.

Folgende Funktionen stehen in der cursororientierten Betriebsart zur Textbearbeitung zur Verfügung:

D **Löschen** des vom Cursor markierten Zeichens

E Übergang vom cursororientierten Kommando-Status in den cursororientierten **Eingabe**-Status; es erscheint die Meldung:

CSR Eingabe

in der Kommandozeile des Bildschirms. Beliebige Texte ab der aktuellen Cursor-Position können eingefügt werden.
Bei Zeilenüberlauf erfolgt die Fehlermeldung

nicht möglich

und der Editor verzweigt automatisch zurück in den Kommando-Status.

Das Beenden der Eingabe kann erfolgen über das Betätigen einer Cursor-Taste oder der RETURN-Taste (Rückkehr zum cursororientierten Kommandostatus)

R Übergang vom cursororientierten Kommando-Status zum cursororientierten **Replace**-Status; es erscheint die Meldung:

CSR Korrektur

auf der Kommandozeile des Bildschirms. Jedes nun eingegebene Zeichen ersetzt das momentan vom Cursor markierte Zeichen. Ist eine Korrektur in der augenblicklichen Cursor-Position nicht zulässig, erscheint die Fehlermeldung

nicht möglich

Dabei erfolgt ein Rücksprung in den cursororientierten Kommando-Status.

Eine Beendigung des Korrekturvorgangs ist (wie aus dem Eingabe-Status heraus) möglich durch Betätigen einer Cursor-Taste oder der RETURN-Taste (Rückkehr zum cursororientierten Kommando-Status).

N verschiebt den dargestellten Text um 1 Zeile in Richtung wachsender Zeilennummern; die absolute Position des Cursors auf dem Bildschirm bleibt unverändert.

U verschiebt den dargestellten Text um 1 Zeile in Richtung fallender Zeilennummern; die absolute Position des Cursors auf dem Bildschirm bleibt unverändert.

2.2.7 Rückkehr zum Betriebssystem

KOS, KOS-

Die Rückkehr zum Betriebssystem mit Abspeichern der neu editierten Datei geschieht durch das Kommando "KOS".

Format: KOS<---

Das Abspeichern unterbleibt bei Verwendung des Kommandos "KOS-".

Format: KOS-<---

Anwendung:

Das Kommando "KOS<---" dient zum Abspeichern der editierten Datei oder zum Fortschreiben einer bestehenden Datei, die korrigiert worden ist. Zu beachten ist, daß die ursprüngliche Datei dabei **immer** überschrieben wird.

Soll dagegen eine **zusätzliche, neue Version** einer Datei abgespeichert werden, und die ursprüngliche Datei erhalten bleiben, benutzt man das S-Kommando und das KOS- Kommando.

Beispiel:

```
EDIT MYFILE.VE1<---
.
Editiervorgang
.
G 0<---              Damit geht man an den Textanfang.
S * MYFILE.VE2<---   Damit speichert man die neue
                     Textdatei MYFILE.VE2
KOS-<---             Damit verläßt man den Editor, ohne
                     die ursprüngliche Version
                     MYFILE.VE1 zu verändern.
```

Grundsätzlich ist beim Editieren von längeren Texten von Zeit zu Zeit ein Abspeichern durch Eingabe von KOS<--- dringend zu empfehlen: ohne Abspeichern ist Ihre Arbeit bei Stromausfall oder Fehlbedienung verloren. Zum Wiedereintritt in der Editor genügt die Eingabe von

```
X dateiname.typ<---
```

Das Kommando "KOS-" gestattet Ihnen, den Editor zu verlassen **ohne** die ursprüngliche Datei zu verändern.

3. Erweiterungen für besondere Anwendungsfälle

In diesem Teil werden Funktionserweiterungen des Texteditors beschrieben, die über die übliche Texterfassung und -Bearbeitung hinausgehen.

3.1 Einstellung des Bildschirmformates

Bei Lieferung ist der Editor auf das Bildschirmformat der Kontron PSI-Systeme eingestellt: das Programm setzt einen Bildschirm mit 25 Zeilen und 80 Spalten voraus.

Die Voreinstellung kann durch das Vx-Kommando geändert werden, um EDIT auch für Bildschirme anderer Einteilung verwendbar zu machen:

Formate: VD<--- Darstellung der Bildschirmparameter

VH n<--- Setzen der Spaltenzahl n, 29< n <256

VV n<--- Setzen der Zeilenzahl n, 1< n <256

VP<--- Schreiben der durch VH, VV geänderten Parameter in die Datei EDIT.COM

Diese Kommandos dienen der Anpassung des Editorprogrammes an abweichende Bildschirmformate.

Sie können auch verwendet werden, um auf Kontron PSI-Systemen andere Text-/Bildschirmformate einzustellen, sofern folgende Einschränkungen berücksichtigt werden:

a) Wenn die logischen und physikalischen Bildschirmformate nicht übereinstimmen, dann sind Cursor-orientierte Textbearbeitungen nicht definiert.

b) Wenn die logischen und physikalischen Bildschirmformate nicht übereinstimmen, dann können zeilen-orientierte Textbearbeitungen nur auf eigenes Risiko hin verwendet werden. Eine Gewährleistung für diese Betriebsart wird nicht gegeben.

c) Wenn mehr als 71 Zeichen/Zeile definiert werden, so kann nur noch im zeilen-orientierten Modus gearbeitet werden.

3.2 Steuer- und Semigrafikzeichen in Texten

EDIT kann zur Bearbeitung von Dateien verwendet werden, die von CP/M-Programmen erzeugt werden. Der Editor fügt an solche Dateien den Ende-Code 0FFh an.

In zeilenorientierter Betriebsart können alle Steuerzeichen außer CTRL-M (RETURN), CTRL-H (Cursor left) und RUBOUT als Parameter in Editor-Kommandos verwendet werden. Dadurch ist es z. B. möglich, alle ASCII-LF (CTRL-J) Zeichen durch das M-Kommando zu löschen, und das CP/M-e.o.f.-Zeichen CTRL-Z einzugeben. Dadurch können z.B. EDIT-Dateien in MBASIC-ASCII-Dateien transferiert werden und umgekehrt.

Semigrafikzeichen können von der Tastatur her oder von externen Dateien eingegeben werden. Diese Zeichen haben das höchstwertige Bit gesetzt.

KONTRON BASIC-Interpreter

Version: 4.33/5.44/5.54/6.05

Mai 1984

Dieser Teil des Handbuchs beschreibt die Kommandos und Anweisungen von BASIC. Es beschreibt die Eigenschaften und den Leistungsumfang dieser BASIC-Implementierung auf KONTRON PSI-Systemen. Anwendern, die mit der Computersprache BASIC nicht gründlich vertraut sind, werden geeignete Lehrbücher empfohlen.

Inhalt

Aufruf und Sprachumfang

BASIC ist ein leistungsfähiger BASIC-Interpreter. Dieses Programm ist auf der KOS-Systemdiskette gespeichert. Der Aufruf erfolgt von KOS aus durch die Eingabe von

```
BASIC<---
```

Sollen im gleichen Aufruf auch bereits Kommandos an den Interpreter mitgegeben werden, so sind diese in Anführungszeichen zu setzen. Das Zeichen "<" stellt dann den Abschluß eines Kommandos dar:

```
BASIC "LOAD DEMO<RUN"<---
```

Kontron BASIC Version 5.31/(4.2/5.2) bzw. 5.32/(6.0) enthält:

- das gesamte **Standard - BASIC**
- Befehle zum **Zugriff auf die Maschinenebene**
- sequentielle und blockorientierte **Bearbeitung von Dateien:**

  von einer geöffneten Datei kann gleichzeitig **Ein- und Ausgabe** sequentiell und blockorientiert realisiert werden
- Anschluß beliebiger Geräte durch **kanalorientierte Ein-/Ausgabe**
- komfortable **Vollgrafik** (256x512 bzw. 400x1024 bei Kontron PSI980 H)
- Bearbeitung von BASIC-Programmen Zusammenbau von Programmoduln
- **Editierfunktionen** im Interpreter
- **Aufruf** von KOS-Systemfunktionen

Ferner steht eine Datei BASKOS.COM zur Verfügung, der auf Adresse 2500H relokierte BASIC-Interpreter. Mit diesem können auch diskresidente KOS-Kommandos aufgerufen werden.

Begriffe

Einige in dieser Beschreibung verwendeten Begriffe sind im folgenden erklärt:

logische Einheit	Bezeichnung für eine außerhalb von BASIC liegende Datenquelle oder ein Datenziel, z.B. Dateien auf Diskette, Treiber
ASCII	Codierungsnorm für Zahlen, Buchstaben und Sonder- sowie Steuerzeichen (siehe Technische Beschreibung Anhang)
String	Folge von ASCII-Zeichen, z.B. ein Text
Treiber	Programm zur Ansteuerung eines Peripheriegerätes, z.B. Drucker, Terminal
Datei	Folge von Daten auf einem Medium (z.B. Diskette), die durch einen Namen ansprechbar ist
Dateizeiger	Bei der Bearbeitung von Dateien auf einem Medium ist dies innerhalb BASIC der Zeiger auf das nächste zu lesende bzw. zu schreibende Zeichen einer Datei
Mastermedium	Das Medium (z.B. Diskettenlaufwerk), auf das zugegriffen wird, wenn keine Mediennummer angegeben wird
Kommando-Modus	Betriebszustand von BASIC zum Eingeben von Kommandos und Anweisungen. Eine Programmzeile wird angenommen und sofort ausgeführt.
Programm-Modus	Ein BASIC-Programm wird ausgeführt. Dieser Modus kann mit der Taste Shift-Minus angehalten bzw. mit der ESC-Taste unterbrochen werden
Grafikbetriebsart	Der Bildschirm befindet sich in der grafischen Betriebsart, wenn der Grafik-Treiber $GRAP dem Kanal 9 zugewiesen und geöffnet (OPEN #9:"$GRAP") ist.

Datentypen

Variablen dürfen beliebig lange Namen aus Großbuchstaben und Ziffern tragen. Damit wird die Lesbarkeit von Programmen erhöht. Die ersten beiden Zeichen kennzeichnen die Variable. Achtung: BASIC-Schlüsselworte werden stets als solche erkannt, z.B. FORI$ ist kein erlaubter Name!

Es werden REAL-,
INTEGER- und
STRING-Variablen unterschieden.

Sie werden am Ende des Variablennamens wie folgt gekennzeichnet:

REAL:	kein zusätzliches Kennzeichen
INTEGER:	%
STRING:	$

Beispiele:	A, Z, AO, Z9	REAL-Variablen
	A%, Z%, AO%, Z9%	INTEGER-Variablen
	A$, Z$, AO$, Z9$	STRING-Variablen

INTEGER-Variablen belegen nur 2 Byte im Speicher und damit nur ein Viertel des Speicherplatzes für REAL-Variablen.

STRING-Variablen wird Speicherplatz dynamisch zugewiesen (keine DIM-Anweisung erforderlich!). Die Länge von Strings ist maximal 65535 Zeichen.

Wertbereiche der Variablen:

REAL:	1E-128..9.999999999999E+126 Zahlen in interner Gleitkommadarstellung, 13 signifikante Stellen der Mantisse, 8 Byte Speicherplatz.
INTEGER:	-32768... +32767 2 Byte Speicherplatz.
STRING:	Alle ASCII-Zeichen, pro Zeichen wird 1 Byte Speicherplatz benötigt. Dynamische Speicherplatzverwaltung.

Datentypen

---------- Fortsetzung

Konstanten:

Ganzzahl-, Festpunkt-, Exponential- oder Stringformat

Stringformat: Strings werden in Anführungszeichen (") eingeschlossen. Für Strings sind alle ASCII-Zeichen zulässig.

Beispiele:
```
1234
3.14159
.241828
-2.09E-39
"Heute scheint die Sonne"
```

Felder:

Es können Felder jeder Variablenart (INTEGER, REAL, STRING) vereinbart werden (siehe DIM-Anweisung).

Anzahl der Dimensionen: 1..255
Indexlaufbereich : 0..65535

Bei STRING-Feldern wird für jeden String pro Index Speicherplatz dynamisch zugewiesen.

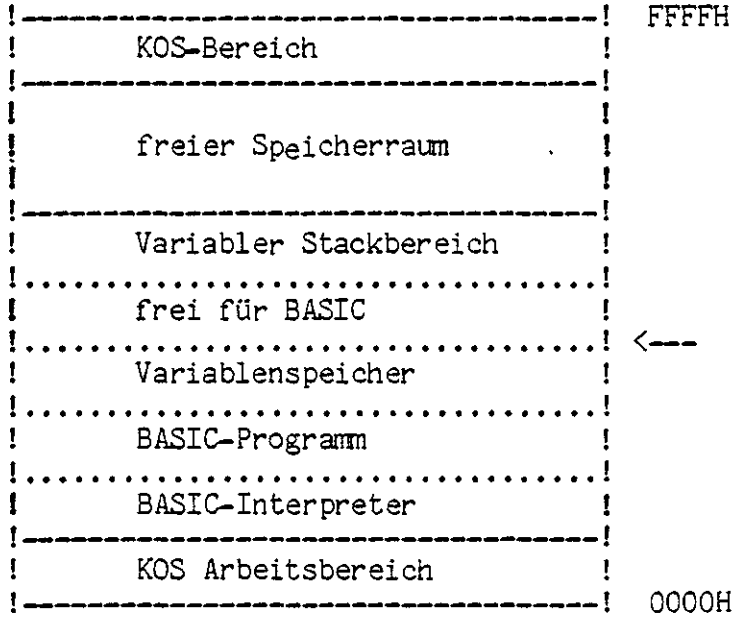
Beispiele:
```
A(N,M)
B9%(2, 50, M, 150)
Namen$ (1000)
```

Datentypen

---------- Fortsetzung

Dynamische Speicherverwaltung für STRING-Variablen

Normalerweise liegen die zu einer STRING-Variablen gehörenden Zeichenketten innerhalb eines lückenlosen Bereichs oberhalb des BASIC-Programms im Anwenderspeicher. Das obere Ende dieses Variablenspeichers markiert ein Zeiger:



Wird einer STRING-Variablen eine Zeichenkette zugewiesen, die länger ist als die alte, so wird zunächst geprüft, ob die alte Zeichenkette an der Obergrenze des Variablenspeichers liegt. Ist dies der Fall, wird die alte Zeichenkette mit der neuen überschrieben und der Zeiger erhöht. Befindet sich die alte Zeichenkette nicht an der Obergrenze, so wird die neue Zeichenkette nach dem Zeiger abgespeichert und der Zeiger auf das Ende der neuen Zeichenkette gesetzt. Der Stringvariablen wird der Speicherbereich der neuen Zeichenkette zugeordnet. Im anfangs lückenlosen Speicherbereich ist an der Stelle der alten Zeichenkette eine Lücke entstanden, die von keiner Variablen mehr benützt werden kann.

Muß der Anwenderspeicher möglichst gut ausgenützt werden (speicherintensive Programme), empfiehlt es sich, der STRING-Variablen zu Beginn des Programms eine genügend lange Zeichenkette zuzuweisen, so daß keine noch längeren Zeichenketten verarbeitet werden müssen. Dies gilt sinngemäß auch für die Elemente von STRING-Feldern.

PROGRAMMERSTELLUNG

- BASIC Aufruf

  Für BASIC: BASIC<---

- Eingabekorrektur
  Die Editor-Kommandos sind beim FETCH-Kommando beschrieben

Cursor an Zeilenanfang setzen	:	HOME
Cursor links und rechts verschieben	:	CTRL-H bzw. CTRL-F
Zeichen einfügen	:	CTRL-A
Zeichen löschen	:	CTRL-D
alle Zeichen vor dem Cursor löschen	:	RUBOUT

- Zeilennummern

  von 1 bis 65534
  die Zeilen werden unabhängig von der Reihenfolge der Eingabe in aufsteigender Zeilennummernfolge sortiert und ausgeführt.

- Mehrfachzeilen

  in einer Zeile können mehrere Befehle untergebracht werden.
  Trennzeichen: ":" Ausgenommen hiervon sind Kommandos.

- Leerzeichen

  BASIC-Programme benötigen keine Leerzeichen.
  In den Beispielen sind jedoch zur besseren Lesbarkeit Leerzeichen eingefügt. Auch beim Auflisten (LIST, PLIST) werden sie automatisch ausgegeben.

- Speicherbelegung

  BASIC belegt den Speicherbereich bis 41FF.
  Der Speicherbereich für das Programm beginnt bei 4200.
  Die obere Grenze des Programmspeichers wird durch die Speicherverwaltung des Betriebssystems bestimmt.
  (Siehe KOS MAP-Kommando)

- Kommando- und Programm-Modus (Benutzung als Tischrechner)

  Wenn kein BASIC-Programm abläuft, so befindet sich BASIC im Kommando-Modus. Alle Befehle (Kommandos, Anweisungen) können in diesem Modus ohne Zeilennummer eingegeben werden und werden dann sofort ausgeführt. Wenn die Kommandozeile fertig abgearbeitet ist, werden alle geöffneten Dateien und die Grafikausgabe automatisch geschlossen.

Programmtest

Zum Test von BASIC-Programmen stehen folgende Hilfsmittel zur Verfügung:

- Einfügen von Kontrollausdrucken (siehe PRINT-Befehl)

  In das Programm können zusätzliche PRINT-Befehle eingefügt werden, um die Werte von Variablen zu kontrollieren.

- Setzen von Unterbrechungspunkten (siehe STOP-Anweisung)

  An beliebigen Stellen im Programm kann eine STOP-Anweisung eingebaut werden. Die Ausführung des Programms wird an dieser Stelle unterbrochen. Alle geöffneten Dateien werden jedoch geschlossen und der Kommando-Modus hergestellt.

- Ausdrucken und Ändern von Variablen

  Bei mit der STOP-Anweisung unterbrochenen Programmen (BASIC ist dann im Kommando-Modus) können mit PRINT und LET Variablen ausgedruckt und geändert werden.

- Fortsetzen von Programmen nach Unterbrechungen

  Das Programm kann mit dem CONT-Kommando fortgesetzt werden, die Variablen werden nicht rückgesetzt. Einschränkungen für Datei-Ein-/Ausgabe und Grafik-Modus beachten.

- Programmablauf-Überwachung

  Mit der TRACE-Anweisung wird die Zeilennummer der gerade bearbeiteten Zeile ausgegeben.

Kommandos

Kommandos sind Anweisungen an den BASIC-Interpreter, die zur Handhabung von BASIC-Programmen dienen. Sie sind ausschließlich im Kommando-Modus zu verwenden.

