

**KONTRON PSI - Computersysteme**

**Betriebssoftware KOS**

**Technische Beschreibung**

# **KONTRON**

## **PSI - SYSTEME**

Diese Technische Beschreibung enthält detaillierte Hard- und Software-Informationen über die Computersysteme der KONTRON PSI-Reihe unter dem Betriebssystem KOS. Sie stellt eine Arbeitsgrundlage für den erfahrenen Computer-Anwender dar, der die Leistungsfähigkeit und Flexibilität dieser Computersysteme auf System- und Prozessor-Ebene voll nutzen möchte. Für den Erstbenutzer der KONTRON PSI-Systeme ist - unabhängig von seiner generellen Erfahrung auf Computersystemen- die sorgfältige Beachtung des Bedienungshandbuches und der einführenden Schriften empfohlen.

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.

Copyright by Kontron Mikrocomputer GmbH,  
Eching  
Alle Rechte vorbehalten  
Juli 1984

DOK-PSI/KOS-D0784

## **Inhaltsverzeichnis der Technischen Beschreibung:**

TB - A	Konzepte und Grundlagen des Betriebssystems KOS
TB - B	Systemkommandos zum Betriebssystem KOS
TB - C	KOS-Betriebssystembeschreibung
TB - D	KOS-Utilities
TB - E	Assembler, Linker und Crossreference-Generator
TB - F	Debugging Module KDM
TB - G	Anhang: Tabellen
TB - H	Stichwortverzeichnis beider Handbücher

## **Literaturhinweise:**

Weitere Informationen entnehmen Sie bitte den folgenden Abschnitten:

### **Bedienungs- anleitung:**

Inbetriebnahme der Kontron PSI-  
Computersysteme  
Einführung in Betriebssystem KOS und  
Systemkommandos  
EDIT: Editor-Beschreibung  
BASIC-Beschreibung  
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 Service Dokumentation

Optionale Software: siehe jeweilige Dokumentation

Kontron ECB-Computer-  
baugruppen: siehe jeweilige Dokumentation

Integrierte Schalt-  
kreise: siehe Dokumentation der jeweiligen Hersteller

**STAND:**

Version 4.33/5.44/5.54/6.05      Juli 1984

**FRÜHERE STÄNDE:**

Version 4.33/5.44/5.54/6.04	August 1983
Version 4.33/5.44/5.54/6.04	April 1983
Version 4.33/5.4/5.5/6.0	Dezember 1982
Version 4.3/5.3	August 1981
Version 3:2	April 1980



**Konzepte und Grundlagen  
Kontron PSI-Computersysteme  
unter dem Betriebssystem KOS**

Stand: Juli 1984

Version: 4.33/5.44/5.54/6.05

Der folgende Teil der Kontron PSI-Dokumentation schildert die dem Entwurf der Systemsoftware KOS zugrundeliegenden Konzeptionen.

Das Verständnis dieser Konzeptionen ist wesentliche Voraussetzung für die volle Ausnutzung der Leistungsfähigkeit der Systeme für Ihre Anwendung. Deswegen steht dieses Kapitel vor den Abschnitten "Systemkommandos" und "Betriebssystem KOS".

Die folgende Bereiche werden hier erläutert:

Aufbau des Betriebssystems KOS  
System-Kommandos und -Kommandosprache  
Dateinamen  
Medien  
Log-in und Working Directories  
Ein-/Ausgabesystem  
Mehrrechnersysteme  
CP/M 2.2-Kompatibilität



## Inhalt

1. Das Kontron Betriebssystem KOS
  - 1.1 Aufbau des Kontron Betriebssystems KOS
  - 1.2 KOS-Systemfunktionen
  - 1.3 Ein-/Ausgabe: logische Kanäle
  - 1.4 Medienkonzept
  - 1.5 Working Directory und Public Files
2. Der Start des Betriebssystems
3. Login: der Start des Systems
4. Systemkommandos von KOS
5. Dateien und Dateiadressen
6. Kontron KOBUS - das Kontron PSI-Mehrrechnersystem
7. CP/M 2.2 Kompatibilität
8. Gewährleistung bei Software

—

—

—

—

## Konzepte und Grundlagen des Betriebssystems Kontron KOS

Für Kontron PSI-Computersysteme existiert Software basierend auf CP/M- und KOS-Betriebssystemen. Die folgende Dokumentation gilt den Grundlagen der Kontron Systemsoftware KOS.

Das Betriebssystem KOS (Kontron-Operating-System) ist in allen Massenspeicher-basierenden Kontron PSI-Systemen implementiert. Es ist als Anwendungs-orientiertes transparentes Betriebssystem konzipiert. Seine besonderen Merkmale sind:

### Medienunabhängigkeit

KOS ist medienunabhängig: externe Massenspeicher werden über logische Kanäle mit wahlfreier Zuordnung verwendet. Für die Medienverwaltung gilt:

- Kapazität pro Medium bis 64 MByte
- automatische Dateisuche auf allen aktiven Medien
- Dateigröße von 0 bis 8 MByte
- Variables Inhaltsverzeichnis mit einer nur durch die Kapazität des Mediums begrenzten Anzahl von Dateien.
- Sequentieller und wahlfreier Zugriff auf alle Sätze einer Datei
- benutzerdefinierbare Dateieigenschaften (file properties) wie Benutzerkennzeichen, schreibgeschützt, löschgeschützt, geheim etc.

### Working Directories

Medien können logisch in Benutzer-spezifische "Working Directories" gegliedert sein (ab KOS5.5x/6.0x).

### Logische Ein-/Ausgabekanäle

Eine komfortable Ein-/Ausgabeverwaltung erlaubt die beliebige Zuordnung von logischen Ein-/Ausgabekanälen zu symbolischen Geräte-kennzeichnungen. Damit ist der Datentransfer von beliebigen Quellen zu beliebigen Zielen von allen Programmen aus möglich, ohne diese selbst zu ändern.

### Tasking

KOS unterstützt den quasisimultanen Ablauf eines Vordergrundprogramms mit bis zu zehn Hintergrundprogrammen (Tasks). Spooling auf Peripheriegeräte ist im standardmäßigen Leistungsumfang enthalten.

### Kompatibilität

Über frei aktivierbare Umsetzermodule wird die Aufwärtskompatibilität zu anderen Betriebssystemen erreicht. Implementiert ist z.Zt. das Anpassungsmodul zu CP/M 2.2. Hierbei ist zu beachten, daß diese Kompatibilität zunächst die definierten CP/M-Systemaufrufe betrifft. Die Gewährleistung des Ablaufs beliebiger CP/M-Software wird nicht übernommen.

### Lokales Netzwerk

Auf Basis des Betriebssystems KOS ist das Mehrrechnersystem Kontron KOBUS mit bis zu 17 Stationen möglich. KOS bietet hier auch die Einbeziehung von Kontron ECB-Computerbaugruppen in dieses lokale Rechnernetz.

## 1. Das Betriebssystem KOS

### 1.1 Aufbau des Kontron Betriebssystems KOS

Das Betriebssystem KOS ist ein Dienstleistungssystem, das Kommandos vom Benutzer oder externen Dateien (Kommandodateien) erhält und diese ausführt. KOS besteht im wesentlichen aus folgenden Teilen:

- dem Systemverwalter
- dem Kommandoentschlüssler
- dem Dateiverwalter
- dem Ein-/Ausgabeverwalter
- dem Speicherverwalter
- dem Taskverwalter
- den internen Systemprogrammen

Der Systemverwalter ist das übergeordnete Softwaremodul des Betriebssystems. Die Hauptfunktion des Systemverwalters ist die Koordination der Abläufe in den übrigen Teilen von KOS.

Der Kommandoentschlüssler unterscheidet KOS-interne von externen (medienresidenten) Systemkommandos und bereitet gleichzeitig den Kommandostring zu einem Parameterblock auf. Dieser wird von anderen Teilen des Betriebssystems weiter verwendet.

Der Dateiverwalter besorgt die Zuordnung zwischen symbolischen und logischen Adressen, z.B. Name <--> logische Satznummer von Informationsaufzeichnungen auf Medien (externen Datenträgern).

Die wesentliche Funktion der E/A-Verwaltung ist die Verwaltung und Verzweigung von und zu logischen Datenübertragungskanälen. KOS und die Dienstprogramme verwenden definierte logische Kanäle zur Ein-/Ausgabe. Logische Kanäle sind gewöhnlich einem Ein-/Ausgabetreiber zugeordnet. Diese Zuordnung ist beliebig und kann jederzeit geändert werden (siehe IODC-Kommando). Damit ergibt sich die Möglichkeit, die Ein-/Ausgaben aller Programme auf beliebige Kanäle zu dirigieren, ohne die Programme selbst ändern zu müssen.

KOS zeichnet sich unter anderem durch eine leistungsfähige Speicher-verwaltung aus. Diese teilt den von der Z80A-CPU adressierbaren Speicherraum von 64 kByte in Segmente zu je 128 Byte ein. Pro Segment wird in einer Tabelle von 64 Byte ein Bit geführt (BITMAP), das gesetzt ist, falls das dazugehörige Segment belegt ist. Im umgekehrten Fall ist es rückgesetzt. Immer dann, wenn Programme Speicherplatz benötigen, wird die Speicherverwaltung beauftragt, den entsprechenden Speicherbereich zu belegen. Ist dieser bereits belegt, so erfolgt eine Fehlermeldung. Vor jeder Ausführung eines Systemprogramms (wie ASM, EDIT, COPY etc.) reserviert KOS auch einen 180H Byte großen Speicherbereich für den Stack des Programms im obersten noch freien Speicher.

Der frei verfügbare Anwenderspeicher in der Bank 0 hängt von der Massenspeicherkonfiguration ab. Er ist maximal bei Diskettenanlagen und wird beim Anschluß von Festplatten geringer. Dies ist zu beachten, wenn Anwendungssysteme von einer Konfiguration auf eine andere Ausbaustufe übertragen werden sollen.

## Belegungspläne

Die folgende Tabelle weist die unter dem KOS-Kommando MAP angezeigte Belegung des Speichers nach Aktivierung bzw. Ansprechen der aufgelisteten Laufwerke aus:

LW-Konfiguration	Typ	! Belegt ! ! ab	! freier ! Anwenderspeicher
2 FD	908/98/980 Q/M2	E500h	55 3/4 K
1 FD, 10 MB HD	908 Q/W10	E000h	54 1/2 K
1 FD, 17.8 MB HD	98/980 Q/W20	DD00h	53 3/4 K
1 FD, 40 MB HD	98/980 Q/W40	D180h	50 7/8 K

Auch die Aktivierung weiterer im Betriebssystem integrierter Treiber benötigt Speicherplatz:

\$VMED benötigt 1 Segment mit 128 Bytes

\$WIN1 (5 MB Wechselplatte) benötigt 6 x 128 Bytes.

Für alle Treiber gilt, daß sie diese zusätzlichen Speichersegmente von der ersten Ansprache an belegen und bis zum nächsten Reset belegt halten. Die erste Ansprache kann ein schreibender oder lesender Zugriff oder das Kommando N mn (mn=Mediennummer) sein. Auch das Kommando 'Status' sorgt bei Vorhandensein des Laufwerkes für die Speicherbelegung.

Extern angeschlossene Speichermedien benötigen entsprechend ihrer Kapazität Speicherplatz in der Bank 0, wobei pro Megabyte ca. 1 Segment von 128 Bytes im Speicher belegt wird. Bei der externen Wechselplatte (IOMEGA/10 MB) kommt zusätzlich noch der für den Treiber benötigte Bereich von 3 K und der dazugehörige BIT MAP Bereich von 1 1/4 K dazu.

## Technische Hintergrund-Informationen

In diesen Speicherbereichen wird die sogenannte "BIT MAP" abgelegt. Die BIT MAP kennzeichnet den Belegungszustand eines Mediums. Pro Block kennzeichnet 1 bit, ob dieser Block belegt ist oder nicht. Bei Anlage von neuen Dateien z.B. wird vom Betriebssystem der erste freie Block anhand der BIT MAP gesucht und dieser freie Block dann der neuen Datei zugewiesen. Die Blockgröße beträgt 2 KB, deswegen gilt die Faustregel: pro MB Kapazität ein Speichersegment von 128 Bytes für die BIT MAP.

## Zusätzliche Belegungen in Bank 0

Zusätzliche Belegungen entstehen selbstverständliche durch Druckertreiber und CPM-Translator. Diese können -bei CP/M-Translator bereits standardmäßig im Lieferumfang- in höhere KOS-Segmente in Bank 1 verlegt werden, siehe Option KOS-IF.

Weitere Belegungen erfolgen durch Kommandodateien und Menü-Programm.

Bei KOBUS-Masterstationen werden durch das Kommando 'KPpload KPP' 2 Speichersegmente, also 256 Bytes, belegt.

Außerdem reserviert KOS einen Stackbereich unterhalb von Top of Memory im Umfang von 256 Bytes. Dieser Stackbereich gilt als dynamisch und wird deswegen durch MAP nicht angezeigt, ebenso wie die bis zu 8 x 128 Bytes, die als temporärer Buffer für File Control Blocks verwendet werden.

Der Taskverwalter von KOS verwaltet bis zu zehn benutzerdefinierbare Tasks, also Programme, die neben der eigentlichen Benutzertask (Editor, Assembler etc.) im Hintergrund ablaufen.



## 1.2 KOS-Systemfunktionen

Die detaillierte Beschreibung der über RST 8-Befehle ansprechbaren KOS-Systemfunktionen finden Sie im Abschnitt Techn. Beschreibung - C. Alle Systemfunktionen (Ein-/Ausgaben, Datei-, Speicherverwaltung) sind über den gemeinsamen Systemeinsprungpunkt auf Adresse 8 (RST 8-Befehl) erreichbar. Dieser Einsprungpunkt ist reentrant, d.h., daß Systemfunktionen sich selbst oder andere Systemfunktionen aufrufen können. Dies bedeutet auch, daß eine Systemfunktion per Interrupt unterbrechbar sein kann und, daß dieselbe Systemfunktion während der Abarbeitung einer Interrupt Service Routine mit den gleichen oder aber anderen Eingangsparametern wieder aufgerufen werden kann.

Auf Anlagen der Reihe Kontron PSI900/9000 unterstützt KOS6 die Organisation des Speichers in 4 Banken mit jeweils 64 KByte. Bank 0 enthält das Anwenderprogramm, eine 2 KByte große IPCA (Inter Program Communication Area) und (gemapped) eine der 8 bis 16 Pages des Betriebssystems aus Bank 1. Das Umschalten der im Z80A-Adreßbereich liegenden Betriebssystem-Page erfolgt über Tabellen im IPCA-Bereich bei RST8-Befehlen und bei Aufruf von Interrupt Service Routinen.

## 1.3 Ein-/Ausgabe: logische Kanäle

Unter KOS sprechen Programme mit der Außenwelt über "logische Kanäle". Es sind 10 Eingabe- und 10 Ausgabekanäle unter ihrer logischen Kanaladresse ansprechbar. Diese Kanaladressen sind durchnummeriert von  
I-0 bis I-9 für Eingabekanäle  
O-0 bis O-9 für Ausgabekanäle

Einem Kanal wird ein Treiberprogramm zugewiesen, das vier wesentliche Funktionen hat:

- Steuerung der physikalischen Geräteschnittstelle (z.B. RS232C)
- Erzeugung bzw. Auswertung von Kommandozeichenfolgen zur Gerätesteuerung
- Filterung bzw. Umsetzung von Zeichen und Steuerbefehlen
- Anpassung an die KOS-Schnittstelle

Ein Treiber wird durch das IODC-Kommando aktiviert, also dem Betriebssystem bekanntgemacht, und einer Kanalnummer zugewiesen.

Vom Start des Systems an sind die Treiber für Bildschirm, Tastatur und Systemmeldungen bereits aktiviert und Kanäle zugewiesen:

I-0	\$KEY	Tastatur
I-1	\$KEY	
O-0	\$MON	Sichtschirm
O-1	\$MON	
O-2	\$MON	
O-3	\$KSM(L)	Systemmeldungen

Die Kanäle I-0, O-0 und O-3 müssen stets mit \$KEY, \$MON und \$KSM(L) verbunden bleiben, alle andere Kanäle sind frei zuordenbar.

Weitere Informationen siehe IODC-Kommando (TB-B), Systemfunktionen (TB-C) und Utilities (TB-D).

#### 1.4 Medienkonzept

Ein KOS-Medium ist ein Datei-orientierter Speicherraum, der durch Kapazität, Inhaltsverzeichnis und Zugriffszeit gekennzeichnet wird. Alle Medien sind in ihrer logischen Schnittstelle identisch. Beispiele für Medien sind Floppy-Disk, Festplatte, Speicherbereiche, oder Kontron KOBUS, die seriell arbeitende Verbindung zu einem anderen Medium.

Der große Vorteil dieses Medienkonzeptes liegt in der absoluten Unabhängigkeit von der physikalischen Organisation eines beliebigen Mediums: Alle Medien bieten die gleiche Benutzerschnittstelle. Dateien werden durch Namen und Identifikation angesprochen, sie bestehen aus logischen Sätzen von 128 Bytes, unabhängig von der physikalischen Länge eines Sektors auf dem Massenspeicher.

Alle zukünftigen Entwicklungen von Massenspeichersystemen sind dadurch direkt in Kontron PSI Einzel- und Mehrplatzcomputer zu integrieren: Plattenlaufwerke größerer Kapazität, Bubble-Speicher, Fest-/Wechselplatten, Bänder, etc.: die Anwendungssysteme werden sofort von neuen Datenträgern Gebrauch machen können.

Medien werden über einen Medientreiber an KOS angeschlossen. Ein Medientreiber kann in KOS integriert sein, dann ist in dieser KOS-Implementierung dieses Medium direkt ansprechbar. Medientreiber können auch explizit aktiviert werden, siehe IODC-Kommando.

Auch das Kontron KOBUS-System beruht auf dem Medienkonzept: Innerhalb von KOBUS-Systemen erfolgt der Zugriff der Computerstationen auf den zentralen Massenspeicher, die Festplatte des Masters, über das verbindende Koaxialkabel. Das SLAVE-Betriebssystem bzw. der Medientreiber \$SLAV stellen einer Slave-Station das virtuelle Medium "KOBUS" zur Verfügung, das sich völlig gleichartig verhält wie ein lokaler Massenspeicher.

Unter KOS werden die Medientreiber den logischen Medienkanälen zugewiesen. Dies erfolgt frei programmierbar, siehe z.B. IODC-Kommando. Anwendungsprogramme und Systemkommandos sprechen diese Medienkanäle unter ihrer Mediennummer mn an und bleiben somit unabhängig von der Art des physikalisch verfügbaren Mediums.

Dateien werden unter KOS auf allen angeschlossenen Medien gesucht, sofern keine Mediennummer explizit angegeben wird. Die Suche beginnt dabei bei dem durch das KOS-Kommando 'M' zum 'Mastermedium' erklärtem Medium. Für CP/M-Programme ist diese automatische Dateisuche nicht möglich (CP/M-Konvention).

### 1.5 Working Directory und Public Files

Jeder Benutzer unter KOS arbeitet zu jeder Zeit in einem bestimmten frei wählbaren "Working Directory".

Dieses Working Directory 'wdir' enthält eine Sammlung aller Benutzerspezifischen Dateien. Diese Dateien können auf verschiedenen Dateiträgern, also Disketten, Festplatten etc. resident sein.

In einem System können nebeneinander beliebig viele Working Directories eingerichtet sein. Kennzeichnet ist jeweils der Name, der aus maximal 15 druckbaren Zeichen besteht. Diese Zeichenfolge wird in einer CRC-Rechnung auf 16 Bits komprimiert und so im Dateieintrag des Inhaltsverzeichnisses gespeichert.

Jede erzeugte Datei führt so als zusätzliche Kennzeichnung das 'Working Directory,' unter dem sie generiert wurde.

Dateien aus anderen Working Directories sind für den Benutzer zunächst unzugänglich, sie sind "Private Dateien".

Der Wechsel des momentanen Working Directory erfolgt durch das diskresidente Systemkommando WDIR (siehe TB-B). Dieses wird unter KOS aufgerufen. Es zeigt das momentan gültige Working Directory an. Jede beliebige Folge von maximal 15 Druckzeichen kann als Name eines neuen Working Directory eingegeben werden. Damit arbeitet der Anwender unter dem neuen Working Directory.

Neben den in verschiedenen Working Directories enthaltenen privaten Dateien stehen die als "Public" deklarierten Dateien allen Benutzern zur Verfügung.

**Public Files sind zugänglich für alle Benutzer.** Dies wird durch das Kennungsbit 'K', einer der in KOS unterstützten Dateieigenschaften, möglich:

durch das Systemkommando DEFP wird eine Datei zu "Public" erklärt. Es findet dabei kein physikalisches Kopieren statt. Die Datei ist gleichzeitig in allen Working Directories "enthalten", d.h., unter jedem Working Directory kann diese Datei direkt angesprochen werden. Der ursprüngliche Working Directory-Eintrag bleibt erhalten.

Wird unter DEFP die Kennung K gelöscht, so ist diese Datei nur mehr im ursprünglichen Working Directory zu erreichen, die Datei "verschwindet" aus allen anderen wdirts.

Public-Dateien können auch ein Schlüsselwort (Benutzerkennzeichen) tragen. Dieses ist eine maximal 15-stellige Folge von druckbaren Zeichen, die unter dem DEFP-Kommando eingegeben wird.

Vor jedem Zugriff auf eine solche Datei fordert das Betriebssystem dann die Eingabe des korrekten Schlüsselwortes an.

Wird das Benutzerkennzeichen (U=User-ID) einer Datei gelöscht, so wird diese dem gerade aktuellen Working Directory zugeordnet.

"Working Directory" und "Public File" sind logische Zuordnungen. Die Dateiverwaltung selbst erfolgt über ein einziges Inhaltsverzeichnis (=Datei mit Typ ".DIR") auf dem Datenträger. In diesem Inhaltsverzeichnis sind alle Dateien dieses physikalischen Mediums enthalten.

### Wie organisiere ich die Medien optimal?

Bei Anlieferung sind alle Dateien in einem Working Directroy "\*" oder "secret" und "public". Von der gesamten Menge der Kontron-System- und Dienstprogramme werden nur diejenigen auf einen Datenträger übernommen, die für die spezielle Anwendung notwendig sind.

Bei der Anlage von Medieninhalten ist folgende Strategie zweckmäßig:

- Betriebssystemdateien (BOOT2.SYS, KOSx.SYS, KOBUS.SYS, KCP.SYS und/oder KCP1.SYS) an den Anfang des Lademediums legen.
- Kennzeichen der Dienstprogramme und der Anwendungsprogramme, die allen zur Verfügung stehen sollen, als public und secret, optional auch zusätzlich schreib- und löschgeschützt.
- Einbringen der benutzer-spezifischen Programme und Daten in getrennte Working Directories, Kennzeichnungen secret etc. sind optional.

Der Vorteil liegt in der größeren Übersichtlichkeit und im leichteren Reorganisieren des Mediums. Außerdem kann jeder Benutzer mit dem MOVE-Kommando seinen Teilbereich z.B. leichter auf Diskette sichern. "Dateileichen", also obsolete Texte, sind im jeweiligen Working Directory leichter zu erkennen.

Bei Diskettensystemen und vergleichbaren Medien ist die Verwendung von Working Directories möglich, im Nutzen jedoch stark anwendungsabhängig.

Bei Festplattermedien und bei KOBUS-Systemen bewirkt das Working Directory-Konzept Übersichtlichkeit und Datensicherheit.

## 2. Der Start des Betriebssystems

Das Betriebssystem KOS wird automatisch nach dem Einschalten des Systems von einem Medium geladen. (Bei KOS4 ist die Eingabe von "K<---" notwendig.)

Folgende Voraussetzungen sind dazu notwendig:

- a) Floppy Disk-basierende Kontron PSI80-Systeme unter KOS4.33:  
Kontron PSI80/S2 u. Kontron PSI80/M2.  
Im Laufwerk 0 (rechtes Laufwerk) muß eine Diskette enthalten sein, die die Dateien 'BOOT2.SYS' und KOS.SYS als erste Einträge hat.  
KOS.SYS muß die Betriebssystem Version KOS4.33 beinhalten.

- b) Floppy Disk-basierende Kontron PSI80/82-Systeme unter KOS5.x:  
Kontron PSI80/S2, PSI80/M2  
Kontron PSI80 D/M2, PSI80 D/S2, PSI82 D/M2  
Kontron PSI80 Q/M2, PSI80 Q/S2  
Kontron PSI80/82-Syste, mit integrierter Festplatte:  
Kontron PSI80 D/W5, PSI80 Q/W5, PSI82 D/W5  
Kontron PSI80 D/W10, PSI80 Q/W10, PSI82 D/W10

Innerhalb der ersten 32 Einträge eines erreichbaren Mediums müssen die Dateien 'BOOT2.SYS' und 'KOS.SYS' hinterlegt sein, wobei 'KOS.SYS' eine für dieses Hardware-System geeignete Version KOS5.x sein muß.

- c) Massenspeicherlose KOBUS-Slave-Station Kontron PSI80/TC.

Innerhalb der ersten 32 Einträge des zentralen KOBUS-Mediums muß die Datei 'KOBUS.SYS' enthalten sein.

- d) Kontron PSI900/9000-Systeme: Auf einem zugänglichen Massenspeicher müssen die Dateien KOS0.SYS bis KOS8.SYS (evtl. bis max. KOSF.SYS) innerhalb der ersten 32 Einträge enthalten sein.

Hinweis: Die Aussage "innerhalb der ersten 32 Einträge" schließt ein, daß die zugehörigen Dateien "am Anfang" des Mediums, zumindest noch vor Record 100h liegen. Dies ist besonders dann zu beachten, wenn in einem bereits gefülltem Medium kurze Dateien durch längere Betriebssystemdateien überschrieben werden. Dann stehen nämlich zwar die Einträge richtig, die zugehörigen Dateien jedoch werden in die erste Lücke passender Größe hinterlegt und sind dann in der Startphase unerreichbar.

### 3. Login: der Start des Systems

Beim Laden von KOS wird optional (KOS5.5x/6.0x) als erstes nach der Benutzer-Identifikation gefragt. Es muß daraufhin eine der gültigen Identifikationen eingegeben werden. Die Gültigkeit wird überprüft, bei Übereinstimmung wird das System freigegeben und gleichzeitig ein zur Benutzer-Identifikation gehörendes Working Directory zugewiesen.

Die Liste der (z.B. in einem KOBUS-System) gültigen Benutzer-Identifikationen und ihrer zugehörigen Working Directories wird bei der Systemgenerierung (s. unten) festgelegt. Bei Auslieferung ist die Benutzer-Identifikation "\*" vordefiniert, ihr ist der Working Directory-Name "\*" zugewiesen.

Sie antworten also auf die Login-Frage mit Eingabe der Taste "\*" und "RETURN", bzw. mit den von Ihnen mit dem Programm SYSGEN (siehe Systemkommandos) eingetragenen Benutzer-Identifikationen.

Als nächstes führt das Betriebssystem eine Kommandodatei KOS.INI (bzw. SLAVE.INI bei Kontron KOBUS Slaves) aus. Diese Kommandodateien können bereits Benutzer-spezifisch sein, wenn sie im zugeordneten Working Directory geführt werden.

In KOS.INI bzw. SLAVE.INI sind die Anfangsaktionen frei vorgebar, indem in diese Dateien mittels des Editors EDIT beliebige Kommandofolgen geschrieben werden können.

Bei KOS-Versionen, die ohne Login-Abfrage generiert wurden, erfolgt nach dem Laden von KOS sofort die Ausführung eines Kommandos "DO KOS.INI".

Der Inhalt der Datei 'KOS.INI' kann durch den Editor EDIT verändert werden. Ist beispielsweise erwünscht, daß das Kontron PSI-System nach jedem Laden von KOS automatisch BASIC aufruft, so muß die Datei 'KOS.INI' auch (oder nur) das Kommando 'BASIC' enthalten.

Für KOBUS-Slave-Stationen Kontron PSI80/TC und Kontron PSI9C heißt diese Kommandodatei 'SLAVE.INI'.

#### 4. Systemkommandos von KOS

Zum Betriebssystem KOS gehören eine Reihe von Kommandos, die als Dienstprogramme die Arbeit mit dem Computersystem Kontron PSI erleichtern. Sie werden im Abschnitt TB-B "Systemkommandos" dieses Handbuches ausführlich dargestellt.

Die Dienstprogramme decken folgende Gebiete ab:

Systemsteuerung	(z.B. TASK)
Medienpflege	(z.B. STATUS, FORMAT)
Dateipflege	(z.B. COPY, MOVE)
Ein-/Ausgabeverwaltung	(z.B. IODC, SPOOL)
Programm- und Texterstellung	(z.B. EDIT, ASM, KDM)

Systemkommandos können Speicherresident oder Medienresident sein.

##### Residente (interne Kommandos)

Diese Kommandogruppe ist immer im Speicher resident, muß also nicht erst von einem Massenspeicher (Floppy Disk, Hard Disk) geladen werden. Die Kommandoidentifikation erfolgt durch einen Buchstaben (groß oder klein geschrieben), implementiert sind Kommandos zum Auflisten des Medieninhaltes, zum Blättern im Bildschirmspeicher etc.

##### Medienresidente (externe) Kommandos

Kommandos dieser Gruppe sind als Dateien auf beliebigen Medien, z.B. Disketten oder Festplatten, resident und werden bei ihrem Aufruf in den Speicher geladen. Dateien, die ausführbare Programme enthalten, sind durch den Typ '.COM' gekennzeichnet. Der Aufruf eines Kommandos erfolgt durch Eingabe des Kommandonamens (ohne Typ '.COM').

## Kommandoeingabe

Kommandos werden durch Eingabe ihres Namens von der Tastatur her oder aus einer Kommandodatei heraus aufgerufen. Im allgemeinen sind Kommandos als Dateien auf Medien hinterlegt.

Kommandos können jederzeit nach dem Promptzeichen

nn/KOS: (KOS6)  
bzw. KOS: (KOS4/5)

eingegeben werden. Hierbei entspricht bei KOS6 'nn' einer zweistelligen Dezimalzahl 0...99, welche die Nummer des derzeitigen Kommandos repräsentiert. Unter dieser Nummer kann ein Kommando zu einem späteren Zeitpunkt wieder aufgerufen werden (siehe History-Kommando, TB-B).

Kommandos und ihre Parameter bestehen aus alphanumerischen Zeichen (A..Z, 0..9) und werden mit der Eingabe von 'RETURN' abgeschlossen.

Steuerzeichen (CTRL-Tasten) während der Kommandoeingabe zeigen entweder eine sofortige Wirkung auf dem Sichtgerät oder aber sie werden bei KOS6 ignoriert.

### a) Steuerzeichen mit sofortiger Wirkung für die Monitoranzeige (KOS4/5/6)

CTRL-G	Bell (Speaker)
CTRL-L	Formularvorschub
CTRL-Q	Character Invert (*)
CTRL-R	Video Invert (*)
CTRL-S	Speed Invert (*)
CTRL-T	Cursor on/off (*)
CTRL-W	Character Blinking on/off (nur KOS6) (*)
CTRL-F	Laufwerkmotoren einschalten (nur KOS6)

(\*) Diese Steuerzeichen wirken jedesmal umschaltend, CTRL-Q invertiert die Darstellung der folgenden Zeichen (und der Füllzeichen bei Tabulation) relativ zum Bildhintergrund.



b) Steuerzeichen für die Kommandoedition bzw. -wiederholung  
(nur KOS6)

- CTRL-A: Again  
Führt die zuletzt eingegebene Kommandozeile nochmals aus.
- CTRL-B: BOOT  
Führt nach einer Rückfrage (y/n) zum Rücksprung in das Promresidente Urladerprom, das Betriebssystem wird erneut geladen.
- CTRL-E: EDIT  
Holt die letzte Kommandozeile in den Eingabepuffer zurück und ermöglicht somit die Modifikation und/oder Erweiterung der letzten Kommandoeingabe. Mit CTRL-E kann auch eine gelöschte Zeile wieder zurückgeholt werden.

Die Steuerzeichen CTRL-A/B/E wirken nur, falls sie als erstes Zeichen nach dem Prompt eingegeben werden. In allen anderen Fällen werden sie ignoriert.

c) Steuerzeichen für die Kommandoedition

- CTRL-H: Backspace bzw. Cursor Left (KOS4/5/6)  
Löscht das zuletzt eingegebene Zeichen und bewegt den Cursor um eine Stelle nach links.
- RUBOUT: Delete Line (KOS4/5/6)  
Löscht die gesamte bisher eingegebene Zeile und bewegt den Cursor an den Zeilenanfang. Versehentlich gelöschte Zeilen können unter KOS6 mit CTRL-E wieder zurückgeholt werden.

Alle hier nicht aufgeführten Steuerzeichen (< 20 Hex) werden bei der Kommandoeingabe unter KOS6 ignoriert, bei KOS4/5 werden die Bildschirm-Steuerzeichen ausgeführt.

Tabulator und nicht druckende Steuerzeichen werden durch CTRL-H ebenfalls als ein Zeichen behandelt. Die Anzeige am Bildschirm kann dadurch bei KOS4/5 vom logischen Inhalt der Kommandozeile abweichen.

**Allgemeines Kommando-Eingabeformat:**

```
/wdir/mn:kommando parameter1...parameterN<---  
/wdir/mn:kommando,<---
```

Zum Abschluß einer Kommandoeingabe muß die Taste 'RETURN' (im folgenden mit '<---' symbolisiert) gedrückt werden. Folgt dem Kommando unmittelbar ein Komma mit darauffolgendem RETURN, so wird das Programm nur geladen und nicht ausgeführt. Die Ausführung kann mit dem Kommando 'X' wiederholt ausgelöst werden, solange das von extern geladene Programm nicht überschrieben oder verändert wird.

Die Angabe von /wdir/ ist optional. Damit wird ein bestimmtes 'Working Directory' (s. unten) selektiert.

Die Mediennummer 'mn' ist ebenfalls optional. KOS sucht die entsprechende Datei zunächst auf dem Mastermedium und, falls dort nicht vorhanden, anschließend auf allen übrigen aktiven Medien. Zahl und Art der Parameter hängen vom aufgerufenen Kommando ab.

Folgen von Kommandos können in einer bis zu 256 Stellen langen Kommandozeile eingegeben werden. Sie werden vom Kommandointerpreter in der Reihenfolge der Eingabe nacheinander abgearbeitet. Bei Eingabe von 'RETURN', spätestens jedoch nach 256 Zeichen, beginnt die Verarbeitung. Trennzeichen zwischen den einzelnen Kommandos ist ein Strichpunkt (;):

```
kommandofeld1;kommandofeld2; ... ;kommandofeldN<---
```

Tastatureingaben, die für die aufgerufenen Kommandos bestimmt sind, können im Kommandoaufruf eingeschlossen sein. Sie sind durch Anführungszeichen (") abgeschlossen. RETURN wird durch "<" symbolisiert, z.B.:

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

In gleicher Syntax können die Kommandoeingaben auch in Kommandodateien hinterlegt und zur Ausführung gebracht werden, siehe DO-Kommando.

Die Medienauswahl geschieht hier auch beim Aufruf von CP/M-Programmen auf der KOS-Ebene, d.h., die KOS-Mediennummern 0...9 sind zur Dateiadressierung zu verwenden. (Innerhalb von CP/M-Programmen gelten die CP/M-Laufwerksbezeichnungen A:, B:....).

### Ausgabesteuerung

Alle Systemprogramme, die Ausgaben auf den Sichtschirm des Kontron PSI-Computers veranlassen, reagieren auf Tastatureingaben wie folgt:

ESCAPE:      Programmabbruch  
CTRL-S:      Geschwindigkeitumschaltung (= 'Speed')  
CTRL-P:      Ausgabeunterbrechung mit der Möglichkeit, mit den  
              Cursortasten auf den zurückliegenden Text (bis zu  
              8 Seiten) zurückzublättern, bis eine beliebige  
              andere Taste gedrückt wird (= 'Paging'). Es gilt:

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

Übrige  
Tasten:      Programmunterbrechung, bis wieder eine beliebige  
              Taste gedrückt wird.

Nach Abschluß der Bildschirmausgaben eines Systemdienstprogrammes können noch vor Programmende bis zu 16 Zeichen (256 bei KOS6) in einen Puffer eingegeben werden. Diese Zeichen werden dann mit dem Promptzeichen 'KOS:' zur Anzeige gebracht.

### Syntaxausdruck (Hilfe-Funktion)

Viele KOS-Dienstprogramme geben bei Aufruf ohne Parameter Hinweise über die erforderliche Syntax des korrekten Kommandoaufrufs aus. Derartige Ausgaben erfolgen in Klartext und enthalten mindestens zwei bis drei Textzeilen. Bei komplexeren Kommandos (z.B. COPY) werden detaillierte Hinweise und Beispiele ausgegeben.

## 5. Dateien und Dateiadressen

Ein Kommandofeld enthält meist eine oder mehrere Dateiadressen als Parameter. Ihr Aufbau wird im folgenden erläutert.

Dateiadressen sind symbolische Namen für Informationen auf Floppy Disks oder anderen Speichermedien. Sie geben die Zuordnung zwischen symbolischer und physikalischer Adresse (Name <--> Spur/Sektor) durch die Dateiverwaltung an. Diese Zuordnung kann eindeutig, oder, zur Ansprache von Dateigruppen, auch mehrdeutig sein.

Dateiadressen bestehen zunächst aus 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 die für Kommandos (A...Z, a...z, 1...9) erlaubten Zeichen. Zwischen Namen und Typ dürfen neben dem Punkt (.) keine weiteren Trennzeichen wie 'Leertaste' oder 'Tabulator' stehen.

Jeder Dateiadresse kann eine Mediennummer mn, gefolgt von einem Doppelpunkt, vorangestellt werden:

```

mediennummer:dateiname.dateityp
*           *           *
*           *           *----- max. 3 Zeichen
*           *----- max. 8 Zeichen
*----- Ziffer 0 bis 8

```

Die Angabe einer Mediennummer ist optional, da KOS beim Eröffnen einer Datei diese auf allen aktiven Medien sucht. Die Suchsequenz beginnt auf dem Mastermedium.

Das Mastermedium ist dasjenige Medium, auf dem bei Dateireferenzen mit fehlender Mediennummer zuerst gesucht wird. Die Umschaltung erfolgt durch das KOS-interne Kommando 'M'.

Vor die Mediennummer kann zur Vervollständigung der Dateiadresse der Working Directory-Name, gekennzeichnet durch Schrägstriche (/) stehen:

```

/wdir/mn:dateiname.dateityp
*
*
*----- max. 15 Stellen (druckbare Zeichen)

```

Diese Angabe entfällt bei 'Public'-Dateien. Ohne Angabe von wdir gilt das gerade aktuelle wdir als ausgewählt.

Dateien sind zusätzlich mit Dateieigenschaften gekennzeichnet. Diese werden ausführlich beim Kommando 'DEFP' (TB-B) beschrieben. In Kommandos werden Sie dann durch "P=...." markiert.

### Eindeutige Dateiadressen

Eindeutige Dateiadressen kennzeichnen in eindeutiger Weise den der (symbolischen) Dateiadresse zugeordneten physikalischen Bereich eines Mediums: eine eindeutige Dateiadresse adressiert genau eine Datei auf einem bestimmten Medium.

Eindeutige Dateiadressen erfordern mindestens die Angabe eines vollständigen Dateinamens. Der Dateityp darf in manchen Fällen fehlen, da verschiedene Programme und Operationen bestimmte Dateitypen voraussetzen oder zumindest einen Vorzugstyp haben. Beispielsweise impliziert der Assembleraufruf die Existenz einer Datei vom Typ '.SRC' (Source = Quelle).

### Mehrdeutige Dateiadressen

Eine mehrdeutige Dateiadresse kennzeichnet eine beliebige Anzahl von Dateien mit einer gemeinsamen Eigenschaft. Eine solche gemeinsame Eigenschaft kann beispielsweise sein:

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

Zur Konstruktion von mehrdeutigen Dateiadressen stehen zwei Sonderzeichen zur Verfügung:

- das Zeichen '\*' (ASCII-Code: 2AH) wirkt als Universalbezeichner. Es kann für eine beliebige Anzahl von Zeichen eines der beiden Teile einer vollständigen Dateiadresse stehen.
- Das Zeichen '?' (ASCII-Code: 3FH) repräsentiert ein beliebiges im jeweiligen Bereich zugelassenes Zeichen einer Dateiadresse.

### Beispiel für mehrdeutige Dateireferenzen:

- a) \*.ASM      alle Dateien vom Typ ASM
- b) ABC.\*      alle Dateien des Namens ABC mit beliebigem Typ
- c) ABC\*.BAS   alle Dateien, die einen mit ABC beginnenden Namen und den Typ BAS haben
- d) X\*.\*      alle Dateien, deren Name mit X beginnt und deren Typ aus einem Zeichen besteht

## 6. Kontron KOBUS - das Kontron PSI-Mehrrechnersystem

In kommerziellen, industriellen und technisch-wissenschaftlichen Anwendungen bieten Multi-User Systeme die bessere Ausnutzung von Massenspeicherraum und Peripherie. Darüber hinaus heben sie die Anwendung in den Bereich der integrierten Datenverarbeitung: gemeinsame Datenbestände werden dezentral bearbeitet, die Daten und Dienstleistungen stehen allen Teilnehmern zur Verfügung.

Kontron KOBUS ist das Multicomputersystem:  
jedem Benutzer seinen eigenen vollständigen Rechner, also dezentrale Intelligenz in einem Multi-Computersystem.

Basis sind Kompaktcomputersysteme Kontron PSI80/98, die Industriesysteme Kontron PSI82/980R, die Systeme der "Ergo Line 900" und Kontron ECB-Systeme.

Mit KOBUS werden Einzelplatzsysteme zu Mehrplatzanlagen, unter voller Programmkompatibilität.

Kontron KOBUS verbindet die Vorteile zentraler und dezentraler Organisationen. Die modernsten Technologien in Hardware und Software bringen Leistungen von Minicomputern und Großrechnern an den Arbeitsplatz des PSI-Benutzers: den Zugriff auf gemeinsame Datenmengen und Programmbibliotheken in einem zentralen Massenspeicher in Festplattentechnologie und den Zugang zu gemeinsam genutzten Peripheriegeräten vom lokalen Terminalcomputersystem aus. Im Industriebereich verbindet KOBUS ECB-Computerplatinsysteme und mechaniklose Kontron PSI82/980R-Computer, die unempfindlich gegen die rauhe Industrieumwelt sind, mit den Massenspeichern und Peripheriegeräten des leistungsfähigen Zentralcomputers. Allen angeschlossenen Systemen gemeinsam ist das durchgehend verwendete Betriebssystem KOS.

Der folgende Abschnitt gilt dem KOBUS4-System, also dem Verbund von Rechnern unter dem gemeinsamen Betriebssystem KOS.

### Aufbau von Multi-PSI-Systemen

Kontron KOBUS verbindet bis zu 17 PSI-Stationen zum Mehrplatzsystem. Alle Stationen sind über eine einzige preiswerte Koaxialleitung miteinander verbunden, die eine Länge von bis zu 2 km haben kann.

An jeder Stelle des Kabels kann eine Slave-Station oder die Masterstation angeschlossen sein. Die Verbindung zwischen Koaxialkabel und Station geschieht durch Koppler und Stichleitungen. Im gesamten System liegt die Übertragungsrate bei 800 kBit/s. Die Stationen arbeiten unabhängig voneinander.

Jede Station ist ein selbständiges Computersystem auf Basis von Kontron PSI80, PSI82, PSI9x oder der KOBUS-ECB-Platine ECB/KCP, für deren vollständige Funktion das Vorhandensein eines Massenspeichers (Floppy, Festplatte) nicht erforderlich ist.

Eine PSI-Masterstation steuert die Kommunikation zwischen Master und bis zu 16 Terminalstationen. Sie stellt dem KOBUS-System den Massenspeicher als Träger der Datenbank und der Programmbibliothek zur Verfügung.

Im Übrigen ist die Masterstation als vollwertiger Arbeitsplatz am KOBUS-System zu verwenden. Die KOBUS-Software belegt maximal 4 kB RAM im Master.

Die Slave-Stationen sind Kontron PSI-Systeme ohne eigenen Massenspeicher oder solche mit lokalen Floppy Disk- oder Festplattenlaufwerken. Auch reine Kontron ECB-Platinensysteme sind anschließbar durch ECB/KCP. Diese Computerplatine integriert autonome ECB-Substationen in das KOBUS4-Konzept und arbeitet in diesem Verbund ebenfalls unter dem durchgängigen Betriebssystem KOS.

### Masterstation

Die Zentral- oder "Master"-Station ist ein erweitertes Kontron PSI-System. Dieses umfaßt:

- den zentralen Massenspeicher (z.B. eine Festplatte mit 20 MioB Kapazität)
- den KOBUS-Preprozessor, ein selbständig arbeitendes Subsystem, integriert in die Master-Station zur Steuerung der Übertragung und zur Unterstützung der Plattenverwaltung.

Der zentrale Massenspeicher ist das physikalische Medium, auf das alle angeschlossenen Stationen Zugriff haben.

Der Preprozessor ist eine Z80A-Computerplatine im Einfach-Europaformat, die als autonomes Subsystem mit 64 kByte RAM, einem Z80A-SIO, der RS422-Schnittstelle und mit Z80A-CPU und Z80A-DMA ausgestattet ist. Die Übertragungssteuerung wird in diesem Preprozessor durch RAM-residente Software durchgeführt, der 64 kByte große Arbeitsspeicherbereich dient als Pufferbereich für zu übertragende Blöcke. Ein Urlader-PROM steuert das Laden der Preprozessorsoftware.

Für Kontron PSI80/82 KOBUS-Systeme mit bis zu 4 Stationen wurde eine Software-Lösung der KOBUS-Steuerung ohne Preprozessor implementiert, die dort einsetzbar ist, wo die Belastung des Masters durch Dateizugriffe über KOBUS gering ist.

### Terminalstationen

Eine Terminalstation ist zunächst ein Kontron PSI80/TC, PSI9C, oder Kontron ECB/KCP Terminal Computer System. Dieses beinhaltet keinen Massenspeicher, besteht aber ansonsten aus einem voll ausgestatteten Computer mit mindestens 64 kByte RAM, Z80A-CPU, und dem KOBUS Anschluß (RS422). Der Transfer zwischen KOBUS und Hauptspeicher ist DMA-gesteuert.

Der Urlader dieser Slavestationen fordert beim Einschalten automatisch über KOBUS das Laden des Betriebssystems an, das auf dem zentralen Massenspeicher als Datei abgelegt ist. Dieses Betriebssystem hat die volle Benutzerfreundlichkeit des von Kontron entwickelten Betriebssystems KOS.

Im Industriebereich bildet die Version PSI82/980R/M0 oder das ECB/KCP-basierende Subsystem die robuste KOBUS-Terminal-Station ohne eigenen Massenspeicher.

Auch Floppy Disk-basierende Kontron PSI-Systeme können über einen Medientreiber \$SLAT an KOBUS angeschlossen werden. Diese Systeme arbeiten zunächst von der integrierten Floppy Disk oder der eigenen Festplatte. Nach Aktivierung des KOBUS-Medientreibers \$SLAT ist ihnen zusätzlich der Zugriff auf die Masterstation möglich.

### **Multi-User-Fähigkeit**

Entscheidend für den Betrieb als Mehrfach-Arbeitsplatzsystem ist die Fähigkeit des Betriebssystems KOS, Mehrfachzugriffe auf gemeinsame Dateien zu verwalten.

Gemeinsame Dateien (Shared Files) können von mehreren Benutzern "gleichzeitig" schreibend und lesend verwendet werden. Gegen Zugriffsfehler schützt ein automatischer Verriegelungsmechanismus, der Sätze von 128 Bytes in gemeinsamen Dateien **zeitweise für andere Benutzer sperrt**. Die Freigabe eines gesperrten Satzes erfolgt ebenfalls automatisch, wenn der Benutzer einen anderen Satz liest oder den gesperrten Satz zurückschreibt. Der automatische Verriegelungsmechanismus tritt nur bei Dateien in Kraft, die mit der Dateieigenschaft 'R' (reserviert, Record Locking) gekennzeichnet sind.

Das Setzen der "R"-Property einer Datei aktiviert folgenden Mechanismus: Bei lesendem Zugriff auf einen logischen/physikalischen Satz von 128 Byte dieser Datei wird die Satzadresse in eine Tabelle des PSI-Systems übernommen, das den Massenspeicher verwaltet. Von da an ist dieser Satz für weitere Zugriffe gesperrt, bis dieser Satz zurückgeschrieben wird oder bis ein anderer Satz der gleichen Datei gelesen wird. Dieser Mechanismus arbeitet auf KOS-Ebene. Logische Satzlängen von anderer Länge als 128 Byte, z.B. aus COBOL-Programmen heraus, können nicht automatisch verwaltet werden, da diese logischen Sätze nicht 1:1 den physikalischen Sätzen zugeordnet sind. Die Umsetzung auf physikalische Sätze erfolgt in einem RUNTIME-Modul, das nicht im Zugriff des Betriebssystems KOS liegt.

Allgemeine Dateien (Public Files) dagegen sind die Utilities, Übersetzer und andere Programme, die allen zur Verfügung stehen. Diese Dienstleistungen werden auf Anforderung in die Station geladen. Ein Verriegelungssystem ist nicht notwendig, da zugehörige Datendateien nur als private Dateien organisiert sein können. Kennzeichen von Public Files ist das Setzen der Dateieigenschaft 'K'.

Anwendungen, die Zugang zu gemeinsamen Dateien (shared files) verwenden, sind von der Benutzerseite aus so zu organisieren, daß Deadlock-Situationen vermieden werden. Dies gilt für den Zugriff auf Dateien oder andere Betriebsmittel wie z.B. Drucker.



## Zentrale KOBUS-Peripherie

Drucker, graphische Peripherie, Modems etc. können lokal an einer KOBUS-Station angeschlossen sein.

Am Master angeschlossene Peripheriegeräte können allen angeschlossenen KOBUS-Stationen zugänglich gemacht werden. Hochwertige Schnelldrucker z.B. können so im Spooling-Betrieb für die einzelnen Arbeitsplätze "quasi-gleichzeitig" arbeiten.

Vorgesehen ist die Erweiterung des Spoolings auf Modem-Anschluß und Datenkommunikation.

Beim Spooling werden vollständige Datensätze (Dateien) von den Stationen zum Druck angemeldet, in eine Warteschlange eingereiht, und nacheinander ausgedruckt.

Diese Warteschlange wird im Master geführt.

Ein Druckauftrag an einem zentralen Drucker lautet (siehe auch 'SPOOL'-Kommando, TB-B):

```
SPOOL dateiname.typ $SLAV $prt
```

Die Angabe von \$SLAV teilt dem Spooler mit, daß der Druckauftrag an einen zentralen Drucker gehen soll (lokales Spooling ist davon unabhängig möglich). \$prt definiert den Ausgabekanal im Master, der z.B. durch den Treiber \$OAP4 belegt ist.

Es können somit mehrere Spool-Kanäle gleichzeitig aktiv sein. Vor dem ersten Druckauftrag an den Master muß am Master die Druckertask aktiviert werden (Kommando "RLOAD PTASK<---").

Die Druckdatei muß auf dem zentralen KOBUS-Massenspeicher stehen.

Lokales Spooling erfordert die Aktivierung der Druckertask im Slave (Kommando "RLOAD PTASK<---"). Der Aufruf lautet:

```
SPOOL dateiname.typ $prt
```

\$prt definiert den Druckertreiber.

## 7. CP/M 2.2 Kompatibilität

Wer mit dem weitverbreiteten Betriebssystem CP/M des Software-Herstellers Digital Research vertraut ist, wird deutliche formale Gemeinsamkeiten mit Kontrons KOS erkennen. KOS ist jedoch benutzerfreundlicher und Hardware-unabhängiger konzipiert, und es bietet in sich den Zugang zum lokalen Netzwerk Kontron KOBUS. Darüber hinaus stehen flexiblere Möglichkeiten in der Datei- und Medienorganisation bei KOS zur Verfügung.

### Formale Gemeinsamkeiten sind:

- '.COM' als Dateityp für Programme
- Startadresse 100 h für Programme voreingestellt
- KOS wie CP/M liegen im obersten Speicherbereich
- Medienanwahl, bei KOS durch Mediennummer 'mn:', bei CP/M durch A:, B:, ...,  
allerdings: KOS kann alle aktiven Medien nach einer Dateireferenz absuchen
- Dateizugriffe über Indexdatei
- interne Kommandos
- 8-stellige Dateinamen, 3-stelliger Dateityp
- '\*' als Universalbezeichner in Dateiadressen
- Systemaufruf über Einsprungvektor möglich

Wegen dieser Gemeinsamkeiten ist der Ablauf von Programmen, die sich an die CP/M-Systemschnittstelle halten, zunächst einmal möglich.

Unterschiede bestehen in

- Medienkonzeption
- Working Directory
- Logische Ein-/Ausgabeorganisation

In diesen Punkten können CP/M-Programme die erweiterten Möglichkeiten von KOS nicht ausnutzen.

Gravierende Unterschiede bestehen jedoch in

- Systemaufrufformat
- Interruptbehandlung
- Zulässigkeit offener Dateien
- direkte Systemeinsprünge
- Zugriff auf Speicheradressen

Diese Unterschiede machen eine Prüfung der Ablauffähigkeit im einzelnen notwendig.

Die wesentliche Anpassung von CP/M-System-Aufrufen bietet der Umsetztreiber \$CPM. Dieses Modul ist als Treiber organisiert und wird wie ein Ein-/Ausgabetreiber aktiviert. Er wirkt als ein dem Betriebssystem KOS vorgeschalteter Umsetzer, indem er die CP/M 2.2-Systemaufrufe in KOS-Systemaufrufe umwandelt (s. Utility/Umsetztreiber für CP/M-Aufrufe \$CPM).

Parallel dazu arbeitet bei KOS5 der Bildschirmvortreiber XMON, der die X-Y-Koordinaten-orientierten Bildschirmausgaben empfängt und in die Hardware-spezifischen relativen Bildspeicheradressierungen umwandelt. XMON ist durch das KOS-Kommando 'RLOAD XMON' zu aktivieren für Programme, die Terminal-ähnliche Bildschirmausgabe voraussetzen, z.B. WordStar etc. In KOS6 ist die XMON-Funktion direkt integriert.

Bisher mit Erfolg erprobt als ablauffähig unter KOS mit CP/M-Treiber sind folgende CP/M-Programme:

MICROSOFT:	MBASIC	Redding:	LYNX
	BASCOM		
	FORTRAN80	SORCIM:	SUPERCALC2
	COBOL80		
	L80	Ashton-Tate:	dBASE II
	LD80		
MICRO PRO:	WORDSTAR	DIGITAL RESEARCH:	Pascal MT+
	WORDINDEX		
	DATASTAR	Micro Focus:	CIS-COBOL
	SUPERSORT		COBOL Level II

und weitere.

**Bei welchen Produkten muß nun mit Anpassungsproblemen gerechnet werden?**

Dies sind:

- Hardware-spezifische Programme
- Programme mit speziellen Terminalbedingungen
- Programme mit Systemeinsparungen außerhalb der Standard CP/M-Systemaufrufe

Diese Anpassungen erfordern Aufwand, die Anpassung ist jedoch zumindest bei Vorliegen des Quellcodes immer lösbar.

**Genauere Prüfung erfordern folgende Umstände:**

- **Speicherbedarf:** Hier schneidet KOS6 besser ab als CP/M, KOS5 jedoch benötigt mehr Speicher als CP/M. Je nach Art des Programms wird die Anpassung möglich oder sehr schwierig sein, falls der Punkt 'Speicher' Probleme aufwirft.
- **Interrupt- und Stack-Probleme:** KOS ist ein Interrupt-orientiertes Betriebssystem. Jeder Interrupt belegt Stackraum. In Programmen, die mehrere kleine Stacks anlegen, treten sporadische Fehlfunktionen auf, wenn der Stack in Programm- oder Datenbereiche überläuft. Eine Programmanpassung ist möglich, wenn der Quelltext vorliegt, sie kann problematisch sein, wenn die Stackadressen dynamisch allokiert werden. Der Stackbereich wird von KOS dynamisch im Speicher verwaltet, die Größe beträgt 256 Bytes.

CP/M kompatible Programme, die den Stackbereich auf eine geringere Größe umdefinieren, können Fehlfunktionen des Gesamtsystems auslösen.

Beim Ablauf von CP/M kompatiblen Programmen gilt daher folgendes zu beachten:

- in welcher Weise wird der Stack verwaltet?
- auf welche Größe wird der Stack definiert?
- hat der Anwender Einfluß auf die Größe der Stacks?

Stellvertretend für eine Reihe von CP/M Software sei hier als Beispiel die Stackverwaltung von PASCAL MT+ Programmen dargelegt:

PASCAL MT+ Programme dimensionieren den Stackbereich auf 128 Byte (PASCAL MT+ Handbuch, Abschnitt 4.11).

Der Hersteller von PASCAL MT+, Digital Research, empfiehlt bei Interrupt betriebenen Systemen den Stack entsprechend zu erhöhen und stellt dafür eine Variable zur Verfügung.

Um Konflikte durch zu klein dimensionierten Stack zu vermeiden, ist in PASCAL MT+ Programmen der voreingestellte Wert von 128 Bytes auf 256 Bytes zu erhöhen.

Bei Kobusanlagen empfiehlt sich eine Stackgröße von 384 Bytes in PASCAL MT+ Programmen zu definieren.

- Offene Dateien und DSB-(FCB-)Zugriffe: KOS verwaltet bis zu 16 offene Dateien. CP/M-Programme haben die Tendenz, beliebig viele Dateien offen zu halten, oder auf geschlossene Dateien wieder zuzugreifen. Der CP/M-Translator umgeht diese Problematik, indem er offene Dateien selbst verwaltet und automatisch schließt, so daß CP/M-Programme keine offenen Dateien zurücklassen können.
- Aufrufverhalten: Der CP/M-Translator schaltet sich beim ersten CP/M-Systemaufruf ein. Dieser darf jedoch kein Aufruf CREATE, OPEN oder DELETE sein. Falls Programme solchermaßen beginnen, stehen folgende Möglichkeiten offen: "Patches" des Programmes, um einen Dummy System Call einzufügen, oder Vorschalten eines Dummy Programmes mit Chaining des eigentlichen Programmes.
- Bei Zugriffsfehlern auf Speichermedien erzeugt \$CPM die Fehlermeldung "\$CPM-KOS I/O-Error:nn", mit nn= 82...86, entsprechend den KOS-Fehlercodes. Die darauffolgende Benutzereingabe "ESC" bricht das laufende Programm ab, durch Eingabe von "RETURN" wird ins laufende Programm zur Fehlerbehandlung zurückverzweigt.

Insgesamt bedeutet die Anpassung von CP/M-Software an KOS dann keine oder wenig Mühe, wenn der Programmierer sich an die definierten CP/M-Systemaufrufe gehalten hat und es vermieden hat, CP/M-interne Abläufe und Speicherzellen zu verwenden. Dies jedoch sind Grundsätze jeder 'sauberen' Programmierung und sollte somit keine Einschränkung bedeuten.

**Welche Vorgehensweise empfehlen wir Ihnen bei CP/M-Software-Problemen?**

Es ist die gleiche, wie bei anderen Software-Problemen auch:

- Versuchen Sie, das Problem in eine einfach reproduzierbare Form zu bringen. Dies führt zu einer klaren Problemdefinition und möglicherweise bereits zur Lösung oder Umgehung des Problems.
- Teilen Sie uns - bitte schriftlich, ein Formularvorschlag dazu ist im Anhang abgedruckt - Ihre Fragestellung mit. Programmbeispiele auf Diskette ermöglichen uns im allgemeinen die schnellere Klärung Ihrer Frage.

**Und als einfachste und doch wichtigste Maßnahmen:**

- Verwenden Sie fehlerfreies, für Kontron PSI-Systeme zugelassenes Diskettermaterial.
- Prüfen Sie die logische Konsistenz und physikalische Funktionalität Ihrer Medien (STATUS, FSCHECK, CHMED, FORMAT P V).
- Verwenden Sie freigegebene aktuelle Software-Stände.
- Beachten Sie die Grundregeln über Medieninitialisierung, Treiberorganisation etc, wie sie in diesem Handbuch beschrieben sind.

**8. Gewährleistung bei Software**

Betriebssysteme, System- und Anwendungsprogramme beinhalten komplexe Aufgabenstellungen. Fehler, Inkonsequenzen und unerwünschte Reaktionen bei Fehlbedienung sind weitgehend durch intensive Tests ausgeschlossen, die Garantie der Fehlerfreiheit kann jedoch kein Hersteller von Rechnern und Software übernehmen. Kontron ist bemüht, Ihnen nach dem Stand der Technik das bestmögliche Produkt zu liefern. Kontron übernimmt keine Gewähr für die Eignung von Software für eine bestimmte Anwendung.

Bei Beachtung der anerkannten Regeln der Kunst des Programmierens und der Rechnertechnologie werden Sie mit Kontron PSI-Rechnern leistungsfähige und benutzerfreundliche Hardware/Software Systeme für Ihre Anwendung aufbauen können.

Bei auftretenden Problemen sind wir bereit, an der Lösung zusammen mit Ihnen zu arbeiten, sei es durch Verbesserung der Dokumentation, durch Schulungskurse oder durch Test und Analyse von Problemstellungen. Weitergehende Haftungen und Ansprüche sind ausgeschlossen.

## **Systemkommandos zum Betriebssystem KOS**

Version 4.33/5.44/5.54/6.05

Juli 1984

Dieser Teil des Handbuches beschreibt die Dienstprogramme der Betriebssysteme KOS4/5/6.

U.a. beinhaltet es Kommandos zur Handhabung von Dateien (COPY, DELeTe, DUMP, MOVE, PRINT, RENaMe), zur Einrichtung von Ein-/Ausgabetreibern (INISER, IODC), zur Systemsteuerung (TASK, SPOOL) und zur Systemüberwachung (STATUS, STOP). Außerdem sind hier die Programme zur Disketten- und Medienverwaltung (COPYM2, FORMAT, IL) beschrieben.

Die Programmsysteme zur Erstellung von Programmen (BASIC, ASSEMBLER, LINKer, EDITor, KDM) und die Hilfsprogramme (Utilities) sind in eigenen Abschnitten dokumentiert

Die für das Verständnis notwendigen Voraussetzungen sind im Abschnitt "Konzepte" dieses Handbuches beschrieben.



**Inhaltsverzeichnis**

1. **Übersicht**  
Verwendete Nomenklatur
  
2. **KOS - interne Systemkommandos**
  - A - Allokation von Speicherbereichen
  - C - Groß-/Kleinumschaltung, close driver/schließe Treiber
  - D - Deallokation von Speicherbereichen
  - H - History
  - I - Inhaltsverzeichnis von Medien
  - M - Mastermedium
  - N - Neuinitialisierung von Medien und Treibern
  - O - Open driver / öffne Treiber
  - P - Paging, Blättern im Bildspeicher
  - R - Repeat, Kommandowiederholung
  - S - Abspeichern in Dateien
  - X - Ausführen von bereits geladenen Programmen
  - Y - SYNC-Kommando
  
3. **KOS - diskresidente Systemkommandos**
  - BACKUP - Sichern von Festplatte auf Diskette
  - BBR - Behandlung von 'Bad Blocks'
  - CEDIT - Zeichensatz definieren (980 H)
  - CHMED - Überprüfen von Medien
  - COLOR - Rand-, Hintergrund-, Vordergrundhelligkeit (980 H)
  - COPY - Kopieren von Dateien und Medien
  - COPY1
  - COPYD2
  - COPYM2
  - COPYWFD
  - CPFILE - Vergleich von Dateiinhalten
  - CS10D - Zeichengenerator laden (980 H)
  - CS16D
  - CS10E
  - CS16E
  - DATE - Systemdatum
  - DEFP - Definition von Dateieigenschaften
  - DEL - Löschen von Dateien
  - DO - Ausführen von Kommandodateien
  - DUMP - Ausdruck von binären Dateien
  - FORMAT - Formatieren von Medien
  - FSCHECK - Logische Überprüfung von Medien
  - IL - Inhaltsverzeichnis von Medien
  - INFO - Bildschirm-orientierter Ausdruck von ASCII-Dateien
  - INISER - Initialisierung der Serienschnittstellen (KOS 4/5)
  - IODC - E/A-Treiber - Steuerung
  - KOKOM, KOKOMS - KOBUS - Überwachung
  - KOSGEN - Konfigurierung der Systemdateien
  - KPPLOAD KPP - Laden der KOBUS-Master Software
  - MAKEFS - Logische Formatierung von Medien
  - MAP - Ausdruck der Speicherbelegung
  - MOVE - Kopieren von Dateigruppen
  - PRINT - Ausdruck von ASCII-Dateien
  - PSTAT - Ausdruck des Programmstatus
  - REN - Umbenennen von Dateien
  - RLOAD - Relokativer Lader
  - RLOAD PTASK - Druckerüberwachung
  - RLOAD SELECT - Kommandomenüs



RLOAD TIME	- Uhrzeittask
SETDATE	- Datum/Uhrzeit setzen
SETTIME	
SPOOL	- Druckausgabesteuerung
STATUS	- Medienstatus
STOP	- Anhalten von Kommandodateien
SYSGEN	- Login-Eintrag in KOS
TASK	- Tasksteuerung
WDIR	- Working Directory-Verwaltung

## 1. Übersicht

Zum Betriebssystem KOS der Kontron PSI Computer gehören eine Reihe von Dienstprogrammen zur Datei- und Massenspeicherbehandlung, zur Ein-/Ausgabesteuerung und zur Systemverwaltung.

Diese Dienstprogramme werden im folgenden als "Kommando" bezeichnet. Ein Kommando wird unter KOS durch Eingabe seines Namens über die Tastatur oder bei der Ausführung einer Kommandodatei zur Ausführung gebracht.

Eine Reihe von Grundkommandos ist bereits in das Betriebssystem KOS integriert und zusammen mit dem Betriebssystem stets im Speicher zugänglich. Diese Kommandos sind KOS-intern oder "resident".

Alle weiteren Kommandos werden zur Ausführung von einem Massenspeicher in den Speicher der Zentraleinheit geladen. Diese "externen" Kommandos stehen gleichberechtigt neben Anwendungsprogrammen.

Die Kommandoeingabe erfolgt nach dem Prompt-Zeichen 'KOS:' durch Eingabe des Kommandonamens und der gewünschten Parameter, gefolgt von der 'RETURN'-Taste. Mehrere Kommandos können in einer Kommandozeile zusammengefaßt sein, dabei werden die einzelnen Kommandos durch ';' voneinander getrennt. Ein Kommandoname, der durch ',' abgeschlossen ist, führt dazu, daß das Programm nur geladen, jedoch nicht ausgeführt wird.

Die meisten Kommandos führen einen Dialog mit dem Anwender und geben Hilfestellung durch Syntaxausgaben bei Fehlbedienung.

Es bestehen keine Unterschiede in der Handhabung der Kommandos für Systeme der Serien PSI80/82 Kontron und PSI9xxx Kontron. Die folgende Beschreibung gilt gleichermaßen für die Verwendung unter KOS4.33/5.33 KOS5.4x/KOS5.5x und KOS6.x. Auf Erweiterungen des Funktionsumfanges wird jeweils explizit verwiesen.

Die hier vorkommenden Begriffe Medium, Working Directory (WDIR), Ein-/Ausgabekanal etc. sind im Kapitel "Konzepte" erläutert. Dort finden Sie auch die detaillierte Beschreibung der Kommandoeingabe und der grundsätzlichen Syntaxregeln.

**Verwendete Nomenklatur:**

In der folgenden Zusammenstellung werden folgende Symbole bzw. Abkürzungen verwendet:

<b>adr</b>	Adressangabe (hex)
<b>date</b>	Datum
<b>dateigruppe</b>	Mehrdeutiger Dateiname, z.B. '1*'
<b>I</b>	Eingabekanal
<b>M</b>	Medienkanal
<b>medn</b>	Medientreibername
<b>mn</b>	Mediennummer
<b>mn1</b>	Mediennummer Quelle
<b>mn2</b>	Mediennummer Ziel
<b>n</b>	Anzahl (hex)
<b>name</b>	Dateiname (bis zu 8 Stellen)
<b>name1</b>	Quelldateiname
<b>name2</b>	Zieldateiname
<b>O</b>	Ausgabekanal
<b>param</b>	Parameterfolge
<b>props</b>	Properties
<b>query</b>	Rückfrage (J/N)
<b>typ</b>	Dateityp (bis zu 3 Stellen)
<b>typ1</b>	Quelldateityp
<b>typ2</b>	Zieldateityp
<b>wdir</b>	Working Directory (bis zu 15 Stellen)
<b>wdir1</b>	Working Directory der Quelle
<b>wdir2</b>	Working Directory des Ziels
<b>x</b>	Kanalnummer
<b>\$iod</b>	Input/Output Driver (Ein-/Ausgabe-Treiber)
<b>\$iodd</b>	Input/Output Driver, Ziel
<b>\$iodn</b>	Input/Output Driver, Name
<b>\$iods</b>	Input/Output Driver, Quelle
<b>&lt;—</b>	RETURN-Taste

## 2. KOS-interne Kommandos

KOS-interne Kommandos werden beim Kaltstart des Betriebssystems mitgeladen. Sie sind Teil des Betriebssystems und stets speicherresident. Alle internen Kommandos verwenden die Kanäle I-1 bzw. O-1 für Ein-/Ausgaben. Interne Kommandos bestehen grundsätzlich aus einem Kennbuchstaben, der sowohl groß, als auch klein geschrieben werden kann.

Bei Kontron PSI80/82-Systemen unter KOS 5.4x/5.5x ist zu beachten, daß aus Gründen der Speicherplatzersparnis in der Kurzversion (short version) des Betriebssystems einige residente Kommandos entfallen. In der Langversion stehen auch diese zur Verfügung.

### A - Kommando (Allocation) (nicht bei KOS5-Kurzversion)

Aufruf: A adr n<---

Funktion:

Belegung von 'n' Speichersegmenten von je 128 Byte ab Adresse 'adr'. Hiermit können beliebige noch freie Speicherbereiche als belegt deklariert werden (siehe MAP-Kommando). Der Versuch, bereits belegte Segmente erneut zu belegen, wird mit der Fehlermeldung 'Speicherbelegungskonflikt' beantwortet. 'adr' und 'n' sind hexadezimale Zahlen.

### C - Kommando (Change Case)

Aufruf: C<---

Funktion:

Umschaltung zwischen Groß- und Kleinschreibung. Nach dem Laden des Betriebssystems ist 'Großschreibung' eingestellt. Als Hinweis auf den gerade aktiven Modus gibt das Betriebssystem die Meldung 'KOS:' klein- oder großgeschrieben aus.

Bei KOS6 wird das Promptzeichen nicht verändert.

**Hinweis:** Bei Kontron PSI980 H muß vorher mit CSxxx der vollständige Zeichensatz geladen werden!

Ab KOS6.03 dient das C-Kommando mit Parametern auch zum Setzen und Löschen von Zählern des TCNT-Kommandos (s. Abschnitt "Utilities").

Ab KOS 6.05 dient das C-Kommando zusammen mit einem Treibernamen (close driver/schließe Treiber) auch zum Positionieren der Winchesterköpfe auf Ihre Landeposition (Transportsicherung), z.B. 'C \$WINO<---'.

### D - Kommando (Deallocation) (nicht bei KOS5-Kurzversion)

Aufruf: D adr n<---

Freigabe von 'n' Speichersegmenten von je 128 Bytes ab Adresse 'adr'. Die Größen 'n' und 'adr' sind hexadezimale Zahlenwerte.

### H - Kommando (History) (nur bei KOS6)

Aufruf: H param<---

#### Funktion:

Dieses Kommando ermöglicht die Auflistung von bisher ausgeführten Kommandozeilen, wobei jeder Zeile eine zweistellige Dezimalzahl zugeordnet ist. Der Pufferbereich des History-Kommandos umfaßt 512 Bytes, was in der Regel ausreicht, um die letzten 40 Kommandozeilen zu speichern.

Neben der einfachen Auflistung kann durch die Angabe einer Nummer die zugehörige Kommandozeile erneut ausgeführt werden.

#### Beispiele:

H<--- ; Auflistung der Kommandos im History-Puffer  
H 10<--- ; Ausführung des Kommandos mit der Nummer 10  
H 10 param<--- ; Ausführung des Kommandos mit der Nummer 10.  
Der String 'param' wird an die Kommando-  
zeile Nummer 10 angefügt.

### I - Kommando (Index) (nicht bei KOS5-Kurzversion)

Aufruf: I /wdir/mn:dateiname.typ<---  
I /wdir/mn:dateigruppe<---

#### Funktion:

Auflistung aller im Parameterfeld des Kommandos spezifizierten Dateien, die nicht geheim (s. DEFP-Kommando) sind. Alle Parameter sind optional. Ist das Parameterfeld nicht besetzt, so werden alle nicht geheimen Dateien des Mastermediums ausgegeben, die im aktuellen wdir oder mit Kennung P=K (public) versehen sind.

#### Beispiele:

I ABC.\*<--- alle nicht geheimen Dateien mit Namen 'ABC'  
I ???B.AS<--- alle nicht geheimen Dateien mit Typ 'BAS'  
I 3:<--- alle nicht geheimen Dateien auf Medium 3  
I<--- alle nicht geheimen Dateien auf dem Master-  
medium

Wird keine Datei der angegebenen Spezifikation gefunden, so antwortet KOS mit der Meldung:

Medium mn: Datei nicht vorhanden.

**M - Kommando (Master Medium)**

Aufruf: M mn&lt;---

**Funktion:**

Definition und/oder Ausgabe des Mastermediums. Ist kein Parameter spezifiziert, wird nur die Nummer des derzeitigen Mastermediums ausgedruckt. Das Mastermedium ist dasjenige Medium, auf dem bei Dateireferenzen mit fehlender Mediennummer zuerst gesucht wird.

**N - Kommando (Neue Initialisierung)**Aufruf: N \$iodn<---  
N mn<---**Funktion:**

Neuinitialisierung der Dateiverwaltung oder eines E/A-Treibers. Die Funktion wird automatisch beim Wechsel von Disketten durchgeführt, sofern die Diskettenamen sich voneinander unterscheiden. Wenn zwei Disketten den gleichen Namen haben, dann muß das N-Kommando beim Wechsel gegeben werden. Es werden dann auf den angesprochenen Medien alle offenen Dateien geschlossen.

Das Kommando 'N \$iodn' führt zur Neuinitialisierung eines E/A-Treibers. Es wird die Routine 'INIT' des E/A-Treibers ausgeführt.

**Beispiele:**

N \$SIOA<---	Initialisierung des E/A-Treibers \$SIOA
N 2<---	Initialisierung des Mediums 2
N<---	Initialisierung aller aktiven Medien

**O - Kommando (Öffne Treiber)**  
(nur bei KOS6)

Aufruf: O \$iodn&lt;---

**Funktion:**

Treiber, die in ein Betriebssystemmodul eingebunden sind (z.B. \$CPM in KOSB.SYS) werden automatisch beim Laden von KOS6 aktiviert, vorausgesetzt, daß das entsprechende Flag in KOS0.SYS gesetzt ist (siehe KOSGEN-Kommando). Vor der ersten Verwendung des Treibers muß dieser mit dem Systemkommando "öffne Treiber" eröffnet werden.

**Beispiel:**

O \$CPM&lt;---

**P - Kommando (Page Mode)**

Aufruf: P<---  
P n<---

**Funktion:**

Blättern in den bisherigen Sichtschirmausgaben. Der Bildwiederhol-speicher des Kontron PSI-Systems speichert ständig die letzten 8 Textseiten. Nach dem P-Kommando kann mit den Cursortasten ein be-liebiger Ausschnitt der letzten 8 Textseiten auf den Sichtschirm gebracht werden.

Durch dieses Kommando können z.B. Bedienfehler oder verpaßte Ausgaben zurückgeholt werden.

**Tasteninterpretation:**

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 Bildaus-schnitt wieder zurück und schließen das P-Kommando ab.

Bei KOS6 stehen vier voneinander unabhängige Seitengruppen mit jeweils 8 Seiten zur Verfügung (1 Seitengruppe mit 32 Seiten bei 980H-Systemen). Das Umschalten der Seitengruppen erfolgt durch das Kommando

P n<---

mit n = 0...3. Beim Umschalten auf eine Seitengruppe n wird der jeweils letzte Bildinhalt dieser Seitengruppe angezeigt. Danach arbeitet das P-Kommando ohne Gruppenanwahl wieder ausschließlich innerhalb der angewählten Seitengruppe. Aus Anwendungsprogrammen heraus bewirkt die Zeichenfolge "ESC/N" (n = 0...3) das Umschalten.

**R - Kommando (Repeat)**  
(nicht bei KOS5-Kurzversion)

Aufruf: R kommando1;kommando2;...kommandoN<---

**Funktion:**

Definition eines Kommandomacros. Die im Parameterfeld des R-Kommandos aufgeführte Kommandofolge wird zwischengespeichert und kann an-schließend beliebig oft durch ein R-Kommando ohne Parameterangabe wiederholt werden. Ein neues Kommandomacro (R mit Parameterangabe) überschreibt ein eventuell bereits vorhandenes Kommandomacro.

**Beispiel:**

R EDIT TEST.SRC;ASM=TEST/L;LINK TEST/N,TEST/P:100/E<---

Die Ausführung der Kommandofolge EDIT-ASM-LINK kann nun beliebig oft durch das Kommando R<--- wiederholt werden.

**S - Kommando (Save)**  
(nicht bei KOS5-Kurzversion)

Aufruf: S n /wdir/mn:dateiname.typ adr<---

**Funktion:**

Abspeichern (Retten) von n x 128 Bytes aus dem Anwenderspeicherbereich in eine Datei. 'n' und 'adr' sind Hexadezimalzahlen. Der Abspeichervorgang beginnt bei Adresse 'adr'. Fehlt dieser Parameter, so wird dafür der Wert 100H eingesetzt.

**Beispiele:**

S A XDATEI.OBJ<---

Abspeichern des Bereichs 100H bis 5FFH unter dem Namen XDATEI.OBJ.

S 28 ABC.DEF<---

Abspeichern des Bereichs 100H bis 14FFH unter dem Namen ABC.DEF.

S 10 PRT.EAT D000<---

Abspeichern des Bereichs D000 bis D7FF unter dem Namen PRT.EAT.

Bei allen aufgeführten Beispielen wird das Mastermedium adressiert. Der Wert von 'adr' wird mit im Inhaltsverzeichnis abgelegt.

**X - Kommando (Execute)**

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

**Funktion:**

Wiederholung eines zuvor geladenen oder ausgeführten Programms ohne erneutes Laden. Die Angabe der Parameter p1...pN ist optional; ihre Art und Anzahl ist vom jeweils wieder zu startenden Programm abhängig.

**Beispiel:**

```
PRINT,<---  
X KOS.INF<---  
X KOS.INI<---
```

Laden des Programms PRINT (das Komma verhindert die sofortige Ausführung - nun ist beispielsweise ein Diskettenwechsel möglich). Anschließend werden die Dateien 'KOS.INF' und 'KOS.INI' ausgedruckt, ohne daß das Dienstprogramm 'PRINT' neuerlich geladen werden muß.

Hinweis: Programme, die auf CP/M-Systemen ablauffähig sind, können i.a. nicht mit dem X-Kommando wieder gestartet werden. Sie müssen zur Initialisierung neu von einem Medium geladen werden.



**Y-Kommando (sYnchronisiere \$WINx)**  
(nur bei KOS6)

Aufruf: Y<---

**Funktion:**

Unter KOS6 arbeiten die Winchester-Treiber mit einem "Look Ahead"-Mechanismus, der den Zugriff auf die Platten optimiert und dadurch beschleunigt. Durch die Optimierung der Zugriffe kann es vorkommen, daß ein Schreibvorgang auf eine Platte nicht sofort stattfindet. Im Betriebssystem KOS wird innerhalb von etwa 10 Sekunden nach dem letzten Plattenzugriff automatisch ein eventuell gepufferter Schreibvorgang ausgeführt.

Das Y-Kommando sorgt dafür, daß sofort alle Plattenpuffer geleert werden. Wird die Frage 'Stop System' mit y (ja) beantwortet, so werden alle Medientreiber geschlossen. Dies bedeutet unter anderem, daß die Köpfe der Festplattenlaufwerke auf die Landeposition gefahren werden (bei Adaptec-Controller). Danach kann das System abgeschaltet oder neu gebootet werden ('BOOT KOS?' y/n). Es wird geraten, vor jedem Ausschalten des Rechners das Kommando 'Y' einzugeben. Nach dem Erlöschen der Plattenbetriebsanzeige kann dann das System ausgeschaltet werden.

### 3. Diskresidente KOS-Kommandos

Kommandos dieser Gruppe sind als Dateien auf beliebigen Medien resident und werden bei ihrem Aufruf in den Speicher geladen. Dateien, die ausführbare Programme enthalten, sind durch den Typ 'COM' gekennzeichnet.

#### **BACKUP - Kommando (Erstellen von Sicherungsdisketten für Festplatten)**

Aufruf: BACKUP<---

E/A-Kanäle: Eingabe ---> I-1  
Ausgabe ---> O-1

#### **Funktion:**

Das Programm BACKUP fertigt ein 1:1 Abbild des von Dateien belegten Teils von Festplattenmedien auf Disketten an. BACKUP betrachtet die Festplatte als Folgen von 1 kByte großen Blöcken, die in der gleichen Reihenfolge wie auf dem Plattenspeicher auf Disketten abgelegt werden. Es ist nicht erforderlich, jedesmal den gesamten Inhalt der Festplatte zu kopieren. Das Betriebssystem vermerkt im Inhaltsverzeichnis welche Blöcke seit dem letzten Backup geändert wurden. BACKUP überträgt dann nur die geänderten Blöcke.

Das Programm arbeitet mit Benutzerführung. Es beinhaltet folgende Möglichkeiten:

- logische Formatierung der Backup Disketten
- Daten von Festplatte auf Floppy Disk schreiben
- Daten von Floppy Disk auf Festplatte zurückschreiben

BACKUP arbeitet laufwerkunabhängig über Medienkanäle. Es errechnet sich die Kapazität des Backup-Mediums in Schritten zu  $2 \text{ hoch } n$  mal 128 kByte ( $n=0, 1, \dots$ )

BACKUP wird bei Übertragungsfehlern mit einer Fehlermeldung abgebrochen. Verwenden Sie zur Datensicherung nur einwandfreies durch das Kommando 'FORMAT P V' geprüftes Diskettermaterial.

#### **Inhalt der Backup-Diskette**

Das Inhaltsverzeichnis aller Disketten ist gleich aufgebaut und besteht aus drei Einträgen.

Inhaltsverzeichnis  
Datei zur Identifikation der Diskette  
Backup-Datei

Die Identifikationsdatei wird während der logischen Formatierung unter BACKUP automatisch erzeugt. Sie enthält die Nummer der laufenden Diskette (beginnend mit Null) und einem Identifikationstext. Der Text darf maximal 100 Zeichen lang sein, jedoch ist darauf zu achten, daß alle Disketten eines Satzes in den ersten 8 Zeichen dieses Textes übereinstimmen.