In BASIC gibt es folgende Kommandos:

Kommando	Bedeutung
LOAD	Laden eines gepackten BASIC-Programms
SAVE	Abspeichern eines gepackten BASIC-Programms
ALOAD	(Dazu-)laden eines BASIC-Programms im ASCII-Code
ASAVE	Abspeichern eines BASIC-Programms im ASCII-Code
LIST, LI	Ausgeben eines BASIC-Programms auf Sichtschirm
PLIST	Ausgeben eines BASIC-Programms auf Drucker (KOS-Kanal 0-4)
NEW	Löschen des im Speicher befindlichen BASIC-Programmes
RUN	Starten eines BASIC-Programmes
CONT	Fortsetzen eines BASIC-Programmes
AUTO	Automatische Zeilennumerierung bei der Programmeingabe
RENUMBER, RNB	Umnumerieren des (Teil-) Programms
DELETE, DE	Löschen des angegebenen Programmteils
FETCH, FE	Bearbeiten (korrigieren) einer Programmzeile

Anweisungen

Ein BASIC-Programm ist aus Anweisungen aufgebaut. Sie stehen in Programmzeilen, die als solche durch eine vorangestellte Zeilennummer gekennzeichnet sind. Sie werden nicht sofort ausgeführt, sondern als Teil eines Programmes gespeichert, das mit RUN gestartet werden kann.

Alle Anweisungen und Programmzeilen können im Kommando-Modus ebenfalls benutzt werden. In diesem Fall wird keine Zeilennummer vorangestellt. Die Kommandozeile wird sofort nach Abschluß mit "RETURN" ausgeführt. Danach werden alle geöffneten Dateien geschlossen bzw. vom Grafik- in den Alpha-Modus zurückgeschaltet (siehe Kommando-Modus, Programmtest, STOP).

In BASIC gibt es folgende Anweisungen:

Ein-/Ausgabe

INPUT	Eingabe von der Tastatur
INPUT#	Eingabe von logischer Einheit
GET	Eingabe eines Zeichens von der Tastatur
PRINT	Ausgabe auf Sichtschirm
PRINT#	Ausgabe auf logischer Einheit
PRINT USING	Formatierte Ausgabe
READ	Eingabe von Daten aus DATA-Anweisung
DATA	Datenspeicherung im Programm
RESTORE	Rücksetzen des DATA-Zeigers
LINELEN	Zeilenlänge verwalten
OPEN	Öffnen einer logischen Einheit (Dateien: 10..13)
CLOSE	Schließen einer logischen Einheit
DEL	Löschen einer logischen Einheit (Datei)
RECORD	Blockweiser Zugriff auf eine Datei (eine Diskette etc.)

Bedingungen

IF..THEN..ELSE	Bedingte Ausführung von Anweisungen

Anweisungen

---------- Fortsetzung

Sprünge, Schleifen

GOTO	Sprung zu einer Programmzeile
GOSUB	Aufrufen eines Unterprogramms
RETURN	Rückkehr von einem Unterprogramm
FOR..TO..STEP	Programmschleife
NEXT	Abschluß einer Programmschleife
ON	Bedingter Sprung

Maschinennahe Programmierung

CALL	Aufruf von Maschinencode-Unterprogrammen
POKE	Schreiben in Speicherzellen
OUT	Ausgabe auf Ports
ON INTR	Interrupthandhabung in BASIC

Grafik

SET	Setzen von Bildpunkten
RES	Löschen von Bildpunkten
INV	Invertieren von Bildpunkten
PLOT	Plotten von Linien
STRING	Schriftzeichen ausgeben (nur im Grafik-Modus)
CLEAR	Bildschirm löschen; Grafik-Modus bleibt

Sonstige Befehle

RANDOM	Zufallsfunktion initialisieren
LET	Zuweisung von Werten zu Variablen
STOP	Unterbrechung im Programmlauf
END	Ende eines Programms
REM, !	Kommentare
DIM	Dimensionieren von Feldern
TRACE	Programmablauf-Überwachung ein-/ausschalten
TRACEOFF	
KOS	Rückkehr ins Betriebssystem
KOS	Aufruf des KOS-Kommando-Interpreters

Funktionen

Funktionen und Operatoren treten in Anweisungen auf und bewirken eine Wertzuweisung, Wertbearbeitung oder eine Wertumwandlung.

Arithmetische Operatoren:

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
^	Potenzierung
()	Klammerung (bis 8-fach)

Vergleichsoperatoren:

=	gleich
<>	ungleich
<=	kleiner oder gleich (auch =<)
>=	größer oder gleich (auch =>)
>	größer
<	kleiner

Logische Operatoren:

NOT	Negation
AND	Und-Verknüpfung
OR	Oder-Verknüpfung
XOR	Exklusiv-Oder-Verknüpfung

Funktions-Operatoren

EXP(X)	ERM(N)	RND(N)
LOG(X)	FRE(N)	PEEK (adr)
SIN(X)	SQR(X)	IN (adr)
COS(X)	ABS(X)	POINT(x,y)
TAN(X)	INT(X)	
ATN(X)	SGN(X)	

String-Operatoren

CHR$(X)	+	STR$(X)
ASC(N$)	LEFT$ (N$,X)	LEN(N$)
VAL(N$)	RIGHT$(N$,X)	MID$(N$,X1,X2)

Übersicht: KOMMANDOS zur Bedienung des Interpreters

Die folgenden Kommandos zur Bedienung des BASIC-Interpreters gestatten das Editieren, Abspeichern, Laden und Ausführen von BASIC-Programmen.

Die Kommandos sind alphabetisch aufgelistet.

Querverweise finden Sie im Register im Anhang.

Kommandos können nur im Kommando-Modus und nur ohne Zeilennummern eingegeben werden.

Anweisungen dagegen sind sowohl innerhalb eines Programmes (mit Zeilennummern) als auch wie Kommandos (ohne Zeilennummern) verwendbar.

Beispiel:

als Anweisung in einem BASIC-Programm:

30 KOS M	bringt das Systemkommando "M" zur Ausführung, anschließend wird das BASIC-Programm fortgesetzt
40 KOS	beendet das BASIC-Programm und führt ins Betriebssystem zurück

als Kommando für den BASIC-Interpreter:

KOS M	zeigt das Mastermedium an und führt in den Kommando-Modus zurück
KOS	führt ins Betriebssystem zurück

ALOAD, ASAVE

Syntax:

ALOAD mn:name
ASAVE mn:name

Wirkung:

Wie SAVE und LOAD; jedoch wird das Programm im ASCII-Code (nicht gepackt) abgelegt. Diese Dateien können mit dem EDITOR bearbeitet werden (Zeilenlänge nicht über 71 Zeichen!). Bei ALOAD hat die Folge der Zeichen der Eingabedatei dieselbe Wirkung wie ein über die Tastatur eingegebenes Programm. Bei fehlerhaften Zeilen erscheint eine Fehlermeldung.

Beim Laden eines Programms mit ALOAD wird das im Speicher befindliche Programm nicht gelöscht, jedoch werden gleiche Zeilennummern überschrieben. Somit ist es möglich, Programmoduln zusammenzusetzen.

Voreinstellungen:

mn : Mastermedium (Masterlaufwerk)
typ: BSC (darf nicht angegeben werden)

Beispiele:

ALOAD TEST

Das Programm TEST.BSC wird vom Mastermedium geladen. Ist es dort nicht vorhanden, so sucht KOS auf allen aktiven Medien.

ASAVE TESTO

Speichert das im Speicher befindliche Programm unter dem Namen TESTO.BSC auf dem Mastermedium ab. Ist ein Programm mit diesem Namen dort nicht vorhanden, so sucht KOS auf allen anderen aktiven Medien nach einer Datei dieses Namens. Wird sie auf einem Medium gefunden, so wird das Programm dort abgespeichert.

Hinweis: Der Dateityp '.BSC' ist die Voreinstellung für mit dem ASAVE-Kommando abgespeicherte Programmtexte. Es ist i.a. nicht zweckmäßig, diese Bezeichnung für andere Dateien mit zu verwenden.

AUTO

Syntax:

AUTO S=schrittweite B=zeilennr

Wirkung:

Es wird für die Programmerstellung nach jeder abgeschlossenen Zeileneingabe automatisch die nächsthöhere Zeilennummer erzeugt. Es kann eine Anfangszeilennummer (zeilennr) und/oder eine Schrittweite vorgegeben werden.
Die Reihenfolge von B= und S= ist beliebig.

Wird ein Parameter nicht angegeben, so gilt dessen **Voreinstellung:**

B = 10
S = 10

Beispiele:

AUTO

Es wird beginnend mit 10 nach jeder neuen Zeile eine um 10 erhöhte Zeilennummer erzeugt.

AUTO B=100 S=5

Als Zeilennummern erscheinen 100, 105, 110, 115 usw.

Hinweis: Der Programmerstellungsmodus 'AUTO' wird durch Eingabe von 'ESC' (ESCAPE) wieder verlassen.

CONT

Syntax:

CONT zeilennr

Wirkung:

Setzt das Programm ab der Zeile 'zeilennr' fort. Die Variablen werden **nicht** gelöscht. Da alle Dateien geschlossen wurden, kann die Datei-Ein-/Ausgabe mit PRINT#, INPUT# nicht fortgesetzt werden. Ähnliches gilt für die Grafikbetriebsart des Bildschirms.

Nach der Meldung "Stop in Zeile ...." anwendbar.

Beispiele:

CONT 2000

Setzt das Programm ab der Zeile 2000 fort. Die Werte der Variablen werden übernommen.

DELETE

Syntax:

```
DELETE  zeilennr1
DELETE  zeilennr1-
DELETE  -zeilennr2
DELETE  zeilennr1-zeilennr2

oder
DE      ...
```

Wirkung:

Der angegebene Programmteil wird gelöscht. Es ist möglich, nur eine Zeile (zeilennr1), alles ab der Anfangszeile (zeilennr1-), alles bis zur Endzeile (-zeilennr2) oder alles zwischen Anfangs- und Endzeile zu löschen.

Wird keine Zeilennr angegeben, wird alles gelöscht.

Genauso kann auch die Kurzform DE verwendet werden.

Beispiele:

DE	Das ganze Programm wird gelöscht
DE 50	Die Zeile 50 wird gelöscht
DELETE 100-	Der Programmteil ab Zeile 100 bis zum Ende wird gelöscht
DELETE -200	Vom Anfang des Programms bis einschließlich der Zeile 200 wird alles gelöscht.
DELETE 250-400	Alle Zeilen mit den Nummern zwischen 250 und 400 (einschließlich) werden gelöscht.

FETCH

Syntax:

FETCH zeilennr
FE zeilennr

Wirkung:

Die Zeile mit der Nummer 'zeilennr' wird in den Eingabepuffer gebracht und auf den Bildschirm ausgegeben, damit sie mit den anschließend beschriebenen Editorkommandos verändert werden kann. Der Cursor steht zunächst am Ende der Zeile. Soll die Korrektur beendet werden, muß die RETURN-Taste betätigt werden. Das ist in jeder Cursorposition möglich. Wird ein Syntax-Fehler von BASIC erkannt, bleibt die alte Zeile unkorrigiert. Wird die Zeilennummer geändert, entsteht eine Kopie der Zeile, wird sie gelöscht, führt BASIC die Befehle der Zeile sofort aus.

BASIC verwaltet die Zeilenlänge. Siehe hierzu die LINELEN-Anweisung. Reicht die vorgegebene Zeilenlänge für die Eingabe nicht aus, versucht BASIC die zu editierende Zeile auszugeben ohne jede Leerzeichen und unter Verwendung aller Abkürzungen (für PRINT ein '?'). Gelingt das nicht, so erscheint eine Fehlermeldung.

Hinweis: Beim Editieren werden Control-Zeichen reflektiert, jedoch nicht ausgeführt.

Programmzeilen mit Zeilennummern, die der Formel n*256-1, n=1,2,... genügen, können erzeugt, jedoch nicht mit 'FETCH' editiert werden.

Das Editieren einer Programmzeile, welche die Anweisung 'KOS <Kommando(s)>' enthält, mit 'FETCH', verlängert diese Zeile mit einem 0FFh, dieses Zeichen kann mit CTRL-D wieder gelöscht werden.

FETCH - EDITOR-Kommandos

---------- Fortsetzung

In jeder Position des Cursors kann das dort stehende Zeichen überschrieben werden. Der Cursor kann mit den folgenden Kommandos bewegt werden:

CTRL-H
CURSOR-LINKS — bewegt den Cursor jeweils einen Schritt rückwärts, ohne ein Zeichen zu verändern.

CTRL-F
CURSOR-RECHTS — Der Cursor wird vorwärts bewegt, ohne ein Zeichen zu verändern.

HOME — Der Cursor wird an den Anfang der zu editierenden Zeile gesetzt.

Die folgenden Kommandos verändern die Zeilenlänge:

CTRL-A — ("Add") Um ein Zeichen an der jeweiligen Schreibstelle des Cursors einfügen zu können, wird zunächst die gesamte Restzeile hinter dem Cursor um eine Stelle nach rechts verschoben.
An der Cursor-Position steht zunächst ein Leerzeichen, das anschließend beliebig überschrieben werden kann.

CTRL-D — ("Delete") Das Zeichen an der jeweiligen Schreibstelle wird gelöscht. Anschließend wird die rechts davon befindliche Restzeile um eine Stelle nach links verschoben.

RUBOUT — Der links vom Cursor stehende Teil der Zeile wird gelöscht. Die Restzeile wird nach links zum Beginn der Zeile verschoben, der Cursor steht bei dem ersten Zeichen.

RETURN — beendet das Eingeben und Editieren einer Zeile. Das ist in jeder beliebigen Cursorposition möglich, die gesamte sichtbare Zeile wird von BASIC bearbeitet. Nach Beenden des Eingebens einer Zeile wird die Syntax geprüft. Steht am Beginn der Zeile eine Zahl, wird sie als Zeilennummer aufgefaßt, die Zeile wird gespeichert. Fehlt die Zeilennummer, werden die Befehle der Zeile sofort ausgeführt (Kommando-Modus).

LIST, PLIST

Syntax:

```
LIST

LIST zeilennr1-
LIST -zeilennr2
LIST zeilennr1-zeilennr2

oder
LI   ...
```

Wirkung:

LIST schreibt im Speicher befindliche Programm auf den Sichtschirm, PLIST gibt das Programm-Listing über den KOS-Kanal 0-4 aus, dem ein Druckertreiber zugeordnet werden kann.

BASIC legt das Programm in einer vorübersetzten Form im Speicher ab. Bei der Ausführung des LIST- bzw. PLIST-Kommandos wird das Programm wieder in den Klartext übersetzt.

Es kann eine Anfangs- (zeilennr1-) und/oder eine Endzeile (-zeilennr2) des Listings angegeben werden.

Anhalten des Ausdrucks durch "_" (Shift-Minus).
Abbrechen der Ausgabe mit ESC-Taste.

Die Kurzform des Kommandos LIST ist LI und kann genauso verwendet werden.

Beispiele:

LIST	Listet das gesamte Programm
LIST 100	Listet nur die Zeile 100
LIST 100-	Listet das Programm ab Zeile 100
LIST -500	Listet das Programm bis Zeile 500
LIST 100-500	Listet das Programm von Zeile 100 bis Zeile 500

LOAD, SAVE

Syntax:

LOAD dateiadresse
SAVE dateiadresse

Wirkung:

- LOAD lädt das Programm mit der Dateiadresse 'dateiadresse' von einem Medium (z.B. Diskette) in den Speicher. Ein vorher geladenes Programm wird gelöscht.

- SAVE speichert das im Speicher befindliche Programm unter der Dateiadresse 'dateiadresse' auf einem Medium ab. Eine bereits existierende Datei dieses Namens wird überschrieben.

  Die Dateiadresse entspricht den KOS-Konventionen:

  (mn:)dateiname.BAS

  Voreinstellungen:

  mn : Mastermedium (Masterlaufwerk)
  typ: BAS (darf beim Aufruf nicht angegeben werden)

Wenn eine Medien- (Laufwerk-) Nummer angegeben ist, wird die Datei nur auf diesem Medium gesucht.

Wird keine Medien-Nummer vorgegeben, wird die Datei zunächst auf dem Mastermedium gesucht und dann, wenn sie dort nicht gefunden wird, auf allen aktiven Medien. Bleibt das Suchen erfolglos, wird beim Kommando LOAD eine Fehlermeldung ausgegeben bzw. bei SAVE die Datei auf das Mastermedium geschrieben.

Das gepackte BASIC-Programm enthält eine Information über die Startadresse des Anwender-Speicherbereichs. Es ist sicherzustellen, daß die mit LOAD und SAVE transferierten Programme sich auf die richtige Anfangsadresse beziehen. Notfalls kann dies mit der Kommandofolge

```
LOAD  dateiadresse
ASAVE dateiadresse
ALOAD dateiadresse
SAVE  dateiadresse
```

realisiert werden.

LOAD, SAVE

---------- Fortsetzung

Beispiele:

LOAD TEST

Das Programm TEST.BAS wird von einem Medium geladen. Die Datei wird auf allen aktiven Medien, beginnend mit dem Mastermedium, gesucht.

SAVE TESTO

Speichert das im Speicher befindliche Programm unter dem Namen TESTO.BAS auf das Mastermedium.

Ist ein Programm mit diesem Namen dort nicht vorhanden, so sucht KOS auf den aktiven Medien nach einer Datei dieses Namens. Ist sie vorhanden, so wird das Programm dort abgespeichert, ansonsten auf dem Mastermedium.

SAVE 1:TEST1

Das im Speicher befindliche Programm wird unter dem Namen TEST1.BAS auf dem Medium 1 gespeichert.

Hinweis: Der Dateityp '.BAS' ist die Voreinstellung für gepackt durch das SAVE-Kommando des BASIC-Interpreters abgespeicherte Programmtexte. Diese Typenbezeichnung sollte nicht für andere Dateiarten verwendet werden. LOAD und ALOAD sind nur auf jeweils passend (mit SAVE bzw. ASAVE) abgespeicherte Programmtexte anzuwenden.

NEW

Syntax:

NEW

Wirkung:

Neuinitialisierung von BASIC.

Das Programm und alle Variablen werden gelöscht.

Beispiel:

```
NEW
```

Hinweis: Mit dem ALOAD-Kommando können Programmteile zusammengesetzt werden. Wenn dies nicht erwünscht ist, kann mittels 'NEW' oder 'DE' ein bestehendes Programm gelöscht werden.

RENUMBER

Syntax:

```
RENUMBER
RENUMBER zeilennr1-              S=schrittweite    B=zeilennr3
RENUMBER -zeilennr2              S=schrittweite    B=zeilennr3
RENUMBER zeilennr1-zeilennr2     S=schrittweite    B=zeilennr3

oder
RNB      ...
```

Wirkung:

Der mit 'zeilennr1' und/oder 'zeilennr2' angegebene Teil des Programms erhält neue Zeilennummern. Es kann auch eine neue Schrittweite und/oder eine neue Anfangszeile (zeilennr3) angegeben werden. Die Reihenfolge der Parameter B und S ist beliebig. Die Reihenfolge der Programmzeilen wird nicht verändert.