Die BACKUP Datei wird ebenfalls während der Formatierung in ihrer vollen Länge (128 kByte bei single density, 256 kByte bei double density, 512 kByte bei double density/double sided Disketten) erzeugt. In diese Datei werden die Blöcke der Festplatte eingetragen.

### **Daten von Festplatte auf BACKUP-Disketten**

Oft wird die volle Kapazität der Festplatte nicht ausgenutzt und deshalb ist es möglich, mit einer kleineren Anzahl von Backup Disketten auszukommen. Sollte sich der Inhalt der Festplatte seit dem letzten Backup vergrößert haben, so müssen weitere Diskette nachformatiert werden. Wurde ein kompletter Satz Backup Disketten neu formatiert, muß nur das erste Mal der vollständige gültige Inhalt von der Festplatte kopiert werden ('alles' oder 'nur Änderungen'). Die Auswahl erfolgt unter Benutzerführung.

BACKUP verlangt die Angabe von Mediennummern von Festplatte und Floppy Disk und die ersten 8 Zeichen des Identifikationstextes. BACKUP fordert die benötigten Disketten nacheinander an. Bei korrekter Disketten-Nummer und übereinstimmendem Text wird der Nachtrag in die Backup-Datei durchgeführt, ansonsten erfolgt eine Fehlermeldung. Dieser Vorgang ist mit allen verlangten Disketten zu wiederholen, bis BACKUP das Ende meldet.

Am Ende des Backup-Vorganges wird auf der Festplatte die Kennzeichnung der geänderten Blöcke gelöscht. Dies ist erforderlich, damit beim nächsten Backup wiederum nur die geänderten Daten berücksichtigt werden. Der Benutzer hat jedoch die Wahl, eine Löschung der Kennzeichnung zu verhindern; auf diese Weise könnte z.B. ein zweiter Satz von Backup-Disketten auf den neuesten Stand gebracht werden.

### **Daten von Backup Disketten auf Festplatte (Restaurieren)**

Auch hier verlangt BACKUP die Angabe der Mediennummern und die ersten 8 Zeichen des Identifikationstextes. Nur bei korrekter Identifikation der Diskette wird der Kopiervorgang ausgeführt, sonst erfolgt eine Fehlermeldung. Da BACKUP nicht feststellen kann, ab welcher Diskettennummer keine neuen Blöcke mehr vorhanden sind, sollte der komplette Satz kopiert werden.

### **Logische Formatierung der Backup Disketten**

Backup setzen physikalisch formatierte Disketten voraus. Dies erfolgt mit dem FORMAT-Kommando durch 'FORMAT P V<---'. Beachten Sie bitte, daß nur einwandfreies, getestetes Diskettermaterial verwendet wird! Ebenso wird vorausgesetzt, daß die Festplatte einwandfrei ist, d.h., daß ihre Statusmeldung weder 'inconsistent' ist noch defekte Records enthält.

Anschließend müssen die Disketten den Formatierungsprozess des BACKUP-Programms durchlaufen, welche die Disketten initialisiert und mit den nötigen Dateien versieht.

Es können alle oder auch nur einzelne Disketten eines Satzes formatiert werden. Die maximal benötigte Anzahl von Disketten hängt von der Kapazität der Floppies und der Festplatte ab. Diese Zahl wird errechnet und als Hinweis ausgedruckt (dezimal). Ist die Festplatte nur schwach ausgelastet, genügt es, eine entsprechend geringe Anzahl von Backup Disketten zu formatieren. Pro 1000 kByte werden ca. 4 Disketten double density oder 2 Disketten double density/double sided benötigt. Wächst später der Inhalt der Festplatte, müssen vor dem Kopiervorgang die zusätzlich benötigten Disketten formatiert werden. Zu beachten ist, daß die ersten 8 Zeichen des Identifikationstextes mit denen der anderen Disketten übereinstimmen. Das Inhaltsverzeichnis, die Identifikationsdatei, sowie die Backup Datei werden automatisch angelegt.

#### **Abbruch des Programms BACKUP**

BACKUP ist so ausgelegt, daß vor jeder größeren Aktivität gefragt wird, ob der Benutzer den Vorgang fortsetzen möchte. Ist dies nicht der Fall, wird auf die vorhergehende Frage zurückgeschaltet. Eventuell fehlerhafte Eingaben können nun wiederholt werden, ohne das ganze Programm von vorne zu starten.

Die Datensicherung ist jedoch nur dann vollständig, wenn BACKUP 'Ende' meldet!

Hinweis: Weitere Sicherungsmöglichkeiten siehe COPYM2-Kommando mit \$XMED und Backup über Wechselplatte in Kontron PSI-Systemen.

**BBR - Kommando (Bad Block Replacement)**  
(nur KOS6)

Aufruf: BBR&lt;---

E/A-Kanäle: Eingabe ----> I-1  
Rückmeldungen ----> O-1  
Protokollausgabe ----> O-2

**Funktion:**

Auffinden und Ersetzen von fehlerhaften Blöcken auf Plattenspeichern nach dem Formatieren der Platte durch das Kommando WFD bzw. FORMATW (siehe Abschnitt Utilities).

'BBR' sucht automatisch das beim Formatieren auf die Platte geschriebene Datenummuster und überprüft alle Sektoren der Platte auf Lesbarkeit und Datenrichtigkeit. Zu diesem Zweck ermittelt 'BBR' interaktiv den Prüfmodus und die logische Mediennummer der Platte.

'BBR' gibt über den Ausgabekanal O-2 ein Protokoll aus, in dem alle aufgefundenen 'Bad Blocks' aufgelistet sind. Es ist zweckmäßig, dem Kanal O-2 einen Druckertreiber zuzuweisen, um so für jedes Plattenlaufwerk ein Protokoll zu erhalten.

'BBR' kann jederzeit mit ESCAPE abgebrochen werden. Nur bei vollständigem Durchlaufen der Prüfphase werden Info-Records über 'Bad Blocks' in einen reservierten Bereich der Platte geschrieben. Werden zu viele Fehler entdeckt, beendet 'BBR' den Programmablauf und schreibt keine Info-Records. Die Info-Records werden von den Medientreibern bei ihrer Initialisierung gelesen und ausgewertet. 'Bad Blocks' werden automatisch durch intakte Blöcke des Reservebereichs ersetzt und treten somit nach außen nicht in Erscheinung.

Nach dem Ablauf von 'BBR' ist das Kommando MAKEFS anzuwenden, um ein logisches Dateisystem auf dem Medium anzulegen.

Hinweis: Ein physikalisches Neuformatieren ist i.a. nur bei Wechselplatten notwendig. Für weitere Informationen siehe WFD und FORMATW Dokumentation im Abschnitt "KOS-Utilities".

**CHMED - Kommando (Überprüfen von Medien auf Lesbarkeit)**

Aufruf: CHMED<---

E/A-Kanäle: Eingabe ---> I-0  
Ausgabe ---> O-1

**Funktion:**

Mit CHMED wird die Lesbarkeit aller Records (= Datensatz mit 128 Bytes) eines Mediums geprüft. Hierzu wird der KOS-Systemaufruf 'Read Logical Record' verwendet. Der Anwender kann somit feststellen, ob alle Records eines beliebigen Mediums grundsätzlich lesbar sind. Dieses Programm sagt jedoch nichts über die Richtigkeit der gelesenen Daten aus, da ein richtiges Arbeiten der Fehlererkennung und Fehlerkorrektur des Massenspeichers vorausgesetzt wird.

Im Dialog werden nur die Mediennummer und der Prüfmodus verlangt. Mit dem Prüfmodus kann angegeben werden, ob das Programm nach Erkennen eines Fehlers anhält und eine Benutzeraktivität verlangt oder ob es nur den Fehler anzeigt und den Prüfvorgang fortsetzt. Nach Erkennen eines Fehlers kann der Record noch einmal gelesen oder das Programm fortgesetzt werden. CHMED kann jederzeit mit der ESC-Taste abgebrochen werden.

Neben der Anzeige des Records, der als nächstes gelesen wird, zeigt CHMED die Anzahl der nicht lesbaren Records an. Wenn ein Record nicht lesbar ist, so wird der Fehlerzähler nicht um 1, sondern um die Anzahl der Records pro physikalischen Block (= Sektor) erhöht. Ebenso wird das Lesen nicht mit dem nächsten Record fortgesetzt, sondern mit dem ersten Record des nachfolgenden Blocks.

Wenn ein Record nicht lesbar ist, so ist der gesamte Block (Sektor), in dem der nicht lesbare Record enthalten ist, nicht lesbar.

**Abhilfe im Fehlerfall:**

Medium neu formatieren und TMED/BBR/MAKEFS anwenden (s. dort).

Hinweis: Das Utility-Programm TMED (Test Medium) erlaubt einen Test mit kritischen Daternustern. Dieses Programm ist dann zu empfehlen, wenn Laufwerke bzw. Disketten und Plattenkassetten auf Beschreibbarkeit getestet werden sollen. Weiteres siehe Teil "Utility".

**CEDIT - Kommando (Zeichensatz definieren)**  
(nur für Kontron PSI980 H)

Aufruf: CEDIT&lt;---

E/A-Kanäle:   Eingabe ---> I-1  
                  Ausgabe ---> O-1

Funktion:

Das Programm CEDIT gestattet die Definition des Zeichensatzes durch den Anwender.

Es wird der jeweils aktuelle, geladene Zeichensatz editiert. Ausgangspunkt ist in der Regel einer der Standard-Zeichensätze CS10D (8x10 Matrix) oder CS16D (13x16 Matrix). Die Zeichen werden unmittelbar im Character-Generator editiert, sodaß die Änderung in Originalgröße kontrolliert werden kann.

CEDIT-Kommandos:

- '?':           Info,  
                gibt eine Übersicht der CEDIT-Kommandos auf dem  
                Bildschirm aus.
- 'S':           setzt den Punkt in der Zeichermatrix, auf dem der  
                Cursor steht.
- 'R':           löscht den Punkt in der Zeichermatrix, auf dem der  
                Cursor steht.
- 'RUBOUT':      löscht das gesamte Zeichen.
- Cursor-Tasten   Home, left, leftup, up, rightup, right, rightdown,  
                 down, leftdown, positionieren den Cursor entsprechend.
- 'RETURN':      nächstes Zeichen editieren.  
                Eingabe des ASCII-Codes sedezimal.
- '!':           speichert den Zeichensatz auf Diskette.  
                Der Dateiname kann vom Benutzer gewählt werden.  
                Voreinstellung des Typs: .COM  
                Fehler beim Beschreiben der Disk werden in der  
                üblichen Weise abgefangen.
- 'CTRL'-'K':    zurück zu KOS.  
                Der editierte Zeichensatz bleibt im Character-Generator

Die von CEDIT erzeugte Datei ist ein lauffähiges Programm und besteht aus einem Ladeprogramm mit der Tabelle der Punktmuster für die Zeichen. Das Ladeprogramm benutzt den System-Call 8dh.

#### Beispiel:

Im Standard-Zeichensatz CS16D soll der Buchstabe A verändert werden.

```
KOS: CS16D      /*Zeichensatz in Character-Generator laden*/
KOS: CEDIT      /*Aufruf des Zeichen-Editors*/
ASCII-Code: 41  /*ASCII-Code des zu editierenden Zeichens A
                 ('A' = 41 HEX). Es werden nur die Tasten 0..9,
                 A..F und ? angenommen.
```

Das Zeichen erscheint nun in Originalgröße in der Eingabezeile und vergrößert in der Arbeits-Matrix.

Mit den Kommandos 'S' und 'R' und den Cursor-Tasten kann das Zeichen editiert werden.

Die Änderungen werden direkt im Character-Generator durchgeführt. Der Cursor befindet sich innerhalb der Arbeitsmatrix.

```
!              /*Kommando zum Schreiben des editierten
                Zeichensatzes auf Disk*/
Dateiname: CTEST /*Typ-Voreinstellung .COM*/
CTRL-K         /*CEDIT beenden*/
```

Das Laden eines Zeichensatzes erfolgt jeweils durch Aufruf der von CEDIT erzeugten Datei, z.B.: "CTEST<---".



Lage der Zeichen in der 13 x 16 Matrix im Standard-Zeichensatz CS16D:

ASCII 41h	ASCII 67h
. . . . .	. . . . .
. . . . .	. . . . .
. . . * * * * *	. . . . .
. . . * * . . . * *	. . . . .
. . * * . . . . * *	. . . . .
. . * * . . . . * *	. . . . .
. . * * . . . . * *	. . . . .
. . * * . . . . * *	. . . * * * * *
. . * * * * * * * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . * * . . . . * *	. . . * * . . . * *
. . . . .	. . . * * . . . * *
. . . . .	. . . * * * * *
. . . . .	. . . . .
. . . . .	. . . . .

Lage der Zeichen in der 8 x 10 Matrix im Standard-Zeichensatz CS10D:

. . . * * *	. . . . .
. . * . . *	. . . . .
. . * . . *	. . . * * * *
. . * . . *	. . * . . *
. . * * * *	. . * . . *
. . * . . *	. . . * * * *
. . * . . *	. . . * * *
. . . . .	. . . . .
. . . . .	. . . . .

**Hinweis:** Die Dateien CS16D.COM bzw. CS10D.COM wurden mit CEDIT erzeugt. Weitere Zeichensätze können mit CEDIT jederzeit generiert und als Datei abgespeichert werden. Die entstehende Datei enthält auch das Programm zum Laden des Zeichensatzes. Englische Zeichensätze: CS16E bzw. CS10E.

**COLOR - Kommando (Einstellen der Rand-, Hintergrund- und Vordergrundhelligkeit)**  
(nur für Kontron PSI980 H)

Aufruf: COLOR<---

E/A-Kanäle: Eingabe ---> I-0  
Ausgabe ---> O-0

Funktion:

Das Programm "COLOR" erlaubt die Definition des Helligkeitsgrades des Bildschirmrandes (FRAME), die Einstellung der Hintergrundhelligkeit (BACKGROUND) und die Einstellung der Schrifthelligkeit (FRONT). Das Programm arbeitet interaktiv mit Benutzerführung (Cursorsteuertasten).

### Achtung!

Die Videoattribute Rand-, Vordergrund- und Hintergrundhelligkeit werden nicht-flüchtig gespeichert (Batteriegepuffertes CMOS-RAM), sodaß beim Wiedereinschalten die zuletzt programmierten Werte eingestellt bleiben.

Falls versehentlich Rand, Vordergrund und Hintergrund gleiche Helligkeitsstufen haben:

1. Reset des Systems
2. '\*<---' eingeben (LOGIN:),  
sofort danach "ESC"-Taste drücken (KOS.INI abbrechen)
3. 'BASIC<---' aufrufen
4. '? CHR\$(27);"F"<---' eingeben (ESC-F an den Monitortreiber schicken)
5. 'KOS<---' eingeben.

**COPY - Kommando (allgemeiner Datentransfer)**

Aufruf: COPY /wdir1/mn1:name.typ /wdir2/mn2:name.typ<---  
 COPY /wdir/mn:quelldatei.typ \$iodd<---  
 COPY \$iodd /wdir/mn:zieldatei.typ<---  
 COPY \$iods \$iodd<---

Voreinst.: mn1 (quelle) = 0  
 = Mastermedium, falls Ziel = \$iodd  
 typ1 (quelle) = COM  
 mn2 (ziel) = 1 falls mn1 = 0 und  
 = 0 falls mn1 = 1  
 = Mastermedium, falls Quelle = \$iods  
 name2 (ziel) = name1  
 typ2 (ziel) = typ1  
 wdir = aktuelles wdir

E/A-Kanäle: Eingabe ---> Kanal I-1  
 Rückmeldungen ---> Kanal O-1  
 Datenquelle (\$iods) ---> Kanal I-5  
 Datenziel (\$iodd) ---> Kanal O-5

**Funktion:**

Das Programm COPY dient zum Transfer von Daten beliebiger Quellen an beliebige Ziele. Als Datenquelle und/oder Datenziel sind möglich:

- eindeutig spezifizierte Dateien auf beliebigen Medien
- E/A-Treiber (gekennzeichnet durch ein \$-Zeichen)

Dies ergibt vier verschiedene Transfermöglichkeiten:

- Kopieren einer Datei auf eine andere Datei
- Kopieren einer Datei an einen E/A-Treiber
- Kopieren von einem E/A-Treiber auf eine Datei
- Kopieren von einem E/A-Treiber auf einen anderen E/A-Treiber

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

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

Bei Eingabe von 'J' und 'Y' wird die vorhandene Datei überschrieben. Nicht zulässig ist das Kopieren einer Datei auf die Datei selbst (gleiches Medium und gleiches wdir).

## Beispiele:

COPY 0:PSI1 1:PSI2<---

Kopiert die Datei PSI1.COM von Medium 0 auf Medium 1, wo sie unter dem Namen PSI2.COM abgelegt wird. Die Mediennummern dieses Beispiels sind voreingestellt und wären deshalb hier nicht notwendig.

COPY PSI1.COM<---

Kopiert die Datei PSI1.COM von Medium 0 auf Medium 1.

COPY /\*/0:TEST.SRC /PROG/0:

Kopiert die Datei TEST.SRC aus dem wdir "\*" in ein wdir "PROG". Beide sind auf Medium 0.

COPY TEST.PRN \$SIOA<---

Kopiert die Datei TEST.PRN auf den E/A-Treiber \$SIOA.

COPY \$KEY \$SIOA<---

Kopiert vom E/A-Treiber \$KEY (Tastatur) auf den E/A-Treiber \$SIOA. Dieser Vorgang endet mit der Eingabe von CNTRL-D (Code 04H, End of transmission). Damit sind z.B. einfache Funktionsüberprüfungen an seriellen Geräten möglich.

COPY \$PSIA TEST.X<---

Kopiert vom E/A-Treiber \$PSIA auf die Datei 'TEST.X'. Damit sind nach Anpassung des Software-Handshakes und der Übertragungsparameter in \$PSIA Kopplungen zwischen Rechnern zur Übertragung von Dateien möglich.

**Hinweise:** Es ist Aufgabe des E/A-Treibers, dem COPY-Kommando das Übertragungsende mitzuteilen (siehe "PSIA" in der Utility-Beschreibung).

Ein Kopieren von großen Dateien auf mehreren Medien kleinerer Kapazität (z.B. eine Datei mit je 616 KB) ist möglich durch Verwendung des Treibers \$XMED. Zur Handhabung siehe 'COPYM2'.

**COPY1 - Kommando (Kopieren einer Datei bei 1-Mediensystem)**

Aufruf: COPY1 /wdir1/name1.typ1 /wdir2/name2.typ2<---

Voreinst.: typ1 (quelle) = COM  
name2 (ziel) = name1 (quelle)  
typ2 (ziel) = typ1 (quelle)  
wdir = aktuelles wdir

E/A-Kanäle: Eingabe ---> Kanal I-1  
Rückmeldungen ---> Kanal O-1

**Funktion:**

Kopieren einzelner Dateien bei Ein-Mediensystemen. Das Programm arbeitet mit Benutzerführung und verlangt zunächst das Einlegen der Quelldiskette. Die Quelldatei wird in den Arbeitsspeicher eingelesen. Danach verlangt COPY1 das Einlegen der Diskette, auf der die Kopie erzeugt werden soll. Der Wechsel zwischen Original und Kopie kann bei kleinem Speicher und großen Dateien mehrmals wiederholt werden müssen.

**Beispiele:**

COPY1 TEST.SRC<---

Kopiert die Datei TEST.SRC auf eine Datei gleichen Namens auf eine zweite Diskette.

COPY1 TEST<---

Kopiert die Datei TEST.COM.

**COPYM2 - Kommando (Kopieren von Medieninhalten)**  
(nur bei KOS5.4x/5.5x/6.x)

Aufruf: COPYM2 mn1>mn2 param<---  
Voreinst.: mn1 (quelle) = Medium 0  
mn2 (ziel) = Medium 1  
param = J ja, rückfragen  
E/A-Kanäle: Eingabe ---> I-1  
Ausgabe ---> O-1

**Funktion:**

COPYM2 kopiert den Inhalt von Medium 'quellmedium' auf Medium 'zielmedium'. Das Zielmedium muß physikalisch formatiert sein. Die Angabe sämtlicher Parameter ist optional. Der Medienname des Quellmediums wird übernommen, falls beim Aufruf des Kommandos die Option 'n' spezifiziert wurde. Andernfalls wird der Medienname des Zielmediums von COPYM2 erfragt. COPYM2 ersetzt das KOS 4.33/5.33-Kommando COPYD2.

**Achtung:** Der Medienname dient der Dateiverwaltung als Medienidentifikation, deren wesentliche Bedeutung darin liegt, einen Datenträger-Wechsel zu erkennen. Gleiche Diskettenamen sind deshalb zu vermeiden, da die erforderliche Neuinitialisierung der Dateiverwaltung nach einem Datenträger-Wechsel bei gleichen Mediennamen nicht stattfindet. Dies führt in der Regel zur logischen Zerstörung der Daten auf Diskette. Die Anzeige dieses Zustandes erfolgt beim STATUS-Kommando durch den Vermerk 'inconsistent'. Falls die Kapazität des Zielmediums kleiner als die Kapazität des Quellmediums ist, wird eine Rückfrage gestellt, ob trotzdem kopiert werden soll. Die Kapazität beider Medien wird berechnet und angezeigt.

Zur Verhinderung von unbeabsichtigtem Löschen von Datenträgern stellt COPYM2 zunächst die Rückfrage:

Original von Medium mn1  
Kopie auf Medium mn2  
Medien bereit? (J/N)

Diese Rückfrage kann durch param = 'N' (nein, nicht rückfragen) umgangen werden. Der Kopiervorgang startet im Falle einer Rückfrage nach der Eingabe eines 'J', wobei Satz für Satz übertragen wird. Nach jedem Satz findet ein automatischer Vergleich zwischen Original und Kopie statt. Fehler führen zu Abbruch und Fehlermeldung. Während des Kopiervorganges erfolgt die Anzeige der gerade kopierten Satznummer. Das Programm kann jederzeit durch die Taste 'ESC' abgebrochen werden.

**Medientreiber \$XMED und COPYM2-Kommando**

Mit Hilfe des speziellen Medientreibers \$XMED kann ein logisches Medium aus mehreren physikalischen Datenträgern (Diskette, Band) bestehen. Es ist dadurch möglich, den Inhalt einer Festplatte auf eine Reihe von Disketten zu kopieren.

Der Medientreiber \$XMED erhöht die logische Kapazität eines beliebigen wechselbaren Mediums auf bis zu 64 MByte. Dazu sind entsprechend viele Einzel-Datenträger (Disketten) notwendig, die von 0 bis n durchnummeriert werden.

\$XMED fragt bei seiner Aktivierung durch IODC, welches (bereits aktivierte) Medium 'verlängert' werden soll. Zur Initialisierung von \$XMED soll der Datenträger Nr. 0 im Laufwerk eingelegt sein, da sich hier der logische Satz 0 befindet (= Anfang des Inhaltsverzeichnis).

Je nach Aktivität fordert \$XMED zum Wechsel des Datenträgers auf. Hier ist äußerste Vorsicht geboten, da eine Überprüfung auf versehentlich verwechselte Datenträger nicht möglich ist.

\$XMED arbeitet zwar unter der Dateiverwaltung von KOS, d.h. einzelne Dateien können gelesen und geschrieben werden, jedoch sollte sich seine Anwendung auf den Backup von größeren nicht wechselbaren Medien beschränken.

In Verbindung mit dem COPYM2-Kommando ergibt sich hier die Möglichkeit, relativ schnell den gesamten Inhalt eines großen Mediums auf Disketten zu kopieren. Insofern ersetzt die Kombination COPYM2-\$XMED das BACKUP-Kommando.

Es wird an dieser Stelle jedoch ausdrücklich auf die Verfügbarkeit von Wechselplattensystemen großer Kapazität für Kontron PSI-Systeme verwiesen.

#### Warnung:

Bei Verwendung des Treibers \$XMED ist es Sache des Benutzers, den richtigen Datenträger einzulegen. Eine Verwechslung wird nicht erkannt.

#### Beispiele:

```
M-0 sei Floppy-Treiber $DSKO
M-1 sei Floppy-Treiber $DSK1
M-2 sei Winchester-Treiber $WINO
M-3 sei Verlängerungstreiber $XMED,
    der seinerseits M-1 aufruft
```

COPYM2<---

Kopiert die Diskette in Laufwerk 0 auf die Diskette in Laufwerk 1.

COPYM2 2>3<---

Kopiert den Inhalt der Festplatte auf einen Satz von Disketten in Laufwerk 1.

COPYM2 3>2<---

Kopiert einen Satz von Disketten von Laufwerk 1 auf Festplatte.

Hinweis: Eine Kopie auf dem gleichen physikalischen Medium darf nicht durchgeführt werden. Der Aufruf 'COPYM2 1>3' ist bei der Medienkanalbelegung des Beispiels unzulässig, da beide Mediennummern das gleiche physikalische Gerät ansprechen.

**COPYWFD - Kommando (Kopieren von Disketten auf/von Festplattenspeicher)**

Aufruf: COPYWFD<---

E/A-Kanäle: Eingabe ---> I-1  
Ausgabe ---> O-1

**Funktion:**

COPYWFD speichert Diskettenabbilder auf Festplatte und zurück auf Diskette. Bei ausreichender Kapazität der Hard-Disk können bis zu 36 Disketteninhalte als Dateien abgelegt werden.

Das Programm besteht aus dem Kommando COPYWFD.COM und der zugehörigen Bildschirmmaske COPYWFD.MAS.

COPYWFD arbeitet mit Benutzerführung. Nach dem Aufruf ist der Benutzer aufgefordert einzugeben, ob er einen Disketteninhalt auf einem größeren Medium (Winchester-Laufwerk) ablegen will ('Save'), oder ob er aus einer abgelegten Datei eine Diskette wiederherstellen will ('Restore').

**SAVE**

Das Programm erfragt die Mediennummern der beiden beteiligten Laufwerke. Es wird eine Datei mit dem Namen 'DISK.\*\*\*' angelegt. \*\*\* steht für die laufende Nummer, die vom Programm automatisch bei jedem neuen Diskettenabbild hochgezählt wird. Die Anzahl der 'DISK'-Files auf der Harddisk ist auf 36 pro wdir beschränkt.

Die ordnungsgemäße Abspeicherung des Disketteninhaltes wird durch 'Disk saved' gemeldet.

Der Abbruch eines Kopiervorganges ist jederzeit durch Drücken der 'Escape'-Taste möglich.

**Fehlermeldungen bei SAVE**

Wrong direction:	Kopieren ist aufgrund der Medien-Kapazitäten in der gewünschten Richtung nicht möglich (z.B. Floppy ---> Floppy oder Hard-Disk ---> Floppy)
Floppy not ready:	Floppy nicht eingelegt oder Laufwerk nicht vorhanden
Read error:	Lesefehler bei Floppy-Disk (Quellmedium)
Write error:	Schreibfehler bei Hard-Disk (Zielmedium)
Medium full:	Hard-Disk voll



**RESTORE**

Nach Eingabe der Mediennummer wird die Nummer der gewünschten Datei DISK.\*\*\* abgefragt. Diese Datei wird dann auf die eingelegte Floppy-Disk kopiert.

Der ordnungsgemäße Kopiervorgang wird mit 'Disk restored' bestätigt. Abbruch ist hier ebenfalls mit der 'Escape'-Taste möglich.

**Fehlermeldungen bei Restore**

Wrong direction:	Kopieren ist aufgrund der Medien-Kapazitäten in der gewünschten Richtung nicht möglich (z.B. Floppy ---> Floppy oder Floppy ---> Hard-Disk)
File not found:	Eingegebene Filenummer existiert nicht auf dem angegebenen Medium
Read error:	Lesefehler bei Hard-Disk
Write error:	Hardwarefehler bei Floppy-Disk
Verify error on floppy disk at record...:	Fehlerhaftes Diskettermaterial

CS16D - Kommando (Zeichengenerator laden)  
CS10D - Kommando  
CS16E - Kommando  
CS10E - Kommando  
(nur für Kontron PSI980 H)

Aufruf: CS16D<---  
CS10D<---  
CS16E<---  
CS10E<---

E/A-Kanäle: Eingabe ---> ---  
Ausgabe ---> ---

#### Funktion:

Der Zeichengenerator-RAM wird durch den Kaltstart nur mit einem Minimal-Zeichensatz geladen. Es ist deshalb erforderlich, eines der Programme CS16x bzw. CS10x zur vollständigen Initialisierung des Zeichengenerators auszuführen (z.B. in der Kommandodatei KOS.INI).

#### Bildschirmformat

25 Zeilen x 80 Zeichen    40 Zeilen x 132 Zeichen

Zeichensatz deutsch:	CS16D	CS10D
Zeichensatz englisch:	CS16E	CS10E

**CPFILE-Kommando (Vergleich von 2 Dateien)**

Aufruf: CPFILE /wdir1/mn1:name1.typ /wdir2/mn2:name2.typ<---

Voreinst.: mn = Mastermedium  
typ = COM  
wdir = aktuelles wdir

E/A-Kanäle: Status ---> Kanal I-1  
Ausgaben ---> Kanal O-1

**Funktion:**

Byteweiser Vergleich des Inhalts von zwei Dateien beliebigen Typs und beliebiger, auch unterschiedlicher Länge.

Die Anzahl und der Inhalt der unterschiedlichen Bytes wird ausgegeben. Abbruch des Kommandos durch 'ESCAPE'.

**Beispiel:**

CPFILE DATEI 1:DATEI<---

Vergleicht den Inhalt der Datei DATEI.COM auf dem Mastermedium mit dem Inhalt der Datei DATEI.COM auf dem Medium 1.

CPFILE 0:KOS.SYS 2:KOS.SYS P=\*<---

Vergleicht die (geheimen) Systemdateien auf Medium 0 und Medium 2.

## DATE-Kommando (Datumseintrag/-ausgabe)

```
Aufruf:      DATE<--- (KOS4/5)
             SETDATE<--- (KOS6)
Voreinst.:   -
```

```
E/A-Kanäle:      Eingabe ---> Kanal I-1
                  Ausgabe ---> Kanal O-1
```

**Funktion:**

Anzeigen und Eintragen eines Datums ins Betriebssystem. Das Programm arbeitet mit Benutzerführung.

Soll ein bestimmtes Datum erzeugt werden, können die notwendigen Parameter auch schon beim Aufruf übergeben werden:

DATE "J ddmmyy"

Die Angaben für das Datum müssen mit führenden Nullen übergeben werden:

dd - Tagesangabe  
mm - Monatsangabe  
yy - Jahresangabe

DATE fragt die Speicherstellen OEH und OFH im KOS-reservierten Bereich des Speichers ab und errechnet daraus das Datum. Dieses Datum kann in einem Dialog verändert werden. Das geänderte Datum wird kodiert in die Speicherstellen OEH/OFH übertragen und auf die dafür vorgesehenen Stellen in das Directory des Mediums 0 geschrieben. Bei Aufruf von DATE wird das eingetragene Datum vom Medium 0 gelesen; wenn das Medium nicht bereit ist (z.B. keine Diskette eingelegt), dann wird als Datum "00.00.00" ausgegeben.

Das beim Einlesen von KOS übernommene bzw. durch DATE bestimmte Systemdatum wird bei Änderung oder Neuanlage von Dateien mit eingetragen. Dies gilt nicht bei unverändertem Kopieren von Dateien durch z.B. MOVE oder COPY.

Beim Auflisten des Inhaltsverzeichnis durch IL wird das Dateidatum bei jeder aufgelisteten Datei mit dargestellt.

Codierung:	OEH:	Bit 0...4	Tag, hexadezimal
		Bit 5...7	Jahr, höhere hex. Ziffer
	OFH:	Bit 0...3	Monat, hexadezimal
		Bit 4...7	Jahr, niedere hex. Ziffer

**Hinweis:** Bei KOS6-Systemen mit Hardware-Uhr ist das Kommando DATE durch SETDATE ersetzt, siehe dort.

**DEFP - Kommando (Dateieigenschaften (Properties) definieren)**

Aufruf: DEFP /wdir/mn:name.typ<---

Voreinstellung: mn = Mastermedium  
typ = COM  
wdir = aktuelles wdir

E/A-Kanäle: Eingabe ---> Kanal I-1  
Ausgabe ---> Kanal O-1

**Funktion:**

Anzeigen und Editieren der Dateieigenschaften.

Die Dateieigenschaften werden beim Editieren unter DEFP in folgender Form mit dem jeweiligen Kennbuchstaben dargestellt:

K - KOS Systemdatei, Public  
W - Datei schreibgeschützt (Write protection)  
E - Datei löschgeschützt (Erase protection)  
P - Properties geschützt  
R - reserviert, Record locking  
D - Directory Dateien  
U - Datei hat Benutzerkennzeichen (USER-ID)  
S - Datei ist geheim (Secret)

Anwahl P=.... (Kennbuchstaben)

P=\* entspricht "allen" Dateieigenschaften

Falls das 'Properties geschützt'-Flag nicht gesetzt ist, erfolgt das Editieren der Properties cursororientiert mit Hilfe folgender Befehle:

S = Setzen  
R = Rücksetzen  
<CR> = Unverändert übernehmen  
K = Rückschreiben der editierten  
Properties und Rückkehr zu KOS

**Bedeutung der einzelnen Property-Flags:**

**K - KOS Systemdatei:** Darunter fallen z.B. die Dateien vom Typ COM, die als Dienstprogramme auf der KOS-Systemdiskette enthalten sind. "Public" Dateien sind keinem speziellen Working-Directory zugeordnet, sondern unter jedem "wdir" erreichbar.

**W - Datei schreibgeschützt:** Ein Versuch, in die Datei zu schreiben, wird von KOS verhindert und mit der Meldung:

Datei schreib-/löschgeschützt

quittiert.

**E - Datei löschgeschützt:** KOS verhindert das Löschen der Datei. Schreibzugriffe auf diese Datei sind gewöhnlich nur über die Random-Funktion möglich.

**P - Properties geschützt:** Das DEFP-Kommando zeigt die Properties zwar an, aber ein Verändern ist nicht mehr möglich. Das Setzen des 'Properties geschützt'-Flag ist endgültig, eine Veränderung der Properties ist daraufhin nicht mehr möglich.

**R - Reserviert, Record locking:** Diese Dateien sind satzweise (128 Byte) gegen Mehrfachzugriff in KOBUS-Systemen verriegelt.

**U - Benutzerkennzeichen:** Falls dieses Flag gesetzt ist, ist ein Zugriff auf die Datei nur nach Angabe des Benutzerkennzeichens möglich. Außerdem ist die Datei schreibgeschützt. Wird dieses Flag im DEFP-Kommando gesetzt, so fordert das System die Eingabe eines Benutzerkennzeichens, das dann der Datei zugewiesen wird. Ändern der Datei im Editor ist nicht möglich (erst U rücksetzen).  
Dateien mit Benutzerkennzeichen sind gleichzeitig "public".

**S - Geheim:** Die Datei wird z.B. beim I- oder IL-Kommando übergangen, wenn nicht im Kommando als Parameter 'P=S' oder 'P=\*' angegeben wird.

**Dateieigenschaften bei den Kommandos IL, MOVE, REN, DEL**

Bei diesen Kommandos kann zur optionalen Spezifikation beliebiger Property-Kombinationen der Parameter 'P=properties' angegeben werden. Für 'properties' steht eine beliebige Folge der Zeichen: K, W, E, P, R, D, U, S, \*.

**Beispiel:**

Alle Dateien, die mindestens schreibgeschützt und 'geheim' sind, sollen mit dem MOVE-Kommando kopiert werden. Dazu ist folgendes Kommando notwendig:

```
MOVE * P=WS<---      oder:
MOVE * P=SW<---
```

Die Reihenfolge der Parameter spielt keine Rolle, solange insgesamt nicht mehr als drei Parameter benötigt werden. Bei mehr als drei Parametern muß die Spezifikation 'P=properties' als letzte eingegeben werden, andernfalls bleibt der letzte Parameter unberücksichtigt. Beispiele für richtige Anwendungen der 'P=properties'-Option:

```
MOVE P=KWS dateigruppe J
MOVE dateigruppe J P=KWS
MOVE 0:dateigruppe 2: J P=KWS
IL P=KWS dateigruppe
DEL 1:dateigruppe P=KWSU
REN 1:*.DIR USER.DIR J P=D
```

Das letzte Beispiel zeigt die Möglichkeit, den bei "FORMAT" oder "COPYM2" angegebenen Mediennamen zu ändern.

Werden die Kommandos IL, MOVE, DEL und REN ohne die 'P=properties'-Option aufgerufen, so bleiben alle Dateien die geheim und/oder Directory Dateien sind, unberücksichtigt.

**DEL - Kommando (Datei löschen)**

Aufruf:           DEL /wdir/mn:name.typ param<---  
                   DEL /wdir/mn:dateigruppe param<---

Voreinst.:       mn       = Mastermedium  
                   typ       = \*  
                   param   = J       = rückfragen  
                               = P       = nicht geheim  
                   wdir   = aktuelles wdir

E/A-Kanäle:      Eingabe ---> Kanal I-1  
                   Ausgabe ---> Kanal O-1

**Funktion:**

Löschen von einzelnen Dateien oder Dateigruppen. Im allgemeinen erfolgt vor dem Start des Löschvorgangs zunächst eine Rückfrage an den Benutzer, ob die Datei tatsächlich gelöscht werden soll (Parameter 'param' = J). Fünf Eingaben sind daraufhin möglich.

- J - Die Datei wird gelöscht
- Y - Die Datei wird gelöscht
- N - Die Datei wird nicht gelöscht
- A - Alle folgenden Dateien der angegebenen Spezifikation werden gelöscht; Rückfragen werden nicht mehr gestellt
- K - Abbruch des Kommandos (KOS-Rücksprung)

Ist param = 'N' im Parameterfeld spezifiziert, so wird von Anfang an keine Rückfrage gestellt. Die gelöschten Dateien werden immer ausgedruckt. Das Kommando kann jederzeit durch die Taste 'ESC' abgebrochen werden.

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

**Beispiele:**

DEL DATEI.ABC<---

Löscht die Datei 'DATEI.ABC' im aktuellen wdir auf dem Masterlaufwerk nach vorheriger Rückfrage bei der Benutzereingabe 'J'. Die Datei 'DATEI.ABC' wird natürlich nur dann gelöscht, wenn sie weder "verborgen" noch "löschgeschützt" ist.

DEL \*.PRN N<---

Löscht alle nicht geheimen Dateien des Typs 'PRN', ohne Rückfragen zu stellen.

DEL \*.COM P=K N

Löscht ohne Rückfrage alle Dateien des Typs 'COM' mit gesetzter Dateieigenschaft "K", die nicht "verborgen" und "löschgeschützt" sind.



**DO - Kommando (Ausführung einer Kommandodatei)**

Aufruf: DO /wdir/mn:name.typ p1 p2 ... p9<---

Voreinst.: mn = Mastermedium  
typ = CMD  
wdir = aktuelles wdir

E/A-Kanäle: Eingaben ---> Kanal I-1  
Ausgaben ---> Kanal O-1

Angabe von 'wdir', 'mn:' und '.typ' ist optional. Die Anzahl der eingegebenen Parameter muß mit ihrer Anzahl in der Kommandodatei übereinstimmen.

**Funktion:**

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

Das DO-Kommando lädt die Kommandodatei in den höchstmöglichen Speicherbereich, schützt diesen und ruft anschließend den Kommandointerpreter von KOS auf. Dieser Speicherbereich wird nach der Ausführung des DO-Kommandos wieder freigegeben.

DO-Kommandos dürfen beliebig ineinander verschachtelt sein. Die Schachteltiefe ist nur durch den zur Verfügung stehenden Arbeitsspeicher begrenzt. Vor der Ausführung der Kommandos wird der Inhalt der Kommandodatei ausgedruckt. Beliebige Parameter können beim Aufruf des DO-Kommandos übergeben werden. Als Parameterrepräsentanten innerhalb einer Kommandodatei dienen die Ausdrücke #1, #2, ... #9. Zugelassen sind maximal neun Parameter.

Durch Eingabe von <ESC> vor dem Ausdruck des Inhaltes der Kommandodatei wird das DO-Kommando abgebrochen. Die spätere Eingabe von <ESC> bricht das gerade laufende Kommando aus der Kommandodatei ab.

Das KOS-Kommando STOP (s. dort) kann in der Kommandodatei enthalten sein. Es hält die Verarbeitung der Kommandodatei an und gibt dem Anwender die Möglichkeit, die Ausführung der Kommandodatei abubrechen oder fortzusetzen.

**Beispiele:**

DO TEST<---

Ausführung der Kommandos in der Datei TEST.CMD. Diese Datei habe folgenden Inhalt:

```
BASIC "LOAD MYFIL<RUN";I 1:*.BAS  
EDIT TEST.CMD;I TEST.*;M
```

DO KDATEI.LIS 0 1<---

Im zweiten Beispiel sind die beiden Parameter '0' und '1' angegeben. Diese ersetzen in der Kommandodatei die Ausdrücke #1 bzw. #2. KDATEI.LIS enthalte beispielsweise diese Kommandofolge:

M #1;I #2:\*.COM

Der Aufruf ergibt dann: M 0;I 1:\*.COM

### Behandlung von Steuerzeichen für \$MON im DO-Kommando

Der Bildschirmtreiber \$MON interpretiert das Steuerzeichen OFH (CTRL-O) als ON/OFF Schalter für nachfolgende Zeichen.