Alle Verzweigungsadressen (nach GOTO, GOSUB, THEN, ON) werden automatisch korrigiert.

Statt RENUMBER kann auch genauso die Kurzform RNB verwendet werden.

Beispiele:

RENUMBER	Das gesamte Programm erhält neue Zeilennummern. Beginnend mit Zeile 10 wird eine Schrittweite von 10 benutzt.
RENUMBER 500- B=1000	Der Teil des Programms ab Zeile 500 bis zum Ende wird umnumeriert, beginnend mit der Zeilenzahl 1000 und Schrittweite 10.
RNB 200-400 S=20 B=500	Das Programm zwischen Zeile 200 und 400 (einschließlich) erhält neue Zeilenzahlen, beginnend mit 500 und Schrittweite 20.

RUN

Syntax:

RUN

RUN zeilennr

Wirkung:

Startet das Programm ab der niedrigsten Zeilennummer, bzw., wenn angegeben, ab 'zeilennr'.

Die Variablen werden vorher gelöscht.

Der Programmablauf kann unterbrochen werden durch:

- Fehler (-> Fehlermeldung)
- ESC-Taste (-> "Stop in Zeile ....")
- STOP-Befehl im Programm (-> "Stop in Zeile ....")

Nach der vollständigen Ausführung eines Programms meldet sich BASIC mit

```
Ready
:
```

Beispiele:

RUN

Löscht alle Variablen und startet das Programm ab der niedrigsten Zeilennummer.

RUN 100

Löscht alle Variablen und startet das Programm mit Zeile 100.

Übersicht

Aus den im folgenden beschriebenen Anweisungen (Statements) können BASIC-Programme aufgebaut werden.

BASIC-Programme bestehen aus Programmzeilen, die jeweils mit einer Zeilennummer beginnen. Die Programmzeilen werden in aufsteigender Reihenfolge ausgeführt. Jede Zeilennummer existiert nur einmal in einem Programm. Lücken in der Zeilennumerierung sind ohne Bedeutung für den Programmablauf.

Die einfache Programmzeile besteht aus Zeilennummer und einer Anweisung. Es können mehrere Anweisungen in einer Programmzeile stehen. Die Anweisungen sind dann durch Doppelpunkt ':' zu trennen.

Zur Verdeutlichung werden jeweils Beispiele angegeben, siehe auch die Beispielprogramme im Anhang.

Zum leichteren Auffinden sind die Anweisungen alphabetisch sortiert; zudem ist der Einsatzbereich im Kopf jeder Seite angegeben.

Querverweise finden Sie im Register im Anhang.

CALL

Syntax:

CALL adr
CALL adr,p1,p2,....,pn

Wirkung:

Ruft ein Unterprogramm in Maschinensprache auf der Speicheradresse 'adr' (dezimal) auf.

'adr' kann eine numerische Konstante, Variable oder Ausdruck sein.

0 <= adr <= 65535

Die Rückkehr von einem Unterprogramm erfolgt durch den Assembler-Befehl

RET.

Parameter von BASIC zum Unterprogramm (p1..pn) sind durch ein vorangestelltes Komma gekennzeichnet.

p kann eine Real-, Integer- oder String-Variable, bzw. auch ein Element eines Feldes sein.

Übergeben werden 16-bit-Pointer, die auf die Speicherplätze der Übergabeparameter zeigen. Sie werden in der angegebenen Reihenfolge p1, p2, .., pn in den Stack geladen (PUSH). Die Unterroutine kann somit über den Stack die benötigten Parameter laden bzw. zurückgeben.

Die Pointer zeigen jeweils auf das erste Byte (A) der Variablen. Je nach Variablentyp sind die weiteren Bytes wie folgt belegt:

Integer (%)	A	LSB-byte
	A+1	MSB-byte
Real	A	Vorz. und MSB-Zahl
	A+1	NSB-Zahl
	.	
	.	
	A+6	LSB-Zahl
	A+7	Exponent mit Vorzeichen
String ($)	A	LSB-byte der aktuellen Länge
	A+1	MSB-byte der aktuellen Länge
	A+2	LSB-byte der Anfangs-Adresse
	A+3	MSB-byte der Anfangs-Adresse

CALL

---------- Fortsetzung

Erzeugt das Unterprogramm eine Zeichenkette, so darf sie nicht länger als die von BASIC übergebene Länge werden. Bei Bedarf kann mit LET B$= "....." vorher eine entsprechend lange Platzhalter-Zeichenkette erzeugt werden.

Beispiele:

```
500 CALL 32768
```

Führt einen Sprung auf die Adresse 32768 dezimal (8000 Hex) aus.

```
600 CALL 49152,P1,P2
```

Führt einen Sprung auf die Adresse 49152 dezimal (C000 Hex) aus. Dabei werden die Pointer (Zeiger) P1 und P2 an das Unterprogramm übergeben.

Das Assembler-Unterprogramm hat die folgende Form:

```
LC000:   POP     BC      ;Ret-Adresse retten
         POP     HL      ;Pointer auf P2
         POP     DE      ;Pointer auf P1
         PUSH    BC      ;Ret-Adresse
         .
         .
         .
         RET             ;ZURÜCK ZU BASIC
         END
```

Hinweis:

Durch die Funktionsweise des Stack ist die Reihenfolge der Parameter im Assembler-Unterprogramm umgekehrt!
Bei "CALL 0" wird die Warmstart-Routine des Betriebssystems ausgeführt. Dadurch wird zwar der BASIC-Interpreter verlassen und sein Speicherbereich freigegeben, die Anwenderbereiche bleiben jedoch allokiert!

CALL

---------- Fortsetzung

Beispiel:

Assembler-Unterprogramm, gelinkt auf die Adresse 7000h (28 672 dezimal):

```
A7000H:  POP BC
         POP HL
         POP DE
         PUSH BC
         LD A,(DE)
         ADD A,01
         LD (DE),A
         INC DE
         LD A,(DE)
         ADD A,02
         LD (DE),A
         LD A,(HL)
         ADD A,03
         LD (HL),A
         INC HL
         LD A,(HL)
         ADD A,04
         LD (HL),A
         RET
         END A7000H
```

BASIC-Programm zum Aufrufen der obenstehenden Assembler-Routine (Parameter-Übergabe):

```
100   INPUT P1,P2
110   CALL 28672,P1,P2
120   PRINT P1,P2
130   CALL 28672,P1,P2
140   PRINT P1,P2
150   END
```

CLEAR 3300

Syntax:

CLEAR

Wirkung:

Im Grafik-Modus wird der Bildschirm gelöscht, der Grafik-Modus bleibt bestehen.

Beispiel:

```
90 CLEAR
```

Bildschirm wird gelöscht. Es wird im Grafik-Modus weitergearbeitet.

CLOSE

Syntax:

CLOSE #n:

Wirkung:

Schließt die logische Einheit n. Möglich für Ausgabe- und bidirektionale Treiber und für Dateien.

Siehe OPEN

Nach Beenden, Abbrechen (ESC-Taste) oder Unterbrechen (STOP-Anweisung) von Programmen oder Befehlsfolgen (Kommando-Modus) werden offene Dateien automatisch geschlossen, der Grafik-Modus wird beendet.

Beispiele:

1000 CLOSE #10:

Schließt die Eingabedatei auf einem Medium.

1010 CLOSE #9:

Schließt den Grafik-Treiber, wenn dieser geöffnet wurde. Dabei wird von der grafischen Betriebsart des Bildschirms in die alphanumerische Betriebsart zurückgeschaltet. Das Bild wird dabei gelöscht.

DATA

Syntax:

DATA liste

Wirkung:

Speichert im Programm die Werte der Liste 'liste'.

'liste' kann numerische- und/oder Stringkonstanten enthalten. Die einzelnen Werte sind durch "," getrennt.

Die Werte der DATA-Anweisungen werden unabhängig von ihrer Lage im Programm nacheinander von den READ-Befehlen gelesen.

(siehe READ,RESTORE)

Beispiele:

```
100 DATA "Ich kauf mir lieber einen Tirolerhut","Donauwalzer"
110 DATA 1,2,3,4,5
120 DATA 2.0E+5,"Wish you were here",500000
```

DEL

Syntax:

DEL #n:

Wirkung:

Die zur logischen Einheit 'n' gehörende Datei wird gelöscht. Für 'n' sind nur die Kanäle 10..13 zulässig. Die Datei muß vorher mit der OPEN-Anweisung geöffnet worden sein. Nach Ausführen der DEL-Anweisung ist die Datei auf dem Medium und in BASIC gelöscht, die logische Kanalnummer ist wieder frei.

Beispiel:

```
 .
 .
 .
100  OPEN#12: "SCRATCH.XXX"
 .
 .
 .
900  DEL#12:
```

Die Datei SCRATCH.XXX wird geöffnet (100), bearbeitet und schließlich wieder gelöscht (900).

DIM

Syntax:

DIM variable(1)

DIM variable(1, k, m, n)

Wirkung:

Reserviert Speicher für Felder.

'1', 'k', 'm' und 'n' sind die oberen Indizes des Feldes, die niederen Indizes sind immer 0.

Index : 0..65535

Anzahl der Dimensionen: 1..255

Bei Überschreiten der Feldgrenzen erfolgt eine Fehlermeldung.
Der Index einer Feldvariablen kann nicht wieder eine Feldvariable sein.

Beispiele:

DIM A(8)

Reserviert ein Feld A mit 9 Variablen.

DIM B(10,10)

Reserviert ein Feld B von 11 mal 11 Variablen.

DIM A$(80)

Reserviert ein Feld für 80 Strings mit jeweils beliebig vielen Zeichen.

END

Syntax:

END

Wirkung:

Beendet das Programm und schließt alle geöffneten Dateien; Rückkehr in den Kommando-Modus.

Die END-Anweisung kann an beliebiger Stelle im Programm stehen. Fehlt sie, so setzt BASIC nach der letzten Zeile des Programms intern eine END-Anweisung.

Bei Verwendung von Unterprogrammen ist dafür zu sorgen, daß diese nur durch GOSUB erreicht werden. Hierzu wird vor dem ersten Unterprogramm ein END-Befehl gesetzt.

Beispiel:

```
100 GOSUB 300
110 END
300 REM -- Unterprogramm --
 .
 .
 .
400 RETURN
```

Das Hauptprogramm ist in Zeile 110 durch die END-Anweisung abgeschlossen, sodaß das Unterprogramm nur durch GOSUB erreicht wird.

FOR..NEXT

Syntax:

```
FOR I=i0 TO i1 (STEP s)
  .....
NEXT I
```

Wirkung:

Der Programmteil zwischen der FOR-Anweisung und dem NEXT wird mindestens einmal ausgeführt und abhängig von der Laufvariablen 'I' wiederholt. Nach dem Ausführen des Programmteils wird 'I' um die Schrittweite 's' erhöht. Dann wird geprüft, ob der Endwert 'i1' überschritten wurde. Hat die Laufvariable den Endwert noch nicht erreicht, wird das Programmteil noch einmal ausgeführt.

Die Laufvariable 'I' muß vom Typ "REAL" sein. Sie darf nicht Element eines Feldes sein.

Wird NEXT ohne Angabe der Laufvariablen 'I' verwendet, so wird das NEXT als zu dem nächsten vorher kommenden FOR... gehörend angenommen.

I	ist eine numerische Variable
i0,i1,s	können numerische Konstanten, Variable oder Ausdrücke sein

Die Schleife wird auf jeden Fall einmal durchlaufen; sie kann verlassen werden, bevor sie vollständig abgearbeitet wurde (GOTO).

Wegen der mit diesem Verfahren verbundenen Unübersichtlichkeit sollte ein 'GOTO' in einer Schleife möglichst vermieden werden.

Überschneidung von Schleifen ist nicht zulässig. Schleifen können bis zu einer Tiefe von 7 geschachtelt werden.

FOR..NEXT

---------- Fortsetzung

Beispiele:

```
1000 FOR I=1 TO 10
1010 PRINT
1020 NEXT I
```

Die Schleife wird 10 mal durchlaufen. Dabei wird 10 mal der Befehl PRINT ausgeführt.
Danach wird das Programm nach dem NEXT-Befehl fortgeführt.

```
3000 FOR K=0 TO K1
3010 FOR L=0 TO L1
3020 LET A(K,L)=K*L
3030 NEXT L
3040 NEXT K
```

Die Elemente des Feldes A mit der Ausdehnung K1 mal L1 werden mit dem Produkt ihrer Indizes belegt.

Dabei wird für jeden Wert des Index K die innere Schleife L1 mal durchlaufen.

Bei 3030 NEXT L und 3040 NEXT K kann L und K weggelassen werden.

```
5000 FOR X7=C*8/N TO C*3/M STEP -S
5010 SET X7,0
5020 NEXT
```

Die Schleife wird von einem Anfangswert zu einem kleineren Endwert mit einer negativen Schrittweite durchlaufen.

GET

Syntax:

GET stringvariable

Wirkung:

Holt **ein** Zeichen von der Tastatur.

Ist kein Zeichen eingegeben worden, so wartet GET nicht auf eine Eingabe, der alte Wert der Stringvariablen bleibt erhalten. Es kann jeweils nur **ein** ASCII-Zeichen eingegeben werden. Es gibt also auch kein Eingabe-Abschluß-Zeichen (wie <CR> bei INPUT). Das Zeichen wird nicht auf dem Bildschirm reflektiert.

Deshalb ist dieser Befehl auch im Grafik-Modus erlaubt. Die Reflektierung der Zeichen sowie das Warten auf eine Eingabe sind mit BASIC-Anweisungen programmierbar.

Beispiele:

```
10 GET A$
```

Holt das anstehende Zeichen von der Tastatur. Steht kein Zeichen an, so läuft das Programm weiter.

```
20 GET B$
30 IF LEN(B$)=0 THEN 20
40 IF ASC(B$)=13 THEN 70
50 LET C$=C$+B$
60 GOTO 20
70 STRING C$,10,10
```

Eine INPUT-Anweisung wird im Grafik-Modus simuliert. Eine Warteschleife (20,30) liest die einzelnen Zeichen (B$) ein. Diese werden zu einem String (C$) verkettet (50). Durch die RETURN-Taste (ASCII-CODE 13) wird die Eingabe abgebrochen (40,70).

GOSUB T30

Syntax:

GOSUB zeilennr

Wirkung:

Ruft das Unterprogramm ab Zeilenummer 'zeilennr' auf.

Ein Unterprogramm wird mit der Anweisung RETURN abgeschlossen. BASIC setzt dann das Programm nach der Zeile mit dem aufrufenden GOSUB fort. Das Unterprogramm darf nur durch GOSUB-Befehle erreicht werden. Wird ohne vorheriges GOSUB ein RETURN erreicht, so erfolgt eine Fehlermeldung.

Siehe RETURN, ON, ON INTR, ON ERROR

Hinweis: Nach GOSUB darf keine weitere Anweisung in der gleichen Programmzeile stehen.

GOSUB

---------- Fortsetzung

Beispiele:

```
10 REM -- Hauptprogramm --
 .
 .
 .
100 GOSUB 1000
110 REM -- Weiter --
 .
 .
 .
999 END

1000 REM  -- Unterprogramm --
 .
 .
 .
1100 RETURN
```

Das Unterprogramm ab Zeilennummer 1000 wird vom Hauptprogramm aufgerufen.
Nach der Zeile 1100 wird die Zeile 110 ausgeführt.

```
100 INPUT "String";A$
110 REM    String invertieren
120 GOSUB 2000
130 PRINT "Invertierter String: ";A$
 .
 .
 .
1999 END

2000 FOR I=LEN(A$)-1 TO 0 STEP -1
2010 LET B$=B$+MID$(A$,I,1)
2020 NEXT I
2030 LET A$=B$
2040 RETURN
```

Liest einen String ein und springt ein Unterprogramm an, das den String invertiert. Dann wird der invertierte String ausgedruckt.

GOTO

Syntax:

GOTO zeilennr

Wirkung:

Setzt das Programm mit der Zeile 'zeilennr' fort. 'zeilennr' ist eine im Programm existierende Zeilennummer.

Siehe ON, ON INTR, ON ERROR

Beispiel:

```
100 GOTO 1000
200 ..
1000 REM          Unterprogrammbeginn
2000 REM RETURN   Unterprogrammende
```

Setzt das Programm mit der Zeilennummer 1000 fort.

Zur besseren Strukturierung von Programmen sollten GOTO-Anweisungen außerhalb von IF- und ON-Anweisungen möglichst vermieden werden.

Hinweis: Nach GOTO darf keine weitere Anweisung in der gleichen Programmzeile stehen.

IF...THEN

Syntax:

```
IF b GOTO zeilennr
IF b THEN zeilennr
IF b THEN anweisung

IF b GOSUB zeilennr

IF b THEN anweisung1    ELSE    anweisung2
IF b THEN zeilennr1     ELSE    zeilennr2
```

Wirkung:

Verzweigt in Abhängigkeit von der Bedingung 'b'.
In den ersten beiden Syntaxvarianten wird nach 'zeilennr' verzweigt, wenn 'b' "wahr" ist.
Auf THEN darf auch eine Anweisung folgen, wie im dritten Fall.

In der vierten Variante wird das Unterprogramm ab 'zeilennr' aufgerufen, wenn 'b' "wahr" ist.

In der letzten Variante wird 'anweisung1' ausgeführt, wenn 'b' "wahr" ist. 'anweisung2' wird ausgeführt, wenn 'b' "falsch" ist.
Werden statt der Befehle Zeilennummern verwendet, so wird nach 'zeilennr1' verzweigt, wenn 'b' "wahr" ist und im anderen Fall das Programm mit 'zeilennr2' fortgesetzt.

'b' kann sein

- numerische Variable, Ausdruck
  0 = "falsch"; -1 = "wahr"

- logische Verknüpfung
  siehe logische Operatoren, Priorität von Operationen

- Vergleichsoperationen mit numerischen oder String-Variablen bzw. Ausdrücken

- Kombination aus dem obigen

Achtung!
Bedingungen, bei denen numerische Werte auf Identität geprüft werden, sind mit Vorsicht anzuwenden, da Zahlen nur mit endlicher Genauigkeit dargestellt werden und somit bereits Rundungsfehler zu Ungleichheit führen. Vorzuziehen sind Abprüfungen auf Wertüberschreitung.

'RETURN'-Anweisungen in 'IF...THEN...(ELSE)'-Anweisungen sollten innerhalb derselben Zeile vermieden werden.