Damit ist es beispielsweise möglich, die Ausgaben von KOS bzw. des DO-Kommandos zu unterdrücken, bis ein bestimmtes Anwenderprogramm gestartet ist und ein zweites CTRL-O an \$MON überträgt. Zu diesem Zweck muß eine Kommandodatei an erster Stelle das Zeichen CTRL-O enthalten. Weitere CTRL-O Zeichen innerhalb einer Kommandodatei bleiben unberücksichtigt.

Beispiel: KOS.INI enthalte folgende Kommandosequenz:

```
CTRL-O
BASIC "LOAD MYFIL<RUN"
```

Damit werden alle Ausgaben auf dem Bildschirm unterdrückt, bis das Programm MYFIL das Steuerzeichen CTRL-O an \$MON überträgt, z.B. durch "PRINT CHR\$(15)".

### Hinweise:

- a) Bei manchen Kommandos mit vielen Parametern kann die Begrenzung auf 71 Zeichen pro Zeile des Editors EDIT störend sein. Hier hilft die Möglichkeit der Parameterübergabe in mehreren Zeilen.

Beispiel:

```
LINK "XXX/N,XYZ,ABC,DEF,GHI,..
KLM,NOP,....."
```

- b) Nach dem Laden von KOS wird als erstes Kommando automatisch das Kommando "DO KOS.INI" ausgeführt. Damit kann die Initialisierung des Systems und der Start des Anwendungssystems automatisch veranlaßt werden. Die Datei KOS.INI kann mit EDIT erstellt werden und in dem von Login (siehe "Konzepte") bestimmten wdirc enthalten sein. Das DO-Kommando selbst und KOS.INI werden auf allen Medien gesucht, die nach dem Systemladen über Medienkanäle ansprechbar sind. In KOBUS-Slave-Stationen ist für die Startkommandodatei der Name "SLAVE.INI" verwendet.

**DUMP - Kommando (Ausdruck binärer Dateiinhalte)**

Aufruf: DUMP /wdir/mn:name.typ param<---

Voreinst.: mn = Mastermedium  
typ = COM  
param = 0=\$MON  
wdir = aktuelles wdir

E/A-Kanäle: Status ---> Kanal I-1  
Fehlermeldungen ---> Kanal O-1  
Dateiausdruck ---> Kanal O-2 oder O-6 falls 0=\$iod

**Funktion:**

Ausdruck des Inhalts einer eindeutig spezifizierten Datei im Hex- und ASCII-Code.

Pro Zeile wird ein 16 (=10H) Byte Block, inklusive der Anfangsadresse (1. Spalte) und der entsprechenden ASCII-Äquivalente ausgedruckt. Nicht darstellbare Codes erscheinen im ASCII-Feld als Punkt. Der Ausgabekanal kann explizit angegeben werden.

**Beispiele:**

DUMP TEST.PRN<---

Ausdruck der Datei TEST.PRN in hexadezimalen Format. Der Ausdruck gelangt auf Kanal O-2, der vom System dem Sichtschirm zugeordnet ist. Diesem Kanal kann durch das IODC-Kommando ein Drucker zugeordnet werden.

DUMP 0=\$SIOA 0:TEST<---

Ausdruck der Datei TEST.COM auf dem Ausgabetreiber \$SIOA. Der Treiber muß zuvor mit dem Kommando

IODC \$SIOA=ACTIVE<---

aktiviert werden; die Zuordnung auf Kanal O-6 erfolgt automatisch.

**FORMAT - Kommando (Diskette formatieren)**

Aufruf:           FORMAT<---

Voreinst.:       -

E/A-Kanäle:       Eingaben ---> Kanal I-1  
                  Ausgaben ---> Kanal O-0

**Funktion:**

Bereitet eine Diskette zur Verwendung auf Kontron PSI-Systemen vor. Alle existierenden Daten auf dieser Diskette werden überschrieben. Ein Inhaltsverzeichnis für 64 Einträge (8kByte) wird angelegt. Format stellt die Rückfrage nach der 'physikalischen' Laufwerksadresse, die sich gewöhnlich von der Mediennummer unterscheidet:

Auf welchem Laufwerk soll formatiert werden? (0/1/2/3)

Dabei gilt: 0:   rechtes oder oberes integriertes FD-Laufwerk  
            1:   linkes oder unteres integriertes FD-Laufwerk  
            2,3: externe FD-Laufwerke

FORMAT fragt nach dem Identifikationsnamen, der die Diskette kennzeichnen soll. Dieser Name darf maximal 8 Zeichen lang sein; gleiche Namen sollten vermieden werden. Bei MAKEFS und COPYM2 wird der bei FORMAT angegebene Name überschrieben. Zum Start muß auf die Frage:

Diskette in Laufwerk n bereit? (J/N)

die Taste 'J' gedrückt werden, andernfalls erfolgt Abbruch (KOS-Rücksprung).

Unter KOS5 werden bei Eingabe von 'FORMAT' und bei Festplatten-Betriebssystemen die Diskettenparameter (ein/zweiseitig, einfache/doppelte Schreibdichte) immer abgefragt.

Bei KOS5 führt ein zusätzlicher Parameter 'V' zur Überprüfung der Diskettenlesbarkeit nach oder anstelle der Formatierung. Durch "FORMAT P V" ist es möglich, auf Kontron PSI80Q-Systemen auch Disketten für Kontron PSI-Systeme mit einseitigem Aufschrieb und/oder einfacher Schreibdichte zu formatieren. Ansonsten stellt sich FORMAT automatisch auf die jeweilige im Betriebssystem in \$DSKO/1 eingestellte Laufwerkskonfiguration ein.

**Beispiele:**

FORMAT 1<---

Formatiert die Diskette in Laufwerk 1 (ohne Rückfrage 1)

FORMAT<---

Formatiert eine Diskette mit allen Rückfragen.

Hinweis: FORMAT hält die Hintergrundverarbeitung an (nicht bei KOS6).  
Ändern der Kapazität des Inhaltsverzeichnisses siehe MAKEFS.

**FSCHECK - Kommando (File System Check)**

Aufruf: FSCHECK<---

E/A-Kanäle: Eingabe ---> I-1  
Ausgabe ---> O-1

Funktion:

Das Programm FSCHECK hilft bei folgenden Situationen:

- Medium hat einen 'Directory Format Fehler'.
- Medium ist 'inconsistent'.
- Es existieren nicht zusammenhängende Dateistücke; erkennbar dadurch, daß 'STATUS' und 'IL' (von Dateien und Directory) eine unterschiedliche Belegung des Mediums anzeigen.
- Den Dateinamen einer Datei finden, die nicht lesbare Records enthält.
- Zu welcher Datei gehört ein bestimmter Record?
- Zu welcher Datei gehört eine bestimmte Directory Blocknummer?

Die Überprüfung eines Mediums setzt keine Kenntnisse des Aufbaus des Inhaltsverzeichnisses (Directory) voraus. Das Kommando FSCHECK verändert ein Medium nicht, es führt keine Fehlerbehebung durch.

Zeigt das Programm FSCHECK jedoch Fehler an, kann eine Fehlerbeseitigung durch das Programm KDM (s. Abschnitt TB-E) bei Kenntnis des Aufbaus des Inhaltsverzeichnisses (Directory), siehe 'Inhaltsverzeichnis' im Teil C, erfolgen. Nach einer Fehlerbeseitigung ist die Neuinitialisierung (N-Kommando) notwendig.

Bei jedem Durchlauf des Programms FSCHECK wird nur ein Fehler angezeigt, auch wenn mehrere Fehler vorliegen. Erst wenn der angegebene Fehler behoben ist, kann der nächste Fehler angezeigt werden. Während der Fehlerbeseitigung darf das betroffene Medium auf keinen Fall als KOBUS-Zentralmedium aktiv sein.

**Fehlermeldungen und Fehlerbeseitigung****'Directory Format Fehler'**

Diese Fehlermeldung kann nach einem N-Kommando mit Angabe der Mediennummer als Parameter erscheinen: "N mn<---", falls ein solcher Fehler auf Medium mn vorliegt.

Falls das Programm 'STATUS mn' nun den Hinweis 'inconsistent' nicht ausdrückt, so hat wahrscheinlich das Byte 0 eines Directory-Eintrags weder den Wert 0 noch den Wert 0E5H. Der Dateiname darf keine Zeichen größer 80H enthalten.

FSCHECK druckt die Recordnummer und die ersten 12 Bytes des fehlerhaften Eintrags aus.

**Fehlerbehebung:** Mit KDM fehlerhaften Record korrigieren. Nach Verlassen von KDM muß unbedingt das Programm STATUS aufgerufen werden. Diese Prozedur darf nicht bei laufenden Rechnernetzen (z.B. KOBUS) durchgeführt werden.

### **'Inconsistent'**

Ist ein Medium inconsistent (STATUS-Kommando), so teilen sich mindestens zwei Dateien ein und denselben Speicherplatz auf dem Medium. Hierbei sind die Inhalte mindestens einer, möglicherweise aber auch aller daran beteiligten Dateien, beschädigt.

FSCHECK findet mehrfach belegte Speicherplätze und druckt die Namen und die Recordnummern der beteiligten Dateien aus.

**Fehlerbehebung:** Falls es möglich ist, sollten alle ausgedruckten Dateien gelöscht werden. Werden nicht alle Dateien gelöscht, kann zwar der Fehler im Inhaltsverzeichnis behoben werden, der Inhalt der verbleibenden Dateien kann allerdings beschädigt sein.

Das Löschen kann mit dem DEL-Kommando erfolgen. Sollte dies nicht möglich sein, weil z.B. das 'Working Directory' unbekannt ist, so kann auch über KDM gelöscht werden. Hierzu ist das erste Byte auf E5H zu setzen, das zweite Byte muß ungleich E5H sein. Wie auch immer gelöscht worden ist, es muß danach das STATUS-Kommando aufgerufen werden. Diese Prozedur darf nicht bei laufenden Rechnernetzen (z.B. KOBUS) durchgeführt werden.

### **Anzahl belegter Records ungleich zwischen STATUS und IL**

Dies festzustellen ist nur auf Einzelplatzsystemen ohne KOBUS und ohne 'Working Directories' auf einfache Weise möglich. Am einfachsten geht es mit dem Programm FSCHECK selbst.

Der mögliche Unterschied in der Medienbelegung erklärt sich dadurch, daß IL nur vollständige Dateien auswertet, STATUS dagegen die gesamte Belegung des Mediums (bei IL darf die Datei Inhaltsverzeichnis (Directory) nicht vergessen werden!). Möglicherweise existieren aber Bruchstücke von Dateien, die zu keiner gültigen Datei mehr gehören.

FSCHECK druckt den Dateinamen, den Dateierweiterungszähler und die Record Nummer des zu keiner Datei mehr gehörenden Directory-Eintrags.

**Fehlerbehebung:** Mit KDM müssen die ersten beiden Bytes des fehlerhaften Directory-Eintrages auf E5H und 41H gesetzt werden. Danach muß unbedingt das Programm STATUS aufgerufen werden. Diese Prozedur darf nicht bei laufenden Rechnernetzen (z.B. KOBUS) durchgeführt werden.

**IL - Kommando (Inhaltsausgabe im Langformat)**

Aufruf: IL /wdir/mn:name.typ param<---  
 IL /wdir/mn:dateigruppe param<---

Voreinst.: mn = Mastermedium  
 name = \*  
 typ = \*  
 param = P=nicht geheim  
 O=\$MON  
 wdir = aktuelles wdir

E/A-Kanäle: Eingaben ---> Kanal I-1  
 Fehlermeldungen ---> Kanal O-1  
 Inhaltsverzeichnis ---> Kanal O-2 oder  
 O-6 falls O=\$iod

**Funktion:**

Ausdruck des Inhaltsverzeichnis eines Mediums in langem Format. Neben dem Dateinamen/-Typ und dem Dateidatum wird die Anzahl der von der jeweiligen Datei belegten Sektoren sowie die dafür auf dem Medium reservierten Blöcke (Vielfache von 1k Byte) ausgegeben. Bei Programmdateien (Typ COM) wird die Ladeadresse mit ausgewiesen. Letzteres gilt nicht für viele CP/M-Programme.

Gewünschte Datei-Properties sind beim Aufruf von IL anzugeben (siehe 'DEFP'-Kommando), der Ausgabekanal kann explizit angegeben werden.

**Beispiele:**

IL TEST<---

Auflistung aller nicht geheimen Dateien des Namens TEST (Typ beliebig) des Mastermediums, die public oder im aktuellen Working Directory sind.

IL 1: P=\* O=\$SIOB

Auflistung aller Dateien von Medium 1 auf den Ausgabetreiber \$SIOB. Die Zuordnung auf Kanal O-6 erfolgt automatisch. \$SIOB muß aktiviert sein.

IL P=D

Ausgabe der Directory-Datei des Mastermediums.

IL /abc/

Ausgabe der Dateien im Working Directory "abc", die nicht geheim sind und auf dem Mastermedium liegen.

IL P=K

Auflistung der Dateien mit Kennung K (public) des Mastermediums.

IL D>11.11.82

Auflistung der nicht verborgenen Dateien des Mastermediums mit Erzeugungsdaten nach dem 11.11.1982, die im aktuellen Working Directory liegen bzw. öffentlich sind.

**INFO - Kommando (Ausgabe einer ASCII-Datei)**

Aufruf: INFO /wdir/mn:name.typ<---

Voreinst.: mn = Mastermedium  
typ = INF  
wdir = aktuelles wdir

E/A-Kanäle: Eingaben ---> I-1  
Fehlermeldungen ---> O-1  
Dateiausdruck ---> O-2

**Funktion:**

Ausgabe einer ASCII-Datei auf dem Sichtschirm des Kontron PSI Computers (oder auf ein beliebiges Peripheriegerät, das dem Kanal 0-2 zugewiesen ist). Die Ausgabe erfolgt Bildschirm-orientiert:

- 1 Seite = 24 Zeilen
- 1 Zeile = Folge von ASCII-Zeichen, mit CR abgeschlossen

**mögliche Eingaben:**

- Leertaste ---> nächste Seite
- Return ---> eine Seite zurück
- ESC ---> Programm abbrechen

Die unterste Bildschirmzeile enthält Status- und Fehlermeldungen:

- ? ---> Datei nicht gefunden
- \* ---> Datei kann nicht gelesen werden
- 'Ende' ---> letzte Seite

**Beispiele:**

INFO KOS<---  
Ausgabe der Datei KOS.INF auf den Bildschirm

INFO<---  
Ausgabe der ersten Datei des Masterlaufwerkes vom Typ '.INF'.

INFO 1:PIO.SRC<---  
Ausgabe der Textdatei PIO.SRC des Mediums 1.



**INISER - Kommando (Neu-Initialisierung von Serienschnittstellen)**  
(nur bei KOS4.33/5.xx)

Aufruf: INISER param<---

Voreinst. param = -

E/A-Kanäle: Eingaben ---> Kanal I-1  
Ausgaben ---> Kanal O-1

**Funktion:**

Programmierung der Serienschnittstellen A und B für asynchrone Übertragung. Alle hierfür relevanten Parameter wie Baudrate, Parity etc. sind einstellbar. 'INISER' arbeitet mit Benutzerführung (falls als Parameter 'param' = P gegeben wurde) und ermöglicht die Abspeicherung von einmal eingegebenen Initialisierungsparametern auf Diskette. Eine neuerliche Initialisierung mit denselben Parametern kann nun beliebig oft ohne neuerliche Eingaben durchgeführt werden durch den Aufruf "INISER<---".

**Beispiele:**

INISER<---

Programmiert eine der Serienschnittstellen (Kanal A oder B) entsprechend der momentan auf Diskette (Platte) abgespeicherten Initialisierungsparameter.

INISER P<---

Alle Parameter werden neu erfragt. Sie können bei Bedarf auf Diskette (Platte) abgelegt werden.

**Hinweise:** INISER ist als Hilfe für den provisorischen Anschluß gedacht. Nach den Testläufen sollten die Schnittstellenparameter in den verwendeten seriellen Treiber durch Editieren und Assemblieren der Treiber-Quellcode-Datei übernommen werden.

INISER ist nach der Aktivierung (IODC) des Treibers auszuführen.

Beim Aktivieren des Treibers nach INISER überschreiben die Schnittstellenparameter des Treibers die durch INISER gegebenen Parameter. Auch mit "N \$iodn" wird die Treiber-eigene Initialisierung erneut übernommen.

## IODC - Kommando (Ein-/Ausgabetreiber-Verwaltung) (bei KOS4.33/5.33: EAK-Kommando)

### Übersicht und Kommandosyntax

Das Programm IODC (Ein-/Ausgabe Steuerung, Input/Output Device Control) ermöglicht folgende Aktivitäten:

- Aktivierung eines Ein-/Ausgabetreibers (E/A-Treiber)
- Deaktivierung eines E/A-Treibers
- Aktivierung eines Medientreibers
- Deaktivierung eines Medientreibers
- Zuordnung einer Kanalnummer für E/A- oder Medientreiber
- Auflistung der aktivierten E/A-Kanäle
- Auflistung der Syntax des Programms IODC

Hierzu stehen folgende IODC-Aufrufe zur Verfügung:

```
IODC $(mn:)iodn=ACTIVE<---  
IODC $(mn:)medn=ACTIVE<---  
IODC $iodn=DEACTIVE<---  
IODC $medn=DEACTIVE<---  
IODC x-n=$iodn<---  
IODC M-n=$medn<---  
IODC LIST<---  
IODC SYNTAX<---
```

E/A-Kanäle:       Eingaben ---> Kanal I-1  
                  Ausgaben ---> Kanal O-1

Bei jedem Aufruf des Programms IODC können bis zu 9 Aktivitäten als Parameter angegeben werden. Fehlt jeglicher Parameter, so werden Syntax und Beispiele für das IODC-Kommando ausgegeben.

Beispiel:

```
IODC $iodn=ACTIVE x-n=$iodn LIST<---
```

Die Abkürzungen hier und im folgenden bedeuten:

```
iodn = Ein-/Ausgabetreiber Name  
medn = Medientreiber Name  
mn   = Mediennummer (Voreinstellung Mastermedium)  
x    = I für Eingabekanal, O für Ausgabekanal  
n    = Kanalnummer (1...9), auch log. Gerätenummer
```

IODC verwendet die Bildschirmmaske IODC.MAS (nur bei KOS6).

**E/A-Treiber Aktivierung**

Aufruf: IODC \$mn:iodn=ACTIVE<---

Voreinst.: mn = Mastermedium

**Funktion:**

Aktivierung des Treibers iodn bzw. medn. Das Kommando ist mit einem 'Relocator' ausgerüstet, der E/A- oder Medientreiber automatisch in den höchstmöglichen Speicherbereich lädt.

Es ist deshalb nicht notwendig, Treiber zu linken. Das Programm IODC sucht zunächst die OBJ-Datei eines Treibers und - erst wenn dieses nicht gefunden wurde - die IOD-Datei. Für den Relocator darf der E/A-Treiber nur aus einem einzigen Modul bestehen und außerdem weder Globals noch Externals enthalten.

Die Aktivierung bewirkt folgendes:

- a) Laden der Datei iodn.OBJ.  
Diese muß den Konventionen für E/A-Treiber  
(s. E/A-Treiberformat, Utility-Beschreibung) entsprechen
- b) Schutz des durch die Datei iodn.OBJ beanspruchten  
Speicherbereichs
- c) Eintrag des Namens iodn in eine Tabelle der E/A-Verwaltung
- d) Ausführung der im E/A-Treiber enthaltenen  
Kanalinitialisierungsroutine ('INIT')

Hinweise: Dem Treiber iodn wird durch die Aktivierung noch keine  
Kanalnummer zugeordnet.  
Bei KOS6 sind vom Start an eine Reihe von E/A- und Medien-  
Kanäle aktiviert.

**E/A-Treiber Deaktivierung**

Aufruf: IODC \$iodn=DEACTIVE<---

**Funktion:**

Deaktivierung des Ein-/Ausgabetreibers iodn. Dieses Kommando bewirkt folgendes:

- a) Freigabe des durch die Datei iodn beanspruchten  
Speicherbereichs
- b) Löschen des Namens iodn aus der E/A-Tabelle der  
E/A - Verwaltung
- c) Ausführung der im E/A-Treiber enthaltenen  
'CLOSE'-Routine

**Auflistung der aktiven Treiber**

Aufruf: IODC LIST<---

**Funktion:**

Auflistung der momentan aktivierten Ein-/Ausgabetreiber, inklusive der dazugehörigen Kanalnummern und Startadressen. Nach dem Laden des Betriebssystems können beispielsweise folgende E/A- bzw. Medientreiber aktiviert sein:

```

$KEY I-0    ---> Tastatureingabe (Keyboard)
$KEY I-1    ---> Tastatureingabe
$MON O-0    ---> Sichtschirmausgabe (Monitor)
$MON O-1    ---> Sichtschirmausgabe
$MON O-2    ---> Sichtschirmausgabe
$KSM O-3    ---> System-/Fehlermeldungen Ausgabe
$DSKO M-0   ---> Floppy Disk (rechtes Laufwerk) bzw. $WINO/$SLAV
$DSK1 M-1   ---> Floppy Disk (linkes Laufwerk) bzw. frei

```

Die Medienkanalvergabe unter KOS5.33/5.44/5.55 bei KONTRON PSI80/82-Systemen hängt vom KOS-Lademedium ab: Nach Start von einer integrierten Festplatte aus ist M-0 der Festplatte mit Treiber \$WINO zugewiesen, bei KONTRON-KOBUS Slavesystemen ohne eigenen Massenspeicher ist M-0 der Slave-Treiber \$SLAV zugewiesen.

Die Kanäle I-0 und O-0 dürfen nicht undefiniert oder deaktiviert werden. Auch Kanal O-3 ist für Systemausgaben festgelegt und darf außer für den Treiber \$KSML nicht anderweitig verwendet bzw. deaktiviert werden. Bei KOS6 ist Kanal O-3 standardmäßig von \$KSML belegt, dieser Treiber leistet die Ausgabe von System-/Fehlermeldungen mit ausführlicherem Text. Bei KOS5 ist \$KSML Teil der Utility-Diskette und optional auf Kanal O-3 aktivierbar.

Die Medientreiberorganisation bei KOS6 hängt ebenfalls in analoger Weise vom KOS6-Lademedium ab. In jedem Fall kommt der Memory-Disk Treiber \$VMED auf Kanal M-4 dazu.

**Kanalzuweisung**

Aufruf: IODC x-n=\$iodn

**Funktion:**

Weist dem Kanal n einem Treiber zu. Die Kanäle I-0 und O-0 sind fest den Treibern \$KEY und \$MON zugewiesen und können nicht undefiniert werden.

**Allgemeines Kommandoformat für die Zuweisung:**

```

IODC I-n=$iodn ; für Eingabekanäle (1 < n < 9)
IODC O-n=$iodn ; für Ausgabekanäle (1 < n < 9)
IODC M-n=$medn ; für Medientreiber (0 < n < 5 schreibend/KOS4/5)
                                   (0 < n < 9 schreibend/KOS6.05)
                                   (0 < n < 9 lesend)

```

**KOKOM - Kommando (KOBUS-Überwachung)**

**KOKOMS - Kommando**

(nur bei Kontron KOBUS-Systemen)

Format:           KOKOM<---  
                  KOKOMS<---

E/A-Kanäle:       Eingabe ---> I-1  
                  Ausgabe ---> O-1

KOKOM, der Kontron KOBUS Monitor, kann nur im Hardware-KOBUS-Master aufgerufen werden, sein Gegenstück ist KOKOMS, der für den Software-basierenden Masterplatz (nur Kontron PSI80/82) ausgelegt ist.

Diese Programme dienen der Überprüfung der KOBUS-Aktivität im Servicefall: Slave-Stationen können vom Master aus angeschlossen oder abgetrennt werden, aktive Slave-Stationen werden angezeigt, etc.

**Der Zweck dieser Programme ist das Austesten von KOBUS-Software und KOBUS-Anlagen. Wie mit jedem Debugging-Programm führt die unbedachte Benutzung zu ernststen Systemstörungen!**

**Es wird keine Gewährleistung für diese Programme übernommen.**

KOKOM und KOKOMS arbeiten Benutzer-führend. Beim Aufruf erscheint die Monitor-Maske, die folgende Eingaben zuläßt:

- 1 - Alle aktiven Slaves anzeigen.  
    Rückkehr ins Kommandomenü durch die Taste 'ESC'
- 2 - Slave Nummer x aktivieren
- 3 - Slave Nummer x deaktivieren
- 4 - Alle Slaves aktivieren
- 5 - Slaveliste sperren: keine Veränderung, z.B.  
    Einloggen eines weiteren Slaves mehr möglich
- 6 - Slaveliste freigeben
- 7 - Reset KPP (KPPLOAD KPP notwendig)
- 8 - Warmstart KPP (Neuinitialisierung)
- 9 - Anzeige KPP-Speicher

Die Slave-Stationen werden durch Hex-Zahlen unterschieden. Jeder Station wird beim Aktivieren eine Stationsnummer zugewiesen.

Die einfachste Anwendung von KOKOM(S) ist nach einem Ausfall und dem erneuten Hochfahren des Masters: durch Auslösen der Funktion 4 (alle Slaves aktivieren) können alle Slave-Stationen ihre Kommunikation fortsetzen. Dies ist dann sinnvoll, wenn die Stationen unabhängig voneinander Dateien bearbeitet haben. Bei Zugriff auf gemeinsame Dateien müssen Anwendungsprogramme in den Stationen unbedingt einen definierten Wiederanlaufpunkt ansteuern.

Eine Gewähr für die logische Integrität der Dateien wird nicht übernommen.

**KOSGEN - Kommando (Konfigurierung der Systemdateien)**  
(nur bei KOS6)

Aufruf: KOSGEN KOSn<---  
KOSGEN KOSn.SYS<---

'KOSGEN' ermöglicht die Konfigurierung der einzelnen KOSn.SYS Dateien bezüglich der wichtigsten Systemparameter durch den Anwender. Dies betrifft vor allen Dingen das Kaltstartverhalten von I/O- und Medientreibern, sowie im Falle von Medientreibern deren Kanalzuordnung.

Jeder konfigurierbaren Datei KOSn.SYS ist eine Bildschirmmaske KOSn.MAS zugeordnet, z.B.

**KOS5.MAS ----> KOS5.SYS**

Folgende KOSn.SYS Dateien können konfiguriert werden:

Datein	Maske
KOS0.SYS (Boot-Segmente)	KOS0.MAS
KOS4.SYS (\$DSK0/1, \$VMED)	KOS4.MAS
KOS5.SYS (\$WIN0/1)	KOS5.MAS
KOS7.SYS (\$MON, \$KEY)	KOS7.MAS
KOS8.SYS (\$SLVn/KOBUS)	KOS8.MAS
KOSA.SYS (\$WIN2/3)	KOS5.MAS
KOSC.SYS (\$SIOA/B, \$PIO)	KOSC.MAS
KOSD.SYS (IOMEGA)	KOSD.MAS

Alle Zahlenwerte müssen dezimal eingegeben werden.

'KOSGEN' wurde gegenüber früheren Versionen erweitert. Es steht nun ein benutzerfreundlicher Kommandointerpreter zur Verfügung.

a) Eingabemodus

- <CR> Beendet Eingabe und geht auf die nächste Eingabeposition
- <ESC> Geht in den Kommandomodus
- <HOME> Geht zurück zur ersten Eingabeposition
- <↑> Geht zurück zur vorhergehenden Eingabeposition

b) Kommandomodus

- <W> Schreibt die gewählte Einstellung auf die Datei KOSn.SYS
- <R> Repeat, lädt die Datei KOSn.SYS erneut und geht in den Eingabemodus
- <HOME> Geht in den Eingabemodus zurück (erste Position)

Eine weitere Erleichterung betrifft die Voreinstellung 'SYS' für den Dateityp beim Anruf des KOSGEN-Kommandos. Die Kommandos 'KOSGEN KOS4.SYS<CR>' bzw. 'KOSGEN KOS4<CR>' sind also äquivalent.

### Konfigurierung von KOS0.SYS

Aufruf: KOSGEN KOS0<---  
KOSGEN KOS0.SYS<---

In KOS0.SYS wird festgelegt, welche Boot-Segmente (KOSn.SYS) geladen (1) werden bzw. nicht geladen (0) werden.

Die Segmente KOS1.SYS, KOS2.SYS und KOS3.SYS müssen immer gebootet werden.

Standardmäßig werden die Systemmodule KOS0.SYS bis KOS7.SYS geladen, optional werden die Module

KOS8.SYS (KOBUS-Modul, muß an Master und Slave geladen werden)

KOSA.SYS (für externe Plattenlaufwerke)

KOSB.SYS (CP/M-Translator \$CPM)

KOSC.SYS (Standardtreiber \$SIOA, \$SIOB, \$PIO)

KOSD.SYS (für Wechselplattenlaufwerke IOMEGA)  
geladen.

**Konfigurierung von KOS4.SYS**

**Aufruf:** KOSGEN KOS4<---  
KOSGEN KOS4.SYS<---

KOS4.SYS enthält die Medientreiber \$DSK0, \$DSK1 und \$VMED.  
Folgende Parameter sind konfigurierbar:

1. Logische Kanalnummer für \$DSK0
2. Logische Kanalnummer für \$DSK1
2. Logische Kanalnummer für \$VMED
4. Kapazität von \$VMED in Kilobytes
5. ECB/D256-Flag

**a) Logische Kanalnummern**

Logische Kanalnummern liegen im Bereich zwischen 0 und 9. Die Zuweisung erfolgt dynamisch während des Kaltstarts des Betriebssystems entsprechend der spezifizierten Werte. Eine besondere Bedeutung hat der Wert 255. Ein so gekennzeichnete Treiber wird nicht aktiviert und erscheint somit auch nicht im Ausdruck des 'IODC LIST' Kommandos.

Voreinstellung:               \$DSK0 ---> M-0  
                              \$DSK1 ---> M-1  
                              \$VMED ---> M-4

**b) Kapazität von \$VMED**

\$VMED verwendet normalerweise 128 kByte des Systemspeichers. Durch Zustecken von Speicherbaugruppen (z.B. ECB/D256) kann der für \$VMED verfügbare Speicher erweitert werden. Die Kapazität von \$VMED wird in Kilobytes spezifiziert.

Voreinstellung:               128 kBytes

**Hinweis:** Die Baugruppen ECB/D256 sind aufgrund unterschiedlicher MMU Hardware bei Kontron PSI80/82 und bei Kontron PSI98/908/980 in zwei verschiedenen Ausführungen kompatibel zur jeweiligen Hardware verfügbar.



**Konfigurierung von KOS5.SYS**

**Aufruf:** KOSGEN KOS5<---  
KOSGEN KOS5.SYS<---

KOS5.SYS enthält die Medientreiber \$WIN0 und \$WIN1. Beide sind für unterschiedliche 5.25" Winchester Plattenlaufwerke geeignet. Folgende Platten-spezifischen Parameter müssen eingestellt werden.

- Step Pulse Width (P1)
- Step Rate (P2)
- Seek Mode (P3)
- Number of Heads (P4)
- Number of Cylinders (P5)
- Starting Cylinder for Write Precompensation (P6)

Desweiteren sind die logischen Kanalnummern für \$WIN0 bzw. \$WIN1 konfigurierbar (Wertebereich: 0...9). Der Wert 255 verhindert die Aktivierung des entsprechenden Treibers.

Die Frage nach dem Interleaving-Faktor wird in jedem Fall mit 1 (ja) beantwortet.

Tabelle 1 enthält die Programmierparameter für die in Serienprodukten der Reihe Kontron PSI 9xx eingebauten Winchester Laufwerke.

System Kontron PSI	980 Q/W10	908 Q/W10	98/980 Q/W20	98/980 Q/W40	
Laufwerk	ST412	3012	4020	3046	
Hersteller	Seagate	Miniscribe		Atasi	Syquest
Step pulse width	3	3	3	3	3
Step rate	1	1	1	1	1
Seek Mode (**)	1	2	2	2	2
Heads	4	2	4	7	2
Cylinders	306	612	480	635	306
Write Precomp.	128	0	0	321	128
Landing Zone (*)	(0)	(44)	(42)	(8)	(0)

(\*) Landing Zone 0...n...127 (für WFD)

Landezone ist n Cylinder innerhalb des innersten Cylinders (0...n...127) oder n-128 Cylinder außerhalb von Cylinder 0 (128...n...255)

Die Informationen zur 'Landing Zone' sind nur für das physikalische Formatieren mit 'WFD' relevant und werden nicht in KOS5.SYS eingestellt. Der 'Landing Zone' Parameter wird entsprechend der Tabelle ausgewählt und beim Formatieren angegeben.

(\*\*) Seek Mode

- 0 - Non buffered Seek - 3.0 ms rate
- 1 - Buffered Seek - 30 us rate
- 2 - Buffered Seek - 14 us rate

Die Frage in KOS5.SYS nach dem Controllertyp ist mit 1 für Adaptec anzugeben.

Tabelle 1: Programmierparameter für serienmäßig eingesetzte Winchester Laufwerke mit Adaptec Controller ACB 4000.

Diese Programmierung darf vom Anwender nicht verändert werden. Bei der Erstausslieferung ist sichergestellt, daß die Laufwerkparameter entsprechend der Systemkonfiguration eingestellt sind. Updates des Betriebssystems werden standardmäßig mit der Einstellung für Miniscribe 4020 verschickt.

Bei anderen Massenspeicherkonfigurationen sind die Parameter entsprechend den vorliegenden Tabellen zu verwenden.

Tabelle 2 enthält die Programmierparameter für die vor Dezember 1983 in Serienprodukten der Reihe Kontron PSI9xx eingebauten Winchester Laufwerke.

Laufwerk: ---> Parameter	Seagate ST412	Miniscribe 4020	Atasi 3046	Syquest
Step pluse width	3	3	3	3
Step rate	1	1	1	1
Seek Mode	0	0	0	0
Heads	4	4	7	2
Cylinders	306	480	635	306
Write Precomp.	128	1	255	128

Nettokapazität: (Miobytes)	10.980	17.244	39.942	5.472
-------------------------------	--------	--------	--------	-------

Die Frage in KOS5.SYS nach dem Controllertyp ist mit 0 für DTC-Controller zu beantworten.

Tabelle 2: Programmierparameter für Winchester Laufwerke mit DTC-Controller.

Unterscheidungsmerkmal bei Kontron PSI9xx Systemen: Rechner, deren integrierte Festplatte mit Adaptec-Controller ausgerüstet ist, melden sich während des Systemboots mit:

PROM BOOT - AUTOLOAD V6.21/A

Rechner, deren integrierte Festplatte mit DTC-Controller ausgerüstet ist, melden sich während des Systemboots mit:

PROM BOOT - AUTOLOAD V6.20/D

## Konfigurierung von KOS7.SYS

Aufruf: KOSGEN KOS7<---  
KOSGEN KOS7.SYS<---

KOS7.SYS enthält die I/O-Treiber \$MON und \$KEY. Konfigurierbar sind:

1. \$MON - ESC Sequenzen
  - zur direkten Cursor Positionierung
  - zum Löschen des Bildschirms (Formfeed)
2. \$KEY - Kaltstartverhalten
  - Groß-/Kleinschreibung (Konvertierung)
  - Funktionstastenzuordnung an/aus
  - Zeichermaskierung

### ESC-Sequenz zur direkten Cursor Positionierung

Diese ESC-Sequenz ist nur dann relevant, wenn das System über ein Terminal (SIOB) betrieben wird (z.B. PSI 9068). Mit bis zu sechs Parametern (CP/P1...CP/P6) kann die für ein bestimmtes Terminal notwendige ESC-Sequenz festgelegt werden. Alle Werte sind dezimal einzugeben. Es bedeutet:

- 0: No Operation (NOP)
- 1: die Row Adresse (Zeile) wird als Binärwert mit dem Offset 20H ausgegeben  
(Beispiel: Zeile 4 ---> 24H)
- 2: die Column Adresse (Spalte) wird als Binärwert mit dem Offset 20H ausgegeben  
(Beispiel: Spalte 20 ---> 34H)
- 3: die Row Adresse (Zeile) wird im ANSI-Format als Folge von ASCII-Digits ausgegeben  
(Beispiel: Zeile 12 ---> 31H, 32H)
- 4: die Column Adresse (Spalte) wird im ANSI-Format als Folge von ASCII-Digits ausgegeben  
(Beispiel: Spalte 74 ---> 37H, 34H)

Alle übrigen Werte werden unverändert als ASCII-Zeichen übertragen.

**Beispiele:** (Parameter: CP/P1...CP/P6)

Terminal		P1	P2	P3	P4	P5	P6
TDV 2220	(*)	27 (ESC)	91 (C/Ä)	3	59 (;)	4	72(H)
Regent 20	(**)	27 (ESC)	89 (Y)	1	2	0	0
CIT-80	(***)	27 (ESC)	89 (Y)	1	2	0	0
CIT-101/101e	(****)	27 (ESC)	91 (C/Ä)	3	59 (;)	4	72(H)
Kontron 9T	(*****)	27 (ESC)	91 (C/Ä)	3	59 (;)	4	72(H)

**ESC-Sequenz zum Löschen des Bildschirms (Formfeed)**

Diese ESC-Sequenz ist ebenfalls nur relevant, wenn das System über ein Terminal (SIOB) betrieben wird. Vier Bytes können spezifiziert werden (FF/P1...FF/P4).

**Beispiele:**

Terminal		FF/P1	FF/P2	FF/P3	FF/P4
TDV 2220	(*)	27 (ESC)	99 (c)	0	0
Regent 20	(**)	12 (M)	0	0	0
CIT-80	(***)	27 (ESC)	72 (H)	27 (ESC)	74 (J)
CIT-101/101e	(****)	27 (ESC)	91 (C/Ä)	50 (2)	74 (J)
Kontron 9T	(*****)	27 (ESC)	91 (C/Ä)	50 (2)	74 (J)

**Hinweis:** \$MON verwendet außer den beiden erwähnten ESC-Sequenzen keine weiteren.

(\*)        Hersteller: Tandberg  
 (\*\*)       Hersteller: ADDS  
 (\*\*\*)     Hersteller: Citoh, kompatibel zu DEC VT52  
 (\*\*\*\*)    Hersteller: Citoh, kompatibel zu DEC VT100  
 (\*\*\*\*\*) Hersteller: Kontron, kompatibel zu DEC VT100

**\$KEY Kaltstartverhalten**

## a) Groß-/Kleinschreibung Konvertierung

(Voreinstellung: aus)

Ist dieses 'Flag' gesetzt, so wandelt \$KEY alle Großbuchstaben (A...Z) in Kleinbuchstaben (a...z) um und umgekehrt.

## b) Funktionstastenzuordnung

(Voreinstellung: ein)

\$KEY ermöglicht die Zuweisung von Zeichenketten zu Funktionstasten (siehe auch SOFTKEY-Kommando). Die standardmäßig in \$KEY realisierte Zuordnung kann durch ein 'Flag' bereits beim Kaltstart aktiviert bzw. gesperrt werden.

## c) Zeichermaskierung

Voreinstellung: 255 (FFH)

Alle eingegebenen Zeichen werden mit der eingestellten Maske maskiert (UND-Verknüpfung). Damit kann beispielsweise das höherwertige Bit ausgeblendet werden (Maske: 127 = 7FH), da dies bei manchen Terminals (z.B. CIT-80) als Parity Bit verwendet wird.

**Konfigurierung von KOS8.SYS**

Aufruf: KOSGEN KOS8<---  
KOSGEN KOS8.SYS<---

KOS8.SYS enthält die Treiber \$SLV0, \$SLV1, \$SLV2 und \$SLV3. Diesen wird in KOS8.SYS eine Mediennummer (0...9) zugeordnet bzw. der Wert 255, falls ein Treiber nicht aktiviert werden soll. Die am Master mit 'KPPLOAD KPP' als 'public' vereinbarten Medien stehen den aktivierten Treibern zur Verfügung. Hierbei können Slave-Systemen ohne Massenspeicher (Kontron PSI9C) andere Medienkanäle zugeordnet werden als Slave-Systemen mit Massenspeichern (Kontron PSI98/908/980).

Für Slaves ohne Massenspeicher werden die Medienkanalzuordnungen für die Slavetreiber \$SLV0 ... \$SLV3 mittels 'KOSGEN KOS8' am Master eingestellt.

Beispiel:

		Kontron PSI98/908/980	!	Kontron PSI9C
	-----	-----	!	-----
\$SLV0	!	255	!	0
\$SLV1	!	255	!	1
\$SLV2	!	255	!	255
\$SLV3	!	255	!	255

255 = deaktiviert

Für Slaves mit Massenspeichern werden die Medienkanalzuordnungen für die Slavetreiber \$SLV0 ... \$SLV3 mittels 'KOSGEN KOS8' am jeweiligen Slave eingestellt.

Beispiel:

		Kontron PSI98/908/980	!	Kontron PSI9C
	-----	-----	!	-----
\$SLV0	!	5	!	255
\$SLV1	!	6	!	255
\$SLV2	!	255	!	255
\$SLV3	!	255	!	255

255 = deaktiviert

Vor dem ersten Ansprechen eines KOBUS-Medienkanals ist der entsprechende Treiber (\$SLV0 ... \$SLV3) mittel O-Kommando zu eröffnen, z.B. 'O \$SLV0<---'

**Hinweis:** Die auf Kontron PSI80 Slavestationen verfügbaren Treiber \$SLAV (Kontron PSI80 ohne Laufwerk) und \$SLAT (mit Laufwerken) können nur auf das erste mit 'KPPLOAD KPP' als 'public' vereinbarte Medium zugreifen.

**Konfigurierung von KOSA.SYS**

**Aufruf:** KOSGEN KOSA  
KOSGEN KOSA.SYS<---

KOSA.SYS enthält die Medientreiber \$WIN2 und \$WIN3 (externe Winchestererweiterung). Die Konfigurierung ist identisch mit derjenigen von KOS5.SYS.