INPUT

Syntax:

```
INPUT liste

INPUT "string",     liste
INPUT "string";     liste
```

Wirkung:

Einlesen von Variablen von der Tastatur.
Die Elemente der Liste 'liste' werden nacheinander von der Tastatur eingelesen.

Elemente können beliebige Variablen sein, sie werden in der 'liste' mit ',' getrennt.

Bei der Ausführung des INPUT-Befehls wird zunächst

?: 'd' nnee .

oder, wenn angegeben

string?:

ausgegeben.

BASIC erwartet dann die Eingabe von Werten, die den Variablen der Liste zugewiesen werden. Bei der Eingabe numerischer Variablen können im Gegensatz zu den Stringvariablen die Werte durch ',' oder durch ' ' getrennt werden. Wenn nicht genügend Werte eingegeben werden, so erscheint nochmals die Eingabeaufforderung "?:". Die Eingabezeile wird durch die RETURN-Taste abgeschlossen.

Ein RETURN als alleinige Eingabe weist der entsprechenden Variablen den Wert 0 bzw. einen Leer-String zu.

Zu beachten ist, daß die Länge der Eingabezeile von der BASIC-internen Zeilenlängenverwaltung bestimmt wird. Die Zeilenlänge kann mit der LINELEN-Anweisung verändert werden.

INPUT liest von dem Betriebssystem-Ausgabekanal 0-1 ein.

INPUT

---------- Fortsetzung

Beispiele:

INPUT A

Liest den Wert der Variablen A ein.
Diese kann als Ganzzahl, als Festkommazahl oder als Exponentialzahl eingegeben werden, z.B.:

12345
3.14159
-2.0E-22

INPUT X,Y,Z

Liest die Werte der Variablen X,Y und Z ein.
Eingabeformat wie oben.

INPUT A$

Liest den Wert der String-Variablen A$ ein.
Abschließen durch RETURN-Taste.

Eingabebeispiel:

Heute scheint die Sonne
ABC-123:ijgh,##

INPUT "Eingabe von Farbe und Größe (Farbe,Größe)"; F$(5), G(5)

Druckt die Stringkonstante aus und liest dann die Werte für die Feldelemente F$(5) und G(5) ein.

INPUT

Syntax:

INPUT #n: liste

Wirkung:

Liest von der logischen Einheit n die in 'liste' angegebenen Variablen nacheinander ein. Die logische Einheit kann eine Datei (n = 10..13) oder ein Treiber (n = 4..9) sein. In 'liste' können beliebige Variablen angegeben sein.

Bei numerischen Variablen (REAL, INTEGER) wird nur eine Eingabezeile erwartet. Die Variablen müssen mit ',' oder ' ' voneinander getrennt werden. Die Eingabezeile wird mit RETURN abgeschlossen.

STRING-Variablen werden mit RETURN getrennt, d.h. es ist für jede STRING-Variable eine ganze Eingabezeile verfügbar. Hier ist die INPUT#-Anweisung erst dann beendet, wenn alle String-Variablen gelesen wurden.

Wird die Anweisung 'INPUT #n:' auf eine leere Datei angewendet und die Fehlermeldung unterdrückt, kann anschließend kein Schreib-/Lesezugriff auf diese Datei erfolgen. Die Datei muß in diesem Fall erst geschlossen (CLOSE) und wieder geöffnet (OPEN) werden.

Ein sogeannter Dummy-Read mit 'INPUT #n:' ist nur innerhalb eines Records (128 Byte) möglich. Mit 'RECORD #n:x' kann auf einen bestimmten Record positioniert werden.

Die Länge der Eingabezeilen wird von BASIC überwacht. Sie ist mit der LINELEN-Anweisung veränderbar. Das Überschreiten der zulässigen Zeilenlänge hat eine Fehlermeldung mit Programmabbruch zur Folge.

Beispiele:

```
50 INPUT #10: X
```

Liest die Variable X von der vorher geöffneten Datei (OPEN #10:..) ein.

```
100 INPUT #11: A$
```

Liest die Variable A$ von der vorher mit OPEN #11:D$ geöffneten Datei ein. Der Dateizeiger steht anschließend auf dem nächsten Eintrag hinter dem eingelesenen. Diese Anweisung kann auch zur Positionierung des Dateizeigers in einer Datei benutzt werden (Dummy-Read).

```
150 INPUT #1: T$
```

Liest vom Betriebsystemkanal I-1 die Stringvariable T$ ein. I-1 ist standardmäßig der Tastaturtreiber $KEY. Die Eingabe wird nicht auf dem Bildschirm reflektiert.

INV,SET,RES

Syntax:

```
SET x,y
RES x,y
INV x,y
```

Wirkung:

Anweisung für Grafik-Modus:

Setzt, löscht oder invertiert den Punkt mit den Koordinaten x,y. 'x' und 'y' können Konstanten, Variable oder Ausdrücke sein.

Grafik-Treiber muß aktiviert und geöffnet sein (siehe Treiber)

Wertebereiche:

0 <= x <= 511 bzw. 1023 bei Kontron PSI980 H
0 <= y <= 255 bzw. 399 bei Kontron PSI980 H

Beispiele:

```
150 SET 100,100
```

Setzt den Punkt 100,100

```
200 RES X1,Y1
```

Löscht den Punkt X1,Y1

```
300 INV X-X0,Y-Y0
```

Invertiert den Punkt X-X0,Y-Y0.

KOS

Syntax:

KOS
KOS kommando

Wirkung:

In der ersten Syntaxform führt die Ausführung der Anweisung 'KOS' zum Verlassen des BASIC-Interpreters und zur Rückkehr ins Betriebssystem KOS. Ein geladenes BASIC-Programm wird nicht gerettet. Der vom Interpreter und vom Anwenderprogramm belegte Speicherraum wird frei.

Der Aufruf der KOS-Anweisung in der zweiten Syntaxform läßt den Seicherraum belegt und ermöglicht die Ausführung von KOS-internen Kommandos, z.B.

KOS M gibt das Mastermedium an

KOS C schaltet die Tastatureingabe zwischen Groß- und Kleinschreibung um

Nicht möglich ist der Aufruf von diskresidenden Kommandos, die in gleichen Speicherbereichen liegen wie der BASIC-Interpreter und dessen Anwendungsspeicher. Passend gelinkte Programme können von Diskette geladen werden:

```
KOS BEISPIEL                                    :
```

führt zum Laden und Ausführen des Assembler-Programms BEISPIEL, das außerhalb des von BASIC belegten Speicherbereichs liegt.

Nach der Ausführung von 'KOS kommando' führt BASIC das Programm weiter aus, bzw., bei Eingabe von 'KOS kommando' im Kommando-Modus, kehrt BASIC in den Kommandomodus zurück. Ein geladenes Programm bleibt bei der Ausführung von 'KOS kommando' unverändert.

Hinweis: Für KOS steht der BASIC-Interpreter als relokierte Version unter dem Namen BASKOS zur Verfügung. Diese Version ermöglicht auch die Ausführung von diskresidenten KOS-Kommandos.

Das KOS-interne Kommando 'I' (KOS I) sollte vermieden werden, eine Alternative ist der Aufruf des KOS-Kommandos 'IL' aus BASKOs.

LET

Syntax:

LET variable=ausdruck

variable=ausdruck

Wirkung:

Wertzuweisung einer Variablen.

Der Variablen 'variable' wird der Wert des Ausdrucks 'ausdruck' zugewiesen.
'variable' kann eine beliebige Variable sein.

Die zweite Syntaxform ohne LET kann genauso verwendet werden.

Beispiele:

```
10 LET X=1
oder
10 X=1
```

Weist der Variablen X den Wert 1 zu.

```
20 LET R=SQR(X*X+Y*Y)
oder
20 R=SQR(X*X+Y*Y)
```

Weist der Variablen R den Wert des Ausdrucks SQR(X*X+Y*Y) zu.

```
30 LET A(I,J)=B(N,M)
```

Weist dem Feldelement A(I,J) den Wert des Feldelements B(N,M) zu.

```
40 A$="String: "+LEFT$(C$,5)+MID$(C$,10,2)
```

Weist der Stringvariablen A$ die Kettung einer Stringkonstanten mit zwei Teilstrings der Variablen C$ zu.

LINELEN

Syntax:

LINELEN m = zeilenlänge

'm' bestimmt die Richtung:
- m := I Eingabe (Input)
- m := O Ausgabe (Output)

'zeilenlänge' = Ausdruck, der die Zeilenlänge angibt
Wertbereich:
zeilenlänge := 20..132
oder, für Ausgabe:
zeilenlänge := * (LINELEN O=*) für unendlich

Wirkung:

BASIC verwaltet die Länge der Zeilen getrennt für Ein- und Ausgabe. Wählbar sind Zeilenlängen zwischen 20 und 132. Jedes Zeichen wird gezählt. Ausgenommen hiervon ist das Backspace (ASCII 08), das den Zeichenzähler um 1 erniedrigt. Die Länge der Ausgabezeilen kann als unendlich vorgegeben sein.

Die Wirkung der LINELEN-Anweisung ist in folgenden Fällen zu unterscheiden:

Eingabe:

Kommandomodus und INPUT — Es wird über den log. Kanal 0 eingegeben mit Reflektion der Zeichen auf den Bildschirm (log. Kanal 0). Die Zeile kann mit den BASIC-Editor-Kommandos bearbeitet werden.

INPUT# — Ein logischer Kanal >0 wird angesprochen. Keine Reflektion der Zeichen. Bei Zeilenende wird die Eingabe abgebrochen und die Fehlermeldung 89 (Zeile zu lang) ausgegeben. Ein Programmlauf wird abgebrochen.

ALOAD — wie INPUT#, nur wird im Fehlerfall das Laden des Programms fortgesetzt.

Hinweis: Es wird nur ein Ausgabezähler geführt, der alle nicht abgeschlossenen Ausgaben (siehe PRINT....;) aufsummiert. Dies beeinflußt die Positionierung durch 'RECORD'. Mitgezählt werden auch Control-Zeichen, z.B. CTRL-G. Diese wirken sich bei der TAB-Funktion (CTRL-I) mit aus.

LINELEN

---------- Fortsetzung

Ausgabe: Vor Überschreiten der maximal erlaubten Zeilenlänge wird Carriage-Return ausgegeben, d.h. die Ausgabe wird auf einer neuen Zeile fortgesetzt.

Ist unendliche Zeilenlänge gewählt, wird kein Carriage-Return ausgegeben. Die Tabulatorfunktion in der PRINT-Anweisung ',' ist immer definiert, die Funktion TAB dagegen nur bis 240 Zeichen nach dem letzten Carriage Return.

Die LINELEN-Anweisung darf beliebig oft verwendet werden.

Voreinstellungen:

```
LINELEN I=80     Eingabezeile : max. 80 Zeichen
LINELEN O=*      Ausgabezeile : unendlich lang
```

Nach jedem NEW wirkt die Voreinstellung der Zeilenlängenverwaltung.

Beispiel:

LINELEN O = *	(Voreinstellung)
LINELEN I = 132	Es ist möglich, Zeilen mit einer Länge von 132 Zeichen einzugeben. Bei dem PSI-Bildschirm mit 80 Zeichen Zeilenlänge geht die Eingabezeile über zwei Zeilen.
LINELEN I = 80	Die Parameter zur Zeilenlängenverwaltung entsprechen wieder der Voreinstellung
LINELEN O = 71 ASAVE dateiadresse	Die Datei (= das Programm) kann mit dem Kontron-Editor EDIT bearbeitet werden, wenn Programmzeilen nicht länger als 71 Zeichen sind. Dabei ist zu beachten, daß jede Programmzeile mit einer Zeilennummer beginnen muß.
LINELEN O = *	Rücksetzen der Parameter entsprechend der Voreinstellung

ON

Syntax:

ON p GOTO liste

ON p GOSUB liste

Wirkung:

Wie Befehle GOTO und GOSUB; jedoch kann in Abhängigkeit von 'p' zu verschiedenen Zielen verzweigt werden.

'p' kann eine Variable oder ein Ausdruck sein.
'p' wird in eine Ganzzahl gewandelt und als Ordnungszahl für die in der Liste 'liste' aufgeführten Zielzeilen verwendet.

'liste' ist eine Liste von im Programm existierenden Zeilenzahlen.

Ist 'p' gleich 1, so wird die 1. Zielzeile angesprungen; ist 'p' gleich 2, so wird die 2. Zielzeile angesprungen usw..

Ist 'p' kleiner als 1 oder größer als die Anzahl der Zielzeilen in 'liste', so wird das Programm mit dem nächsten Befehl fortgesetzt.

Für 'p' dürfen nicht die Variablen INTR oder ERROR verwendet werden!

Siehe ON INTR, ON ERROR.

Hinweis: Nach GOTO bzw. GOSUB darf in der gleichen Zeile keine weitere Anweisung folgen.

ON

---------- Fortsetzung

Beispiele:

```
10 A=3
20 ON A GOTO 50,60,80,90
```

Verzweigt wird nach Zeile 80.

```
100 ON N GOTO 200,300,400
```

Verzweigt abhängig von N.
N wird mit der Funktion INT in eine Ganzzahl gewandelt.
Es gilt als Zielzeile für

```
1 <= N < 2      GOTO 200
2 <= N < 3      GOTO 300
3 <= N < 4      GOTO 400
```

Für alle anderen Werte wird das Programm mit dem nächsten Befehl fortgeführt.

```
200 ON 2+SGN(X) GOSUB 1000,2000,3000
```

Ruft abhängig von X verschiedene Unterprogramme auf.

```
X < 0  GOSUB 1000
X = 0  GOSUB 2000
X > 0  GOSUB 3000
```

(siehe Funktion SGN)

ON ERROR

Syntax:

```
ON ERROR GOTO
ON ERROR GOSUB
ON ERROR RETURN
```

Wirkung:

ON ERROR GOTO
Diese Anweisung veranlaßt zunächst keine Aktion. BASIC wird hiermit informiert, daß im Fehlerfall keine Meldung ausgegeben, sondern bei GOTO zu der angegebenen Zeile verzweigt werden soll.

ON ERROR GOSUB
In gleicher Weise wird bei GOSUB im Fehlerfall eine Unterroutine aufgerufen und anschließend nach der RETURN-Anweisung die auf die fehlerhafte Zeile folgende Programmzeile ausgeführt.

Nach einer Fehlerbehandlung mit der ON ERROR Anweisung wird ON ERROR unwirksam. Der nächste Fehler führt wieder zu einer Meldung und zum Abbruch der Programmausführung. Damit ist sichergestellt, daß BASIC nicht in eine Endlosschleife gerät, wenn der im Fehlerfall auszuführende Programmteil ebenfalls einen Fehler enthält. Der Anwender sollte also die ON ERROR Anweisung in die Programmschleife einbauen oder gezielt am Ende der Fehlerbehandlungsroutine eine neue ON ERROR Anweisung einfügen.

ON ERROR RETURN
nimmt die im vorigen Abschnitt festgelegte Fehlerbehandlung wieder zurück. Es erfolgt im Fehlerfall eine Meldung.

Hinweis: Wird ein Programmlauf unterbrochen (ESCAPE, STOP-Anweisung) und ist eine 'ON ERROR'-Anweisung noch nicht abgeschlossen (kein Fehler aufgetreten und kein 'ON ERROR RETURN'), wird bei einem Fehler während der Kommandoeingabe diese 'ON ERROR..'-Anweisung ausgeführt. Abhilfe bietet das vorsorgliche Eingeben des Kommandos 'ON ERROR RETURN'.

Beispiele:

```
    .....
200 ON ERROR GOTO 300
210 INPUT #11:A$(N)
220 N=N+1:GOTO 210
300 IF ERM(0)=87 THEN PRINT "DATEI-ENDE" ELSE STOP
310 GOTO....
```

Mit dieser Routine soll eine Datei unbekannter Länge bis zum Ende gelesen werden. Um die sonst unvermeidliche Fehlermeldung zu umgehen, wird in diesem Fall zur Zeile 300 verzweigt. Die Fehlerbehandlungsroutine ab Zeile 300 prüft, ob der Fehler wirklich durch ein Datei-Ende verursacht wurde und schreibt in diesem Fall eine eigene Meldung. Im anderen Fall wird die Programmausführung beendet.

ON INTR (INTERRUPT)

Syntax:

ON INTR GOTO zeilennr

ON INTR GOSUB zeilennr

ON INTR RETURN

Wirkung:

Führt abhängig von Software-Interrupts BASIC-Anweisungen aus.

In der ersten Syntaxvariante wird das Programm bei diesem Befehl so lange angehalten, bis ein Interrupt erfolgt. Dann wird das Programm ab Zeile 'zeilennr' fortgesetzt.

In der zweiten Variante wird, nachdem dieser Befehl durchlaufen ist, bei jedem Interrupt das Unterprogramm ab Zeile 'zeilennr' aufgerufen. Das Programm wird danach an der unterbrochenen Stelle fortgeführt (BASIC-Interrupt-Service-Routine).

Durch die dritte Syntaxvariante wird die Wirkung dieses Befehls wieder aufgehoben. Ab dieser Zeile werden Interrupts ignoriert.

Um einen Interrupt auf Maschinenebene verarbeiten zu können, ist eine Assembler-Interrupt-Service-Routine notwendig. Dieses, durch einen Hardware-Interrupt aktivierte Programm, hat das BASIC-Interrupt-Flag-Byte auf Adresse 109H auf OFFH zu setzen und löst damit den Software-Interrupt für den BASIC-Interpreter aus.

OPEN

Syntax:

OPEN #n: string

string: STRING-Variable, Ausdruck, STRING-Konstante

Wirkung:

Öffnet die logische Einheit, die sich aus 'string' ergibt, und ordnet sie Kanal n zu. Unter dieser Kanalnummer ist die logische Einheit dann in PRINT # und INPUT # ansprechbar.

Es gibt 3 Arten von logischen Einheiten:

- Treiber mit beliebiger Kanalzuordnung

  Drucker-Treiber, IEC-Treiber können auf die Kanäle 3..8 gelegt werden.
  string: "$xxxx"

  Der Name eines Treibers besteht aus einem "$"-Zeichen und **genau** 4 alphanumerischen Zeichen. Hat der Name weniger als 4 Zeichen, so sind die restlichen Stellen mit Leerzeichen aufzufüllen.

  Achtung: Auf Kanal 0-3 liegt $KSM (System Messages)

- Treiber mit fester Kanalzuordnung

  Grafik-Treiber Kanal 9
  string: "$GRAP"

- Dateizugriff

  Ein-/Ausgabe auf den Kanälen 10..13. Nach OPEN ist das Schreiben und Lesen vom Dateianfang an gemischt möglich. Vier Dateien können gleichzeitig offen sein. Das Medium muß Random Access erlauben.

  dateiadresse: (mn:)dateiname(.typ)

  z.B. TEST 1:TEST TEST.ABC 0:TEST.ABC

Voreinstellung:

mn : Mastermedium (Masterlaufwerk)
typ: DAT

OPEN

Achtung: Falls eine zu öffnende Datei noch nicht existiert, so wirken sich

```
OPEN #10: "DATEI"
```

und `OPEN #10: "1:DATEI"`

unterschiedlich aus:

```
200 OPEN #10: "DATEI"
```

Die Datei wird auf allen aktiven Medien, beginnend mit dem Mastermedium gesucht. Ist sie nicht vorhanden, so wird sie auf dem Mastermedium angelegt.

Ist die Datei 'DATEI.DAT' auf einem Medium vorhanden, so wird sie eröffnet und es wird ihr die logische Einheit 10 zugeordnet. Der Dateizeiger weist auf das erste Zeichen der Datei.

```
200 OPEN #10: "1:DATEI"
```

sucht die Datei "DATEI.DAT" auf Medium 1 und eröffnet sie. Falls sie nicht existent war, so existiert nach dem Befehl eine zunächst leere Datei dieses Namens auf Medium 1.

```
200 LET N$="1:KUNDEN.001"
300 OPEN #13: N$
```

Öffnet auf Medium 1 die Datei KUNDEN.001 und weist dieser die logische Einheit 13 zu.

```
400 OPEN #5: "$SIO "
```

Öffnet den Treiber $SIO und weist ihm die logische Einheit 5 zu.

```
500 OPEN #9: "$GRAP"
```

Öffnet den Grafik-Treiber und löscht den Bildschirm.

Hinweis: Die Zuweisungen auf logische Kanäle aus BASIC heraus ist unabhängig vom IODC-Kommando.

OUT

Syntax:

OUT adr,x

Wirkung:

Ausgabe über die E/A-Adressen der Z80A-CPU.

OUT gibt ein Byte (x) auf den Port mit der Adresse 'adr' aus.

'adr' ist eine existierende E/A-Adresse (dezimal).
'x' kann eine numerische Konstante, Variable oder Ausdruck sein.

```
0 <= adr <= 255
0 <=  x  <= 255
```

Beispiele:

```
20 OUT 4,Z+48
```

Gibt ein Byte (Z+48) auf den Port mit der Adresse 4 aus.

PLOT

Syntax:

PLOT x,y,p

Wirkung:

Plotten im Grafik-Modus auf dem Bildschirm

Der Grafik-Treiber muß aktiviert und geöffnet sein (siehe Treiber).

PLOT fährt einen imaginären Schreibstift von der letzten PLOT-Position zu dem Punkt 'x','y'.

Für 'p' gilt:

p=0:	Strecke nicht zeichnen (nur Positionierung)
p=1:	Strecke zeichnen
p=2:	Strecke löschen
p=3:	Strecke invertieren
p=4:	Zeichnen eines Rechteckrahmens
p=5:	Löschen eines Rechteckrahmens
p=6:	Invertieren eines Rechteckrahmens
p=7:	Zeichnen einer Rechteckfläche
p=8:	Löschen einer Rechteckfläche
p=9:	Invertieren einer Rechteckfläche

0 <= x <= 511 bzw. 1023 bei Kontron PSI980 H
0 <= y <= 255 bzw. 399 bei Kontron PSI980 H

'x', 'y' und 'p' können Ausdrücke sein.

Hinweis: Die Variablennamen der Koordinaten dürfen höchstens 2-stellig sein.

PLOT

---------- Fortsetzung

Beispiele:

```
50 PLOT 0,0,0
```

Fährt die Position 0,0 an, ohne zu zeichnen.
(Linke untere Ecke des Bildschirms)

```
60 PLOT X1,Y1,1
```

Zeichnet die Strecke vom letzten PLOT-Punkt zu dem Punkt mit den Koordinaten X1,Y1.

```
70 PLOT C*(X-X0),C*(Y-Y0),1
```

Zeichnet die Strecke vom letzten PLOT-Punkt zu dem Punkt mit den Koordinaten C*(X-X0),C*(Y-Y0).

```
80 PLOT 10,10,0
90 PLOT 50,60,9
```

Punktweises Invertieren der Rechteckfläche mit den Koordinaten

10,10 ; 50,10 ; 50,60 ; 10,60

POKE

Syntax:

POKE adr,x

Wirkung:

Schreibt 'x' in den Speicherplatz mit der Adresse 'adr'.

'adr' ist eine Adresse aus dem Speicher des Computers in dezimaler Schreibweise. 'adr' kann eine numerische Konstante, eine Variable oder ein Ausdruck sein.

'x' ist eine numerische Konstante, Variable oder Ausdruck.

```
0 <= x <= 255
0 <= adr <= 65535
```

Beispiele:

```
300 POKE 2000,0
```

Schreibt in die Adresse 2000 (dezimal) den Wert 0.

```
400 POKE A,X+64
```

Schreibt in den Speicherplatz mit der Adresse A den Wert X + 64.
(65 = "A", 66 = "B",....)

PRINT

Syntax:

PRINT liste

oder

? liste

Wirkung:

Gibt die Elemente der Liste 'liste' auf den Bildschirm aus.

Elemente der Liste können Konstanten, Variablen oder Ausdrücke sein.

Das Ausgabeformat kann durch die Trennzeichen zwischen den Elementen gesteuert werden. BASIC verwaltet die Zeilenlänge für Ausgabe, siehe LINELEN.

,	Tabulation; 20 Zeichen pro Element
;	Kein Zwischenraum, kein Zeilenvorschub; bei Zahlen wird jedoch ein Zeichen als Vorzeichen reserviert. Zeichenzähler wird inkrementiert.
TAB(I)	Tabulation bis zur I. Position Liegt I vor der aktuellen Spalte, so wird TAB ignoriert. Bei unendlich langer Ausgabezeile (LINELEN 0=*) ist TAB nur bis zur 240. Position nach Carriage Return definiert.
PRINT	ohne 'liste' setzt die Schreibmarke auf den Anfang der nächsten Zeile.

PRINT gibt auf den Betriebssystemkanal 0-1 aus.

Anhalten der Ausgabe durch "_" (Shift-Minus).

Siehe auch: PRINT#
PRINT USING

PRINT

---------- Fortsetzung

Beispiele:

Ausgabe einer numerischen Konstanten
```
10 PRINT 12345
```

Ausgabe:
```
12345
```

Ausgabe einer String-Konstanten
```
20 PRINT "DAS IST EIN STRING"
```

Ausgabe:
```
DAS IST EIN STRING
```

Ausgabe einer Variablen
```
25 P=3.1416
30 PRINT P
```

Ausgabe:
```
3.1416
```

Ausgabe eines Ausdrucks
```
40 R=5
50 PRINT 2*P*R
```

Ausgabe:
```
31.416
```

Ausgabe mehrerer Elemente
```
60 PRINT "Radius:";R;" mm "," Umfang:";2*P*R;" mm"
```

Ausgabe:
```
Radius: 5 mm       Umfang: 31.416 mm
```

Abgekürzte Schreibweise
```
100?X,Y
```

ist gleichwertig mit
```
100 PRINT X,Y
```

PRINT

Syntax:

PRINT #n: liste

Wirkung:

Schreibt auf die logische Einheit n die Liste 'liste'.
Siehe PRINT.

Die Ausgabe erfolgt im ASCII-Code.

Die logische Einheit kann eine Datei auf einem Medium oder ein Treiber sein.

Beispiele:

```
50 PRINT #11: X
```

Schreibt die Variable X auf eine Datei, welche vorher geöffnet worden sein muß (OPEN).

```
90 X=1: Y=2
100 PRINT #3: "X =";X;"   Y =";Y
```

Schreibt auf die logische Einheit 3:

X = 1 Y = 2

PRINT USING

Syntax:

PRINT USING string trennz liste
'trennz' ist "," oder ";" (Wirkung wie bei PRINT)

Wirkung:

Jedes Element der Liste 'liste' wird mit dem String 'string' formatiert ausgegeben. 'string' ist eine Stringkonstante und besteht aus einer Folge von Platzhaltern, die das Format einer Zahl festlegen.

Mögliche Elemente von 'string':

#

Platzhalter für eine Ziffer.
Bei Überlauf rechts wird gerundet; bei Überlauf links erscheint die Fehlermeldung '??'. Nicht besetzte Stellen vor dem Komma ergeben Leerzeichen; nach dem Komma 0.

.

Platzhalter für den Dezimalpunkt

*****

Nicht besetzte Stellen vor dem Komma werden durch '*' ersetzt.
(Z.B. beim Ausfüllen von Schecks vor dem Betrag: ***100.-DM)
* zählt nicht als Platzhalter.

+

Platzhalter für das Vorzeichen
Kann vor oder hinter der Zahl stehen.
Positiver Wert ergibt +
Negativer Wert ergibt -

-

Platzhalter für das Vorzeichen
Kann vor oder hinter der Zahl stehen.
Positiver Wert ergibt Leerzeichen
Negativer Wert ergibt -

'string' kann eine String-Konstante oder eine String-Variable sein.

PRINT USING

---------- Fortsetzung

Beispiele:

```
5 PRINT USING "####.#",1
```

Ausgabe:

```
   1.0
```

```
100 PRINT #11: USING "******",1;2;3;4,5
```

Ausgabe auf Datei:

```
*****1*****2*****3*****4                *****5
```

```
150 P1=1.13*99.80 :!Preis plus MWST
160 P2=3*P1         !Endbetrag
190 PRINT "Einzelpreis:","Endbetrag"
200 PRINT "  ";USING "*####.##";P1;" DM";
210 PRINT USING "*######.##",P2;" DM"
```

Ausgabe:

```
Einzelpreis:        Endbetrag
  *112.77 DM        ***338.32 DM
```

```
300 PRINT USING "##",100
```

Ausgabe:

??

RANDOM

Syntax:

RANDOM

Wirkung:

Nach einem RANDOM-Befehl ist der Anfangswert der RND-Funktion zufällig. Wird diese Anweisung nicht gegeben, so erscheint bei jedem Programmlauf dieselbe Folge von Zufallszahlen bei den RND-Funktionsaufrufen.

Beispiel:

```
10 RANDOM
20 A=RND(0)
```

READ

Syntax:

READ liste

Wirkung:

Liest im Programm gespeicherte Daten ein.

Wie INPUT, jedoch werden die Werte nicht von der Tastatur, sondern aus DATA-Befehlen eingelesen.

Der erste READ-Befehl liest beginnend bei dem ersten Wert des ersten DATA-Befehls im Programm. Danach werden die Werte aus den folgenden DATA-Befehlen gelesen.

Nach einem RESTORE-Befehl beginnt READ wieder mit dem ersten DATA-Befehl.

DATA-Befehle dürfen an beliebiger Stelle im Programm stehen (siehe DATA, RESTORE).

Beispiele:

```
100 DATA 1,"STRING"
110 READ X,A$,Y
120 DATA 2.0E6
```

Nach diesen Programmzeilen hat X den Wert 1, A$ den Wert "STRING" und Y den Wert 2000000.

RECORD

Syntax:

RECORD #n: x

'x' = math. Ausdruck im Bereich 0..65535 oder das Zeichen '*'
'n' = 10..13 (siehe OPEN)

Wirkung:

Setzt den Dateizeiger für die Datei mit der logischen Kanalnummer 'n' auf den Block 'x'. Nach diesem Befehl kann mit den Befehlen INPUT# und PRINT# weiter sequentiell von der Datei gelesen bzw. auf die Datei geschrieben werden. Es kann auch mit INPUT# (Dummy) auf ein beliebiges Zeichen innerhalb einer Datei positioniert werden.

Mit RECORD #n: * kann eine Datei ab Dateiende sequentiell weiter beschrieben werden (PRINT#).

Ein Block entspricht der Länge eines logischen Records auf dem Medium (128 Zeichen). Der Dateizeiger steht nach diesem Befehl auf dem x*128+1. Zeichen der Datei. Achtung: 'PRINT...;'-Ausgaben werden mitgezählt. Damit kann ebenfalls positioniert werden.

Wird beim Zugriff auf beliebige Blöcke einer Datei ihre Länge überschritten, so erfolgt eine Fehlermeldung (siehe 'ON ERROR').

Beispiele:

```
100 RECORD #11: 3
```

Setzt den Dateizeiger der Datei mit der logischen Einheit 11 auf den Block 3. Bei der nächsten Eingabe- oder Ausgabeanweisung auf diese logische Einheit wird ab dem 1.Zeichen dieses Blocks gelesen/-geschrieben.

```
120 RECORD #13: *
```

Setzt den Dateizeiger auf das 1. Zeichen nach dem Dateiende.

```
150 RECORD #10: X-3*A1
```

Setzt den Dateizeiger der Datei mit der logischen Nummer 10 auf den Block, der sich durch den Ausdruck X-3*A1 ergibt. Werte kleiner Null führen zu einer Fehlermeldung.

RESTORE

Syntax:

RESTORE

Wirkung:

Der nächste READ-Befehl liest wieder vom ersten DATA-Befehl, unabhängig von dessen Lage im Programm. Positionieren auf DATA-Anweisungen erfolgt durch wiederholtes 'Dummy-READ' (siehe READ,DATA).

Beispiel:

```
100 RESTORE
```

Beispiel für Dummy-Read:

```
100 RESTORE
200 READ A$,A$,A$
300 READ B$
400 DATA "String", "String"
500 DATA "String", "ABC"
```

Nach diesen Befehlen nimmt B$ den Wert "ABC" an.

RES,SET,INV

Syntax:

```
SET x,y
RES x,y
INV x,y
```

Wirkung:

Anweisung für Grafik-Modus.
Setzt, löscht oder invertiert den Punkt mit den Koordinaten x,y. 'x' und 'y' können Konstanten, Variable oder Ausdrücke sein.

Grafik-Treiber muß aktiviert und geöffnet sein (siehe Treiber).

Wertebereiche:

```
0 <= x <= 511 bzw. 1023 bei Kontron PSI980 H
0 <= y <= 255 bzw.  399 bei Kontron PSI980 H
```

Beispiele:

```
150 SET 100,100
```

Setzt den Punkt 100,100

```
200 RES X1,Y1
```

Löscht den Punkt X1,Y1

```
300 INV X-X0,Y-Y0
```

Invertiert den Punkt X-X0,Y-Y0.

STOP

Syntax:

STOP

Wirkung:

Unterbricht das Programm. BASIC kehrt in den Kommando-Modus zurück.

Mit dieser Anweisung kann das Programm an beliebiger Stelle unterbrochen werden. Es können im Kommando-Modus Variablen ausgedruckt (PRINT) und verändert werden. Mit der Anweisung KOS P kann z.B. in den acht Bildseiten beliebig hin- und hergeblättert werden, um z.B. die Zeilennummern von der TRACE-Anweisung zu überprüfen.

Es werden jedoch alle Dateien geschlossen.

Das Programm kann mit der CONT-Anweisung fortgesetzt (Einschränkungen der Datei-Ein-/Ausgabe beachten), oder mit RUN neu gestartet werden.

Nach Erreichen eines STOP-Befehls erfolgt die Meldung:

STOP in Zeile ...

SET,RES,INV

Syntax:

```
SET x,y
RES x,y
INV x,y
```

Wirkung:

Anweisung für Grafik-Modus.

Setzt, löscht oder invertiert den Punkt mit den Koordinaten x,y.
'x' und 'y' können Konstanten, Variable oder Ausdrücke sein.

Grafik-Treiber muß aktiviert und geöffnet sein (siehe Treiber)

Wertebereiche:

```
0 <= x <= 511 bzw. 1023 bei Kontron PSI980 H
0 <= y <= 255 bzw.  399 bei Kontron PSI980 H
```

Beispiele:

```
150 SET 100,100
```

Setzt den Punkt 100,100

```
200 RES X1,Y1
```

Löscht den Punkt X1,Y1

```
300 INV X-X0,Y-Y0
```

Invertiert den Punkt X-X0,Y-Y0.

STRING

Syntax:

STRING string,x,y

STRING string,x,y,f,r

Wirkung:

Ausgabe von ASCII-Zeichen im Grafik-Modus.

Grafik-Treiber muß aktiviert und geöffnet sein (siehe Treiber).

Schreibt den String 'string' um den Faktor 'f' vergrößert in der Richtung 'r' auf den Bildschirm (Grafik-Modus).
Die linke untere Ecke des ersten Buchstabens hat die Koordinaten x,y.
Möglich sind Großbuchstaben und Ziffern.

'x','y','f' und 'r' können Ausdrücke sein.
'string' ist eine String-Variable.

Wertebereiche der Parameter:

0 <= x <= 511 bzw. 1023 bei Kontron PSI980 H
0 <= y <= 255 bzw. 399 bei Kontron PSI980 H
1 <= f <= 15
(jeweils INTEGER)
r= (0,90,180,270) <Grad>

Bei Überschreitung dieser Wertebereiche erfolgt eine Fehlermeldung.

'r' und 'f' können weggelassen werden.
Voreinstellung: f=1 r=0

Strings, die die Bildschirmgrenzen überschreiten, werden nicht unterdrückt.

Das Löschen von Strings auf dem Bildschirm im Grafikmodus erfolgt durch 'Rechteck löschen', siehe PLOT-Anweisung.

STOP

---------- Fortsetzung

Beispiele:

```
200 IF B>=0 THEN 220
210 STOP
220 X=SQR(B)
```

Ist B negativ, so wird das Programm durch den STOP-Befehl unterbrochen.

```
300 STOP

310 X=A+B*(C1-C2)
```

Vor der Berechnung des Ausdrucks in Zeile 310 ist ein STOP-Befehl eingefügt worden. Nach der Meldung

```
STOP in Zeile 300
```

kann nun mit

```
PRINT A,B,C1,C2
```

der Wert der einzelnen Operanden ausgegeben und wenn gewünscht auch verändert werden. Z.B.:

```
B=5
```

Dadurch wird B der Wert 5 zugewiesen.

Anschließend kann das Programm mit

```
CONT 310
```

mit den geänderten Werten fortgesetzt werden.

TRACE, TRACEOFF

Syntax:

TRACE

TRACEOFF

TRACE — Vor jeder Ausführung einer neuen Zeile bei RUN wird die Nummer der gerade bearbeiteten Zeile in der Form <zeilennr.> auf dem Bildschirm ausgegeben.

Der Grafik-Modus wird in den alphanumerischen Modus rückgesetzt.

TRACEOFF — unterbindet wieder die Ausgabe von TRACE.
TRACE und TRACEOFF können wie alle Anweisungen auch im Kommando-Modus verwendet werden.