**Hinweis:** KOSA.SYS wird nur 'gebootet', wenn KOSO.SYS entsprechend konfiguriert wurde.

**Konfigurierung von KOSC.SYS**

**Aufruf:** KOSGEN KOSC<---  
KOSGEN KOSC.SYS<---

KOSC.SYS enthält die Treiber \$SIOA, \$SIOB und \$PIO. Diese können beliebigen verfügbaren E/A-Kanälen zugeordnet werden.

**Hinweis:** KOSC.SYS wird nur 'gebootet', wenn KOSO.SYS entsprechend konfiguriert wurde.

**Konfigurierung von KOSD.SYS**

**Aufruf:** KOSGEN KOSD<---  
KOSGEN KOSD.SYS<---

KOSD.SYS enthält die Treiber \$IOM0 und \$IOM1 für ein 10 MB Wechselplattenlaufwerk (IOMega).

In KOSD.SYS werden den zu aktivierenden Treibern die gewünschten/verfügbaren Mediennummern (0...9) zugeordnet bzw. der Wert 255, wenn ein Treiber nicht aktiviert werden soll.

Der Treiber \$IOM0 wird verwendet, wenn das Wechselplattenlaufwerk das einzige externe Plattenlaufwerk ist (ECB/SASI-Adresse 78h), der Treiber \$IOM1 wird verwendet, wenn das Wechselplattenlaufwerk als zweites externes Plattenlaufwerk an dieselbe ECB/SASI (Adresse 78h) angeschlossen wird. Sonstige in KOSO.SYS vorhandene Treiber müssen deaktiviert sein (255).

**KPPLOAD KPP - Kommando (nur für KOBUS-Anlagen)**

Aufruf: KPPLOAD KPP<---

E/A-Kanäle: Eingabe ---> I-1

**Funktion:**

Dieses Kommando lädt die Datei KPP.COM in die Steuerhardware ECB/KPP eines KOBUS-Hardwaremasters und bringt das Programm KPP.COM innerhalb der ECB/KPP zur Ausführung. Der Benutzer hat die Möglichkeit bis zu 4 'Public Media' zu spezifizieren. Für jedes Medium (0...9) ist die Frage "Media M-n public? (y/n):" entsprechend zu beantworten. Werden mehr als 4 dieser Fragen mit "y" beantwortet, so werden nur die ersten 4 gewertet. Kontron PSI80-Slavestationen haben nur Zugriff auf das erste vereinbarte 'Public Medium'. Die Computerbaugruppe ECB/KPP umfaßt eine eigene Z80A-CPU, 64 KB RAM, Z80A-DMA, Z80A-SIO und die RS422-Schnittstelle. Die Baugruppe wird in den Erweiterungsrahmen des Kontron PSI-Systems eingesteckt, das als KOBUS-Master konfiguriert wird.

KPP.COM überwacht und steuert die Aktivität auf der KOBUS-Verbindung.

Im einzelnen leistet KPP.COM:

- Polling der angeschlossenen KOBUS-Slave-Stationen
- Entgegennahme der Anforderungen auf Übertragung
- Durchführung der Übertragung in Blöcken von 128 Bytes im SDLC-Format
- Puffern der ersten ca. 50 KByte des zentralen Massenspeichers (=Inhaltsverzeichnis) zur Beschleunigung der KOBUS-Zugriffe

Die Datenübertragung über den ECB-Bus erfolgt DMA-gesteuert.

Zusätzliche Informationen sind in der KOBUS-4-Dokumentation enthalten.



**MAKEFS - Kommando (Make File System)**

Aufruf: MAKEFS<---

E/A-Kanäle: Eingabe ---> I-1  
Ausgabe ---> O-1

**Funktion:**

MAKEFS bietet die Möglichkeit, auf KOS-Medien Inhaltsverzeichnisse (Directories) mit Längen in beliebigen Vielfachen von 32 Einträgen anzulegen. Es löst das KOS5.33-Programm SOFTFORM ab.

Damit besteht für den Benutzer die Möglichkeit, auf seinen Medien das Fassungsvermögen des Inhaltsverzeichnisses dem jeweiligen Verwendungszweck anzupassen.

Das Kommando 'MAKEFS' schreibt auf ein bereits formatiertes KOS-Medium ein neues Inhaltsverzeichnis (Directory). Dabei bestimmt der Benutzer interaktiv die Mediennummer, den Namen des Inhaltsverzeichnisses (= den Mediennamen) und die Anzahl der Dateien, für die es ausgelegt wird. MAKEFS rundet die eingegebene Anzahl automatisch auf Vielfache von 32 auf. Falls diese Anzahl später nicht ausreicht, wird das Directory vom Betriebssystem selbstständig erweitert. Diese Erweiterungen stehen aber im allgemeinen nicht mehr am Anfang des Mediums und verlängern so Suchvorgänge.

Nach dem Schreiben des Inhaltsverzeichnisses wird das Medium neu initialisiert, so daß unmittelbar danach Dateien darauf abgelegt werden können.

**Anwendungsbeispiele:**

1. Auf einer Harddisk, die als KOBUS-Mastermedium in der Entwicklungsabteilung verwendet wird, werden viele relativ kurze Dateien von verschiedenen Benutzern abgespeichert. Wählt man hier eine große Directory-Länge (z.B. für 1024 Einträge), so vermeidet man über die Disk verstreute Directory-Erweiterungen und beschleunigt dadurch Dateisuchsequenzen beachtlich.
2. Im kommerziellen Einsatz eines Systems kann der Fall auftreten, daß nur einige, dafür aber relativ große Dateien auf einem Medium abgelegt sind. Hier ist es sinnvoll, das Directory klein zu halten (z.B. 128 Einträge). Dadurch werden ebenfalls schnelle Suchsequenzen erzielt. Daneben wird auch zusätzliche Medienkapazität für den Anwender freigehalten.

**MAP - Kommando (Auskunft über Speicherbelegung)**

Aufruf: MAP<---

Voreinst.: -

E/A-Kanäle: Eingaben: ---> Kanal I-1  
Ausgaben: ---> Kanal O-1

**Funktion:**

Ausdruck der aktuellen Speicherbelegung von KOS. Zur Verwaltung des Speichers teilt KOS den vorhandenen Speicher in Segmente zu je 80H Byte ein. Ein 'B' im Ausdruck kennzeichnet ein belegtes, ein '.' ein freies Segment. Bei folgenden Gelegenheiten werden Speichersegmente belegt:

- Laden eines Programms
- Aktivierung eines E/A-Treibers
- Ausführung einer Kommandodatei (DO-Kommando)
- Ausführung des A-Kommandos
- Beim ersten Ansprechen eines Medientreibers
- Beim Abspeichern einer Kommandodatei
- Durch die Menü-Auswahl RLOAD SELECT
- Beim Aktivieren von Tasks

Die Freigabe der entsprechenden Segmente erfolgt analog dazu bei:

- der Rückkehr von einem Programm
- der Deaktivierung eines E/A-Treibers
- dem Ende einer Kommandodatei-Abarbeitung
- der Ausführung des D-Kommandos
- dem Ende des DO-Kommandos
- dem Verlassen des SELECT-Menüs
- beim dauernden Deaktivieren von Tasks

**Hinweis:** Das MAP-Kommando selbst belegt den Speicher wie andere Programmdateien auch ab Adresse 100H. Es kann deswegen nicht zum Ausdruck des Speicherbelegungsplans verwendet werden, wenn ein anderes Programm mit Ladeadresse 100H geladen ist. Der durch solche Programme belegte Speicherraum kann aus der Auflistung des IL-Kommandos entnommen werden.

**MOVE - Kommando (Kopieren von Dateien)**

Aufruf:            MOVE /wdir1/mn1:name.typ /wdir2/mn2 param<---  
                  MOVE /wdir1/mn1:dateigruppe /wdir2/mn2 param<---

Voreinst.:        quellmedium mn1 = 0  
                  name                = \*  
                  typ                  = \*  
                  zielmedium mn2 = 1    falls quellmedium = 0  
                                     bzw. 0    falls quellmedium = 1  
                  param                = N    nein, nicht rückfragen  
                                         P=nicht geheim  
                  wdir                 = aktuelles wdir

E/A-Kanäle:       Eingabe ---> Kanal I-1  
                  Status ---> Kanal I-1  
                  Ausgabe ---> Kanal O-1

**Funktion:**

Kopieren von einzelnen Dateien oder ganzen Dateigruppen. Der Dateiname bleibt hierbei ohne Rückmeldung erhalten. Bereits vorhandene Dateien des angegebenen Namens werden überschrieben. Die Angabe von 'quellmedium:', 'name', 'typ', 'zielmedium' und 'param' ist optional, die Reihenfolge von zielmedium und param ist beliebig. Das Kopieren einer Datei auf sich selbst ist nicht zulässig.

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

- J - Die Datei wird kopiert
- Y - Die Datei wird kopiert
- N - Die Datei wird nicht kopiert
- A - Alle folgenden Dateien der angegebenen Spezifikation werden kopiert. Rückfragen werden nicht mehr gestellt.
- K - Abbruch des Kommandos (KOS-Rücksprung)  
       (dies ist auch mit der Taste 'ESC' möglich)

Im Unterschied zum COPY-Kommando können mit dem MOVE-Kommando neben einzelnen Dateien auch Dateigruppen kopiert werden. Bei Dateien mit Properties muß der Parameter 'P=properties' angegeben werden (siehe 'DEFP'-Kommando).

**Die Angabe des Namens oder Typs der Zieldatei ist nicht zulässig!**

## Beispiele:

MOVE \*<---

Kopiert alle nicht geheimen Dateien von Medium 0 auf Medium 1.

MOVE 1:TEST.\* J P=<---

Kopiert alle Dateien des Namens 'TEST' (Typ beliebig) von Medium 1 auf Medium 0, param = 'J' bewirkt eine Rückfrage, so daß der Anwender entscheiden kann, ob eine bestimmte Datei übertragen werden soll oder nicht.

MOVE /abc/2:\* /def/3:

Kopiert alle Dateien aus Working Directory "abc" des Mediums 2 ins wdir "def" des Mediums 3.

Hinweis: Bei Dateien, die unter KOS4.33/5.33 erstellt wurden, kann ein Eintrag 'wdir' bereits vorliegen. Diese Dateien sind zunächst nicht sichtbar.

**A**hilfe: Durch WDIR-Kommando 'Super User Modus' einstellen (s. dort), dann MOVE \* /abc/mn: J<---

**PRINT - Kommando (Ausgabe einer ASCII-Datei)**

Aufruf: PRINT /wdir/mn:name.typ param<---

Voreinst.: mn = Mastermedium  
typ = PRN  
param = 0=\$MON  
wdir = aktuelles wdir

Dateispez.: EDA

E/A-Kanäle: Status ---> Kanal I-1  
Fehlermeldungen ---> Kanal O-1  
Dateiausdruck ---> Kanal O-2 oder O-6  
falls 0=\$iodn

**Funktion:**

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

Es wird überprüft, ob die spezifizierte Datei ausschließlich ASCII-Zeichen enthält. Ist dies nicht der Fall, so erfolgt der Ausdruck bis zum ersten Nicht-ASCII-Zeichen und der Hinweis:

---> PRINT: Datei enthält unzulässige (NICHT-ASCII-) Zeichen

**Beispiele:**

PRINT 1:TEST.SRC<---

Ausdruck der Datei TEST.SRC von Medium 1 auf den Bildschirm.

PRINT TEST.PRN 0=\$SIOA<---

Ausdruck der Datei TEST.PRN auf den Ausgabetreiber \$SIOA. Die Zuweisung auf Kanal O-6 erfolgt automatisch.

**PSTAT - Kommando (Ausgabe des Programmstatus)**

Aufruf:           PSTAT mn<---  
Voreinst.:       mn   = Mastermedium  
E/A-Kanäle:      Eingabe ---> Kanal I-1  
                  Ausgabe ---> Kanal O-2

**Funktion:**

Zur automatischen Auswertung des Status aller COM- und OBJ-Dateien einer Diskette steht unter KOS das Programm 'PSTAT' zur Verfügung. Folgende Informationen werden aufgelistet:

- a) Programmname und Typ
- b) Versionsnummer
- c) Datum der Erstellung bzw. letzten Änderung
- d) Version des Betriebssystems, ab der das Programm ablauffähig ist.

Die Ausgabe der Information erfolgt auf Kanal O-2 und kann somit leicht durch Zuweisen eines Druckertreibers auf O-2 (IODC-Kommando) auf den Drucker dirigiert werden.

PSTAT setzt ein bestimmtes Format des auszuwertenden Programms voraus. Dies ist:

```
Programmanfang:  jr Start
                  defm 'Name x.y(4.3/5.3/6.0)-'
                  defm 'Datum'
                  defw 0ff00h

Start:           ; hier beginnt das eigentliche
                  Programm
```

Bei E/A- und Medientreiber muß obige Information nach dem Einsprungspunkt stehen.

Es bedeutet:	x.y	-	Versionsnummer des Programms
	4.3/5.3/6.0	-	ablauffähig ab KOS4.3/KOS5.3/KOS6.0
	Datum	-	Format: dd.mm.jjjj

Stimmt das Programmformat nicht mit dem eben beschriebenen überein, so gibt PSTAT anstelle der Versionsnummer und des Datums Striche aus (Beispiel: BASIC, wo das Format aus verschiedenen Gründen nicht eingehalten werden kann). Beachten Sie bitte, daß Fremdsoftware (z.B. von Microsoft) gewöhnlich nicht mit PSTAT ausgewertet werden kann.

Hinweis: Der Status einzelner COM-Dateien kann auch mit dem PRINT-Kommando ausgegeben werden.

**REN - Kommando (RENAME = Umbenennung einer Datei)**

Aufruf:               REN /wdir/mn:name1.typ1 name2.typ2 param<---

Voreinst.:           mn       = Mastermedium  
                   typ1     = \*  
                   typ2     = typ1  
                   param    = N (nein, nicht rückfragen)  
                             = P nicht geheim  
                   wdir     = aktuelles wdir

**Funktion:**

Umbenennung der Datei name1.typ (alté Dateiadresse) in name2.typ (neue Dateiadresse). Bei der Dateiadresse kann der Typ durch den Universalbezeichner '\*' ersetzt werden. In solchen Fällen werden alle Dateien des Namens name1 umbenannt. Die Namen der umbenannten Dateien werden ausgegeben. Ist für param = J angegeben, so erfolgt vor der Umbenennung eine Rückfrage an den Benutzer, worauf fünf Eingaben möglich sind:

- J - Die Umbenennung wird durchgeführt
- Y - Die Umbenennung wird durchgeführt
- N - Die Umbenennung wird nicht durchgeführt
- A - Alle folgenden Dateien der angegebenen Spezifikation werden (ohne neuerliche Rückfragen) umbenannt.
- K - Abbruch des Kommandos (KOS-Rücksprung)

Die Datei-Properties müssen angegeben werden (siehe 'DEFP'-Kommando).

**Beispiele:**

REN BASIC.COM BASICNEU<---

Die Datei 'BASIC.COM' des Mastermediums wird in 'BASICNEU.COM' umbenannt. Eine Rückfrage wird nicht gestellt.

REN /abc/1:TEST.\* TEST1 J<---

Alle Dateien des Namens 'TEST' (Typ beliebig) im wdir 'abc' auf Medium 1 werden in 'TEST1' umbenannt, wobei zuvor jeweils eine Rückfrage gestellt wird.

REN \*.SRC \*.MAC<---

Alle Dateien mit Typ 'SRC' werden in Dateien mit Typ 'MAC' umbenannt.

REN ALT.DIR NEU P=D<---

Ändern des Mediennamens.

**Wichtiger Hinweis zum letzten Beispiel:**

KOS erkennt am Mediennamen einen evtl. Diskettenwechsel. Bei gleichem Medienamen der gewechselten Diskette muß der Diskettenwechsel durch das KOS-Kommando 'N' angezeigt werden.

**RLOAD - Kommando (Lade relocativ)**

Aufruf: RLOAD /wdir/mn:name.typ

Voreinst.: mn = Mastermedium  
typ = OBJ  
wdir = aktuelles wdir

E/A-Kanäle: Eingabe ---> Kanal I-1  
Rückmeldungen ---> Kanal O-1

**Funktion:**

Die spezifizierte Objektdatei 'name.typ' wird geladen und relokiert (gelinkt). Bei der Objektdatei muß es sich um ein Modul ohne Externalis und Globals handeln. Der vom Relokator erzeugte Code wird so hoch wie möglich im Speicher abgelegt und der entsprechende Speicherplatz wird belegt.

Abschließend wird das erzeugte Programm gestartet.



**RLOAD PTASK-Kommando (Ausgabe von ASCII-Dateien in der Hintergrundverarbeitung)**

Aufruf: RLOAD PTASK param<---

Voreinst.: param = J

E/A-Kanäle: Ausgabetreiber ---> 0-8  
Ausgabe ---> 0-1

**Funktion:**

Aktivierung der Task 'SPOOL' und Ausgabe von ASCII-Dateien in der Hintergrundverarbeitung durch Ausgabetreiber.

Vor dem erstmaligen Aufruf des Kommandos 'SPOOL' muß durch 'RLOAD PTASK' die Task 'SPOOL' aktiviert werden. Die Datei- und Treibernamen für die Ausgabe werden durch das Programm 'SPOOL' in eine Warteschlange eingetragen. 'PTASK' verwaltet diese Warteschlange. (Die Task 'SPOOL' wird durch das Kommando 'RLOAD PTASK' eingerichtet und darf nicht mit dem 'SPOOL'-Kommando verwechselt werden).

Durch die Task 'SPOOL' werden die Datei und der Treiber des jeweils ersten Auftrags in der Warteschlange geöffnet. Dieser Auftrag wird dann in der Warteschlange gestrichen. Bei jeder Taskausführung werden dem Treiber Zeichen zur Ausgabe übergeben, bis dieser 'Busy' meldet. Wenn EOF (Dateiende) der Datei erkannt wird, werden die Datei und der Treiber geschlossen. Der nächste Auftrag wird ausgeführt.

Vor jede Druckdatei wird ein Kopf gedruckt, der den Dateinamen und das Login enthält.

Durch param = N wird dieser Kopf unterdrückt.

Die Task 'SPOOL' belegt ca. 2 kByte Speicherplatz. Dieser wird bei der Deaktivierung der Task durch das Kommando TASK wieder freigegeben.

**Beispiel:**

RLOAD PTASK<---

Die Task 'SPOOL' wird aktiviert. Die Liste der Aufträge in der Warteschlange ist leer. Erst wenn durch das Kommando SPOOL Aufträge übergeben werden, werden diese im Hintergrund abgearbeitet.

RLOAD PTASK N<---

Drucken ohne automatisch vorgeschalteten Kopf.

**RLOAD SELECT - Kommando (Menügesteuerte Programmverzweigung)**

Aufruf: RLOAD SELECT /wdir/mn:programmliste.typ

Voreinst.: mn = Mastermedium  
 programmliste = PROGLIST  
 typ = SEL  
 wdir = aktuelles wdir

E/A-Kanäle: Eingabe ---> Kanal I-1  
 Rückmeldungen ---> Kanal O-1

**Funktion:**

Ermöglicht unter Benutzerführung das Verzweigen in bis zu 10 verschiedene Kommandofolgen. Die Programmliste ist eine eigenständige Datei und wird über den Editor erstellt. In der einfachen Ausführung enthält diese Liste für jedes auszuführende Programm zwei Zeilen. In der ersten Zeile steht ein beliebiger benutzerdefinierbarer Text, der vom SELECT-Kommando angezeigt wird. In der zweiten Zeile steht die dazugehörige KOS-Kommandozeile. Enthält die Programmliste mehr als 10 Anzeige-Zeilen, so ist die Datei für das SELECT-Kommando nicht auswertbar.

In der erweiterten Ausführung (ab Mai 1982) der Menüliste können Zeilen mit einem Sonderzeichen beginnen. Dabei bedeutet:

- ; Anzuzeigender Titel, maximal 71 Zeichen
- = Hervorhebungszeichen, wenn nicht angegeben: CTRL-Q
- + Verlängerungszeile zu einer Kommandozeile
- Kommandozeile mit Stop nach Ausführung
- ! Aufruf eines anderen Selektionsmenüs, enthält nur den Namen der aufzurufenden Menüliste

Zeilen ohne Sonderzeichen sind weiterhin paarweise Text- bzw. Kommandozeilen.

Die Anwahl innerhalb des Menüs erfolgt durch die Pfeiltasten (Cursor up/down), die gerade angewählte Zeile ist durch Invertierung hervorgehoben. Das Ausführen der gerade angewählten Zeile erfolgt durch die Taste "RETURN"!

Die Darstellung am Bildschirm wird durch eine Maske bestimmt, die als Datei MENU.MAS (KOS5.44/5.54) bzw. MENU16.MAS und MENU10.MAS (KOSV6.05) abgelegt ist. Diese Datei ist durch EDIT editierbar. Innerhalb der Maskendatei ist die Hervorhebung der gerade angewählten Kommandozeile einstellbar. Steht in der ersten Zeile das Zeichen '=', gefolgt von einem Code, dann wird dieser Code zur Hervorhebung dargestellt. Die Voreinstellung ist CTRL-Q (Invertierung).

Eine baumartige Verschachtelung des SELECT-Kommandos ist möglich. Die jeweilige Schachtelungstiefe wird durch eine Zahl in der Titelzeile angezeigt.

Das SELECT-Programm kann nur durch Eingabe von CTRL-K verlassen werden. Dabei wird jeweils eine Schachtelungstiefe zurückgeschaltet. Bei Aufruf des höheren Menüs wird die Schachtelungstiefe ebenfalls zurückgeschaltet.

Beispiel einer einfachen Programmliste:

```
Neueste Informationen
INFO 0:KOS
Serielle Schnittstelle SIOA initialisieren
IODC $SIOA=ACTIVE
CP/M-Translator aktivieren
IODC $CPM=ACTIVE
Submenü auswählen
RLOAD SELECT SUB
```

Der Name dieser Datei sei 'LIST.SEL', damit lautet der Aufruf:

```
RLOAD SELECT LIST<---
```

Beispiel für ein erweitertes Menü:

```
;Hauptmenü
Neueste Informationen
INFO KOS
Textverarbeitung
IODC $CPM=ACTIVE;RLOAD XMON (RLOAD XMON nur bei KOS4/5)
+C;WS;C;N
Speicherbelegung darstellen
-MAP
Submenü anwählen
!SUB
```

Das SELECT-Kommando, wie auch das DO-Kommando reduzieren den für andere Programme verfügbaren Speicher. Unter KOS5 ist dabei die Sonderstellung des Bereichs 8000-BFFF zu beachten: dieser Bereich wird zeitweise zur Bildspeicherbehandlung gemapped. Deswegen dürfen in diesem Bereich keine Programme liegen, die zur Interruptverarbeitung dienen. Ein Beispiel dafür ist Grafiktask GRAPTASK. Solche Programme sollten außerhalb des SELECT-Menüs aktiviert werden, möglichst bereits in KOS.INI.

Eine andere Möglichkeit besteht darin, die Datei SELECT.OBJ fest als Programm mit Startadresse 8000 hex zu linken:

```
LINK SELECT/P:8000/E<---
```

Dadurch entsteht die Programmdatei SELECT.OOM, die durch 'SELECT<---' aufgerufen wird.

**RLOAD TIME - Kommando (Ausgabe der Uhrzeit auf dem Bildschirm)**

Aufruf: RLOAD TIME<---

E/A-Kanäle: Eingaben ---> Kanal I-1  
Ausgaben ---> Kanal O-1

**Funktion:**

KOS4/5: Aktivierung des Programms 'TIME', die in der Hintergrundverarbeitung eine Uhr softwaremäßig nachbildet. Die Task meldet sich mit der Aufforderung:

Bitte Uhrzeit eingeben (hh:mm:ss):

Die Eingabe von Minuten und Sekunden ist nicht zwingend vorgeschrieben. Als Trennzeichen sind ':' und ' ' zulässig.

Hinweis: Mit 'RLOAD TIME' werden die Tasks 'CLOCK' und 'DISPLAY' aktiviert. Die Task 'CLOCK' sollte die am höchsten priorisierte Task sein. Dies wird automatisch erreicht, wenn 'TIME' vor anderen Hintergrundprogrammen aufgerufen wird.

KOS6: Das Programm bringt die Uhrzeit zur Anzeige. Setzen der Uhrzeit siehe "SETTIME" in "KOS-Utilities".

Allgemeines: Die am Bildschirm angezeigte Uhrzeit ist im Arbeitsbereich von KOS unter folgenden Adressen hinterlegt:

Adresse (hex)	Inhalt (hex)
001C	Sekunden
001D	Minuten
001E	Stunden

Der Quellcode dieses Programms ist als Beispiel für ein TASK-orientiertes Programm auf der Utility-Diskette enthalten. Zusätzliche Informationen siehe "KOS-Utilities".

**SETDATE** - Kommando (Datum setzen)  
**SETTIME** - Kommando (Uhrzeit setzen)

Aufruf: SETDATE<---  
          SETTIME<---

Ab KOS6.04 werden beim Kaltstart Uhrzeit und Datum aus dem 'Real Time Clock'-Baustein der Computer Hardware ausgelesen, wobei automatisch die zugrunde liegende Hardwarekonfiguration verschiedener Systeme (PSI980 mit TCB/IOV, PSI908/9C/98 mit KDT6) ermittelt wird. Uhrzeit und Datum werden in die dafür reservierten Systemspeicherstellen übertragen (Datum: OE/OFh, Uhrzeit 1C/1D/1Eh), wo sie auf Wunsch im Sekundentakt verändert werden können. Dem Anwender stehen somit Datum und Uhrzeit in hardwareunabhängiger Form zur Verfügung.

Die Programme SETDATE und SETTIME erlauben benutzerführend das Setzen von Datum und Zeit. Das Datum der Uhrenbausteine ist unabhängig vom Datum eines Mediums, das über das Systemkommando DATE verändert werden kann.

Bei den Uhrenbausteinen auf KDT6-basierenden Systemen (Kontron PSI908/9C/98) wird die Jahreszahl nicht mitgeführt, bei Jahreswechsel muß die Jahreszahl mittels 'SETDATE' erhöht werden und mit CTRL-Y auf das Systemmedium geschrieben werden.

**SPOOL - Kommando (Ausgabeaufträge annehmen)**

Aufruf: SPOOL /wdir/mn:name.typ \$iodn<---  
SPOOL =LIST<---  
SPOOL =DELETE<---

Voreinst.: mn = Mastermedium  
wdir = aktuelles wdir

E/A-Kanäle: Fehlermeldung 0-1

**Funktion:**

Einfügen eines Ausgabeauftrages in die Warteschlange. Die Datei wird eröffnet. Es wird geprüft, ob der E/A-Treiber aktiviert ist. Anschließend wird der Zeiger auf den DSB der Datei und der Treibername an die TASK 'SPOOL' übergeben. Maximal können 10 Aufträge in die Warteschlange aufgenommen werden. Die Task 'SPOOL' muß mit dem Kommando 'RLOAD PTASK' aktiviert sein. Diese Task gibt die Dateien aus der Warteschlange nacheinander über die angegebenen Treiber in der Hintergrundverarbeitung aus. Die Treiber müssen spoolfähig sein.

Mit dem Kommando 'SPOOL =LIST' wird die Liste der Dateien in der Warteschlange ausgegeben.

Mit dem Kommando 'SPOOL =DELETE' wird das Drucken angehalten. Die laufenden Druckaufträge werden durchnummeriert aufgelistet. Durch Angabe der Nummer wird dieser Auftrag aus der Warteschlange gelöscht.

**Anmerkungen:**

1. Während des Ausdrucks der Datei darf diese nicht im Vordergrund geschlossen werden. Das Kommando 'N' schließt alle Dateien auf allen Medien.
2. Geöffnete Dateien sollten nicht anderweitig bearbeitet werden. Es ist z.B. zwar möglich, aber unsinnig, die Datei TEST.PRN im Hintergrund auszudrucken, und gleichzeitig im Vordergrund die Datei TEST.SRC mit dem Befehl 'ASM =TEST/L' zu assemblieren.
3. Die Ausgabedateien werden mit einem Vorspann gedruckt, der Dateiname und Login enthält. Dieser Vorspann kann durch entsprechende PTASK-Aktivierung unterdrückt werden: Siehe "RLOAD PTASK NK---".

## Spooling in Kontron KOBUS System

Die bisher beschriebenen SPOOL-Formate gelten für Kontron PSI-Einzelplatzsystem und für lokales Spooling von Kontron KOBUS-Stationen.

KOBUS-Slavestationen können Spool-Aufträge an die Masterstation und deren Peripherie absetzen. Die Druckdatei liegt immer auf dem zentralen Massenspeicher des Masters.

Der Aufruf lautet:

```
SPOOL /wdir/mn:name.typ $SLxx $iodn<---  
SPOOL =LIST $SLxx<---
```

mit:

```
$SLxx=$SLAV bei Kontron PSI80 ohne Laufwerke  
=$SLAT bei Kontron PSI80 mit Laufwerken  
=$SLV0 bei Kontron PSI9xx  
=$SLV1 bei Kontron PSI9xx  
=$SLV2 bei Kontron PSI9xx  
=$SLV3 bei Kontron PSI9xx
```

Durch Angabe von '\$SLxx' wird der Spoolauftrag in die Warteschlange der Masterstation eingetragen. Die Liste der gerade laufenden Druckaufträge wird am Slave angezeigt. mn ist die Mediumnummer von \$SLxx und kann entfallen, wenn \$SLxx das Mastermedium ist, ebenso ist die Angabe von wdir optional.

Das Löschen von Druckaufträgen ist nur an der KOBUS-Master-Station möglich ('SPOOL =DELETE').

**Hinweis:** Von Kontron PSI80 Slavestationen aus kann nur auf das erste von 4 möglichen 'Public Media' zugegriffen werden.

Beispiele:

```
SPOOL 1:TEST.PRN $SIO
```

Die Datei TEST.PRN soll durch den lokalen E/A-Treiber '\$SIO' ausgegeben werden. Sie wird in die Warteschlange aufgenommen. Die Druckausgabe erfolgt ohne Druckkopf, sofern das PTASK-Kommando entsprechend ausgeführt wurde (RLOAD PTASK N<---).

```
SPOOL =LIST
```

Mit diesem Kommando wird die Liste der noch wartenden Aufträge ausgegeben.

```
SPOOL 3:TEST.SRC $SLxx $PIO
```

Das SPOOL-Kommando im KOBUS Slave löst das Ausdrucken der Datei TEST.SRC am Master aus. \$PIO ist der spoolfähige Ausgabetreiber im Masterplatz. Medienkanal 3 ist in diesem Beispiel \$SLxx. Die Datei TEST.SRC befindet sich bereits auf dem zentralen Massenspeicher.

**STATUS - Kommando (Information über Medienbelegung)**

Aufruf: STATUS mn<---

E/A-Kanäle: Ausgaben ---> Kanal 0-1

**Funktion:**

Ausgabe der Medienbelegung entsprechend des untenstehenden Beispiels. Vor dem Ausdruck führt 'STATUS' automatisch eine Neuinitialisierung aller aktiven Medien durch. Dabei werden die offenen Dateien geschlossen. Bei Angabe der Mediennummer mn ( $0 \leq mn < 9$ ) wird nur der Status dieses Mediums ausgegeben.

**Beispiel:**

STATUS<---  
Ausgabe des Status aller aktiven Medien.

STATUS 2<---  
Ausgabe des Status von Medium 2.

**Beispiel:**

STATUS<---

**Tabelle der Speichermedien**

Kanal	Treiber	Medien-Name	Kapazität	belegt	frei	Recordl.
0	\$DSK0	DISK1	308k	275	33	256 Byte
1	\$DSK1	DISK2	308k	175	133	256 Byte



**STOP - Kommando (Programmierte Unterbrechung von Kommandodateien)**

Aufruf: STOP param<---

Voreinst.: param = J

E/A-Kanäle: Eingaben ---> Kanal I-1  
Ausgaben ---> Kanal O-1

**Funktion:**

Das Kommando 'STOP' führt eine Unterbrechung der Ausführung von Kommandodateien oder Kommandozeilen durch und ermöglicht danach einen Abbruch der Abarbeitung der Kommandozeile oder -datei. Hierzu stellt 'STOP' die Rückfrage:

<--- STOP: Kommandozeile/datei abbrechen? (J/ESC)

und wartet anschließend auf eine Benutzereingabe. Bei den Tasten 'J' und 'ESC' werden weitere Kommandos in der Kommandozeile oder -datei nicht mehr ausgeführt. Bei allen anderen Zeichen werden die Kommandos nach dem STOP-Kommando weiter bearbeitet. Ist die Option 'N' angegeben, so unterbricht 'STOP' nur dann, wenn vorher irgendeine Taste gedrückt war.

**Beispiele:**

EDIT TEST.SRC;ASM=TEST/L;STOP;LINK TEST/N,TEST/E<---

Das LINK-Kommando wird nicht ausgeführt, wenn beim STOP-Kommando die Tasten 'J' oder 'ESC' gedrückt werden. (Sinnvoll, wenn beim ASM-Lauf Fehler auftraten).

DRUCK.CMD  
EDIT TEST.SRC;ASM=TEST/L  
STOP  
LINK TEST/N,TEST/E  
STOP N  
TEST

Eine weitere Möglichkeit, die Fortführung eines DO-Kommandos zu verhindern, ist die Eingabe von CTRL-C.

**SYSGEN - Kommando (Eintragung der gültigen Benutzernamen)**  
(nur bei KOS5.5x und KOS6.xx)

Aufruf:           SYSGEN KOSO<---   (nur bei KOS6.xx) für KOSO.SYS  
                  SYSGEN KOS<---    (nur bei KOS5.5x) für KOS.SYS  
                  SYSGEN KOBUS<---   (nur bei KOS5.5x) für KOBUS.SYS  
                  SYSGEN KCP<---    (nur bei KOS5.5x) für KCP.SYS

E/A-Kanäle:       Eingabe ---> I-1  
                  Ausgabe ---> O-1

Das Kommando SYSGEN gestattet das Eintragen von Benutzer-Identifikationen zum Login und zugehöriger Working-Directories in die Dateien KOSO.SYS, KOS.SYS, KOBUS.SYS bzw. KCP.SYS auf dem Mastermedium. SYSGEN benötigt die Maskendatei SYSGEN.MAS zum Ablauf.

SYSGEN arbeitet mit Benutzerführung. Die möglichen Eingaben sind:

- E     Editiere ID-Liste.** Die Eingabe der 'RETURN'-Taste läßt den bestehenden Eintrag unverändert. Namen können in beliebige Positionen der Liste eingegeben werden, jeweils paarweise (Login und zugehöriges wdir). Mit 'ESC'-Taste im Login-Feld wird der Editiermodus verlassen.
- G     Blättern in der Liste.** Eine Gruppe von 8 Einträgen wird angezeigt. 4 Gruppen sind verfügbar.
- W     Speichern (WRITE) der neu editierten Liste.** Erst mit diesem Kommando werden die Einträge in die Datei KOS.SYS des Mastermediums abgelegt. Dabei wird noch einmal die Bestätigung durch Eingabe von 'Y' gefordert.
- H     Ausgabe der Hilfstexte zur Erklärung der Eingaben.** Weiterschalten mit 'Y'.
- K     Rückkehr zu KOS.**

Fehlerzustände werden im Quittierungsfeld durch "FF" angezeigt.

Hinweis: Es ist zweckmäßig, anschließend an SYSGEN die '.SYS'-Dateien als schreibgeschützt, geheim, public und löschgeschützt zu definieren. (DEFP-Kommando mit Setzen der Dateieigenschaften S, K, E, W).

**TASK - Kommando (Information über Hintergrundverarbeitung)**

Aufruf: TASK<---

E/A-Kanäle: Eingabe ---> I-0  
Ausgabe ---> O-0

**Funktion:**

Information über Hintergrundverarbeitung und Funktionsausführungen. Die Liste der aktiven Hintergrund-Tasks wird angezeigt. Außer dem Namen werden die Parameter STATUS, AMS, PCNT, TEP und TLA einer TASK ausgegeben. Die Bedeutung der Parameter sind im Kapitel Hintergrundverarbeitung, Technische Beschreibung Teil C "Betriebssystem KOS" beschrieben.

Anschließend wird ein Funktionscode abgefragt, mit dem verschiedene Aktionen, wie Deaktivieren oder Reaktivieren, aufgerufen werden können:

- 1 = **Nach KOS zurückspringen**  
Benden des Programms TASK und Rückkehr zum Betriebssystem KOS
- 2 = **Task deaktivieren**  
Task-Nr. wird abgefragt und anschließend deaktiviert, d.h. aus der Liste der Hintergrund-Tasks gestrichen und der belegte Speicherbereich freigegeben.
- 3 = **Task temporär deaktivieren**  
Task-Nr. wird abgefragt und anschließend nur zeitweise deaktiviert, d.h. sie wird nicht aus der Liste gestrichen, sondern vorübergehend nicht ausgeführt.
- 4 = **Task reaktivieren**  
Task-Nr. wird abgefragt und eine temporär deaktivierte Task wird wieder aktiviert, d.h. die Task wird wieder ausgeführt.
- 5 = **Prioritäten ändern**  
Die Prioritätsliste wird abgefragt und danach die Tasks unsortiert. Die Prioritätsliste besteht aus den Task-Nummern in der gewünschten Reihenfolge getrennt durch Komma oder Leerzeichen. Nicht angegebene Tasks werden angeschlossen.
- 6 = **Parameter setzen**  
Task-Nr. wird abgefragt und eine Parameterübergabe an die Task wird ausgeführt.
- K = **zurück zu KOS**

**Beispiel:**

TASK<---

Gibt die Liste der Hintergrund-Tasks aus. Anschließend können die verschiedenen Funktionen angewählt werden.

**WDIR - Kommando (Wahl des Working Directory)**  
(nur bei KOS5.5x/6.xx)

Format: WDIR&lt;---

E/A-Kanäle: Eingabe ---> I-1  
Ausgabe ---> O-1**Funktion:**

In einem System können nebeneinander beliebig viele Working Directories verwendet sein. Kennzeichnend ist jeweils der Name, der aus maximal 15 druckbaren Zeichen besteht. Diese Zeichenfolge wird in einer CRC-Rechnung auf 16 Bits komprimiert und so im Dateieintrag des Inhaltsverzeichnisses gespeichert. Maßgebend ist dabei das Working Directory, unter dem die Datei generiert wurde. Dateien mit einem Benutzerkennzeichen (s. DEFP-Kommando) haben statt des 'wdir' die Benutzeridentifikation eingetragen.

Das aktuell gültige 'wdir' wird im Betriebssystem festgehalten. Nach dem Start des Systems wird das dem jeweiligen Login zugeordnete 'wdir' als aktuelles 'wdir' eingetragen. Der Wechsel des momentanen Working Directories erfolgt durch das diskresidente Systemkommando WDIR. Es zeigt das momentan gültige Working Directory an. Jede beliebige Folge von maximal 15 Druckzeichen kann als Name eines neuen Working Directory eingegeben werden. Damit arbeitet der Anwender unter dem neuen Working Directory.

Das Programm WDIR arbeitet mit Benutzerführung. Folgende Eingaben sind möglich:

- CNTR-S** Super User-Modus einschalten. Alle nicht geheimen Dateien werden unabhängig von ihrem Working Directory sichtbar.
- C** Change: Working Directory umschalten. WDIR fordert den Namen des neu angewählten WDIR's an, der durch 'RETURN' abgeschlossen wird.
- P** Ausdruck (Print) der Liste der bisher erfolgten Working Directory-Einstellungen.
- K** Zurück zu KOS

Bei jedem Wechsel des Working Directories wird eine Datei WDIR.LST auf dem aktuellen Masterlaufwerk fortgeschrieben. Diese Datei enthält die in der Vergangenheit angewählten Working Directories. Sie darf nicht als 'schreibgeschützt' gekennzeichnet sein. Sie sollte "Public" sein, damit sie nicht mehrfach in verschiedenen wdirs angelegt wird.

Mit der Eingabe 'P' unter WDIR wird diese Liste durchgeblättert. WDIR.LST kann gelöscht werden mit dem DEL-Kommando.

Zum Programm WDIR gehört die Maskendatei WDIR.MAS, die auf dem gleichen Medium existieren muß. Falls diese Maske nicht mehrfach (in verschiedenen Working-Directories) angelegt werden soll, empfiehlt es sich, WDIR.MAS als P=KS zu definieren (siehe DEFP-Kommando).

## Dateitransfer zwischen Working Directories

Zum Dateitransfer zwischen Working Directories wird die erweiterte Dateiadresse verwendet.

Die vollständige Dateiadresse lautet:

/wdir/mn:dateiname.dateityp

mit	wdir	Name des Working Directory, max. 15 Stellen
	mn	Mediennummer
	dateiname	Name der Datei, max. 8 Stellen
	dateityp	Typ der Datei, max. 3 Stellen

Für wdir, dateiname und dateityp sind alle druckbaren Zeichen zugelassen, das erste Zeichen sollte A..Z oder a..z sein.