Beispiel:

```
100 TRACE
110 GOSUB 200          ) von diesem Teil des Programms
120 TRACEOFF           ) wird der Ablauf überwacht.
```

STRING

---------- Fortsetzung

Beispiele:

```
10 STRING C$,0,0
```

Schreibt den String C$ ab dem Punkt 0,0 auf den Bildschirm.

```
10 X=10:Y=200:F=2:R=270
20 STRING D$,X,Y,F,R
```

Schreibt den String D$ ab dem Punkt 10,200 um den Faktor 2 vergrössert und um 270 Grad gedreht (von oben nach unten).

```
30 GET A$ : IF LEN(A$)=0 THEN 30
40 STRING A$,200,200
```

Liest von der Tastatur ein Zeichen ein und weist es der Stringvariablen A$ zu. Diese wird ab dem Punkt 200,200 geschrieben. Auf diese Weise ist es möglich, im Grafik-Modus Eingaben von der Tastatur zu machen und diese auszugeben.

Arithmetische Operatoren

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
^	Potenzierung
()	Klammerung (bis 8-fach)

Ausdrücke mit verschiedenen Operatoren werden entsprechend den Regeln der Algebra aufgelöst.

Durch Klammerung kann eine beliebige Auflösungsreihenfolge erreicht werden.

Beispiele:

```
100 Y=C1*X^3+C2*X*X+(C1-C0)*X+C0

110 Y=((X-3)*((X+5)-(X+1))-7)*(X+2)
```

Übersicht: Operatoren

Operatoren bewirken in Anweisungen Wertzuweisungen zu Variablen.

Aus Operatoren und Funktionsoperatoren werden Ausdrücke gebildet.

In Ausdrücken können Operatoren der verschiedenen Arten gemischt verwendet werden.

Arten von Operatoren:

- Arithmetische Operatoren
- Vergleichsoperatoren
- Logische Operatoren
- Funktionsoperatoren
- String-Operatoren

Priorität von Operatoren:

Ausdrücke werden mit folgender Reihenfolge der Priorität der Operatoren ausgeführt:

1. Klammerung
2. Funktionsoperatoren
3. Potenzierung
4. Multiplikation und Division
5. Addition und Subtraktion
6. Vorzeichen (z.B. -SIN(X))
7. Vergleichsoperatoren
8. NOT
9. AND
10. OR
11. XOR

Logische Operatoren

NOT	Negation
AND	Und-Verknüpfung
OR	Oder-Verknüpfung
XOR	Exklusiv-Oder-Verknüpfung

Wirkung der logischen Operatoren:

Werden die logischen Werte x,y (0="falsch";1="wahr") mit den Operatoren verknüpft, so ergibt sich:

x=	0	0	1	1
y=	0	1	0	1
NOT x	1	1	0	0
x AND y	0	0	0	1
x OR y	0	1	1	1
x XOR y	0	1	1	0

Die Werte der zu verknüpfenden Variablen werden in 16-Bit Werte gewandelt (-32768 bis +32767) und bitweise verglichen. Wird dieser Zahlenbereich überschritten, so erfolgt eine Fehlermeldung.

Beispiele:

```
100 A=4 OR 8
```

Ergebnis: A=12

```
110 B=127 AND 3
```

Ergebnis: B=3

```
120 X=5+(7 AND B*(C OR NOT D))
```

Auswertung dieses Ausdrucks siehe Priorität von Operationen.

Vergleichsoperatoren

=	gleich
<>	ungleich
<=	kleiner oder gleich (auch =<)
>=	größer oder gleich (auch =>)
>	größer
<	kleiner

Vergleichsoperatoren können gemischt mit arithmetischen und Funktionsoperatoren eingesetzt werden.

Vergleichsoperatoren können auch auf Strings angewendet werden. Diese werden zeichenweise verglichen.

Beispiele:

```
50 IF X>Y-3 THEN 100

60 IF (A*A)^B-5<C OR A=0 THEN 100

70 IF D<>0 THEN 100

80 IF A$="ENDE" THEN 999
```

String-Operatoren

X - numerische Variable, Ausdruck oder Konstante
N$ - String-Variable, -Ausdruck oder -Konstante

CHR$(X)	Umwandlung einer dezimalen Zahl in ihr ASCII-Zeichen entsprechend dem ASCII-Code. (auch für Steuerzeichen: siehe Tabelle im Anhang)
ASC(N$)	Umwandlung des 1. Zeichens von N$ in eine Zahl (0..255)
VAL(N$)	Extrahiert die erste in einem String enthaltene Zahl Achtung: 'A-B' und '- 123' führt zu Fehlermeldungen!
STR$(X)	Wandelt einen numerischen Wert in einen String, Darstellung im Exponentialformat!
LEN(N$)	aktuelle Länge eines Strings
+	Kettung von Strings
LEFT$ (N$,X)	ergibt die X ersten Zeichen eines Strings
RIGHT$(N$,X)	ergibt die X letzten Zeichen eines Strings
MID$(N$,X1,X2)	bei dieser Funktion werden von der Stringvariablen N$ X1 Zeichen ignoriert und die folgenden X2 Zeichen als Ergebnis übernommen.

Das Verketten von Stringfunktionen mit der 'STR$(..)'-Funktion sollte vermieden werden.

Beispiel: A$=STR$(I):B$=LEFT$(A$,3) anstelle von B$=LEFT$(STR$(I),3)

Funktions-Operatoren

EXP(X)	Exponentialfunktion zur Basis e
LOG(X)	natürlicher Logarithmus (Basis e)
SIN(X)	Sinus (X im Bogenmaß)
COS(X)	Cosinus
TAN(X)	Tangens
ATN(X)	Arcus Tangens (Ergebnis im Bogenmaß)
SQR(X)	Wurzel (Quadratwurzel) Das Radizieren einer negativen Zahl mit 'SQR(..)' erzeugt die Quadratwurzel des Absolutbetrages der Zahl.
ABS(X)	Absolutwert
INT(X)	Integer: ergibt Ganzzahl ohne Kommastellen INT(X)<=X
SGN(X)	Signum ergibt 1 wenn X>0 -1 wenn X<0 0 wenn X=0
RND(1)	Zufallszahl zwischen 0 und 1 ((1) ist Platzhalter)
PEEK(adr)	liest Inhalt der Speicherzelle mit der dezimalen Adresse 'adr'
IN(adr)	liest Wert vom Port mit der dezimalen Adresse 'adr'
FRE(1)	gibt die Anzahl der restlichen freien Speicherplätze innerhalb des BASIC-Anwenderspeichers (1) ist Platzhaltergröße
POINT(x,y)	prüft, ob der Punkt mit den Koordinaten x und y gesetzt ist. Das Ergebnis ist 0 oder -1, wenn der Punkt rückgesetzt oder gesetzt ist.
ERM(1)	gibt die Codenummer der letzten Fehlermeldung an. (1) ist Platzhaltergröße.

Beispiele:

```
100 Y=EXP(Y)

110 X=R*COS(W)

120 Z1=INT(Z)

130 IF POINT (X,Y) THEN 200 ELSE 300
```

Übersicht: Treiber

Um periphere Geräte wie z.B. Drucker von BASIC aus bedienen zu können, ist ein Programm zur Ansteuerung des speziellen Gerätes erforderlich. Solche Programme nennt man Treiber.

Treiber stellen die Verbindung her zwischen dem logischen Ein-/Ausgabekanal, auf die ein Benutzerprogramm zugreift, und der Hardwareschnittstelle, an der ein spezifisches Ein-/Ausgabegerät betrieben wird.

Treiber werden vom Betriebssystem KOS verwaltet, deswegen ist vor dem Start eines BASIC-Programms die Treiberaktivierung durch das Dienstprogramm IODC durchzuführen.

Erfolgt auch die Zuweisung des Ein- oder Ausgabekanals über das Dienstprogramm IODC, dann bleibt das Benutzerprogramm völlig unabhängig vom verwendeten Peripheriegerät, solange sich die Art des Gerätes nicht grundsätzlich ändert; so können z.B. Programme unterschiedliche Arten von Druckern über eine logische Schnittstelle verwenden.

Grundsätzlich ist die Kanalzuordnung beliebig, zu beachten ist nur, daß die Kanäle O-0..O-3, I-0..I-1 und O-9 für spezielle Aufgaben reserviert sind.

String-Operatoren

Beispiele:

```
200 PRINT CHR$(12)
```

Gibt das ASCII-Zeichen Formfeed (0C Hex) aus.

```
210 X=ASC(X$)
```

Wandelt das erste Zeichen der Stringvariablen X$ in eine Zahl (ASCII).

```
220 L=LEN(A$)
```

Weist L die Anzahl der Zeichen in A$ zu.

```
230 T$="Heute ist der 24.Dezember"
240 T=VAL(T$)
```

Extrahiert aus dem String T$ die Zahl 24 und weist sie T zu.

```
250 Y=1
260 Y$=STR$(Y)
```

Wandelt den Wert von Y in einen String.
Ergebnis: "1.000000000000E+000"

```
270 LET K$=A$+X$+Y$+"Salat"
```

Die einzelnen Strings werden im String K$ verkettet.

```
280 A$="ABCDEFGH"
290 LET T$=MID$(A$,3,2)
300 U$ = LEFT $(A$,2)
400 V$ = RIGHT$ (A$,2)
```

T$ erhält den Wert "DE", U$ den Wert "AB", V$ den Wert "GH".

Grafik-Treiber GRAP

Sämtliche Grafik-Befehle benutzen den Grafik-Treiber und können nur ausgeführt werden, wenn dieser aktiviert und geöffnet ist.

Der Grafik-Treiber liegt als ".OBJ"-Datei vor und wird vom IODC-Kommando (s. KOS-Kommandos) auf die höchstmögliche Anfangsadresse geladen.

Aufruf:

```
IODC $GRAP=ACTIVE      (KOS 4.x)
```

bzw.

```
RLOAD GRAPTASK
IODC $GRAP=ACTIVE      (KOS5.x)
```

Im Betriebssystem KOS 6.x sind die Graphikroutinen bereits im Beriebssystem integriert. Die Treiber GRAPTASK und $GRAP werden hier nicht benötigt.

In BASIC ist vor der Anwendung von Grafik-Befehlen die grafische Betriebsart des Bildschirms zu initialisieren. Dies geschieht durch Öffnen des Grafik-Treibers. Dabei wird der gesamte Bildspeicherinhalt gelöscht.

Beispiel: 100 OPEN #9:"$GRAP"

Der Grafik-Treiber kann nur auf die logische Einheit 9 gelegt werden!

Während sich der Bildschirm in der grafischen Betriebsart befindet, darf keine alphanumerische Ausgabe auf den Bildschirm gemacht werden , erlaubt sind: GET, STRING. In diesem Fall schaltet der Bildschirm automatisch auf die alphanumerische Betriebsart zurück. Dasselbe gilt bei Fehlermeldungen vom Betriebssystem KOS oder von BASIC.

Die grafische Betriebsart wird verlassen durch Schließen des Grafik-Treibers.

Aufruf:

```
200 CLOSE #9:
```

Dabei wird der gesamte Bildspeicher gelöscht.

Drucker-Treiber

Um einen Drucker von BASIC aus ansprechen zu können, sind folgende Schritte erforderlich:

1. Aktivieren

Der Treiber wird dabei von der Diskette in den Speicher geladen und initialisiert. Dies geschieht im Betriebsystem am Beispiel des Treibers für den O-Kanal des Z80A-SIO-Bausteins SIO-1 mit dem IODC-Kommando:

```
IODC $SIOA=ACTIVE
```

2. Kanal zuordnen

Dem Treiber wird nun ein I/O-Kanal des Betriebssystems zugeordnet.

```
IODC O-4=$SIOA
```

Dies kann auch in BASIC entweder im Kommando- oder im Programm-Modus geschehen:

```
OPEN #4: "$SIOA"
```

3. Auf Kanal ausgeben

Nach dem Aufruf von BASIC kann nun durch einen PRINT-Befehl über diese Kanalnummer auf den Drucker ausgegeben werden:

```
100 PRINT #4: "Ausgabe auf den Drucker"
110 PRINT #4: USING "###.##",P1,P2
```

Fehlermeldungen

---------- Fortsetzung

91..95	<Nicht definierter Fehlercode>
96	<KOS E/A-Fehler>
97	<Nicht definierter Fehlercode>
98	<Gerät nicht bereit>
99	<Treiber nicht aktiviert>
100	<Datenübertragungsfehler>
101	<Datei-Suchfehler>
102	<Medium schreibgeschützt>
103	<falsche Datenrichtung>
104	<Medium voll>
105	<Speicherbelegungskonflikt>
106	<Zu viele Dateien sind geöffnet>
107	<Directory-Format Fehler>
108	<Datei schreibgeschützt>
109	<Datei löschgeschützt>
110	<Datei inkonsistent>
111	<Nicht definierter Fehlercode>
112	<Fehler (für E/A-Treiber)>

Fehlermeldungen

Code (dezimal)	Fehlertext
58	<Nicht definierter Fehlercode>
59	<Unzulässige logische Einheit>
60	<Datei nicht vorhanden>
61	<Ende der Datei>
62	<Zahl ist nicht zulässig>
63	<Zeilenüberschreitung>
64	<Fehler nicht rücksetzbar>
65	<Speicherüberlauf>
66	<Schleifenschachtelung>
67	<FOR...NEXT Syntax>
68	<Variable doppelt in DIM>
69	<zu viele Variablen>
70	<Index zu groß>
71	<GOSUB ohne RETURN oder FOR ohne NEXT>
72	<Zeilenzahl zur groß>
73	<Unbekanntes Sprungziel>
74	<FOR...NEXT Schleifenüberschneidung>
75	<Klammerfehler>
76	<Zeilenanfang nicht identifizierbar>
77	<Speicherüberlauf>
78	<Variable nicht bekannt>
79	<math. Zahlenüberlauf>
80	<Index größer als Feld>
81	<Division durch 0 oder SQR/LOG mit neg. Zahl>
82	<RETURN ohne GOSUB oder NEXT ohne FOR>
83	<Syntaxfehler>
84	<Stackfehler nach CALL>
85	<Verknüpfung nicht kompatibler Typen>
86	<Befehl oder Operator unbekannt>
87	<Keine Daten mehr in DATA/Ende der Datei>
88	<Überlauf bei 16-bit Wandlung>
89	<Zeile zu lang>
90	<Klammer nach Funktion fehlt>

Beispiele

---------- Fortsetzung

```
10 PRINT CHR$(12)
20 PRINT  TAB (20);" 2)Ändern einer Datei "
30 PRINT : PRINT : PRINT
40 INPUT "Name der Datei ";N$
50 OPEN #10:N$
60 PRINT : INPUT "Welcher Block (9999=Schluß) ";N
70 IF N=9999 THEN 280
80 IF N<1 OR N>1152 THEN 60
90 ON ERROR GOTO 310
100 RECORD #10:N-1: REM Auf Record positionieren
110 PRINT : INPUT "Welcher Satz";M: REM Auf Satz positionieren
120 FOR I=1 TO M: INPUT #10:T$: NEXT I
130 L=LEN(T$)
140 PRINT : PRINT "Alter Stand": PRINT : PRINT T$: PRINT
150 INPUT "Neuer Stand";T$
160 IF LEN(T$)=0 THEN 60
170 RECORD #10:N-1: REM Auf Record positionieren
180 IF M=1 THEN 210
190 FOR I=1 TO M-1: INPUT #10:X$: REM Auf Satz pos.
200 NEXT I
210 PRINT #10:T$;
220 L1=LEN(T$)
230 FOR I=L1 TO L-2
240 PRINT #10:" ";
250 NEXT I
260 PRINT #10:" "
270 GOTO 60
280 CLOSE #10:
290 PRINT : PRINT "Datei ";N$;" geschlossen"
300 END
310 IF ERM(0)<>61 THEN 350
320 PRINT : PRINT "Datei existiert nicht"
330 PRINT : PRINT : PRINT
340 GOTO 60
350 PRINT : PRINT : PRINT
360 PRINT "     Fehler:   ";ERM(0): GOTO 300
```

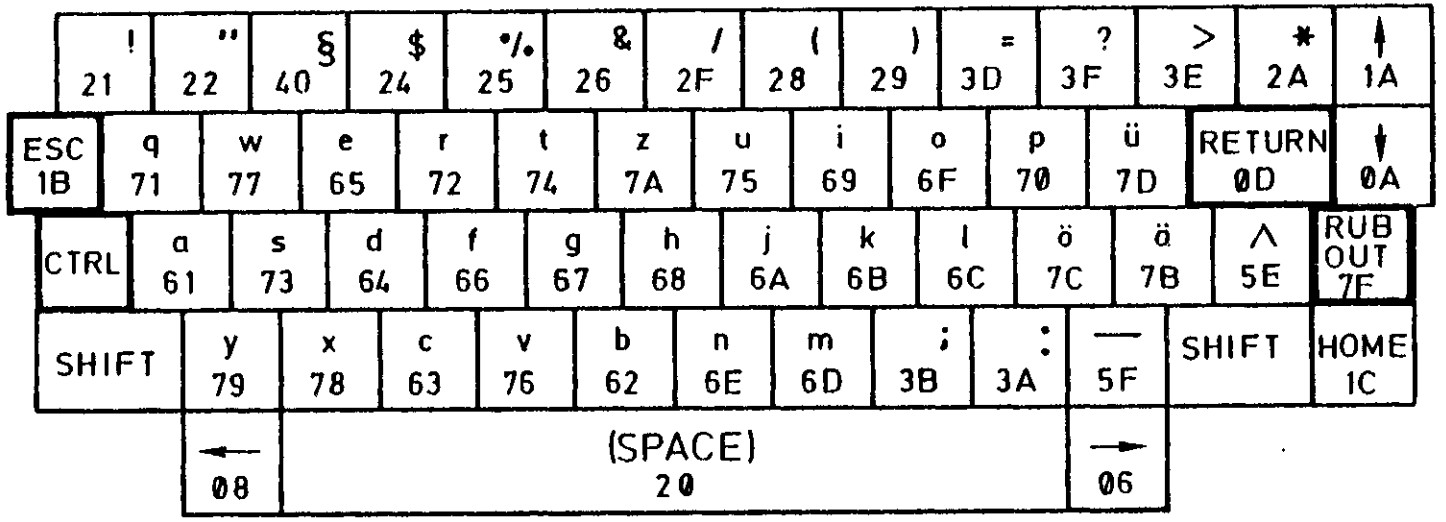
Beispiele

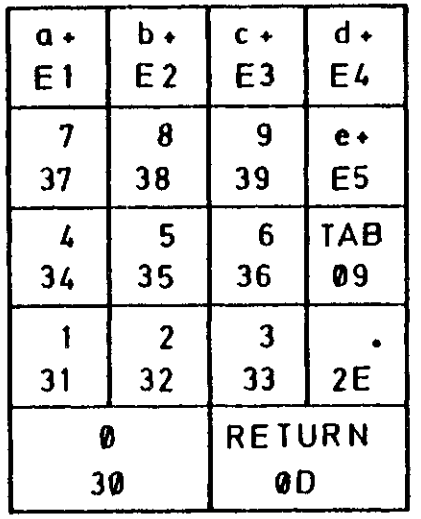
```
10 PRINT CHR$(12)
20 PRINT : PRINT  TAB (20);"  1)Anlegen einer Datei "
30 PRINT : PRINT
40 PRINT "Datei = N Blöcke zu je M Sätzen": PRINT : PRINT
50 PRINT "Ein Block = 128 Zeichen; ein Satz = 12 bis 128/M Zeichen"
60 PRINT : INPUT "Name der neuen Datei ";N$
70 PRINT : INPUT "Wieviele Blöcke ";N
80 PRINT : INPUT "Wieviele Sätze ";M
90 PRINT : INPUT "Satzlänge ";L
100 IF L<12 THEN 90
110 IF M*L>125 THEN 70
120 PRINT : PRINT
130 OPEN #10:N$
140 FOR I=1 TO N
150 FOR J=1 TO M
160 PRINT #10:"(";
170 PRINT #10:USING"####",I;
180 PRINT #10:".";
190 PRINT #10:USING"##",J;: PRINT #10:")";
200 FOR K=1 TO L-11: PRINT #10:" ";: NEXT K: PRINT #10:" "
210 NEXT J
220 FOR K=1 TO 126-M*L: PRINT #10:" ";
230 NEXT K: PRINT #10:" "
240 PRINT I;".Block"
250 NEXT I
260 CLOSE #10:
270 PRINT : PRINT "Datei ";N$;" geschlossen"
```