Transfer zwischen Directories:

**MOVE** /wdir1/mn1:dateiname.typ /wdir2/mn2:<---

wdir1	Quell-Working Directory Vorbesetzung ist das aktuelle Working Directory
mn1	Quellmediennummer Vorbesetzung ist 0 (siehe MOVE-Kommando)
dateiname.typ	eindeutiger Quell-Dateiname oder -Dateigruppenname ("*" für "alle")
wdir2	Ziel-Working Directory Vorbesetzung ist das aktuelle Working Directory
mn2	Zielmediennummer Vorbesetzung ist 1

**COPY** /wdir1/mn1:dateiname.typ1 /wdir2/mn2:dateiname2.typ2<---

wdir1	Quell-Working Directory Vorbesetzung ist das aktuelle Working Directory
mn1	Quellmediennummer Vorbesetzung ist 0 (siehe MOVE-Kommando)
dateiname1	eindeutige Bezeichnung der Quelldatei
typ 1	eindeutiger Typ der Quelldatei, Vorbesetzung ist .COM
wdir2	Ziel-Working Directory Vorbesetzung ist das aktuelle Working Directory
mn2	Zielmediennummer Vorbesetzung ist 1
dateiname2	Name der Zieldatei, Vorbesetzung ist dateiname1
typ2	Type der Zieldatei, Vorbesetzung ist typ1

Durch COPY bzw. MOVE werden auf dem Ziel-Medium neue Dateien erzeugt, es findet also auch ein physikalisches Kopieren der Dateien statt, bzw. wenn die Zieldatei im Ziel-wdir bereits existiert, dann wird diese überschrieben, sofern sie nicht schreibgeschützt (Dateikennung P=W) ist.

Bei allen KOS-Kommandos, mit Ausnahme von BASIC/ASM/LINK/CROSS kann die Option /wdir/ verwendet werden.

### Beispiel zur Auswirkung des Working Directory: IL-Kommando

```
IL      /wdir/mn:dateiname.typ    P=parameter<---
```

wdir	Working Directory Voraussetzung ist das aktuelle Working Directory
mn	Mediennummer Voraussetzung ist das Mastermedium
dateiname.typ	Dateiname oder -Dateigruppenname ("*" für "alle")
parameter	Dateikennzeichnungen (S=Secret, W=Write protected, E=Erase protected, K=Public etc, s. DEFP-Kommando).

### Beispiele:

Es wird angenommen, daß alle öffentlichen Dateien (Public Files) die Kennzeichnung "secret" tragen.

```
IL /SLAVE1/*      P=*<---
```

Alle Dateien des Working Directories SLAVE1 und alle öffentlichen Dateien werden aufgelistet.

```
IL *<---
```

Alle nicht als "secret" gekennzeichnete Dateien des aktuellen Working Directories werden aufgelistet.

```
IL * P=K
```

Alle öffentlichen Dateien (Public Files) werden aufgelistet.

## **Betriebssystem Kontron KOS**

### **Technische Beschreibung**

Version: Rev. 4.33/5.44/5.54/6.05

Juli 1984

Diese Beschreibung dokumentiert das Betriebssystem KOS der Kontron PSI80/82 und PSI90/900-Computer in Organisation, Anwendung und den Schnittstellen für Anwendungsprogramme.

Sie gibt sehr detaillierte Hinweise über das Betriebssystem KOS, die für Anwender nützlich sind, die die volle Leistung dieses Betriebssystems aus ihren Anwenderprogrammen heraus nutzen wollen.

Für alle anderen Anwendungsfälle ist das Kontron PSI-Bedienungshandbuch zusammen mit dem Abschnitt 'KOS-Systemkommandos' in dieser Technischen Beschreibung voll ausreichend.





**Betriebssystem K O S****Technische Beschreibung****Inhalt**

1. Einführung
  - 1.1 Die Entwicklung von KOS
  - 1.2 Kompatibilität von KOS-bezogener Software
  - 1.3 Software-Gewährleistung
  - 1.4 Organisation von KOS
2. Organisation und Verwendung des Speichers
  - 2.1 Reservierte Speicheradressen im Bereich von 0 bis FFH
  - 2.2 Betriebssystem- und Anwenderspeicher unter KOS6
    - 2.2.1 Anwenderspeicher in Bank 1
    - 2.2.2 Anwenderspeicher und Halbleiter-Floppy Disk
  - 2.3 Betriebssystem- und Anwenderspeicher unter KOS4/5
  - 2.4 Interrupttabelle
3. E/A-Treiber
  - 3.1 Allgemeines
  - 3.2 Kompatibilität zwischen KOS4/5- und KOS6-Treibern
  - 3.3 Residente E/A-Treiber in KOS6
  - 3.4 Funktionen des Bildschirmtreibers \$MON
4. Konfigurierung von KOS
  - 4.1 Änderung der Standardkonfiguration bei KOS6
  - 4.2 Konfigurierung mit anwenderspezifischen residenten E/A-Treibern bei KOS6
5. KOS4/5/6-Systemfunktionen
  - 5.1 Übersicht und Definitionen
    - 5.1.1 Aufrufkonventionen
    - 5.1.2 Systemaufrufvektor (IX-Vektor)
    - 5.1.3 Parameterübergabe in Registern
    - 5.1.4 Rückmeldecode (Return Code)
  - 5.2 Basis Ein-/Ausgabe Funktionen
  - 5.3 Dateiverwaltungsfunktionen
    - 5.3.1 Der Dateispezifikationsblock
    - 5.3.2 Beschreibung der Dateiverwaltungsfunktionen
    - 5.3.3 Dateiverwaltungsfunktionen in KOBUS-Slaves
  - 5.4 Allgemeine Systemfunktionen und Tasking

## 1. Einführung

KOS ist Kontron's Betriebssystem für die Rechnersysteme der Reihe Kontron PSI.

Für diese Rechner steht daneben Standard CP/M 2.2 zur Verfügung.

KOS ist ein voll in Z80A-Code programmierter Superset zu CP/M 2.2, ausgerichtet auf Benutzerfreundlichkeit, Netzwerkfähigkeit und Transparenz.

### 1.1 die Entwicklung von KOS:

- |          |  |
|----------|--|
| KOS 3.2  | entwickelt 1978/79 für Kontron PSI80-KDT4:<br>Disketten-orientiertes Betriebssystem  |
| KOS 4.33 | entwickelt 1980 für Kontron PSI80-KDT4:<br>Disketten- und Plattenorientiertes Betriebssystem,<br>Nachfolger von KOS 3.2  |
| KOS 5.33 | entwickelt 1980 für Kontron PSI80-KDT5:<br>Disketten- und Plattenorientiertes Betriebssystem,<br>funktionsgleich mit KOS 4.33  |
| KOS 5.54 | entwickelt 1981/82 für Kontron PSI80/82-KDT5:<br>Anschluß an Kontron KOBUS, Netzwerk-orientiertes<br>Betriebssystem mit Working Directories und Login,<br>aufwärtskompatibel zu KOS 5.33 |
| KOS 5.44 | entwickelt 1982 aus KOS 5.54 für Kontron PSI80/82-KDT5:<br>funktionsgleich zu KOS 5.54, jedoch ohne Login, für<br>Einzelplatz-Diskettensysteme, aufwärtskompatibel zu KOS<br>5.33        |
| KOS 6.0  | entwickelt 1982 aus KOS 5.54 für Kontron PSI9xx-KDT6<br>bzw. Z80/TCB:<br>Netzwerk-orientiert, aufwärtskompatibel zu KOS 5.54   |

Wie aus dieser Übersicht hervorgeht, gibt die erste Stelle der Versionsnummer den Hinweis auf die zugrunde liegende Hardware, die zweite Stelle charakterisiert den Funktionsumfang, die dritte Stelle zeigt den Änderungsstand bei gleichbleibendem Funktionsumfang.

## 1.2 Kompatibilität von KOS-bezogener Software

Die KOS-Betriebssysteme sind untereinander aufwärtskompatibel. Folgende Unterschiede sind jedoch zu beachten:

- KOS 3.2 gegenüber KOS 4/5/6:  
logisches Dateiformat geändert. Unter KOS 4/5 liest ein Umsetztreiber \$MICP (\$D32) KOS3-Dateien. Neu-Übersetzung von Programmtexten empfohlen. Treiber für Peripherie sind zu ändern.
- KOS 3/4 gegenüber KOS 5/6:  
Hardwareänderung beachten. Treiber für Peripherie anpassen. Bei direkten Hardwarezugriffen Programänderungen.
- KOS 5.33 gegenüber KOS 5.4x/5.5x:  
Erweiterung um 'Working Directories'. Transfer von KOS 5.33-Dateien nach KOS 5.4x/5.5x unter KOS 5.4x/5.5x mit wdir="super user" durch MOVE-Kommando. Treiber überprüfen. Speicherbedarf von KOS 5.4x/5.5x größer als bei 5.33 Diskettentreiber modifiziert.
- KOS 4/5 gegenüber KOS 6:  
physikalisch anderes Diskettenformat. Übertragung von Dateien über Serialschnittstelle durch \$PSIA/B oder über Kontron KOBUS. Treiber überprüfen. KOS6 bietet mehr Speicherplatz für Anwenderprogramme. Die Änderungen der Hardware betreffen vorwiegend drei Bereiche:
- Grafik und Bildwiederholtspeicher: Programme, die direkte Hardwarezugriffe in dem Bildwiederholtspeicher ausführen, müssen auf das neue einfachere Zugriffsverfahren umgestellt werden. Programme, welche die standardmäßigen Grafikroutinen der Utility-Diskette (GRAPH A, GRAPH B, GRAPH V) enthalten, müssen neu gelinkt werden. Programme, die Ausgaben auf den Grafiktreiber \$GRAP durchführen bedürfen keiner Änderung, sofern das Eröffnen korrekt durchgeführt wird, in Kontron BASIC z. B. durch "100 OPEN #9:"\$GRAP".
  - Statusport: Unter KOS6 umfassen der Statusport und seine Abbildung im Speicher 2 Bytes. Das Grafik-Kennungsbit ist geändert.
  - Zusätzliche Ein-/Ausgabeadressen sind unter KOS6 belegt, entsprechend den zusätzlichen Möglichkeiten der Hardware.

Die Systemkommandos zeigen unter dem PSTAT-Kommando, auf welche Betriebssysteme sie abgestimmt sind. Kontron ist bemüht, neuere Versionen der Systemkommandos auch unter älteren Betriebssystemversionen ablauffähig zu halten. Umgekehrt gilt dies nicht: ältere Versionen der Systemkommandos dürfen wegen des erweiterten Funktionsumfangs nicht unter neueren KOS-Ständen ausgeführt werden.

### 1.3 Software-Gewährleistung

Im Rahmen der Software-Gewährleistung werden derzeit von Kontron unterstützt:

KOS 4.33  
KOS 5.54/5.44  
KOS 6.05

mit den jeweiligen Systemdienstprogrammen und Utilities.

KOS ist ein seit Jahren erprobtes und kontinuierlich weiterentwickeltes Betriebssystem. Trotzdem kann Kontron nicht ausschließen, daß sich Fehlfunktionen, Unterschiede in der Funktion gegenüber der Dokumentation oder inkonsistente Systemreaktionen auf Befehls- und Systemaufrufsequenzen ergeben.

Kontron ist bemüht, zu schriftlich eingehenden, nachvollziehbaren Problembeschreibungen über Produkte, die unter die Software-Gewährleistung fallen, mit der kürzest möglichen Reaktionszeit Stellung zu nehmen und Hinweise zur Problembeseitigung oder zur Vermeidung der Problemstellung zu geben, und im Rahmen des technisch Möglichen Korrekturen oder Änderungen an den freigegebenen Softwareversionen vorzunehmen.

Kontron behält sich vor, Änderungen durch neue Freigabestände der Systemsoftware zugänglich zu machen, die auch Funktionserweiterungen beinhalten können. Davon abhängige Parameter, wie Speicherbedarf oder das Zeitverhalten etc. können dadurch Änderungen unterworfen sein.

## 1.4 Organisation von KOS

Die Kontron PSI-Systeme zum Betrieb unter KOS sind ausgestattet mit einem PROM-residenten Urlader ("BOOT1")

Dieser Urlader wird durch Eingabe von 'K<---' (KOS4) oder automatisch (KOS5/6) aktiv und sucht eine Datei BOOT2.SYS zyklisch auf allen erreichbaren Medien, beginnend mit der integrierten Festplatte, anschließend Floppy Disk-Laufwerk 0 (je nach Einbauart rechts bzw. oben) und Floppy Disk-Laufwerk 1.

Dieser Dateieintrag muß den KOS-Konventionen entsprechen und er muß innerhalb der ersten 32 Einträge eines der zugänglichen Medien stehen.

BOOT2.SYS wird geladen, gestartet und sucht nun seinerseits eine Datei KOS.SYS (KOS5) bzw. KOSO.SYS (KOS6).

Diese Datei muß wiederum in den ersten 32 Einträgen eines zugänglichen KOS-Mediums stehen. KOS bzw. KOSO wird geladen und gestartet. Die Hardware wird entsprechend initialisiert, die KOS-Systemmeldung mit der Versionsnummer erscheint am Bildschirm. Bei KOS 5.5x und KOS 6.0x wird das "Login" abgefragt, überprüft und das zugehörige "wdir" eingestellt. Danach wird der dafür benötigte Speicherbereich freigegeben und als erstes Kommando automatisch 'DO KOS.INI' ausgeführt.

Bei KOBUS-Slave-Stationen ohne eigenen Massenspeicher entfällt das Laden von BOOT2.SYS, statt dessen wird die Datei KOBUS.SYS (KOS 5.5x) bzw. KOSO.SYS (KOS 6.0x) geladen und ausgeführt, wodurch nach der entsprechenden Initialisierung der Hardware wieder das "Login" erscheint. Als erstes Kommando wird in KOBUS-Slaves 'DO SLAVE.INI' ausgeführt.

### KOS5:

KOS 5.xx wird standardmäßig in folgenden Versionen geliefert:  
KOS 5.4x (die Version ohne Login für Einzelplatz-Diskettensysteme)

Dateiname	Klassifizierung nach Medien
F0001y.D4x	2 FD-Laufwerke DD/SS (308 KB)
F0011y.D4x	2 FD-Laufwerke DD/DS (616 KB)
B0000y.D4x	Bubble-Speicher (min 128 KB)

KOS 5.5x (die Version mit Login für KOBUS-Mehrplatz- und Festplattensysteme)

F0001y.D5x	2 FD-Laufwerke DD/SS (308 KB)
F0011y.D5x	2 FD-Laufwerke DD/DS (616 KB)
K0000y.D5x	KOBUS-Slave
W0000y.D5x	integrierte Festplatte 5 MioB

Dabei bedeutet:

- F = Floppy Disk - System
- K = KOBUS-Slave-System
- W = Winchesterplatten-System
- B = Bubble-Medium
- D = deutsche Version
- x = 4 (KOS 5.44) oder 5 (KOS 5.54)
- y = S(hort)/L(ong)-Version

Das Systemlademedium (Diskette, Platte oder Bubble-Speicher) muß also Dateien mit den Namen BOOT2.SYS und KOS.SYS enthalten (für KOBUS-Slaves zusätzlich KOBUS.SYS). Die Datei KOS.SYS muß dabei der vorliegenden Hardware-Konfiguration entsprechen, da sonst das Lademedium nach dem Laden von KOS nicht mehr angesprochen werden kann. Beispiel: Nach Laden von B0000S.D44 unter dem Namen KOS.SYS ist dem Betriebssystem ausschließlich das Bubble-Medium bekannt. Von dort her können dann aber zusätzliche Medientreiber geladen und aktiviert werden, wenn das IODC-Kommando und der Medientreiber, z.B. MIDD.OBJ, auf dem Bubble-Medium verfügbar sind.

Diese richtige Betriebssystemversion wird durch das COPY-Kommando auf das Lademedium gebracht, z.B. COPY B0000S.D44 2:KOS.SYS. Dabei sei das Medium 2 dem Treiber BM.OBJ zugewiesen.

#### KOS6:

Wesentlicher Unterschied zwischen KOS4/5 und KOS6 ist, daß KOS4/5 in einer Datei enthalten ist, während KOS6 in 8 bis 16 "Pages" mit jeweils maximal 4 KByte residenten Codes aufgeteilt ist. Diese "Pages" sind in den Dateien KOS0.SYS bis KOS8.SYS bzw. (maximal) KOSF.SYS enthalten. KOS0.SYS bestimmt dabei die Initialisierung und erzeugt die dem Lademedium entsprechende Belegung der Medienkanäle. KOS5.SYS bestimmt die Organisation des Festplatten-Massenspeichers. KOS8.SYS ist notwendig für KOBUS-Systeme.

Für den Benutzer von KOS6 ergeben sich folgende in Z80-Systemen wichtige Vorteile:

- bis zu 58 KByte direkt erreichbarer Speicher für Anwenderprogramme
- integrierte Halbleiter Floppy (virtuelles Medium) mit 128 KByte
- integrierte E/A-Treiber
- integrierbare Anwender-E/A-Treiber
- keine Restriktionen für die Lage von Interrupt Service Routinen
- zusätzliche Speichererweiterung durch Verwendung einer Hardware MMU (Memory Management Unit)

Im folgenden werden Organisation, Speicherbelegung und Einsprungspunkte von KOS dargestellt. Die ebenfalls in KOS enthaltenen residenten Kommandos A,C,D,H,I,M,N,O,P,R,S,X,Y sind der Übersichtlichkeit halber im Kapitel "Systemkommandos" zusammengefaßt beschrieben.

## 2. Organisation und Verwendung des Speichers

### 2.1 Reservierte Speicheradressen

Reservierte Speicheradressen befinden sich im Adreßbereich zwischen 0 und FFH.

Diese Übersicht gilt für alle KOS Versionen.

Adresse (hex)	Bedeutung
00 - 02	KOS Warmstart Einsprungpunkt; wird von KDM bzw. \$CPM modifiziert; ermöglicht in jedem Fall den Rücksprung zu KOS bzw. KDM; Stack wird automatisch korrigiert.
03	Statusbyte (für KDT Rev. 5, 6 und TCB/Z80); spiegelt den Inhalt des Statusports (Adresse 1CH) wieder, ist immer FF in KOS4.xx (KDT Rev. 4)
04	reserviert für \$CPM
05	Einsprungpunkt für CP/M-Systemaufrufe (CALL 5); enthält einen RST 10H Befehl
06 - 07	MEMTOP; Adresse der ersten nicht mehr überschreibbaren Speicherstelle für Programme, welche die KOS-Speicherverwaltung nicht verwenden (z.B.: CP/M-Programme)
08 - 0A	Einsprungpunkt für KOS-Systemaufrufe (RST 8-Befehl)
0B - 0D	Einsprungpunkt für den Ausdruck von KOS-Fehlermeldungen; führt anschließend einen KOS-Warmstart aus.
0E - 0F	enthält das aktuelle Systemdatum (hexadezimal) OE: Bit 0...4 Tag Bit 5...7 Jahrzehnt OF: Bit 0...3 Monat Bit 4...7 Jahr
10 - 12	Einsprungpunkt nach \$CPM; wird über Adresse 5 (CP/M-Systemaufruf) angesprungen.
13	Statusflag für alle Floppy Disk Treiber (\$DSKO/\$DSK1/\$MIDS etc.)
14 - 15	Pufferpointer für \$MON/\$GRAP/\$KEY; wird verwendet bei: \$MON/\$GRAP in KOS 4.xx \$GRAP in KOS 5.xx \$KEY ab KOS 6.03
16 - 17	reserviert für KONTRON PASCAL; frei verfügbar außerhalb PASCAL.

18 - 1A	RST 18H Einsprungpunkt; frei verfügbar
<hr/>	
1B	reserviert für \$GRAP
1C - 1E	Uhrzeit, falls die Task 'Clock' aktiviert ist (RLOAD TIME); ansonsten nicht verwendet. 1C/1D/1E ---> Sek./Min./Std.
1F	Intervallzähler des Task Schedulers; wird inkrementiert im 20 ms Takt des Task-Schedulers
<hr/>	
20 - 22	frei für RST 20H Einsprungpunkt (nicht bei KOBUS-Master)
23 - 27	reserviert für KOS
28 - 2A	frei für RST 28H Einsprungpunkt in KOS4.xx/5.xx reserviert für KOS in KOS6.xx
2B - 2C	reserviert für TCNT/PCNT (KOS6)
2D	reserviert für KOBUS; falls Bit n rückgesetzt, wird Medium n nicht reinitialisiert (N-Kommando)
2E	KOBUS-Flag; enthält 0 - in KOBUS Systemen FF - in allen anderen Systemen
2F	reserviert für \$SLV0/\$SLAV (während Boot-Phase)
<hr/>	
30 - 32	frei für RST 30H Einsprungpunkt in KOS4.xx/5.xx; nicht verfügbar in KOS6.xx; wird dort als Einsprungpunkt zu User Programmen verwendet.
33 - 36	reserviert für KOBUS 33/34: Verzweigungsadresse 35: Flagregister 36: Enable Flag für 'active' Task



37	Break Enable Flag für CTRL-K Taste 0 - CTRL-K ist unwirksam 1 - CTRL-K bewirkt BREAK (Sprung auf 38H)
38 - 3A	Breakpoint Einsprungpunkt für KDM (RST 38H = Code FF)
3B - 3C	reserviert für KOS
3D	reserviert für \$WINx
3E	frei in KOS4.xx/5.xx; DMA busy Flag in KOS6 0 - DMA wird derzeit nicht verwendet >0 - DMA derzeit aktiv
3F	frei in KOS4.xx/5.xx Motor On/Off Flag für Floppy Disk Treiber in KOS6 0 - Motor Off Task ist 'disabled' 1 - Motor Off Task ist 'enabled' (Motor schaltet nach etwa 10 sec. ab)
<hr/>	
40 - 4F	nicht vorbelegt - frei für Anwender
<hr/>	
50 - 5F	DSB1 (für KOS Parameteraufbereitung)
60 - 6F	DSB2 (für KOS Parameteraufbereitung)
70 - 7F	DSB3 (für KOS Parameteraufbereitung)
<hr/>	
80 - FF	Default Sector Buffer
<hr/>	

Hinweis: Benutzerprogramme dürfen, falls erforderlich, nur den Bereich zwischen 40 und 4F verwenden, um Konflikte mit eventuellen zukünftigen KOS-Versionen auszuschließen.

Von besonderer Bedeutung für den Systemprogrammierer ist das Byte 03, das Abbild des Statusports (KDT5, Kontron PSI80/82) bzw. des Statusports 0 (KDT6 bzw. TCB/Z80 und TCB/IOV, Kontron PSI900/9000), in den unter der I/O-Adresse 1ch Statusumschaltungen eingeschrieben werden. Der Statusport 0 kann nicht ausgelesen werden. Es wird deswegen von der Betriebssoftware und von anwenderseitig erstellter Systemsoftware bei Veränderung des Statusport das Statusbyte mit der Adresse 03 mit verändert. Das Auslegen der Adresse 03 und das Schreiben des geänderten Status in die Speicheradresse 03 und die I/O-Adresse 1ch erfolgt insgesamt unter Interruptsperre.

Für das Statusbyte und den zugehörigen Statusport gelten folgende Zuordnungen:

	KDT5	KDT6 oder TCB/Z80 bzw. TCB/IOV
ST0	MAPO off/on	Watchdog off/on (optionale Hardware)
ST1	MAP1 off/on	Taktfrequenz 0.5/1.0 x Phi
ST2	Sound Trigger off/on	Audiokanal off/on
ST3	Video Invert off/on	Zeichensatz 0/1 (nur KDT6)
ST4	Alpha/Grafik	DMA-Quelle FDC.DRQ/SIOA.RDY
ST5	Prom on/off	Prom on/off
ST6	Standard/Mini-FD	Standard/Mini-FD
ST7	FD-Motor off/on	FD-Motor off/on

Weitergehende Erläuterungen der Signalbedeutung sind im Kapitel 3.2 der jeweiligen Hardware-Beschreibung enthalten.

Wie aus der Tabelle ersichtlich ist, betreffen Änderungen zwischen KDT5 einerseits und KDT6 bzw. TCB/Z80 und TCB/IOV andererseits dies Mapping des Speichers und die Video-Ausgabe. Hinzu kommen Hardware-Erweiterungen.

Anwendungsprogramme mit Zugriff auf den Statusport sind in diesen Punkten zu überprüfen.

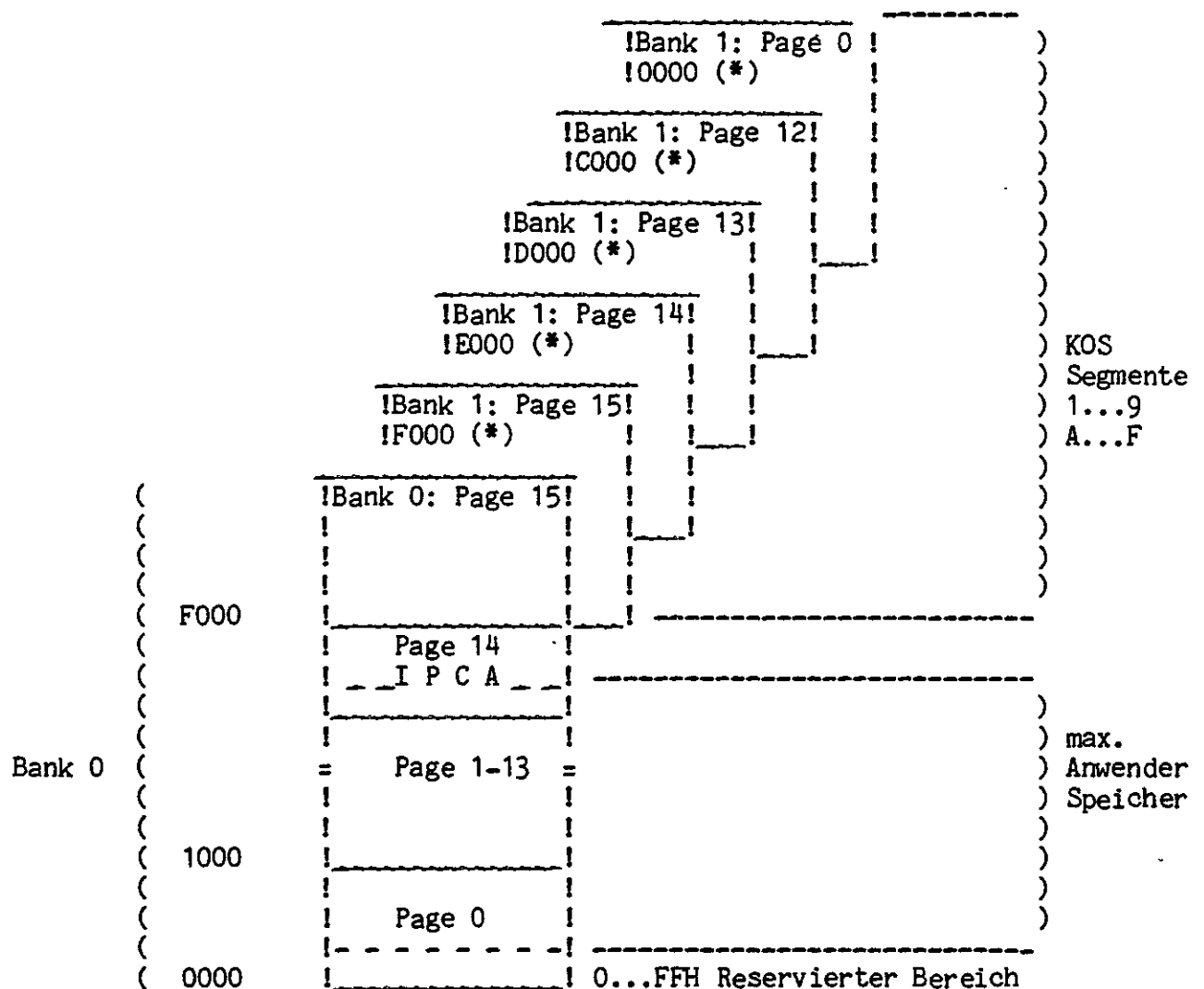
## 2.2 Organisation und Verwendung des Speichers unter KOS6

KOS verwendet und verwaltet 256 kByte Schreib-/Lesespeicher, der bereits standardmäßig auf der zugrunde liegenden Hardware der Computersysteme Kontron PSI900/9000 enthalten ist.

Der Speicher besteht aus vier 64 kByte Bänken zu je sechzehn 4 kByte Pages. Die Hardware MMU (Memory Management Unit) bietet die Möglichkeit, jede dieser Pages mit einer beliebigen logischen Adresse innerhalb des 64 kByte großen CPU-Adressraums anzusprechen. Von dieser Möglichkeit macht KOS Gebrauch. Es ist in mehrere 4 kByte große Segmente aufgeteilt, die jeweils ab der logischen Adresse F000 (hex) in verschiedenen Pages der Bank 1 resident sind.

### 2.2.1 Betriebssystem- und Anwenderspeicher unter KOS6

Das Betriebssystem residiert zum größten Teil in Bank 1. Dem Anwender steht der überwiegende Teil von Bank 0 zur Verfügung. Etwa 6 kByte sind in dieser Bank für KOS reserviert. Das folgende Bild zeigt die schematische Darstellung dieser Speicherorganisation.



Schematische Darstellung der Speicherorganisation von KOS.0

In Bank 0, Page 14 befindet sich zwischen E800H und EFFFH die "Inter Page Communication Area" (abgekürzt: IPCA). Dieser Bereich enthält alle Programmteile und Tabellen, die immer resident sein müssen. Dazu zählen beispielsweise:

- der Stackbereich von KOS6
- die Interrupt Tabelle von KOS6
- alle Interrupt Service Routinen von KOS6
- alle Systemeinsprungpunkte, sowie
- alle 'Globalen Speicherbereiche', die von mehreren Pages aus zugänglich sein müssen.

Je nach Bedarf tauscht KOS6 Page 15 durch entsprechende Programmierung der Hardware MMU aus (siehe Hardware-Beschreibung). Diese Art der Speicherorganisation ermöglicht eine weitgehende Kompatibilität zu früheren KOS-Versionen. Geringfügige Unterschiede ergeben sich lediglich bezüglich E/A-Treibern (siehe Abschnitt 3).

Von der Sicht des Anwenderprogramms aus betrachtet umfaßt der Speicher 64 kByte, von dem 6 kByte von KOS benötigt werden. Im Gegensatz zu früheren KOS-Versionen gibt es in KOS keine Einschränkungen bezüglich der Lage von Benutzer-definierten Interrupt Service Routinen mit Ausnahme der Forderung, daß sie immer im Anwenderspeicher der Bank 0 resident sein müssen.

Bank 2 und Bank 3 sind von KOS als "virtuelles Medium", als schnelle RAM-Floppy Disk mit 128 kByte Kapazität organisiert. Diese Bänke können alternativ auch durch Mapping vom Anwender anderweitig verwendet werden.

Der Bildwiederholtspeicher umfaßt zusätzlich 64 kByte. Innerhalb des Betriebssystems wird er vom Bildschirmtreiber \$MON und den Grafik-routinen verwendet.

Lagen der KOSx.SYS-Segmente:

Bank 0, Page 15: KOS0.SYS  
Bank 1, Page 0 : KOS1.SYS  
Bank 1, Page 1 : KOS2.SYS  
...  
Bank 1, Page 7 : KOS8.SYS  
...

### 2.2.2 Anwenderspeicher in Bank 1 unter KOS6

Abhängig von der jeweiligen Konfiguration des Betriebssystems steht dem Anwender in Bank 1 mehr oder weniger Speicher zur Verfügung. Die Standardversion von KOS beansprucht minimal 7 Segmente in Bank 1; es bleiben somit maximal 32 kByte für den Anwender. Hierbei handelt es sich um die Pages 0 bis 6 (physikalischer Adreßbereich 0000...7FFF).

Wie in Abschnitt 4 gezeigt wird, kann KOS durch den Anwender konfiguriert und erweitert werden. Maximal können durch den Urlader 16 Segmente je 4 kByte geladen werden. Entsprechend der Anzahl der KOS-Segmente verringert sich die Anzahl der frei verwendbaren Pages in Bank 1.

Hinweis: Es ist Sache des Anwenderprogramms, die MMU bei Zugriffen auf Bank 1 korrekt zu programmieren. Beispiele hierfür befinden sich in der Hardwarebeschreibung. Auf keinen Fall dürfen die Pages 0, 14 und 15 von Bank 0 verlagert werden.

### 2.2.3 Anwenderspeicher und Halbleiterfloppy unter KOS6

KOS6 enthält standardmäßig den Medientreiber \$VMED, der die Bänke 2 und 3 des Speichers als 'Halbleiterfloppy' verwendet. Die Kapazität dieses 'virtuellen' Mediums beträgt 128 kByte.

Selbstverständlich können die Bänke 2 und 3 auch von Benutzerprogrammen verwendet werden. Voraussetzung ist, daß KOS4.SYS ohne \$VMED konfiguriert wurde.

Auch hierbei muß die MMU korrekt programmiert werden.

### 2.3 Betriebssystem- und Anwenderspeicher unter KOS 4/5

Die Betriebssysteme KOS 4/5 benötigen etwa 12 kByte und residieren grundsätzlich im obersten zur Verfügung stehenden Speicherbereich. Beim Kaltstart von KOS wird folgende Speicherorganisation aufgebaut:

- Der PROM-Bereich der Kontron PSI80/82-Zentralplatine wird abgeschaltet, so daß nur Schreib-/Lesespeicher vorhanden ist.
- Der Adreßbereich 0 - FFH (256 Byte) wird für Systemparameter reserviert und enthält auf der Adresse 8 den Systemeinsprungpunkt KOSCAL; dort erfolgt die Verzweigung zu den einzelnen Funktionen des Betriebssystems aufgrund einer Funktionsnummer 'n'.
- Für das Betriebssystem wird der Bereich von xx00H bis FFFFH reserviert.

#### KOS 4/5-Speicherorganisation

!-----! FFFFH	
! K O S 4/5 !	
!-----! xx00H+700H	
! Systemvariable ! (Systemabhängig)	
! und !	
! KOS-Stack !	
!-----! xx00H+100H	
! Interrupt Tabelle !	
!-----! xx00H (xx=I-Register)	
! A n w e n d e r - !	
! s p e i c h e r - !	
! b e r e i c h !	
!-----! 0100H	
! KOS-Einsprungpunkte !	
!-----! 0000H	

Es wird empfohlen, den Wert xx grundsätzlich aus dem I-Register der CPU zu entnehmen. Dies garantiert Kompatibilität bei Änderungen der Lage der Interrupt-Tabelle.

## 2.4 Interrupttabelle

Die Interrupttabelle befindet sich im Standardsystem ab der Adresse EA00H (KOS6) (100 Hex Bytes entsprechend 128 Service-Routinen). Das I-Register der CPU unter KOS6 und KOS 4/5 ist als Referenz bei Zugriffen auf die Interrupttabelle zu verwenden. Die folgende Assembler-Routine ist dafür geeignet:

```
get.inttab.addr: ld a,i
                  ld h,a
                  ld l,0
                  ret
```

Alle nicht benötigten Einträge sind mit einem Zeiger auf eine 'Dummy ISR' vorbelegt.

```
Dummy.ISR:      ei
                  reti
```

Folgende Einträge sind für folgende I/O-Bausteine reserviert:

xx78...xx7F	DMA auf ECB/SASI	(**)
xx80...xx8F	Z8038-FIO on ECB/C16	(**)
xx90...xx93	PIO (KDT6 or TCB/Z80)	
xx94...xx9B	reserved	
xx9C...xx9F	PIO (TCB/IOV)	
xxA0...xxAF	DART1	(TCB/IOV) (*)
xxB0...xxBF	CTC1/CTC2	(TCB/IOV) (*)
xxC0...xxCF	SIOA/SIOB	(KDT6 oder TCB/Z80 oder KDT5)
xxD0...xxDF	DART2	(TCB/IOV) (*)
xxE0...xxEF	CTC1/CTC2	(KDT6 oder TCB/Z80 oder KDT5)
xxF0...xxF7	reserviert	für KOBUS-Master ECB/KPP (**)
xxF8...xxFF	DMA	(KDT5 oder TCB/Z80 oder KDT6)

Die Einträge von xx00...xx9F sind frei verwendbar. Der Wert xx ist aus dem I-Register der CPU ablesbar (OEAH im Standardsystem).

- (\*) Diese Baugruppe ist nur in den Systemen Kontron PSI980/9800 enthalten. Die zugehörigen Interrupt-Einträge sind in allen Systemen für KOS reserviert.
- (\*\*) Diese Baugruppen gehören zu speziellen Systemerweiterungen, die Interrupteinträge gelten jedoch immer als reserviert.

**Hinweis:** Bei nicht-DMA-gesteuertem Floppy-Disk-Datentransfer, nach Lesen oder Schreiben eines Records, darf die di/ei-Zeit maximal 13 usec. betragen (z.B. bei Interrupt Service Routine).

### 3. E/A-Treiber (I/O-Treiber)

#### 3.1 Allgemeines

KOS unterscheidet formal zwischen zwei Treibergruppen:

- a) der Gruppe von 'residenten' Treibern
- b) der Gruppe von 'aktivierbaren' Treibern

wobei in Punkto Programmcode bei KOS6 kein Unterschied zwischen beiden Gruppen besteht.

'Residente' Treiber sind bereits in das Betriebssystem integriert. Sie laufen bei KOS6 in Speicherbank 1 ab und benötigen deshalb keinen Anwenderspeicher.

'Aktivierbare' Treiber liegen bei KOS5 und KOS6 als relocatives Objektmodul auf Diskette etc. vor und werden bei Bedarf durch das Dienstprogramm IODC (I/O-Driver Control) aktiviert. In diesem Fall wird Anwenderspeicher benötigt; der Treiber läuft auch bei KOS6 in Speicherbank 0 ab.

KOS6 bietet als Option KOS-IF die Möglichkeit, beliebige Treiber aus der Gruppe der 'aktivierbaren' Treiber in das Betriebssystem zu integrieren, also 'resident' zu machen. Ein Hinweis hierzu ist in Abschnitt 4 gegeben.

#### 3.2 Kompatibilität zwischen KOS4/5- und KOS6-Treibern

E/A-Treiber, die für KOS5 geschrieben wurden, sind unverändert unter KOS ablauffähig, vorausgesetzt, daß untenstehende Funktionen implementiert sind. Diese Funktionen waren bereits unter KOS5 definiert, dort allerdings nicht in jedem Fall notwendig.

Um der Tatsache Rechnung zu tragen, daß E/A-Treiber bei KOS6 in verschiedenen Pages resident sein können, und deshalb für Anwender- und Systemprogramme nicht ohne weiteres 'sichtbar' sind, ergeben sich folgende Konsequenzen für E/A-Treiber:

- a) die bei KOS5 notwendige Anfangs-Sprungtabelle kann entfallen. Ein E/A-Treiber hat somit nur einen einzigen Einsprungpunkt, der über eine Reihe von 'System Calls' erreichbar ist. Die Verzweigung zu den einzelnen Funktionen eines Treibers (STATUS, INIT, OPEN, CLOSE) erfolgt auf Grund des 'Request Codes' unter (IX+1).



b) Folgende Funktionen müssen in E/A-Treibern implementiert sein:

Eingabe Treiber:

-----

Funktion 82 - Input Driver Status  
Funktion 84 - Character Input  
Funktion 8C - Init/Open/Close Input Driver

Ausgabe Treiber:

-----

Funktion 81 - Output Driver Status  
Funktion 86 - Character Output  
Funktion 8D - Init/Open/Close Output Driver

Medien Treiber:

-----

Funktion 80 - Return Media Identification  
Funktion 8A - Logical Record Read  
Funktion 8B - Logical Record Write  
Funktion 9C - Init/Open/Close Read  
Funktion 9D - Init/Open/Close Write  
Funktion A0 - Logical Sector Read  
Funktion A1 - Logical Sector Write

Zu beachten ist, daß die aufgeführten Funktionen bei KOS6 nicht mehr über eine Sprungtabelle erreichbar sind. Alle für KOS4/KOS5 geschriebenen Programme mit direkten Treibereinsprünge müssen entsprechend abgeändert werden.

Beispiele zur Treiberprogrammierung sind im Abschnitt "Utility" gegeben.

### 3.3 Residente E/A-Treiber in KOS

Residente E/A-Treiber sind Teil des Betriebssystems und sind nach dem BOOT (Kaltstart) von KOS bereits aktiv. Sie dürfen durch das IODC-Kommando nicht deaktiviert werden.

Bei KOS6 befinden sich alle residenten E/A-Treiber in den verschiedenen KOS-Segmenten und benötigen deshalb keinen Anwenderspeicher. Durch die Option KOS-IF können Anzahl und Zuordnung der residenten Treiber vom Anwender festgelegt werden. Ein Hinweis dazu wird in Kapitel 4 "Konfigurierung" gegeben.

Die Standardversion von KOS hat folgende E/A-Treiber:

Kanal:	Name:	Funktion:	resident bei:
I-0	\$KEY	Keyboard Eingabe	KOS4/5/6
I-1	\$KEY	Keyboard Eingabe	KOS4/5/6
O-0	\$MON	Monitor Ausgabe	KOS4/5/6
O-1	\$MON	Monitor Ausgabe	KOS4/5/6
O-2	\$MON	Monitor Ausgabe	KOS4/5/6
O-3	\$KSM	System-/Fehlermeldungen Ausgabe	KOS4/5/6 (KOS6:KSML)
O-9	\$GRAP	Graphik Ausgabe	KOS6
	\$DSKO (*)	Floppy Disk 0	KOS6; KOS4/5 bei FD-Boot
	\$DSK1 (*)	Floppy Disk 1	" " " "
	\$WINO	Mini Winchester (Festplatte)	KOS6; KOS4/5 bei HD-Boot
	\$WIN1	Mini Winchester (Wechselplatte)	KOS6
	\$VMED	Halbleiter Floppy	KOS6
	\$SLxx	KOBUS Slave Treiber	KOS5/KOS6 bei KOBUS-Boot
		xx=AV/VO	

Diese Standardversion von KOS ist auf allen Systemen mit entsprechender Hardware und der jeweiligen Massenspeicherkonfiguration ablauffähig.