Stichwortverzeichnis

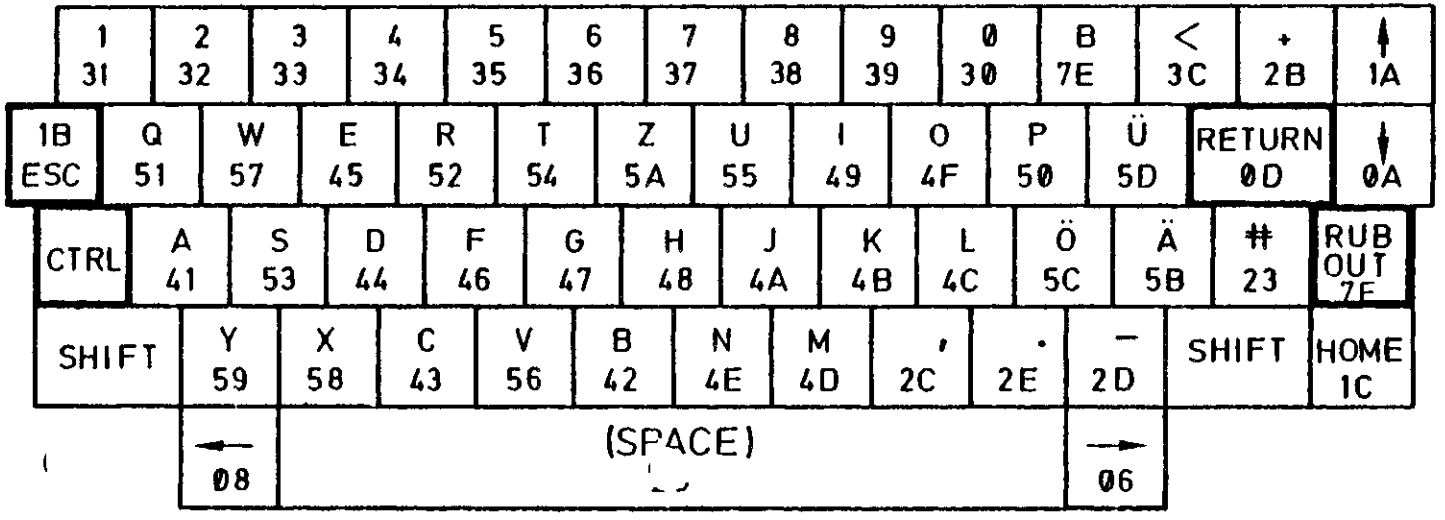
A N H A N G

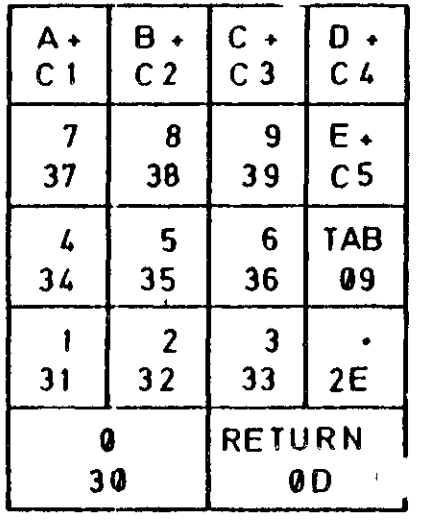
SHIFT



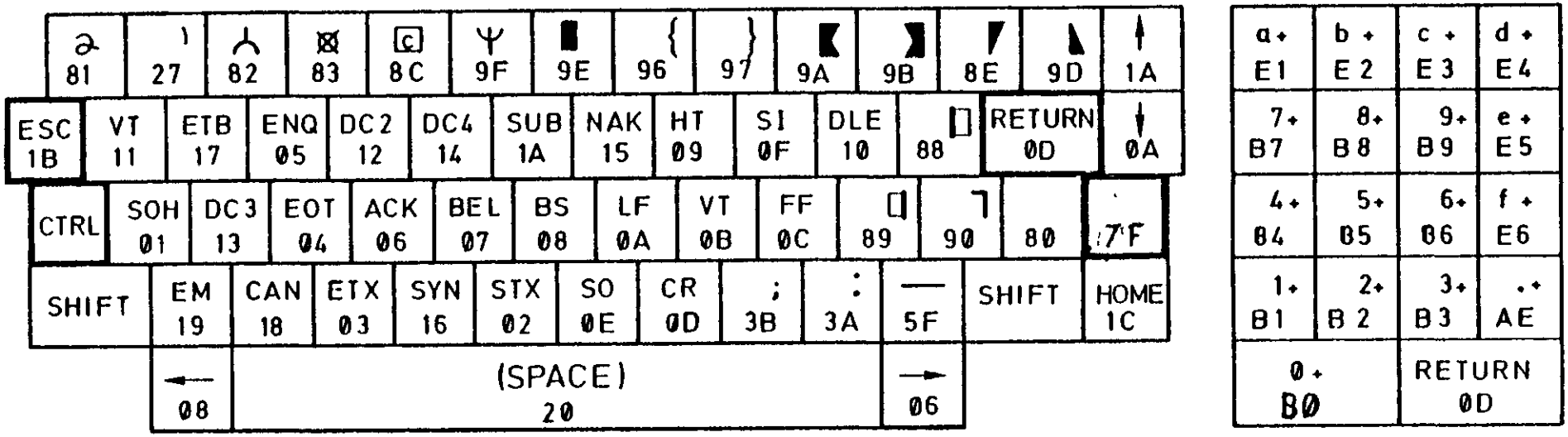


UNSHIFT

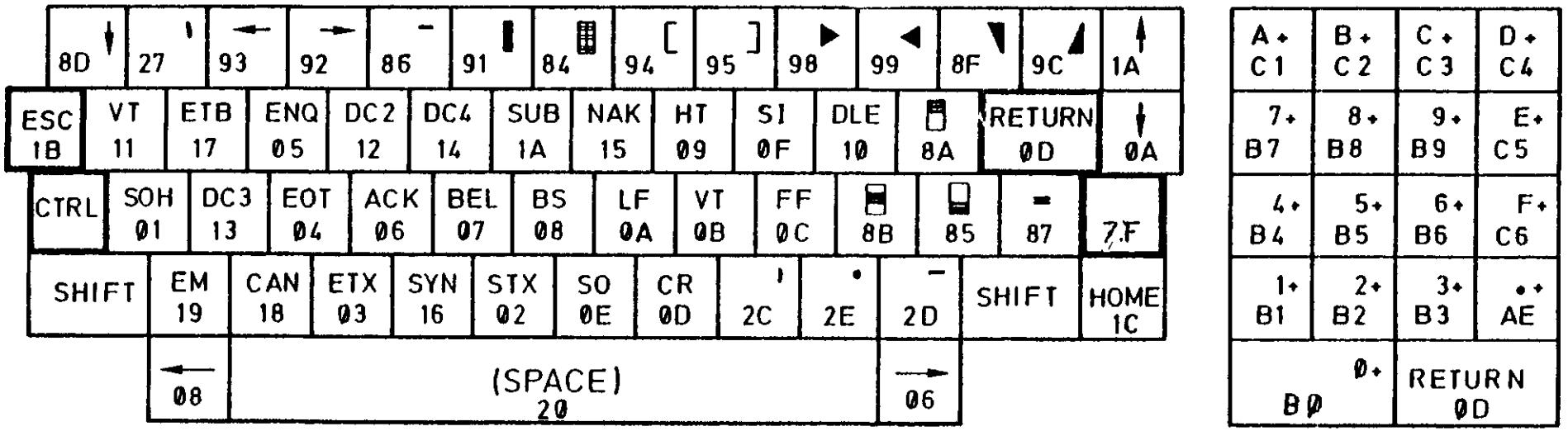




SHIFT-CONTROL



UNSHIFT-CONTROL



Kontron "Ergo Line"-Tastatur

Die "Ergo Line"-Tastatur ist Bestandteil der Systeme Kontron PSI908/9C/98/980. Diese Tastatur umfaßt neben dem alphanumerischen Feld nach DIN 2137 einen Cursor-Block, einen 10er-Block und eine Reihe von Funktionstasten. Die Funktionen der einzelnen Felder werden im folgenden beschrieben.

1. Alphanumerisches Feld

Das alphanumerische Feld ist von seiner Grundeinstellung her in Kleinschreibung. Unter dem Kontron Betriebssystem KOS kann diese Grundeinstellung umgeschaltet werden durch das KOS-interne Kommande "C". Dieses Kommando kann direkt in die Initialisierungsdatei KOS.INI eingebunden werden. Durch die Voreinstellung "Kleinschreibung" ist die größte Nähe zur Schreibmaschinentastatur gegeben.

Das alphanumerische Feld umfaßt neben den Buchstaben, Ziffern und deutschen Sonderzeichen folgende Tasten:

ESC	Escapetaste
RUBOUT	Löschtaste, löscht eingegebene Zeile
TAB	Tabulation
LF	Linefeed
CTRL	Controltaste, bewirkt, daß bei niedergehaltener Controltaste Controlzeichen von den Alphatasten abgeschickt werden.
LOCK	Schaltet fest um auf Großschreibung
SHIFT	Schaltet um auf Groß-/Kleinschreibung, löst LOCK-Funktion
CR	Return Taste, schließt Eingabe ab

2. Cursorfeld

Rechts von der alphanumerischen Tastatur befindet sich das Cursorfeld. Links oben die Taste DIN schaltet um auf den nationalen Modus. Die Taste DIN zeigt diesen Zustand durch den roten Leuchtpunkt an. In der Betriebsart "DIN" sind die deutschen Sonderzeichen der Tastatur freigegeben und die internationalen Zeichen (eckige Klammer etc.) gesperrt, bei der Betriebsart "International" (Lampe DIN dunkel) sind entsprechend die deutschen Sonderzeichen gesperrt.

Die Eingabe von gesperrten Zeichen gibt ein akustisches Fehlersignal.

Bei der Version 2.1 sendet die DIN-Taste einen Code aus, der vom Betriebssystem KOS6.04/6.05 zur Umschaltung ausgenutzt wird. Diese Tastaturen benötigen KOS6.04/6.05.

Belegung der Tasten DEL, INS, ER CHAR: siehe Belegungspläne der Versionen 1.1/2.0 bzw. 2.1. Mit den Cursorsteuerungstasten HOME, Pfeil links, Pfeil rechts, Pfeil oben, Pfeil unten kann der Cursor über den Bildschirm bewegt werden, Voraussetzung ist jedoch, daß das gerade ablaufende Programm diese Eingaben berücksichtigt.

ASCII-Code-Tabelle

Deutsche Referenz-Version (mit Umlauten)

b_6	b_5	b_4	b_3	b_2	b_1	Zeile \ Spalte	0	1	2	3	4	5	6	7
0	0	0	0	0		0	NUL	TC_7 (DLE)	SP	0	§	P		p
0	0	0	1	1		1	TC_1 (SOH)	DC_1	!	1	A	Q	a	q
0	0	1	0	2		2	TC_2 (STX)	DC_2	"	2	B	R	b	r
0	0	1	1	3		3	TC_3 (ETX)	DC_3	#	3	C	S	c	s
0	1	0	0	4		4	TC_4 (EOT)	DC_4	$	4	D	T	d	t
0	1	0	1	5		5	TC_5 (ENQ)	TC_8 (NAK)	%	5	E	U	e	u
0	1	1	0	6		6	TC_6 (ACK)	TC_9 (SYN)	&	6	F	V	f	v
0	1	1	1	7		7	BEL	TC_{10} (ETB)	'	7	G	W	g	w
1	0	0	0	8		8	FE_0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9		9	FE_1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	10		10	FE_2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	11		11	FE_3 (VT)	ESC	+	;	K	Ä	k	ä
1	1	0	0	12		12	FE_4 (FF)	IS_4 (FS)	,	<	L	Ö	l	ö
1	1	0	1	13		13	FE_5 (CR)	IS_3 (GS)	-	=	M	Ü	m	ü
1	1	1	0	14		14	SO	IS_2 (RS)	.	>	N	^	n	ß
1	1	1	1	15		15	SI	IS_1 (US)	/	?	O	_	o	DEL

HEXADECIMAL COLUMNS												
6		5		4		3		2		1		
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	
0	0	0	0	0	0	0	0	0	0	0	0	
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1	
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2	
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3	
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4	
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5	
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6	
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7	
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8	
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9	
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10	
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11	
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12	
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13	
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14	
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15	
0123		4567		0123		4567		0123		4567		
BYTE				BYTE				BYTE				

POWERS OF 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$

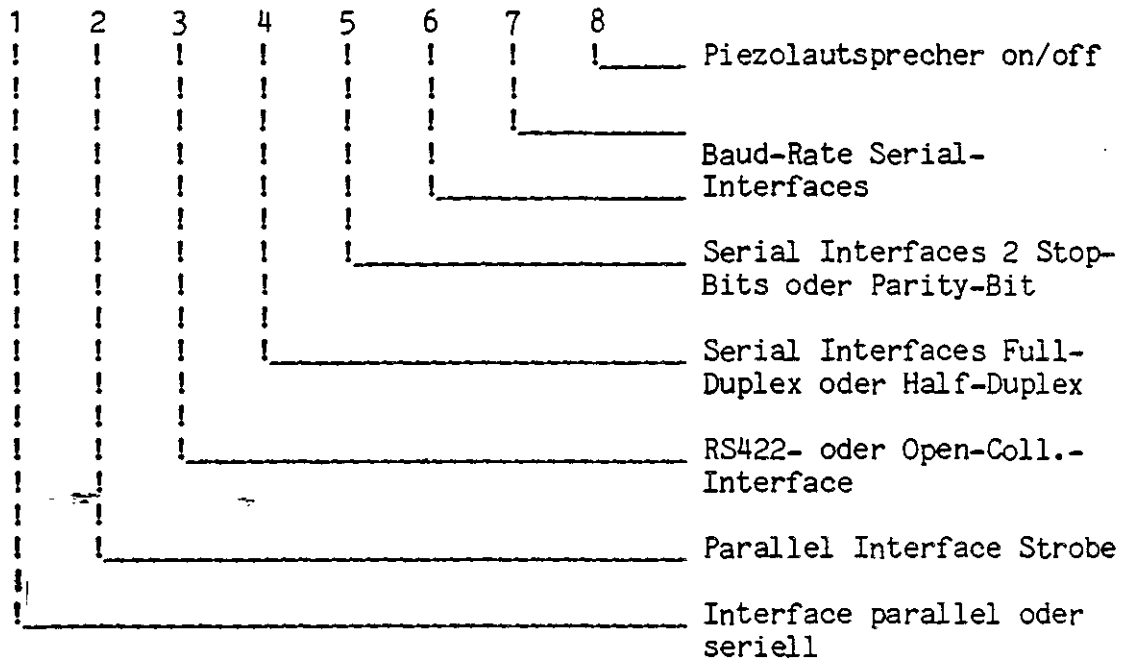
POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

5. Einstellbare Funktionen

Nach Lösen der 4 Schrauben der Bodenplattenbefestigung wird ein DIL-Schalter zugänglich, der 8 Schalter umfaßt. Damit können folgende Parameter eingestellt werden:

Version 1.1
Switch SW 1:



(open = "H")

SW1.1 = "L" Serial Interface
"H" Parallel Interface

SW1.2 = "L" Keyboard-Strobe KBSTRB Active-Low (Parallel-
"H" Keyboard-Strobe KBSTRB Active-High Interface)

SW1.3 = "L" RS-422-Interface selected (Serial-
"H" Open-Collector-Interface selected Interface)

SW1.4 = "L" Serial Interface Full-Duplex
"H" Serial Interface Half-Duplex

SW1.5 = "L" Serial Interface 2 Stop-Bits, no Parity-Bits
"H" 1 Parity-Bit, 1 Stop-Bit

SW1.6	SW1.7	Serial Interface
"L"	"L"	1200 Baud
"L"	"H"	2400 Baud
"H"	"L"	4800 Baud
"H"	"H"	9600 Baud

SW1.8 = "L" Piezolautsprecher Rückmeldung bei jedem Tastendruck und Fehlermeldungen
"H" Piezolautsprecher nur bei Fehlermeldungen aktiv.

Diese Tastatur wird standardmäßig für den seriellen Anschluß mit 9600 Baud ausgeliefert. Die Betriebssoftware der Kontron PSI90/900-Systeme ist für diese Betriebsart ausgelegt (ab KOS 6.03).

3. 10er-Block

Rechts außen befindet sich der Zehnerblock zur schnellen Eingabe von numerischen Daten. Dieser besteht aus den Ziffern von 0 bis 9, im Dezimalpunkt und der RETURN-Taste (CR), die wiederum die Eingabe abschließt. Die Taste SP erzeugt einen Leerschritt. Mit der Taste CLEAR wird die zuletzt eingegebene Zahl gelöscht, sie wirkt genauso wie die Taste RUBOUT. Die vier Tasten für die Grundrechnungsarten senden die zugehörigen Zeichen aus, sie sind in diesem Sinn parallel geschaltet zu den entsprechenden Tasten aus dem alphanumerischen Feld. Die Auswirkung dieser Zeichen hängt vom laufenden Programm ab.