Die Medienkanalordnung ist vom jeweiligen 'Boot-Device' abhängig:

	Floppy-Boot	Hard Disk-Boot	KOBUS-Boot
M-0	\$DSKO	\$WINO	\$SLxx x
M-1	\$DSK1	\$WIN1 *	\$VMED *
M-2	\$WINO *	\$DSKO *	-
M-3	\$WIN1 *	\$DSK1 *	-
M-4	\$VMED *	\$VMED *	-
M-5	\$SLxx *x	\$SLxx *x	-

(\*) nur bei KOS6

(x) \$SLxx = \$SLV0..\$SLV3 für Kontron PSI9xx-Systeme  
           = \$SLV0 für Kontron PSI9C-Systeme bei KOBUS-Boot  
           = \$SLAV für Kontron PSI80/TC-Systeme bei KOBUS-Boot

(\*) Auf Kontron PSI9xx Systemdisketten sind außer dem Betriebssystemmodul KOS4.SYS, welches die Diskettentreiber \$DSKO, \$DSK1 enthält, noch die Module KOS4.FST und KOS4.SLW enthalten. KOS4.FST beinhaltet Diskettentreiber für Standardlaufwerke, während KOS4.SLW langsame Diskettentreiber für staubgeschützte FD-Laufwerke für Kontron PSI980 R-Systeme beinhaltet. Standardmäßig ist auf Systemdisketten für Kontron PSI9xx das Modul KOS4.FST als KOS4.SYS vorhanden, während bei Kontron PSI980R das Modul KOS4.SLW als KOS4.SYS ausgeliefert wird.

Bei KOS5 gibt es unterschiedliche KOS-Dateien, die sich jeweils in den integrierten Massenspeichertreibern unterscheiden (siehe 1.4). Die Namensgebung zeigt dies:

F.... für 2 x Floppy Disk-Systeme  
W.... für Festplattensysteme  
K.... für KOBUS-Slaves (Kontron PSI80/TC)  
B.... für Bubble Memory Systeme

Bei KOS6 unterscheidet sich KOS5.SYS je nach dem integrierten Plattenspeicher, KOSA.SYS wird entsprechend der eingesetzten externen Plattensysteme konfiguriert.

### 3.4 Funktionen des Bildschirmtreibers \$MON

\$MON verwaltet den Bildwiederholtspeicher des Computers. Neben der Darstellung von alphanumerischen und semigraphischen Zeichen verarbeitet \$MON eine Reihe von Steuerzeichen und ESC-Sequenzen. Letztere werden von KOS6 direkt ausgewertet, unter KOS5 ist der Converter XMON zu laden.

#### Steuerzeichen

Als Steuerzeichen werden alle ASCII-Zeichen im Wertebereich zwischen 0 und 1FH bezeichnet. \$MON reagiert auf alle Steuerzeichen der folgenden Tabelle. Alle übrigen werden ignoriert.

Folgende Anmerkungen sind zu beachten:

- Eine Reihe von Steuerzeichen wirkt bei jedem Senden an den Bildschirm umschaltend, so z.B. CTRL-O, CTRL-S etc. Diese sind in der folgenden Tabelle durch "\*" gekennzeichnet.
- Die Grundeinstellung des Bildschirms ist "helle Zeichen auf dunklem Grund". CTRL-R schaltet den Inversbetrieb (dunkle Zeichen auf hellem Grund) ein oder aus.
- Die zeichenweise Invertierung wird durch CTRL-Q eingeleitet: alle folgenden sichtbaren Zeichen (auch die durch Leerstellen aufgefüllten Tabulatorstellen) werden relativ zum eingestellten Hintergrund (CTRL-R) invertiert.
- Die Ausgabe von Steuerzeichen (z.B. CTRL-G) und von alphanumerischen Zeichen ist im Grafikmodus unzulässig.

## Liste aller Steuerzeichen mit Wirkung für \$MON

Code	Taste	Wirkung
01h	CTRL-A	Setzt den Cursor auf den Anfang der (**) letzten Zeile
06h	CTRL-F	Bewegt den Cursor um eine Stelle nach rechts
07h	CTRL-G	Aktiviert die Akustikausgabe
08h	CTRL-H	Bewegt den Cursor um eine Stelle nach links
09h	CTRL-I	Setzt den Cursor auf die nächste Tab Position
0Ah	CTRL-J	Bewegt den Cursor um eine Zeile nach unten (Linefeed)
0Ch	CTRL-L	Setzt den Cursor an den Anfang einer neuen Seite (Formfeed, clear screen)
0Dh	RETURN	Setzt den Cursor an den Anfang der aktuellen Zeile (Carriage Return)
0Fh	CTRL-O	Schaltet Monitorausgabe ein bzw. aus.
11h	CTRL-Q	Invertiert die nachfolgenden Zeichen (*)
12h	CTRL-R	Invertiert das gesamte Bild (*)
13h	CTRL-S	Schaltet die Ausgabegeschwindigkeit von normal auf langsam bzw. umgekehrt (*)
14h	CTRL-T	Schaltet den Cursor ein bzw. aus (*)
17h	CTRL-W	Schaltet die Blinkfunktion ein bzw. aus (*) (nur bei KOS6)
1Ah	CTRL-Z	Bewegt Cursor um eine Zeile nach oben
1Bh	ESC	Startet eine ESC-Sequenz (bei KOS5 nur über XMON)
1Ch	HOME	Setzt den Cursor auf den Anfang der ersten Zeile

(\*) Dies sind 'Flipflop'-Funktionen, die auf den jeweils anderen Zustand umschalten.

(\*\*) Hat Kommandofunktion, wenn im KOS6-Kommandoeingabemodus als erstes Zeichen nach Prompt-Zeichen 'KOS:' eingegeben (Wiederholung).

**ESCape-Sequenzen**

ESC-Sequenzen werden durch die Ausgabe des Codes 1BH bzw. 27 dezimal (ESC-Taste) an \$MON (KOS6) bzw. XMON (KOS5) eingeleitet. Sie haben folgendes Format:

**ESC cmd (par1 par2)**

ESC-Sequenzen dienen zur direkten Cursor Adressierung und zum Editieren des Bildschirminhalts. Folgende Kommandos sind implementiert:

**Liste aller ESC-Sequenzen**

Kommando	Parameter	Wirkung
ESC/=	row/col	Position Cursor
ESC/R	-	Delete Line at Cursor
ESC/T	-	Erase to End of Line at Cursor
ESC/t	-	Erase to End of Line at Cursor
ESC/Y	-	Erase to End of Screen
ESC/y	-	Erase to End of Screen
ESC/E	-	Insert Line at Cursor
ESC/C	-	Insert Character at Cursor
ESC/-	-	Invert Video Mode On
ESC/j	-	Invert Video Mode On
ESC/e	-	Invert Video Mode On
ESC/q	-	Invert Video Mode Off
ESC/k	-	Invert Video Mode Off
ESC/m	-	Invert Video Mode Off
ESC/a	-	Cursor On
ESC/b	-	Cursor Off
ESC/0	-	Schalten auf Videosegment 0
ESC/1	-	Schalten auf Videosegment 1
ESC/2	-	Schalten auf Videosegment 2
ESC/3	-	Schalten auf Videosegment 3

Hinweis: Das Zeichen '/' dient hier nur zur Trennung.

Mit diesen Sequenzen emuliert \$MON die Terminals Lear Siegler ADM-3A und SOROC IQ.

## a) Position Cursor: ESC/= /row/col

Wirkung: Positioniert den Cursor in die Spalte 'col' der Zeile 'row'.

Der Wertebereich für 'row' liegt zwischen 1 und 25, der den ASCII-Codes 20H bis 38H zugeordnet ist, also:

20H (Blank)	--->	Zeile 1
21H (!)	--->	Zeile 2
.		
.		
38H (8)	--->	Zeile 25

Der Wertebereich für 'col' liegt zwischen 1 und 80, der den ASCII-Codes 20H bis 6FH zugeordnet ist, also:

20H (Blank)	--->	Spalte 1
21H (!)	--->	Spalte 2
.		
.		
6FH (o)	--->	Spalte 80

Das 'Position Cursor'-Kommando wird ignoriert, falls einer der Parameter (row oder col) außerhalb des gültigen Wertebereichs liegt.

## b) Delete Line at Cursor: ESC/R

Wirkung: Löscht die gesamte durch den Cursor bestimmte Zeile. Die Position des Cursors wird nicht verändert.

## c) Erase to End of Line: ESC/T oder ESC/t

Wirkung: Löscht die aktuelle Zeile, ab der momentanen Cursor Position. Diese bleibt unverändert.

## d) Erase to End of Screen: ESC/Y oder ESC/y

Wirkung: Löscht den gesamten Bildschirminhalt ab der momentanen Cursor Position. Diese bleibt unverändert.

## e) Insert Line at Cursor: ESC/E

Wirkung: Schiebt den gesamten Bildschirminhalt ab der durch den Cursor bestimmten Zeile um eine Zeile nach unten. Die Cursor Position bleibt unverändert.

## f) Insert Character at Cursor: ESC/C

Wirkung: Schiebt in der durch den Cursor bestimmten Zeile alle Zeichen ab der Cursor Position um eine Stelle nach rechts. Die Position des Cursors bleibt unverändert.

## g) Video Invert Control

Invert On: ESC/- oder ESC/e oder ESC/j

Invert Off: ESC/q oder ESC/k oder ESC/m

Wirkung: Schaltet den Invers-Video-Modus ein oder aus.

## h) Cursor Control

Cursor On: ESC/a

Cursor Off: ESC/b

Wirkung: Schaltet den Cursor ein bzw. aus.

Nicht implementierte ESC-Sequenzen werden ignoriert.

Hinweis: Während der Abarbeitung von Programmen, die ESC-Sequenzen für \$MON/XMON verwenden, dürfen keine Ausgaben an \$MON von Background Tasks erfolgen. Dies betrifft alle 'druckbaren' Zeichen (größer 20H).

#### 4. Konfigurierung von KOS

Bei KOS5.5x wird durch das SYSGEN-Kommando die Liste der Logins und der zugehörigen Wdirs editiert. Bei Auslieferung sind beide Werte auf "" eingestellt.

Das Betriebssystem KOS6 kann bzw. muß in folgenden Punkten konfiguriert werden:

- a) Sprache: deutsch ist Standard, englisch/französisch ist Option
- b) Festplatte (20 Mbyte ist Standard)
- c) Liste der zugelassenen 'Logins'
- d) Bestimmung der optionalen Boot-Segmente (KOS8.SYS...KOSF.SYS)
- e) Automatischer 'Login' oder 'Login'-Abprüfung
- f) Erweitern um anwendungsspezifische residente Treiber

Bei Auslieferung ist die KOS-Systemdiskette folgendermaßen konfiguriert:

- a) deutsche Version
- b) 20 Mbyte Festplatte
- c) \* ist das einzig gültige 'Login'
- d) keine optionalen Boot-Segmente werden geladen (Standard-Boot: KOS0.SYS...KOS7.SYS)
- e) 'Login' wird geprüft
- f) keine anwendungsspezifischen residenten Treiber.

##### 4.1 Änderung der Standardkonfiguration bei KOS6

Eine Änderung der Standardkonfiguration ist in der Regel nicht notwendig. Die Systeme werden fertig konfiguriert ausgeliefert.

Bei späteren Updates ist KOS6 für die jeweilige Massenspeicherkonfiguration zu generieren, siehe KOSGEN-Kommando.



Mit dem Kommando SYSGEN werden die gültigen "Logins" und "Working-Directories" eingestellt (genaue Beschreibung im Teil "Systemkommandos"):

**- Liste der gültigen 'Logins' und der zugehörigen 'Working Directories'**

Verwenden Sie hierzu das Kommando E (EDIT) von SYSGEN.  
Mit dem Kommando KOSGEN KOSO werden die zu ladenden Boot Segmente definiert.

**- Bestimmung der optionalen Boot-Segmente**

Das Betriebssystem besteht aus bis zu 16 Segmenten  
KOSO.SYS...KOSF.SYS. Standardmäßig werden die Segmente  
KOSO.SYS...KOS7.SYS geladen.

**Anwendungsbeispiele:**

Soll ein System als KOBUS-Teilnehmer (Master/Slave) fungieren, so muß Segment KOS8.SYS mitgeladen werden. KOSA.SYS ist für externe Plattensysteme notwendig. Auch Anwender-Programme können resident gemacht werden, näheres dazu im folgenden Abschnitt.

**- Automatischer 'Login'**

Setzen Sie hierzu das "Auto Login Flag" in KOSO.SYS entsprechend:

Eingabe von "0": Login wird abgefragt.  
Eingabe von "1": Login wird nicht abgefragt, das System geht direkt in die Ausführung von KOS.INI über. Es gilt das als erstes eingetragene Working Directory.

**Wichtiger Hinweis:** Das Ergebnis einer jeden Änderung durch SYSGEN wird erst durch das Kommando 'W' (Write), und die Bestätigung "Y" auf Diskette geschrieben.

KOS ist nun fertig konfiguriert. Nach dem erneuten Booten wird die geänderte Version geladen.

#### 4.2 Konfigurierung mit anwendungsspezifischen residenten E/A-Treiber bei KOS6.

Optional erhältlich (Option KOS-IF) sind die Quelldateien der KOS-Konfigurierungsschnittstelle. Auf dieser Diskette mit Source- und Objectdateien befindet sich unter anderem die Datei 'IOTABLE.SRC'. Durch Editieren dieser Datei kann die standardmäßige Treiberkonfiguration von KOS den Kundenwünschen angepaßt werden, um beispielsweise

- die Kanalzuordnung zu ändern
- nicht benötigte Treiber zu streichen
- neue Treiber hinzuzufügen

Das ASM-Programm IOTABLE.SRC enthält im wesentlichen drei Sprungtabellen, sowie die Namenstabelle aller residenten E/A-Treiber. Über die Sprungtabellen mit jeweils 10 Einträgen, erfolgt die Verzweigung zu den einzelnen E/A-Kanälen, entsprechend der Kanalnummer 0...9. Die Reihenfolge der Sprungtabellen (Eingabe-, Ausgabe-, Medienkanäle) darf nicht verändert werden.

Das Quellprogramm ist ausführlich dokumentiert, weshalb sich eine detaillierte Dokumentation an dieser Stelle erübrigt. Ein Listing der Datei IOTABLE.SRC ist im Lieferumfang der Option KOS-IF enthalten.

Im wesentlichen werden die Änderungen von KOS durch Editieren und Assemblieren von IOTABLE.SRC durchgeführt. Anschließend wird IOTABLE.OBJ zur Systemdatei KOSO.SYS gelinkt.

Alle erforderlichen Quell- und Objectdateien sind im Lieferumfang von KOS-IF enthalten.

## 5. KOS - SYSTEMFUNKTIONEN

Im folgenden werden die Systemfunktionen im Stand von KOS 5.44, 5.54 und 6.05 dargestellt!

### 5.1 Übersicht und Definitionen

KOS bietet eine Reihe von Diensten an, auf die jedes Programm auf einfache Art und Weise zurückgreifen kann. Solche Dienste werden im folgenden als Systemfunktionen bezeichnet. Über Systemaufrufe (engl.: System Call) stehen sie jedem Programm zur Verfügung.

Alle Systemaufrufe werden über einen gemeinsamen Einsprungpunkt an das Betriebssystem weitergeleitet. Dort erfolgt die Verzweigung aufgrund eines mitgelieferten Funktionscodes (engl.: Request Code).

Es wird zwischen drei Gruppen von Systemfunktionen unterschieden:

- 1) Basis Ein-/Ausgabe Funktionen zur zeichen- oder blockorientierten Datenkommunikation mit Ein-/Ausgabekanälen und Massenspeichern.
- 2) Dateiverwaltungsfunktionen zur 'satzorientierten' Datenkommunikation mit Massenspeichern auf der Ebene logischer Größen, wie Dateien.
- 3) Allgemeine Systemfunktionen zur Verwaltung von Betriebsmitteln wie Speicher, Rechenzeit (Multitasking) etc.

### 5.1 Aufrufkonventionen

Systemfunktionen sind formal als Unterprogramme zu betrachten, erreichbar über den Systemeinsprungpunkt auf Adresse 8 durch den 'Ein-Byte-Call-Befehl' RST 8

Grundsätzlich erfordert jeder Systemaufruf Eingangsparameter und liefert Ausgangsparameter zurück. Eingangsparameter sind:

- 1) Der Systemaufrufvektor (engl.: System Call Vector), bestehend aus 8 oder 16 Bytes, auf den das ix-Register der CPU zeigt. Dieser Vektor wird oftmals auch als ix-Vektor bezeichnet.
- 2) Die spezifischen Eingangsgrößen einer Funktion, die entsprechend folgender Konventionen übergeben werden:
  - a) 8 Bit Größen im Register A der CPU
  - b) 16 Bit Größen im Registerpaar HL der CPU

Sind weitere Eingangsparameter erforderlich, so werden hierfür entweder Teile des ix-Vektors, weitere Register oder beides gleichzeitig verwendet.

Die Übergabe von Ausgangsparametern richtet sich ebenfalls nach diesen Vereinbarungen, also:

- a) 8 Bit Größen im Register A der CPU
- b) 16 Bit Größen im Registerpaar HL der CPU

### 5.1.2 Systemaufrufvektor (ix-Vektor)

Der ix-Vektor ist allen Systemfunktionen gemeinsam. Er zeigt auf einen Speicherbereich, dessen Informationen teilweise vom Betriebssystem, teilweise von der aufgerufenen Funktion ausgewertet werden. Normalerweise umfaßt der ix-Vektor 8 Bytes, in einigen Sonderfällen, gekennzeichnet durch Bit 4 von (ix+0), auch 16 Byte.

- (ix+0) Request Modifier Bits (RMB0...RMB7)
  - RMB0...RMB3: logische Kanalnummer für Basis E/A-Funktionen (Wertebereich 0...9)
  - RMB4: funktionsabhängig (\*)
  - RMB5: 0 - Vektor umfaßt 8 Byte  
1 - Vektor umfaßt 16 Byte
  - RMB6: funktionsabhängig (\*)
  - RMB7: funktionsabhängig (\*)
- (ix+1) Funktionscode (Request Code)
- (ix+2) funktionsabhängiger Eingangsparameter (\*)
- (ix+3) funktionsabhängiger Eingangsparameter (\*)
- (ix+4) funktionsabhängiger Eingangsparameter (\*)
- (ix+5) Rückmeldungscode (Return Code) des Betriebssystems an das aufrufende Programm. Dieser enthält:
  - a) den Wert 0 im Normalfall
  - b) einen Wert 40H bis 4FH zur Kennzeichnung von 'Sonderfällen'
  - c) einen Wert 80H bis 8FH zur Kennzeichnung von 'Fehlerfällen'
- (ix+6/7) Rückkehradresse im Fehlerfall: (ix+5) = 80H..8FH, wird nur berücksichtigt, falls ungleich 0
- (ix+8) funktionsabhängiger Eingangsparameter (\*)
- bis
- (ix+15) funktionsabhängiger Eingangsparameter (\*)

(\*) Diese Eingangsparameter müssen immer den Wert 0 enthalten, falls sie von der aufgerufenen Funktion nicht ausgewertet werden. Dies sichert Aufwärtskompatibilität zu eventuellen zukünftigen Erweiterungen des Betriebssystems.

### 5.1.3 Parameterübergabe in Registern

Bei jedem Systemaufruf rettet das Betriebssystem die Inhalte sämtlicher CPU-Register in den Stack. Gleichzeitig wird das IY-Register mit dem Wert des Stackpointers geladen. Dieser 'IY-Stack' ist im allgemeinen von Systemfunktionen für die Register-orientierte Parameterübergabe zu verwenden. Der zweite Registersatz der CPU wird vom Betriebssystem nicht verwendet.

Der Aufbau des 'IY-Stacks' ist wie folgt:

(IY-6)	Returnadresse	(low byte)
(IY-5)	Returnadresse	(high byte)
(IY-4)	frei verwendbar, mit Null vorbelegt(*)	
(IY-3)	frei verwendbar, mit Null vorbelegt(*)	
(IY-2)	frei verwendbar, mit Null vorbelegt(*)	
(IY-1)	frei verwendbar, mit Null vorbelegt(*)	
-----		
(IY+0)	L-Register	
(IY+1)	H-Register	
(IY+2)	E-Register	
(IY+3)	D-Register	
(IY+4)	C-Register	
(IY+5)	B-Register	
(IY+6)	F-Register	
(IY+7)	A-Register	
(IY+8)	IY-Register	(low byte)
(IY+9)	IY-Register	(high byte)

(\*) Diese mit Null vorbelegten Speicherstellen können von einer Systemfunktion (Treiber etc.) als temporäre Softwareregister verwendet werden.

Einer Systemfunktion - dazu zählen auch Anwender-geschriebene E/A-Treiber - stehen somit jederzeit die ursprünglichen Registerinhalte zur Verfügung. Beim Einsprung in eine Systemfunktion enthalten die Register A und HL bereits die ursprünglichen Werte, sind also mit (IY + 7) bzw. (IY + 0/1) identisch.

Systemfunktionen, die Parameter in Registern zurückliefern, tun dies ebenfalls über den 'IY-Stack'.

Beispiel: Ein Eingabetreiber, der ein Zeichen im Register A zurückliefert, muß dies unter (IY+7) mit folgendem ASM-Statement tun:

```
ld (iy+7),a
```

#### 5.1.4 Rückmeldecode (Return Code)

Der 'Return Code' zeigt dem aufrufenden Programm an, ob der Systemaufruf korrekt ausgeführt werden konnte. Während der Wert 0 auf die fehlerfreie Ausführung eines Systemaufrufs hinweist, deutet ein Wert ungleich 0 eine Ausnahmesituation an. Fehlerfälle sind grundsätzlich durch Bit 7 im Return Code gekennzeichnet. In der folgenden Aufstellung ist jeweils angegeben, ob die Fehlerursache auf logischer oder physikalischer Ebene lokalisiert ist. Alle aufgeführten Beispiele von Fehlern 'physikalischer' Natur beruhen auf Disketten-Laufwerken.

KOS definiert folgende 'Return Codes':

80H	wie 81H
81H	Eingangsparameter 'out of range' Ursache: logisch Beispiel: Sektor/Spurnummer überschritten, Funktion nicht implementiert
82H	E/A-Gerät nicht bereit ('online') Ursache: physikalisch Beispiel: Diskette nicht eingelegt
83H	E/A- oder Medientreiber nicht aktiviert Ursache: logisch
84H	Datenübertragung fehlerhaft durchgeführt Ursache: physikalisch Beispiel: CRC-Fehler (Cyclic Redundancy Check Error) bei Diskettenzugriffen
85H	Datenübertragung nicht durchgeführt Ursache: physikalisch Beispiel: Sektor/Spur nicht gefunden
86H	Datenübertragung nicht durchgeführt: Richtungskonflikt Ursache: physikalisch/logisch Beispiel: Diskette mechanisch schreibgeschützt, deshalb 'Read Only' Richtung
87H	nicht definiert
88H	Medium (Massenspeicher) voll Ursache: logisch
89H	Betriebsmittel wie Speicher, Rechenzeit für Tasks etc. nicht verfügbar Ursache: logisch
8AH	Maximale Anzahl der gleichzeitig offenen Dateien überschritten (zulässig sind 16) Ursache: logisch
8BH	Formatfehler im Inhaltsverzeichnis eines Massenspeichers (Diskette etc.) Ursache: logisch / physikalisch

- 8CH            Dateizugriffskonflikt: Datei ist schreibgeschützt  
Ursache: logisch
- 8DH            Dateizugriffskonflikt: Datei ist löschgeschützt  
Ursache: logisch
- 8EH            Dateisystem ist inkonsistent  
Ursache: zwei oder mehrere Dateien belegen ein und denselben physikalischen Bereich einer Diskette
- 8FH            Zugriff auf den gewünschten Satz einer Datei derzeit durch einen anderen Benutzer blockiert (nur in KOBUS-Systemen möglich)

Im Fehlerfall kann KOS optional zu einer unter (ix+6/7) spezifizierten Adresse zurückkehren, um so beispielsweise automatisch eine Fehlerbehandlungsroutine des aufrufenden Programms anzuspringen. Dieser Mechanismus tritt allerdings nur in Kraft, falls (ix+6/7) einen Wert ungleich 0 enthalten.

Bestimmte Systemfunktionen und KOS selbst kennzeichnen nicht fehlerbedingte Sondersituationen ebenfalls in (ix+5) mit Werten zwischen 40H und 4FH. Bislang sind lediglich die Werte zwischen 40H und 45H definiert.

- 40H            **Anfang einer Datenübertragung**  
Kennzeichnet das erste Byte einer beginnenden Übertragung und wird zur MODEM-Steuerung benötigt.
- 41H            **Ende einer Datenübertragung**  
Wird von allen Eingabetreibern mit der Übertragung des letzten Datenbytes erwartet und beispielsweise vom COPY-Kommando als Abbruch-Kriterium betrachtet.
- 42H            **Systemfunktion busy**  
Der 'System Call Entry Point' von KOS ist reentrant. Dies bedeutet, daß Systemaufrufe zu beliebigen Zeitpunkten von Tasks oder Interruptserviceroutinen unterbrochen werden können, die ihrerseits wieder Systemaufrufe durchführen. KOS antwortet mit dem Return Code 42H, falls die aufgerufene Funktion momentan nicht ausführbar ist, also 'Busy' ist. Dies kann nur bei Systemaufrufen innerhalb von 'Backgroundtasks' oder Interrupt Service Routinen vorkommen.
- 43H            **Datei bereits eröffnet**  
Eine bereits offene Datei wurde erneut eröffnet.
- 44H            **Speicher bleibt allokiert (belegt)**  
Kehrt ein Programm oder eine Task mit dem Wert 44H unter (ix+5) in das Betriebssystem zurück, so bleibt der für dieses Programm reservierte Speicher geschützt (allokiert).

- 45H**            **Datei logisch nicht vorhanden**  
 Rückmeldung der Systemfunktion 'Open File', falls eine Datei mit Benutzerkennzeichen logisch nicht vorhanden ist, da ein falsches Benutzerkennzeichen eingegeben wurde.
- 46H**            **Beende Kommandozeile**  
 Kehrt ein Anwenderprogramm mit 46H unter (IX+5) nach KOS zurück, dann werden weitere Kommandos in dieser Kommandozeile nicht mehr ausgeführt.

## 5.2 Basis Ein-/Ausgabe Funktionen

Basis Ein-/Ausgabe-Funktionen dienen zur zeichen- oder blockorientierten Datenkommunikation mit Ein-/Ausgabekanälen und externen Massenspeichern (sog. Medienkanälen). Der ix-Vektor umfaßt grundsätzlich 8 Byte:

- |          |  |
|----------|--|
| (ix+0)   | RMB0...3: logische Kanalnummer (0...9)   |
|          | RMB4: funktionsabhängig(*)               |
|          | RMB5: 0                                  |
|          | RMB6: funktionsabhängig(*)               |
|          | RMB7: 0                                  |
| (ix+1)   | Funktionscode 80H...FFH                  |
| (ix+2)   | funktionsabhängiger Eingangsparameter(*) |
| (ix+3)   | funktionsabhängiger Eingangsparameter(*) |
| (ix+4)   | funktionsabhängiger Eingangsparameter(*) |
| (ix+5)   | Return Code                              |
| (ix+6/7) | Rückkehradresse im Fehlerfall (optional) |

(\*) Diese Eingangsparameter müssen immer den Wert 0 enthalten falls sie von der aufgerufenen Funktion nicht ausgewertet werden. Dies sichert Aufwärtskompatibilität zu eventuellen zukünftigen Erweiterungen des Betriebssystems.

Basis Ein-/Ausgabe Funktionen sind in der Regel in Ein-/Ausgabe- oder Medientreibern implementiert. Zur Vereinfachung von Treibern setzt KOS die komplexeren Funktionen dieser Gruppe in einfache Grundfunktionen um. Beispielsweise wird die Ausgabe einer Zeichenfolge (String) auf die Ausgabe von Einzelzeichen zurückgeführt. Die Gruppe der Basis Ein-/Ausgabe Funktionen gliedert sich auf in drei Untergruppen:

- In Treibern implementierte E/A-Funktionen (Level I E/A).
- Im Betriebssystem implementierte E/A-Funktionen (Level II E/A).
- Im Betriebssystem implementierte Verwaltungsfunktionen für E/A-Treiber.

In der folgenden Tabelle sind die Basis E/A-Funktionen mit Ein- und Ausgangsparametern zusammengestellt. Alle 'Return Codes' zwischen 80H und 86H sind denkbar.



**Tabelle der Ein-/Ausgabe Funktionen**

Request Code: Funktion

---

80H:	Return Media Driver Identification
81H:	Output Channel Status
82H:	Input Channel Status
83H:	KOS Output Control
84H:	Character/Byte Input
85H:	String Input
86H:	Character/Byte Output
87H:	String Output
88H:	Hex Output
89H:	Read/Write Mailbox
8AH:	Read Logical Record
8BH:	Write Logical Record
8CH:	Special Functions: Input Driver
8DH:	Special Functions: Output Driver
8EH:	Read Cursor Address
8FH:	Read Scroll Address
90H:	Set Cursor Address
91H:	Set Scroll Address
92H:	Return IOD-Control Table
93H:	Assign Logical Channel Number
9CH:	Special Functions: Media Read
9DH:	Special Functions: Media Write
A0H:	Read Logical Block
A1H:	Write Logical Block

Im folgenden werden die hier aufgelisteten Funktionen näher beschrieben. Im 'Kopf' der Beschreibung sind jeweils Ein- und Ausgangsparameter angegeben.

Alle aufgeführten Programmbeispiele sind als Hilfsmittel zu betrachten, nicht aber als vollständige Module. Insbesondere fehlt in der Regel die Fehlerbehandlung.

## Return Media Driver Identification

## Funktion 80H

---

Gruppe: Medientreiber Basisfunktion

Eingang: (ix+0) - RMB0...3: log. Kanalnummer  
HL - Pufferzeiger

Ausgang: -

Basisfunktion eines Medientreibers (\$DSKx, \$VMED etc.), über die sich insbesondere die Dateiverwaltung von KOS die notwendigen Informationen über den Treiber beschafft. Ein Medientreiber überträgt zur Identifikation 4 Byte in den durch Registerpaar <HL> bestimmten Pufferbereich:

Byte 0: Bit 0...3: physikalische Satzlänge in  $128 \times 2^{\text{exp } n}$   
Bit 4...6: nicht verwendet  
Bit 7: ext. Medium (wie \$SLAV)

Byte 1-3: maximale Satz-nummer (19 Bit)

Dieser Wert kennzeichnet die Kapazität eines Mediums, also die Anzahl der dort speicherbaren 128 Byte Einheiten (Records). KOS verwaltet Medien bis zu 64 MByte.

Byte 1: high byte (3 bit)  
Byte 2: medium byte  
Byte 3: low byte

Funktion 80H wird von Anwenderprogrammen gewöhnlich nicht verwendet.

## Output Channel Status

## Funktion 81H

---

Gruppe:           Ausgabetreiber Basisfunktion

Eingang:        (ix+0) - RMB0...3: log. Kanalnummer  
                  A       - Sekundärer Funktionscode (SFC)  
                          0: nur Status  
                          1: Status + Übertragung  
                  HL       - Datenbyte (<L>)

Ausgang:        Z        - CPU Zero-Flag  
                          0: Zeichen übertragen  
                          1: Treiber 'busy'

Basisfunktion eines Ausgabetreibers (\$MON, \$SIOx etc.), die in Abhängigkeit von Register <A>, entweder nur den Status des Ausgabekanals anzeigt (A=0) oder, falls möglich, das Datenbyte im L-Register gleichzeitig überträgt (A>0). Diese Funktion dient insbesondere zur 'Task-gesteuerten' Datenübertragung an Ausgabekanäle und wird beispielsweise vom 'SPOOLER' des Betriebssystems verwendet.

Unabhängig vom Eingangswert von Register <A> zeigt in jedem Fall auch das 'Zero-Flag' der CPU den momentanten Status des Ausgabekanals an:

Zero-Flag = 1:           Der Treiber war nicht in der Lage, ein Zeichen zu übertragen. In diesem Fall enthält zusätzlich (ix+5) den Wert 42H (Funktion 'busy').

Zero-Flag = 0:           Der Treiber ist oder war bereit, ein Zeichen zu übertragen. Das Zeichen im L-Register wurde also übertragen, sofern <A> beim Aufruf der Funktion einen Wert ungleich 0 hatte.

## Input Channel Status

Funktion 82H

---

Gruppe:            Eingabetreiber Basisfunktion

Eingang:        (ix+0) - RMB0...3: log. Kanalnummer  
                 A        - Sekundärer Funktionscode (SFC)  
                            0: nur Status  
                            1: Status + Übertragung

Ausgang:        A        - Datenbyte, falls vorhanden  
                 Z        - CPU Zero-Flag  
                            1: Kein Zeichen vorhanden  
                            0: Zeichen vorhanden

Basisfunktion eines Eingabetreibers (\$KEY, \$SIOx etc.), die zur Abfrage des momentanen Status des Eingabekanals dient. Der Status eines Eingabekanals sagt aus, ob ein Byte vorliegt oder nicht. Beim Aufruf dieser Funktion bestimmt der 'Sekundäre Funktionscode' im Register <A>, ob im Falle eines vorhandenen Bytes dieses gleich mit übertragen werden soll. Ist ein Byte vorhanden, so wird dieses im Register <A> übergeben. Es bleibt jedoch logisch vorhanden, sofern der sekundäre Funktionscode den Wert 0 hatte.

Unabhängig davon zeigt auch das 'Zero-Flag' der CPU den momentanen Status des Eingabekanals an:

Zero-Flag = 1:            Es liegt kein Zeichen vor.

Zero-Flag = 0:            Es liegt ein Byte vor. Dieses wurde in Register <A> übertragen, bleibt aber logisch weiter vorhanden, falls <A> beim Aufruf der Funktion den Wert 0 hatte.

## KOS Output Control

## Funktion 83H

Gruppe: Level II E/A-Funktion des Betriebssystems

Eingang: (ix+0) - RMB0...3: log. Kanalnummer

Ausgang: Z - CPU-Zero Flag  
0: ESC-Taste war gedrückt  
1: ESC-Taste war nicht gedrückt

Funktion des Betriebssystems zur Steuerung von Ausgaben auf die Bedienkonsole (im allgemeinen ein CRT-Monitor). Neben dem programm-gesteuerten Anhalten, Verlangsamen und Abbrechen von Textausgaben ermöglicht diese Funktion auch das 'Hin- und Herblättern' im Bildwiederholtspeicher der zugrundeliegenden Hardware. Die Wirkung dieser Funktion wird durch folgende Tasten bestimmt:

- a) Eine beliebige Taste außer CTRL-S, CTRL-P und ESC führt zum Warten, bis eine beliebige weitere Eingabe erfolgt.
- b) Die Tastenkombination CTRL-S führt zur Verlangsamung/Beschleunigung der Schreibgeschwindigkeit auf dem Sichtgerät.
- c) Die Tastenkombination CTRL-P führt in den sogenannten 'Page Modus' des Sichtschirmtreibers (\$MON). Mit Hilfe der Cursortasten kann zeilen- oder seitenweise innerhalb der letzten 8 Seiten des Bildwiederholtspeichers 'Hin- und Hergeblättert' werden:

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

Als Ergebnis von Funktion 83H zeigt das Zero-Flag, ob die ESC-Taste gedrückt wurde:

Z-Flag = 1:	ESC-Taste war nicht gedrückt
Z-Flag = 0:	ESC-Taste war gedrückt

Funktion 83H wird von vielen KOS-Systemkommandos verwendet. Die ESC-Taste führt üblicherweise zum Abbruch der gerade laufenden Aktivität.

KOS führt die Funktion 83H automatisch auf die Basisfunktionen 82H, 84H und 86H zurück.

CTRL-S und CTRL-P arbeiten nur zusammen mit dem Ausgabetreiber \$MON, nicht jedoch bei anderen Ausgabekanälen.

## Character/Byte Input

## Funktion 84H

---

Gruppe:        Eingabetreiber Basisfunktion  
Eingang:       (ix+0) - RMB0...3: log. Kanalnummer  
Ausgang:       A        - Datenbyte

Basisfunktion eines Eingabetreibers (\$KEY, \$SIOx etc.) zur zeichen- bzw. byteorientierten Datenübertragung von Peripheriegeräten. Diese Funktion wartet in einer Schleife, bis ein Datenbyte vorliegt und übergibt dieses im Register <A>.

Folgende Sonderfälle werden unter (ix+5) signalisiert:

- a) (ix+5) = 40H:        Start einer Datenübertragung. Dies wird nur von speziellen Treibern erwartet (z.B. Modem-ansteuerung)
- b) (ix+5) = 41H:        Ende einer Datenübertragung. (ix+5) wird vom COPY-Kommando ausgewertet, um das 'Kopieren' von einem Eingabekanal abubrechen. Im allgemeinen meldet ein Treiber dann 'End of Transmission', wenn CTRL-D (ASCII-Code: 4) erkannt wurde. Dies gilt beispielsweise für \$KEY.

## String Input

## Funktion 85H

---

Gruppe: Level II E/A-Funktion des Betriebssystems

Eingang: (ix+0) - RMB0...3: log. Kanalnummer  
RMB4: Direct I/O  
A - Anzahl der maximal einzulesenden Zeichen  
HL - Puffer Zeiger

Ausgang: A - Anzahl der eingelesenen Zeichen  
HL - Zeiger auf das erste Zeichen

Funktion zum Einlesen von maximal <A>-Zeichen in einen Pufferbereich ab Adresse <HL>. Das Zeichen 'RETURN' (ASCII-Code: 0DH) beendet die Eingabe vorzeitig. Die Größe des Pufferbereichs muß <A>+3 sein, da drei Speicherstellen für temporäre Werte verwendet werden. Alle eingegebenen Zeichen werden automatisch reflektiert. Korrekturen sind mit den Tasten CURSOR-LEFT bzw. RUBOUT (DEL) möglich:

CURSOR-LEFT (CTRL-H): Löscht das zuletzt eingegebene Zeichen und bewegt den Cursor um eine Stelle nach links. Pro zu löschendes Zeichen wird die Zeichenfolge BACKSPACE-BLANK-BACKSPACE' (Code: 8-20H-8) übertragen.

RUBOUT (DEL): Löscht alle bisher eingegebenen Zeichen und setzt den Cursor auf die ursprüngliche Anfangsposition. Pro zu löschendes Zeichen wird die Zeichenfolge BACKSPACE-BLANK-BACKSPACE' (Code: 8-20H-8) übertragen.

Nach der Rückkehr enthält Register <A> die Anzahl der tatsächlich eingegebenen Zeichen; Registerpaar <HL> zeigt auf das erste eingegebene Zeichen.

Im allgemeinen führt KOS die Funktion 85H auf die beiden Basisfunktionen 84H und 86H zurück, es sei denn, Bit 4 von (ix+0) ist gesetzt (Request Modifier Bit: RMB4). In diesem Fall wird der Systemaufruf direkt an den entsprechenden Eingabetreiber weitergeleitet. Der Benutzer hat damit die Möglichkeit, diese Funktion in spezifischen Treibern selbst zu implementieren und seinen Anforderungen anzupassen.

## Character/Byte Output

## Funktion 86H

---

Gruppe:       Ausgabetreiber Basisfunktion

Eingang:       (ix+0) - RMB0...3: log. Kanalnummer  
              A       - Datenbyte

Ausgang:       -

Basisfunktion eines Ausgabetreibers (\$MON, \$SIOx etc.) zur zeichen- bzw. byteorientierten Datenübertragung an Peripheriegeräte.

f86.example:

```
        .comment  #output of <B> bytes
                <HL> is byte pointer #

        ld (ix+0),4    ;lets take channel #4
        ld (ix+1),86h
loop.86h:
        ld a,(hl)
        rst 8          ;System call 86h
        inc hl
        djnz loop.86h ;loop <B> times
        ret
```



## String Output

## Funktion 87H

---

Gruppe: Level II E/A-Funktion des Betriebssystems

Eingang: (ix+0) - RMB0...3: log. Kanalnummer  
          RMB4: Direct I/O  
          HL - Zeiger auf Zeichenkette (String)

Ausgang: -

Funktion zur Ausgabe von Zeichenketten (Strings). Die Zeichenkette muß mit Null abgeschlossen sein (RMB4=0) oder es muß ihre Länge unter (ix+2) spezifiziert sein (RMB4=1).

Im allgemeinen führt KOS die Funktion 87H auf die Basisfunktion 86H zurück, es sei denn, das 'Request Modifier Bit' RMB4 ist gesetzt. In diesem Fall wird der Systemaufruf direkt an den entsprechenden Ausgabekanal weitergeleitet. Der Benutzer hat damit die Möglichkeit, diese Funktion in spezifischen Treibern selbst zu implementieren und seinen Anforderungen anzupassen, um damit beispielsweise beliebige Bytefolgen übertragen zu können. Es wird folgendes Verhalten erwartet, falls Funktion 87H direkt in einem Treiber implementiert ist:

(ix+2) bestimmt die Anzahl der zu übertragenden Bytes, falls es ungleich 00H ist. Ansonsten gilt der String-Delimiter '0'.

## f87.example:

```
ld (ix+0),2      ;load logical unit
ld (ix+1), 87h
ld hl, msg       ;load string pointer
rst 8
ret

msg:             defw 0a0dh      ;line feed and return
                defm 'This is a string'
                defb 0          ;delimiter
```

## Hex Output

Funktion 88H

---

Gruppe: Level II E/A-Funktion des Betriebssystems

Eingang: (ix+0) - RMB0...3: log. Kanalnummer  
A - Hexadezimalzahl

Ausgang: -

Funktion zur Ausgabe der Hexadezimalzahl in Register <A> als zwei ASCII-Zeichen. KOS führt diese Funktion auf die Basisfunktion 86H (Character output) zurück.