Die Tasten STOP und BREAK senden die Codes 1BH (entspricht der ESC-Taste) bzw. FFH. Die Auswertung dieser Codes ist ebenfalls programmabhängig.

4. Funktionstasten

Die Funktionstasten F1 bis F13 oberhalb des alphanumerischen Feldes sind in 3 Ebenen belegt. Nach dem Einschalten der Tastatur zeigt eine rote Lampe an, daß der unterste Belegungsstreifen aktiv ist. Die ausgesendeten Codes sind von Programmen auswertbar, sie entsprechen den Codierungen 80H bis 8CH.

Durch die F-Taste wird jeweils um eine Ebene weitergeschaltet. Die jeweils dazugehörige Lampe leuchtet auf.

In der mittleren Ebene ist die Tastenbelegung wieder von Programmen auswertbar, sie senden die Codes 90H bis 9CH aus.

In der obersten Ebene wird eine Belegung mit Control-Zeichen aktiv, dies ist z.B. geeignet für Programme wie WordStar, welche diese Ebene belegen.

Rechts außen im Funktionstastenstreifen befindet sich die Taste CAPS. Diese Taste schaltet das alphanumerische Feld um auf Großschreibung. Die Tasten SHIFT und LOCK sind ohne Wirkung auf die Buchstabentasten, sie erlauben jedoch das Umschalten zwischen Ziffern und Sonderzeichen. Die Taste CAPS zeigt durch eine eingebaute Lampe an, das dieser Zustand eingestellt ist. Durch nochmaliges Drücken der Taste CAPS wird dieser Zustand wieder ausgeschaltet.

SW1.6 : nur bei seriellem Interface
"L" 2 Stop-Bits
"H" 1 Parity-Bit/1 Stop-Bit

SW1.6	SW1.7	nur bei seriellem Interface
"L"	"L"	9600 Baud
"L"	"H"	4800 Baud
"H"	"L"	2400 Baud
"H"	"H"	1200 Baud

Die Tastatur wird standardmäßig für die serielle Schnittstelle RS422 ausgeliefert. Dabei sind alle Schalter geschlossen (SW1.1...SW1.8 ="L"):
RS422-Interface mit 9600 Baud, 2 Stop-Bits

Zu beachten ist bei allen Softwareversionen, daß bei Auswahl einer anderen Schnittstelle nicht nur der DIL-Schalter SW1 umgeschaltet, sondern auch ein anderes Interface-Kabel in die Tastatur eingelötet werden muß!

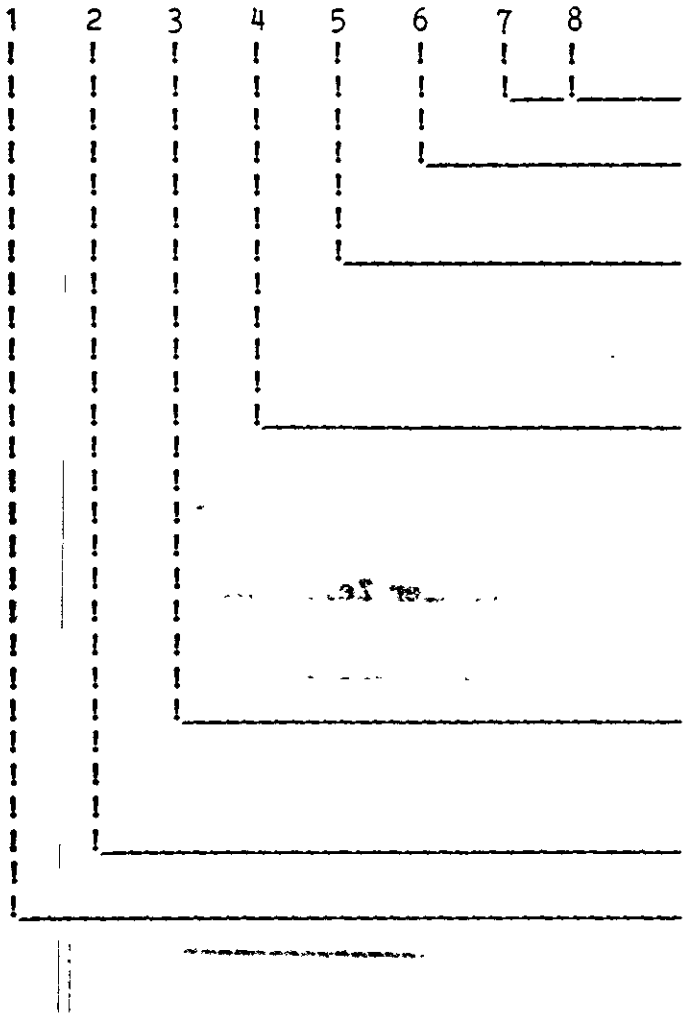
6. Umschaltung Deutscher/Internationaler Zeichensatz

Die Umschaltung erfolgt mit Taste "DIN" (KG16):

NATIONAl (deutsch)	(Code HEX)	INTERNATIONAL (ASCII)
Ä	5B	
ä	7B	
Ü	5D	
ü	7D	
Ö	5C	
ö	7C	
§	40	
ß	7E	

Version 2.0/2.1

Die DIL-Schalterbelegung wurde gegenüber Version 1.1 geändert, da einige Funktionen neu hinzugekommen sind:



Einstellung der Schalter

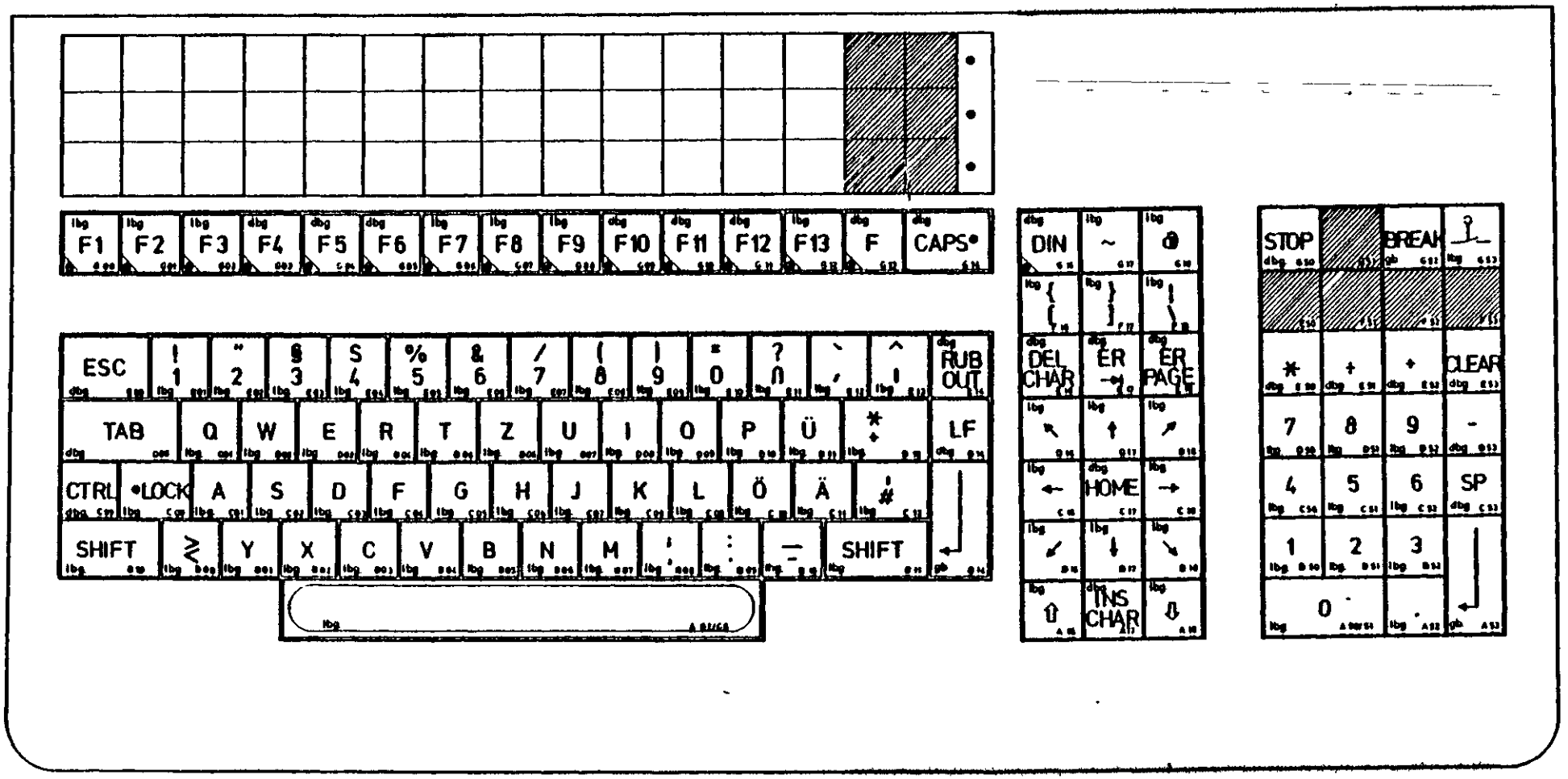
OPEN = "H" am DIL-Schalter SW1

SW1.1 = "L" Testmode "aus" ---> Normalbetrieb Tastatur
"H" Testmode "ein" ---> nur für Prüffeld/Service

SW1.2 = "L" Piezolautsprecher gibt Rückmeldung bei jedem Tastendruck und Fehlermeldung
"H" Piezolautsprecher ist nur bei Fehlermeldungen aktiv

SW1.3 = "L" Serielles Interface aktiv (RS422 bzw. OPEN COLLECTOR)
"H" Paralleles Interface aktiv

SW1.4 : wenn mit Switch 1.3 serielles Interface gewählt:
"L" = RS422
"H" = OPEN CONTROLLER
wenn mit Switch 1.4 paralleles Interface gewählt
"L" = Keyboardstrobe "active-low"
"H" = Keyboardstrobe "active-high"



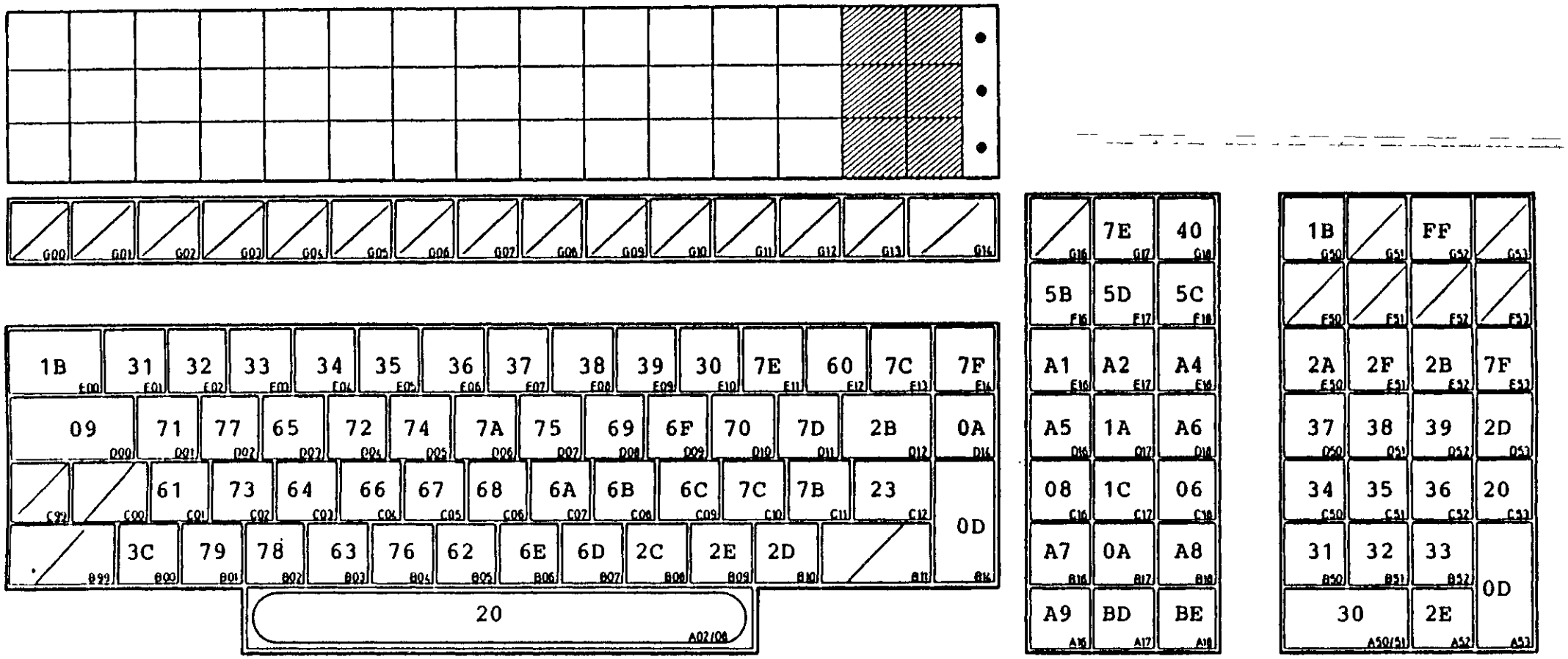
LEGEND: ● - LED

KEY-LOCKED SWITCH

COLOUR OF KEYS: lbg - light beige
dbg - dark beige
gb - greybrown

KEYBOARD
VERSION - MICROCOMPUTER

285 - 0007



VERSION: MICROCOMPUTER-GERMAN MCG 2.1

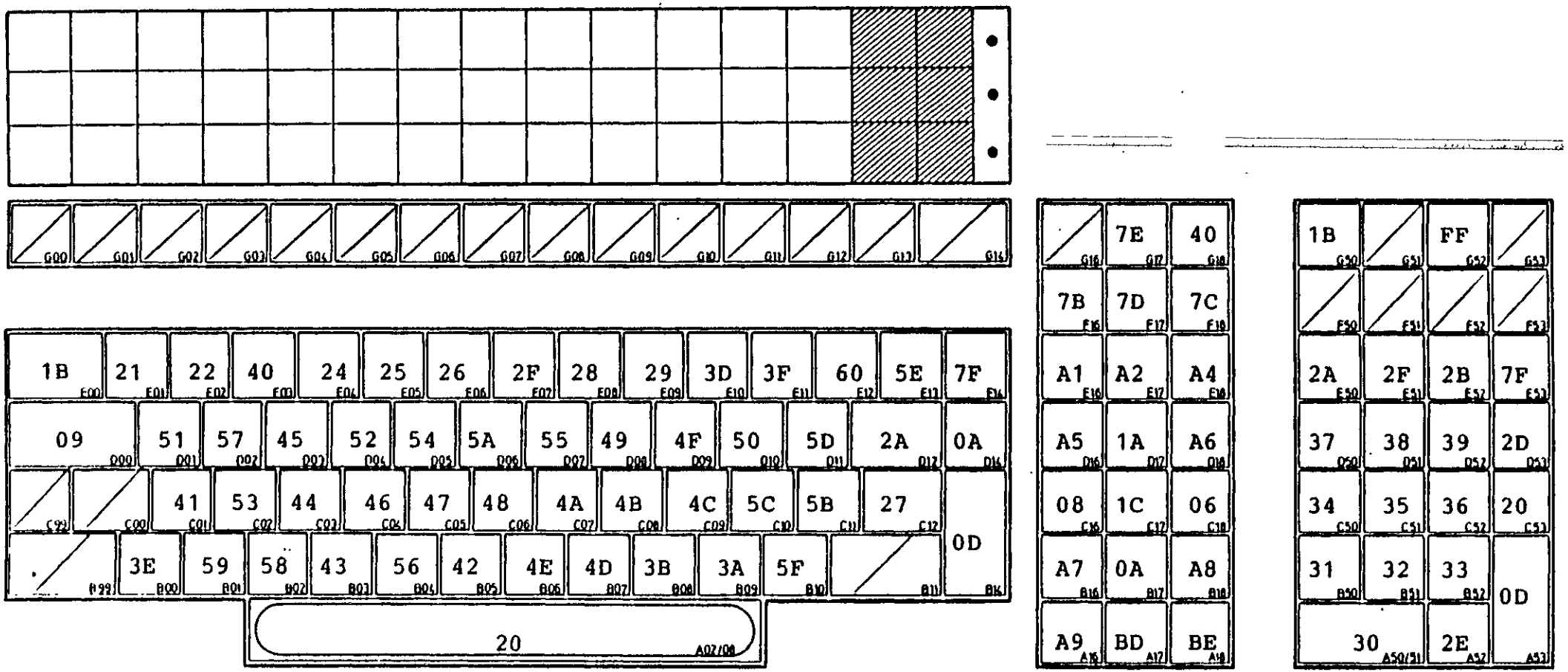
PAGE: 1/7

CHANGES: NEW LAYOUT

TABLE: UNSHIFT

DATE: 08/07/83

				Tag	Name	Benennung: KEYBOARD
				Bearb. 21.2.83	Hansel	
				Gepr. 22.2.83		
					KONTRON ELECTRONIC	Zeichn.-Nr. 1019/1035
1.1	Nr. 178	22.2.83				Blatt-Nr. 1
Rev.	Änderungs-Nr.	Tag	Name			
						Zu Gerät: Zu Anlage:



VERSION: MICROCOMPUTER-GERMAN MCG 2.1

PAGE: 2/7

CHANGES: NEW LAYOUT

TABLE: SHIFT

DATE: 08/07/83

				Tag	Name	KEYBOARD
			Bearb	21.2.83	Hansel	
			Gepr	22.2.83		
			KONTRON ELECTRONIC			1019/1035
Nr. 178		22.2.83				
Rev	Änderungs-Nr	Tag	Name			

Kontron PSI80/98/900 Systeme

Problembeschreibung/Verbesserungsvorschlag

Absender: Name ..... Datum .....

Firma .....

Abteilung .....

Adresse ..... Tel. .....

An
Kontron Mikrocomputer GmbH
Applikation
Breslauer Str. 2

D-8057 Eching b. München

Dies ist
- ein Verbesserungsvorschlag
- eine Problembeschreibung

für
- Hardware
- Software
- Dokumentation

Benutztes System: (genaue Beschreibung der Systemumgebung ist unbedingt erforderlich)

Hardware: ..... Serien Nr. .....

Zusätzliche Hardware: .....

Software: Version .....

Andere Programme .....

KSS (Kontron Software Service) Ja/Nein

Haben Sie sich bereits mit einer Kontron Geschäftsstelle in Verbindung gesetzt?
Wer war Ihr Ansprechpartner?

.....

Problembeschreibung/Verbesserungsvorschlag:

Anlagen: Diskette mit Beispielprogrammen .....
(beschleunigt die Bearbeitung)