Das zuerst ausgegebene Zeichen entspricht dem höherwertigen Halbbyte des hexadezimalen Zahlenwertes.

f88.example:

```
.comment #  
Output of a 4 digit hex number in  
registerpair <HL> to channel 2  
#  
  
ld (ix+0),2  
ld (ix+1),88h  
ld a,h           ;Most significant 2 digits  
rst 8  
ld a,l           ;Least significant 2 digits  
rst 8  
ret
```

## Mailbox

---

In Kontron KOBUS Systemen steht für jeden KOBUS-Teilnehmer erreichbar ein Mailboxregistersatz von 8 Registern zu je 128 Byte (Zeichen) zur Verfügung.

Jedes an KOBUS angeschlossene System kann jedes dieser Register auslesen oder beschreiben.

Das Lesen und Schreiben der Mailboxregister erfolgt über Aufrufe an das Betriebssystem, bei denen der KOBUS Medienkanal angesprochen wird.

### KOBUS Medienkanäle:

	KOS5	KOS6
Master	\$PUB	\$WIN0 oder anderes KOBUS Medium
Slave	\$SLAV/\$SLAT	\$SLV0 ... \$SLV3

## Read Mailbox

## Funktion 89H

Gruppe: Medientreiber KOBUS Funktion

Eingang: (ix+0) - RMB0...3: log. Kanalnummer KOBUS Treiber  
 A - 8 (Unterfunktion Mailbox lesen)  
 HL - Pufferzeiger  
 Puffer+0 - Registernummer (0...7)

Ausgang: -

Funktion eines KOBUS Medientreibers (\$SLVx,\$WINx,\$SLAV,\$SLAT,\$PUB) zum Auslesen eines der 8 Mailboxregister. Im ersten Byte des Lesepuffers steht beim Aufruf die Registernummer. Nach dem Aufruf steht im Lesepuffer ab dem 2. Zeichen der Inhalt des Mailboxregisters (128 Byte).

## mailbox.example:

```
ld    (ix+0),kobus.chan    ; KOBUS media channel
ld    (ix+1),89h
ld    a,8
ld    hl,rx.buffer        ; receive buffer
ld    (hl),0              ; register 0
rst    8                  ; KOS call
ld    hl,rx.buffer+1      ; pointer to mailbox contents
ld    a,(ix+5)            ; check error
and    a                  ; zero flag reset if error
ret
```

## rx.buffer:

```
defs 129                  ; 1 byte reg.no, 128 byte data
```

## Write Mailbox

## Funktion 89H

Gruppe: Medientreiber KOBUS Funktion

Eingang: (ix+0) - RMB0...3: log. Kanalnummer KOBUS Treiber  
 A - 9 (Unterfunktion Mailbox schreiben)  
 HL - Pufferzeichen  
 Puffer+0 - Registernummer (0...7)

Ausgang: -

Funktion eines KOBUS Medientreibers (\$SLVx,\$WINx,\$SLAV,\$SLAT,\$PUB) zum Beschreiben eines der 8 Mailboxregister. Im ersten Byte des Schreibpuffers steht beim Aufruf die Registernummer und ab dem 2. Zeichen der Inhalt des Mailboxregisters (128 Byte).

mailbox.example:

```

ld (ix+0),kobus.chan      ; wr.buffer contains data
ld (ix+1),89h             ; starting at wr.buffer+1
ld a,9                   ; KOBUS media channel
ld hl,wr.buffer           ; write buffer
ld (hl),0                ; register 0
rst 8                    ; KOS call
ld a,(ix+5)               ; check error
and a                    ; zero flag reset if error
ret
```

```

wr.buffer:
  defs 129                ; 1 byte reg.no, 128 byte data
```

Read Logical Record  
Write Logical Record

Funktion 8AH  
Funktion 8BH

---

Gruppe: Medientreiber Basisfunktion

Eingang: (ix+0) - RMB0...3: log. Kanalnummer  
RMB6: Reserve Bereich  
(ix+2/3/4) - log. Satz-(Record-) Nummer  
HL - Pufferbereich

Ausgang: -

Basisfunktion eines Medientreibers (\$DSKx, \$WINO, \$VMED etc.) über die ein log. Satz, bestehend aus 128 Byte, gelesen bzw. geschrieben wird.

Registerpaar <HL> definiert den 128 Byte Record-Puffer. Die Dateiverwaltung von KOS verwendet diese Funktion zur Kommunikation mit dem 'logischen Gebilde Medium'. Zur Adressierung einzelner Records dienen 19 Bits, entsprechend einer maximalen Medienkapazität von 8 x 65636 Records (= 64 MByte).

Ein Medium besitzt im allgemeinen Fall eine bestimmte Anzahl von Reserve Records, die alternativ für defekte Records (Bad Blocks) des normalen Bereichs verwendet werden. Reserve Records sind normalerweise über Systemaufrufe nicht zugänglich, es sei denn, RMB6 ist gesetzt.

Die Record-Nummern für 'Reserve Records' schließen sich unmittelbar den Record-Nummern des 'normalen Bereichs' an.

Medientreiber, deren zugrundeliegendes Medium eine physikalische Satzlänge ungleich der logischen Satzlänge von 128 Bytes hat, benötigen geeignete 'Deblocking' und 'Blocking' Mechanismen. Derartige Routinen sind in allen Medientreibern von KOS implementiert. Sie führen in der Regel dazu, daß ein Aufruf der Funktion 8BH zunächst keinen physikalischen Zugriff auf ein Medium bewirkt.

Special Functions: Input Driver  
 Special Functions: Output Driver

Funktion 8CH  
 Funktion 8DH

Gruppe: Ein-/Ausgabetreiber Basisfunktion

Eingang: (ix+0) - RMBO...3: log. Kanalnummer  
 A - Sekundärer Funktionscode (SFC)

Ausgang: A - Treiberidentifikation, falls SFC=0

Basisfunktion eines Ein-/Ausgabetreibers (\$KEY, \$MON, \$SIOx etc.), die in Abhängigkeit des Sekundären Funktionscodes in Register <A> folgende Routinen aufruft:

A = 0: INIT-Routine des Treibers  
 A = 1: OPEN-Routine des Treibers  
 A = 2: CLOSE-Routine des Treibers  
 A = 3: Statusinformation 8 Byte  
 (nur bei 980 H-Systemen)

Die spezifische Wirkung dieser Funktionen hängt von der jeweiligen Treiberimplementation ab.

In Anwender-geschriebenen Treibern können jederzeit weitere sekundäre Funktionen implementiert werden.

Die Funktionen INIT/OPEN/CLOSE werden beispielsweise bei folgenden Aktivitäten aufgerufen:

INIT: Treiberaktivierung (IODC \$IOD=ACTIVE)  
 Treiberneuinitialisierung (N \$IOD)

OPEN: Datenübertragung Anfang: COPY \$IOD file  
 PRINT file 0=\$IOD

CLOSE: Datenübertragung Ende: COPY \$IOD file  
 PRINT file 0=\$IOD

#### Hinweis für Kontron PSI980 H-Systeme:

Der Treiber \$MON liefert mit Funktion 8D (SFC=3) 8 Bytes an Statusinformation in einen durch Registerpaar <HL> definierten Speicherbereich. Die Bedeutung ist wie folgt:

Byte 0: Aktuelles Bildschirmformat - Zeilen/Bild  
 Byte 1: Aktuelles Bildschirmformat - Zeichen/Zeile  
 Byte 2: Auflösung  
           0 - 512 x 256  
           1 - 1024 x 400 (Kontron PSI980 H)  
 Byte 3: Video Attribute (Kontron PSI980 H)  
 Byte 4: reserviert (0)  
 .  
 .  
 .  
 Byte 7: reserviert (0)

Diese Funktion wird beispielsweise von EDIT und BASIC ausgewertet.

Read Cursor Address	Funktion 8EH
Read Scroll Address	Funktion 8FH
Set Cursor Address	Funktion 90H
Set Scroll Address	Funktion 91H

---

Gruppe: Basisfunktion des Ausgabetreibers \$MON

Eingang: HL - relative Cursor/Scroll Adresse (8EH/8FH)

Ausgang: HL - relative Cursor/Scroll Adresse (90H/91H)

Basisfunktion des Ausgabetreibers \$MON, über die der aktuelle Stand von Cursor und Scroll Adresse ermittelt bzw. verändert werden kann. Beide Werte repräsentieren eine relative Adresse des Bildwiederhol-speichers.

- a) Die Cursor Adresse, bestimmt die Position des nächsten darzu-stellenden Zeichens.

Wertebereich: 0 - 47D0H

- b) Die Scroll Adresse, bestimmt die dem Bildanfang entsprechende Adresse des Bildwiederhol-speichers.

Wertebereich: 0 - 3FF0H

Da 'Scrolling' nur zeilenweise erfolgt, ist die relative Scroll Adresse normalerweise ein Vielfaches von 80 (50H)

Hinweis: \$MON verwaltet 16 kByte des Bildwiederhol-speichers als zirkulierenden Speicher auf Basis der beiden Parameter 'Cursor- bzw. Scrolladresse'. Da die Cursor Adresse in jedem Fall größer/gleich der Scroll Adresse sein muß, geht der Wertebereich der Cursor Adresse scheinbar über 16 kByte (4000H) hinaus.



## Return IOD-Control Table

Funktion 92H

---

Gruppe: E/A-Treiber Verwaltung  
Eingang: -  
Ausgang: HL - Zeiger auf 'IOD-Control Table'

Funktion zur Bestimmung der Lage verschiedener Tabellen innerhalb des Betriebssystems.

Registerpaar <HL> zeigt auf eine Tabelle bestehend aus 4 Adreßeinträgen, die ihrerseits wieder auf Tabellen zeigen.

1. Namenstabelle aller E/A- und Medientreiber mit 8 Byte pro Treiber und Platz für 20 Einträge.

Bedeutung der einzelnen Bytes:

Byte 1: Anzahl der vom Treiber belegten Speichersegmente  
Byte 2/3: Startadresse des Treibers  
Byte 4: Treiberidentifikation  
0 - Ausgabetreiber  
1 - Eingabetreiber  
2 - Bidirektionaler Treiber  
3 - Medientreiber  
Byte 5..8: Treibername

2. Sprungtabelle für E/A- und Medienkanäle mit folgender Organisation:

JP I-0 bis JP I-9 ; Input  
JP O-0 bis JP O-9 ; Output  
JP M-0 bis JP M-9 ; Media

3. wie Punkt 1.

4. Sprungtabelle für Medienkanäle entsprechend der Organisation in Punkt 2.

## Assign logical Channel Number

## Funktion 93H

Gruppe: E/A-Treiber Verwaltung

Eingang: A - Bits 0...3: log. Kanalnummer  
 Bits 4...7: 0: Medientreiber  
 A: Ausgabetreiber  
 E: Eingabetreiber

HL - Zeiger auf den Treibernamen; vierstellig, mit  
 Blanks an den nicht benutzten Stellen

Ausgang: A - Treiberidentifikation  
 0: Ausgabetreiber  
 1: Eingabetreiber  
 2: bidirektionaler Treiber  
 3: Medientreiber

HL - siehe Text

DE - siehe Text

Über Funktion 93H weisen die Dienst- und Anwenderprogramme Treibern logische Kanalnummern zu. Eine Zuweisung ist nur dann möglich, wenn die Treiberidentifikation dies zuläßt. Ausgabetreiber können beispielsweise nicht auf Eingabekanäle gelegt werden. Falls dies trotzdem versucht wird, antwortet Funktion 93H mit 80H unter (ix+5).

Die Ausgabeparameter <HL> und <DE> sind abhängig von verschiedenen Fällen folgendermaßen definiert:

DE - Startadresse des Treibers, falls aktiviert, sonst unverändert

Fall 1: HL - Startadresse des Treibers, falls (ix+5)=0

Fall 2: HL - undefiniert, falls (ix+5)=83H  
 ---> Treiber ist nicht aktiviert

Fall 3: HL - Zeiger auf Namenseintrag einer Tabelle in KOS, falls (ix+5)=80H

Byte 1: Anzahl der belegten Speichersegmente  
 Byte 2/3: Startadresse  
 Byte 4: Treiberidentifikation  
 Byte 5-8: Treibername

Fall 1 ist der Normalfall, die Funktion konnte ordnungsgemäß ausgeführt werden. Fall 2 sagt aus, daß der Treiber nicht aktiviert war, während Fall 3 darauf hinweist, daß beispielsweise versucht wurde, einen Eingabetreiber einem Ausgabekanal zuzuweisen.

Special Functions: Media Read  
 Special Functions: Media Write

Funktion 9CH  
 Funktion 9DH

Gruppe: Medientreiber Basisfunktion

Eingang: (ix+0) - RMB0...3: log. Kanalnummer  
 A - Sekundärer Funktionscode (SFC)

Ausgang: A - Treiberidentifikation, falls SFC=0

Basisfunktion eines Medientreibers (\$DSK0, \$WIN0 etc.), die in Abhängigkeit des sekundären Funktionscodes in Register <A> folgende Routinen aufruft:

A = 0:	INIT-Routine des Treibers	
A = 1:	OPEN-Routine des Treibers	
A = 2:	CLOSE-Routine des Treibers	
A = 3:	Statusanzeige für die 'Bad Block' Verwaltung	(*)
A = 4:	SYNC-Routine des Treibers	(*)
A = 5:	FORMAT-Track	(*)

(\*) Diese Routinen müssen nicht notwendigerweise in einem Medientreiber implementiert sein.

A = 3:	für BBR-Kommando notwendig
A = 4:	in \$WINx (KOS6) implementiert
A = 5:	in \$DSK0/1 (KOS6) implementiert

Die spezifische Wirkung dieser Funktion hängt von der jeweiligen Treiberimplementation ab.

In Anwender-geschriebenen Treibern können jederzeit weitere sekundäre Funktionen implementiert werden.

Die Funktion INIT wird beispielsweise bei folgenden Aktivitäten aufgerufen:

INIT: Treiberaktivierung (IODC \$IOD=ACTIVE)  
 Treiberinitialisierung (N \$IOD)

Read logical Block  
Write logical Block

Funktion A0H  
Funktion A1H

Gruppe: Medientreiber Basisfunktion

Eingang:	(ix+0)	- RMB0...3: log. Kanalnummer
		RMB6: Reserve Bereich
		RMB7: Multi Sector Transfer
	(ix+2/3/4)	- log. Blocknummer
	HL	- Pufferbereich
	DE	- Anzahl der Bytes (*)
	A	- Speicherbank (0...3) (*)

Ausgang: —

(\*) nur relevant, falls RMB7 = 1

Basisfunktion eines Medientreibers (\$DSKx, \$WINO etc.), über die ein logischer Block bestehend aus  $n \times 128$  Byte gelesen bzw. geschrieben wird. Der Wert  $n$  ergibt sich aus der physikalischen Blocklänge des betreffenden Mediums.

Diese beiden Funktionen werden von KOS nicht verwendet. Sie dienen spezifischen Anwender- oder Systemprogrammen zum schnellen Zugriff auf einen Massenspeicher.

Bei Medien mit einer physikalischen Blocklänge von 128 Byte sind die Funktionen AOH/A1H identisch mit den Funktionen 8AH/8BH (Read/Write logical Record).

Logische Blöcke werden in Inkrementen von  $2 \exp n$  adressiert. Ein Medium, dessen physikalische Blocklänge 512 Byte beträgt, hat demnach folgende logische Blocknummern:

0, 4, 8, 12 etc.

### 5.3 Dateiverwaltungsfunktionen

Dateiverwaltungsfunktionen dienen der Satz- (Record-) orientierten Bearbeitung von Dateien, die auf beliebigen Medien (ext. Massenspeichern) gespeichert sind.

Eine Datei ist eindeutig beschrieben durch den sogenannten Dateispezifikationsblock (engl.: File control block), abgekürzt: DSB. Der DSB entspricht einem Eintrag im Inhaltsverzeichnis (engl.: Directory) eines Mediums und umfaßt 128 Byte. Dateiverwaltungsfunktionen erfordern in der Regel einen Zeiger auf den DSB der zu bearbeitenden Datei als Eingangsparameter.

Der ix-Vektor umfaßt gewöhnlich 8 Byte, in einigen Fällen optional auch 16 Byte.

(ix+0)	RMB0...3:	0
	RMB4:	funktionsabhängig (*)
	RMB5:	0 - Vektor umfaßt 8 Byte
		1 - Vektor umfaßt 16 Byte
	RMB6:	funktionsabhängig (*)
	RMB7:	reserviert für Dateiverwaltung
(ix+1)	Funktionscode:	40H...7FH
(ix+2)	optionale Pufferadresse (low Byte)	(*)
(ix+3)	optionale Pufferadresse (high Byte)	(*)
(ix+4)	nicht verwendet	(*)
(ix+5)	Return Code	
(ix+6/7)	optionale Rückkehradresse im Fehlerfall	
(ix+8/9)	optionale Spezifikation des Working Directory's;	nur gültig, falls RMB5 gesetzt ist.
(ix+10/15)	nicht verwendet	(*)

(\*) Diese Eingangsparameter müssen immer den Wert 0 enthalten, falls sie von der aufgerufenen Funktion nicht ausgewertet werden. Dies sichert Aufwärtskompatibilität zu eventuellen zukünftigen Erweiterungen des Betriebssystems.

Mit wenigen Ausnahmen verwenden alle Dateiverwaltungsfunktionen das Register <A> für Rückmeldungen an das aufrufende Programm, wobei zwischen zwei Fällen unterschieden wird:

- <A> = 0:       Funktion wurde erfolgreich ausgeführt
- <A> = FFH:     Funktion wurde nicht ausgeführt. Dies kann seine Ursachen sowohl auf der logischen als auch auf der physikalischen Ebene haben.

**Beispiel:       Eröffnen einer Datei**

- <A> = 0:       Datei wurde eröffnet, ist also vorhanden
- <A> = FFH:     Datei wurde nicht eröffnet, da sie
- a) entweder nicht vorhanden ist, oder
  - b) ein Zugriffsfehler beim Durchsuchen des Directory's auftrat. In diesem Fall hätte (ix+5) den entsprechenden Fehlercode.

Beim Aufruf von Dateiverwaltungsfunktionen sind alle Fehlermeldungen zwischen 80H und 8FH möglich. Hierbei resultieren die Fehler 82H bis 87H gewöhnlich von einer tieferen Ebene, dem Medientreiber. Derartige Fehlermeldungen werden unverändert an das aufrufende Programm weitergeleitet.

**Hinweis:**

Die in KOS-Versionen 5.xx/4.xx definierten Dateiverwaltungsfunktionen 40H...4FH sind in KOS weiterhin implementiert, so daß bestehende Programme unverändert ablauffähig bleiben.

Für Neuentwicklungen wird dringend empfohlen, ausschließlich die hier aufgeführten Funktionen zu verwenden.

### 5.3.1 Dateispezifikationsblock (DSB)

Wie erwähnt, umfaßt der DSB 128 Byte und entspricht exakt einem Eintrag im Inhaltsverzeichnis eines Mediums. Beim Eröffnen und Generieren einer Datei muß das aufrufende Programm lediglich die ersten 16 Byte bereitstellen (DSB16). Die Dateiverwaltung sucht daraufhin automatisch ein freies Speichersegment im Systemspeicher und generiert dort den gesamten 128 Byte DSB. Einer Datei wird beim Eröffnen eine Nummer zugewiesen (0 bis 15), die im oberen Halbbyte des ersten Bytes im DSB abgelegt wird.

Die Bedeutung der einzelnen Bytes des DSB ist wie folgt:

- |                     |   |
|---------------------|---|
| <b>Byte 0</b>       | Mediennummer (niederwertiges Halbbyte); log. Dateinummer (höherwertiges Halbbyte). Diese wird von der Dateiverwaltung beim Eröffnen eingetragen.<br>Byte 0 wird nicht auf dem Medium abgelegt. Dort steht an dieser Stelle immer Null, mit Ausnahme bei gelöschten Dateien, wo an dieser Stelle der Wert E5H steht (siehe 5.4 Seite C-79: FCB-Generator, Funktion 02H, DSB-Statusbyte 's').   |
| <b>Byte 1 - 8</b>   | Name der Datei (8-stellig mit Leerzeichen auf den nicht besetzten Stellen).   |
| <b>Byte 9 - 11</b>  | Typ der Datei (3-stellig mit Leerzeichen auf den nicht besetzten Stellen).  |
| <b>Byte 12</b>      | Dateierweiterungszähler (Extension Counter)<br>Der DSB von KOS beschreibt 48 kByte einer Datei. Bei größeren Dateien ist pro 48 kByte Segment ein eigener Eintrag im Inhaltsverzeichnis erforderlich. Byte 12 enthält die Nummer des von diesem DSB beschriebenen Segment.  |
| <b>Byte 13 - 14</b> | Ladeadresse für ablauffähige Maschinenprogramme (nur für das erste 48 kByte Segment abgelegt).<br>Bei Dateien größer 48 kByte enthält der Inhaltsverzeichnis-Eintrag für das n-te Segment in den Bytes 13/14 die log. Satznummer des Inhaltsverzeichnis-Eintrags für das (n-1)-te Segment (Backpointer).<br>Bei Medien mit einer Gesamtkapazität von mehr als 8 MByte umfaßt der Backpointer 19 Bit. Die höherwertigen 3 Bits sind in den Bitstellen 4-6 von Byte 27 enthalten. |
| <b>Byte 15</b>      | Dateieigenschaften (Properties)<br><br>Bit 0 - Public File (Systemdatei)<br>Bit 1 - Datei schreibgeschützt<br>Bit 2 - Datei löschgeschützt<br>Bit 3 - Properties gesperrt<br>Bit 4 - Record Locking (KOBUS)<br>Bit 5 - Directory Datei<br>Bit 6 - Datei hat Benutzerkennzeichen<br>Bit 7 - Datei ist 'verborgen'  |
| <b>Byte 16 - 17</b> | Datum, an dem die Datei generiert wurde.  |

- Byte 18 - 19** Working Directory (codiert), dem diese Datei zugehört, bzw. User-Identifikation.
- Byte 20 - 25** Backup Flags.
- Byte 26 - 27** Anzahl der Sätze in der Datei(erweiterung). Dies ist ein Wert zwischen 0 und 180H. Der Wert 180H (9 Bit) deutet darauf hin, daß ein weiteres 48 kByte Segment vorhanden ist. Die Bits 1-6 von Byte 27 enthalten die höherwertigen 3 Bits des Fore- und Backpointers.
- Byte 28 - 29** Satznummer des nächsten Eintrags im Inhaltsverzeichnis. Ist eine Datei(erweiterung) größer als 48 kByte, so enthalten diese Bytes die log. Satznummer des nächsten Directorysatzes für diese Datei (Forepointer). Bei Medien mit einer Gesamtkapazität von mehr als 8 MByte umfaßt der Forepointer 19 Bit. Die höherwertigen 3 Bits sind in den Bitstellen 1-3 von Byte 27 enthalten.
- Byte 30 - 31** Satzzähler für Schreib-/Lesezugriffe. Dieser bestimmt die Nummer des Satzes, der bei Schreib-/Lesezugriffen adressiert wird. Dies ist ein Wert zwischen 0 und 180H bei sequentiellen Dateizugriffen (Funktionen: 67H/68H). Bei wahlfreien Dateizugriffen (Funktionen: 77H/78H) ist dies ein beliebiger Wert zwischen 0 und FFFF.
- Byte 32 - 127** Logische Blocknummern, die dieser Datei zugeordnet sind (1 Block enthält 8 Sätze). Ist eine Datei(erweiterung) kleiner als 48 kByte, so bestimmt der Satzzähler (Byte 26 - 27) die Anzahl der relevanten Einträge im Bereich der Bytes 32 - 127. Die Blocknummer umfaßt 2 Byte. KOS kann somit Medien bis 65535 Blöcke (entsprechend 64 Megabyte) verwalten.



### 5.3.2 Beschreibung der Dateiverwaltungsfunktionen

#### Tabelle der Dateiverwaltungsfunktionen

Request Code: Funktion

-----

60H:	Init File Management
61H:	Define Master Media
62H:	Open File
63H:	Close File after write
64H:	Search File
65H:	Search next File
66H:	Delete File
67H:	Read File Sequential
68H:	Write File Sequential
69H:	Make new File
6AH:	Rename File
6BH:	Define Record Buffer
6CH:	Return Master Media Number
6DH:	Close File after Read
70H:	Return FM Table Pointers
71H:	Define Media Parameter
77H:	Read File Random
78H:	Write File Random
79H:	Position End of File

Im folgenden werden die hier aufgelisteten Funktionen näher beschrieben. Im 'Kopf' der Beschreibung sind jeweils Ein- und Ausgangsparameter angegeben.

Alle aufgeführten Programmbeispiele sind als Hilfsmittel zu betrachten, nicht aber als vollständige Module. Insbesondere fehlt in der Regel die Fehlerbehandlung.

## Init File Management

## Funktion 60H

---

Eingang:        A        -    Mediennummer n = 0...9 definiert ein  
                             bestimmtes Medium,  
                             n=\*:    für alle Medien

Ausgang:        -

Funktion zur Initialisierung der Dateiverwaltung. Bei dieser Gelegenheit werden je nach Eingangsparameter alle aktiven oder nur ein bestimmtes Medium neu initialisiert. Bei der Initialisierung eines Mediums wird das Inhaltsverzeichnis gelesen und daraus der Gesamtbelegungsplan des Mediums errechnet und im Systemspeicher zusammen mit dem Namen der Datei 'Inhaltsverzeichnis' abgelegt. Dieser Name dient der Dateiverwaltung als Mediumidentifikation.

Die Initialisierung eines Mediums durch Funktion 60H wird durch das KOS-interne Kommando N aufgerufen. Sie erfolgt automatisch, wenn beim Eröffnen einer Datei ein Wechsel der Identifikation des adressierten Mediums festgestellt wird.

Ein wichtiger Nebeneffekt dieser Funktion ist das Schließen aller offenen Dateien des adressierten Mediums.

## Define Master Media

Funktion 61H

---

Eingang:        A        -    Mediennummer n

Ausgang:       -

Funktion zur Bestimmung des Mastermediums. Definitionsgemäß wird damit dasjenige Medium festgelegt, das bei Dateireferenzen ohne Mediennummer adressiert wird.

Der Wertebereich von n liegt zwischen 0 und 9.

Funktion 61H wird beispielsweise von dem KOS-internen Kommando M aufgerufen.

## Open File

## Funktion 62H

---

Eingang:      (ix+0) - RMB5  
                            0 - aktuelles Working Directory  
                            1 - spezifiziertes Working Directory  
            HL        - DSB16

Ausgang:      A        - 0: Datei vorhanden und geöffnet  
                            FFH: Datei nicht vorhanden  
            HL        - DSB, falls <A>=0

Funktion zum Eröffnen von Dateien. Ist keine Mediennummer angegeben (Byte 0 des DSB = 0), so wird eine automatische Dateisuchsequenz auf allen aktiven Medien (beginnend mit dem Mastermedium) gestartet. Ist die Datei vorhanden, so wird der entsprechende Eintrag des Inhaltsverzeichnis in ein freies Speichersegment des Systems geladen. Nach der Rückkehr zum aufrufenden Programm zeigt das Registerpaar <HL> auf das erste Byte des 128 Byte DSB.

Byte 0 enthält hierbei in den Bits D0 bis D3 die Nummer des Mediums, auf dem die Datei eröffnet wurde. Die Bits D4 bis D7 werden von der Dateiverwaltung für interne Zwecke verwendet (Dateinummern). Byte 0 des DSB darf vom Anwenderprogramm nach dem Eröffnen nicht verändert werden.

Alle Dateiverwaltungsfunktionen mit Ausnahme von 'Search' und 'Search-next' erfordern das vorausgehende Eröffnen der gewünschten Datei. Da die Dateiverwaltung für jede geöffnete Datei Speicher belegt, ist am Ende einer Dateibearbeitung in jedem Fall die Funktion Close File erforderlich (Close after write bzw. Close after read).

Hat eine Datei ein Benutzerkennzeichen, so führt der Versuch, diese zu eröffnen zur Aufforderung, das Benutzerkennzeichen einzugeben. Die Datei ist logisch nicht vorhanden, falls das eingegebene Benutzerkennzeichen mit dem gespeicherten nicht übereinstimmt. In diesem Fall enthält (ix+5) den Wert 45H.

Nach dem Eröffnen einer Datei wird der Satzzähler (Byte 30/31) auf Null gestellt. Sequentielles Schreiben/Lesen ist deshalb ohne 'Manipulation' des Satzzählers möglich.

Ein Beispiel für die 'Open File' Funktion befindet sich im Anschluß unter 'Close File...'.

Close File after write  
Close File after read

Funktion 63H  
Funktion 6DH

Eingang:        (ix+0) - RMB5  
                              0 - aktuelles Working Directory  
                              1 - spezifiziertes Working Directory  
                  HL        - DSB

Ausgang:        A        - 0: Datei geschlossen  
                              FFH: Datei nicht geschlossen

Funktion zum Schließen von Dateien.

Bei dieser Gelegenheit erfolgt der Eintrag des DSB in das Inhaltsverzeichnis eines Medium. Außerdem wird die geschlossene Datei aus der Liste der geöffneten Dateien gestrichen und das durch den DSB belegte Speichersegment freigegeben. Die Funktion 'Close File after write' ist am Ende eines Dateischreibvorgangs (Funktion 67H) notwendig, nicht aber am Ende eines Lesevorgangs. Das logische Schließen einer Datei nach einem Lesevorgang erfolgt mit der Funktion 6DH ('Close File after read').

f62.example:

```

ld (ix+0),0           ;current 'Working Directroy'
ld (ix+1),62h
ld hl,dsb16           ;pointer to 16 byte DSB
rst 8
and a                 ;file found?
jr z,found            ;if found: <A>=0
.                     ;error routine: file not found
.
.
ret

found:
.
.
.
ret

dsb16:
defb 0                ;means master media
defm 'Myfile typ'
defb 0,0,0,0
```

Search File  
Search next File

Funktion 64H  
Funktion 65H

---

Eingang:	(ix+0)	-	RMB5
			0 - aktuelles Working Directory
			1 - spezifiziertes Working Directory
	(ix+2/3)	-	optionale Pufferadresse
	HL	-	DSB16
Ausgang:	A	-	0: Datei gefunden
			FFH: keine Datei (mehr) gefunden
	HL	-	DSB, falls <A>=0

Sucht die erste bzw. nächste Datei, die dem 16 Byte DSB beim Aufruf entspricht. Bei diesen Funktionen dient das Fragezeichen (Code: 3FH) als Universalbezeichner im Bereich der Bytes 1 bis 11 (Name/Typ) des DSB16.

Unter (ix+2/3) kann optional eine Pufferadresse für den DSB spezifiziert werden. Dazu muß (ix+2/3) ungleich Null sein. Ist keine Adresse angegeben, so wird automatisch ein für diese Funktionen reserviertes Speichersegment mit dem DSB beschrieben.

Beim Aufruf dieser Funktion von Hintergrundprogrammen (Background Tasks) ist es notwendig (ix+2/3) zu spezifizieren, da es sonst zu Konflikten kommen kann, falls im Vordergrund gleichzeitig eine der Funktionen 64H/65H aufgerufen wurde.

Die Funktionen 64H/65H berücksichtigen die Eigenschaften einer Datei (File Properties). Dies hat zur Folge, daß beim Aufruf einer dieser Funktionen Bedingungen bezüglich der Properties spezifiziert werden können, so daß nur Dateien mit bestimmten Properties oder Kombinationen von Properties gefunden werden. Die Bedingung wird in Form einer Maske in Byte 15 des DSB erwartet. Steht dort der Wert 0, so wird eine Datei nur dann gefunden, wenn sie nicht 'verborgen' ist. Durch das Setzen entsprechender Bits können Dateien, die nicht mindestens die in der Maske spezifizierten Properties haben, ausgeklammert werden.

## Delete File

Funktion 66H

---

Eingang:        (ix+0) - RMB5  
                      0 - aktuelles Working Directory  
                      1 - spezifiziertes Working Directory  
                  HL        - DSB16 falls Datei nicht 'offen'  
                              DSB, falls Datei 'offen'

Ausgang:        A        - 0: Datei gelöscht  
                          FFH: Operation nicht ausgeführt  
                              (Datei nicht vorhanden)

Funktion zum Löschen von Dateien. Ein vorheriges Eröffnen der zu löschenden Datei ist nicht erforderlich, aber zulässig. Falls die Datei bereits eröffnet war, so wird sie automatisch geschlossen.

Die Dateiverwaltung kennzeichnet gelöschte Dateien durch den Hexadezimalwert E5H in Byte 0 des DSB. Funktion 66H wird nicht ausgeführt bei schreib- und/oder löschgeschützten Dateien.





## Make new File

Funktion 69H

---

Eingang:        (ix+0) - RMB5  
                               0 - aktuelles Working Directory  
                               1 - spezifiziertes Working Directory  
                   HL        - DSB16

Ausgang:        A        - 0: Datei angelegt  
                               FFH: Operation nicht ausgeführt  
                                       (Medium voll)  
                   HL        - DSB

Funktion zur Generierung einer neuen Datei. Diese bleibt automatisch eröffnet.

Die Funktion 'Make new file' reserviert Platz im Inhaltsverzeichnis für die zu generierende Datei, wobei diese zunächst leer bleibt. In die Bytestellen 16/17 wird automatisch das Systemdatum eingetragen; in die Bytestellen 18/19 der Code des aktuellen (RMB5=0) bzw. spezifizierten (RMB5=1) Working Directory's.

Nach dieser Funktion kann eine Datei durch ein- oder mehrmalige Aufrufe der Funktion 68H (Write File Sequential) gefüllt werden.

f69.example:

```
ld (ix+0),0      ;current 'Working Directory'
ld (ix+1),69h
ld hl,my.dsb     ;pointer to DSB16
rst 8
and a            ;must be 0,if no error
jr nz,make.error
:
```

my.dsb:

```
defb 1           ;means: Media #0
defm 'Myfile typ'
defb 0,0,0,0
```

## Rename File

## Funktion 6AH

---

Eingang:        (ix+0) - RMB5  
                         0 - aktuelles Working Directory  
                         1 - spezifiziertes Working Directory  
                 HL    - DSB32

Ausgang:        A        - 0: Datei umbenannt  
                         FFH: Operation nicht ausgeführt  
                              (Datei nicht vorhanden)

Funktion zum Umbenennen einer Datei. Hierfür ist ein spezieller 32 Byte DSB erforderlich, der in den Bytes 1 bis 11 den alten Dateinamen und in den Bytes 17 bis 27 den neuen Dateinamen beschreibt. Die Bytes 12 bis 15, sowie 28 bis 31 sind irrelevant.

f6a.example:

```
ld (ix+0),0
ld (ix+1),6ah
ld hl,dsb.32
rst 8
and a
jr nz,rename.error
ret
```

```
dsb.32:  defb 0
         defm 'Oldfile typ'
         defw 0,0
         defb 0
         defm 'Newfile typ'
         defw 0,0
```

## Define Record Buffer

Funktion 6BH

---

Eingang:     (ix+0) - 0  
              HL     - Pufferadresse

Ausgang:     -

Funktion zur Bestimmung der Pufferadresse für nachfolgende sequentielle Schreib-/Lesefunktionen (Funktionen 67H/68H: Read/Write File Sequential).

Die Funktion ist redundant, wenn stattdessen die optionale Spezifikation der Pufferadresse unter (ix+2/3) im Aufrufvektor der Read/Write File Sequential Funktion verwendet wird.

## Return Master Media Number

Funktion 6CH

---

Eingang: (ix+0) - 0

Ausgang: A - Nummer des aktuellen Mastermediums

Funktion zur Abfrage der Nummer des derzeitigen Mastermediums. Der Wertebereich für diese Nummer liegt zwischen 0 und 9.

Das Mastermedium ist definitionsgemäß dasjenige Medium, das bei Dateireferenzen ohne Angabe einer Mediennummer adressiert wird.

## Return FM Table Pointers

Funktion 70H

Eingang: (ix+0) - 0

Ausgang: HL - Tabellenzeiger

Diese Funktion liefert einen Zeiger auf folgende Pointertabelle (jeweils 2 Byte) der Dateiverwaltung.

a) Zeiger auf die Tabelle der DSB der momentan geöffneten Dateien. Die Einträge sind nach der logischen Dateinummer geordnet. Hat ein Eintrag den Wert 0, so ist die entsprechende logische Dateinummer frei. Es sind maximal 16 Einträge zu je 2 Byte (= 16 geöffnete Dateien) möglich.

b) Zeiger auf die Tabelle der Anfangsadressen der Medienbelegungspläne. Die Einträge sind nach folgendem Schema angeordnet:

1. Eintrag: reserviert (2 Byte)
2. Eintrag: Adresse des Belegungsplans von Medium 0 (2 Byte)
3. Eintrag: Adresse des Belegungsplans von Medium 1 (2 Byte)
- .
- .
11. Eintrag: Adresse des Belegungsplans von Medium 9 (2 Byte)

Die ersten 8 Byte des Belegungsplans enthalten den Namen der Datei 'Inhaltsverzeichnis' des entsprechenden Mediums. Pro verwendeter Blocknummer wird im Belegungsplan ein Bit gesetzt. Die Länge des Belegungsplans errechnet sich aus der Kapazität des Mediums.

c) Zeiger auf die Tabelle, welche die Anzahl der logischen Sätze (1 Satz = 128 Byte) eines Speichermediums enthält, also dessen Kapazität bestimmt. Hierfür werden jeweils 3 Byte benötigt und nach folgendem Schema eingetragen:

1. Eintrag: reserviert (3 Byte)
2. Eintrag: Kapazität Medium 0 (3 Byte)
3. Eintrag: Kapazität Medium 1 (3 Byte)
- .
- .
11. Eintrag: Kapazität Medium 9 (3 Byte)

d) Zeiger auf die Tabelle der Identifikationsbytes der Speichermedien (siehe Funktion 80H: Return Media Identification).

- |                                      |          |
|--------------------------------------|----------|
| 1. Eintrag: reserviert               | (1 Byte) |
| 2. Eintrag: Identifikation Medium 0  | (1 Byte) |
| .                                    |          |
| .                                    |          |
| 11. Eintrag: Identifikation Medium 9 | (1 Byte) |

e) Zeiger auf die Tabelle, die die Satznummer des DSB der momentan offenen Datei enthält. Die Einträge sind nach der logischen Dateinummer geordnet und benötigen jeweils 3 Byte.

**Alle Tabellen dürfen von Anwenderprogrammen nicht verändert werden!**

## Define Media Parameter

## Funktion 71H

---

Eingang:     A       - Mediennummer  
              HL      - Zeiger auf Parameterblock

Ausgang:     -

Funktion zur Definition der spezifischen Parameter eines Mediums, welche sind:

      cd - Treiberidentifikation  
maxrec - Anzahl der verfügbaren log. Sätze (128 Byte) eines Mediums.

Beim Aufruf der Funktion 71H zeigt <HL> auf einen 4 Byte-Parameterblock. Register <A> enthält die Nummer des Mediums, dem diese Parameter zugeordnet werden sollen.

Parameterblock:     cd           - Treiberidentifikation  
                                  maxrec   - Anzahl der log. Sätze auf dem Medium (3 Byte).

Siehe hierzu auch die Funktion 80H (Return Media Driver Identification).

Read File Random  
Write File Random

Funktion 77H  
Funktion 78H

---

Eingang:        (ix+0)        - RMB4: siehe Abschnitt 5.3.3 (Record Locking)  
                                      RMB6: 'Write Protection' wird  
  ignoriert  
                      (ix+2/3) - Pufferadresse  
                      HL        - DSB

Ausgang:        A        - 0: Satz gelesen/geschrieben  
                                      FFH: Operation nicht ausgeführt  
  (Dateiende überschritten)

Funktion zum wahlfreien Schreib-/Lesezugriff auf einen beliebigen Satz einer Datei. Hierzu muß der Satzzähler in den Bytestellen 30/31 des DSB auf den gewünschten Wert gesetzt werden. Die Spezifikation der Pufferadresse unter (ix+2/3) ist obligatorisch.

Nach jedem Aufruf der Funktion 77H/78H wird der Satzzähler automatisch um eins inkrementiert. Sequentielles Lesen/Schreiben ist deshalb auch mit diesen Funktionen möglich.

Die Funktion 78H wird nicht ausgeführt bei schreibgeschützten Dateien, es sei denn RMB6 ist gesetzt.



## Position End of File

Funktion 79H

---

Eingang:        (ix+0) - 0  
                 HL     - DSB

Ausgang:        A        - Nummer der Dateierweiterung  
                 HL        - nächste Satznummer innerhalb der letzten  
                                 Dateierweiterung

Diese Funktion setzt den Extension- und Recordcounter der spezifizierten Datei auf das Dateiende. Die Datei muß zuvor eröffnet werden. Nach dieser Funktion kann eine Datei unmittelbar mit der Funktion 68H (Write File Sequential) erweitert werden.

## 5.3.3 Dateiverwaltungsfunktionen in KOBUS-Slaves

## a) Blockverwaltung

KOBUS-Slaves haben zwar Zugriff auf den zentralen Massenspeicher, verwalten diesen aber nicht selbst. Es wurden deshalb speziell für diesen Zweck die Funktionen 72H und 89H definiert. Diese sind hier der Vollständigkeit halber angegeben, haben aber absolut keine Bedeutung in Benutzerprogrammen.

72/89	! (IX+0)	! (IX+2)-(IX+4)	! (IX+5)	! A	! HL	! Funktion
Eingang	! Medium	! Recordnummer	! -	! 1	! DSB	! Directory
	!	!	!	!	!	! Record belegen
	! Medium	! -	! -	! 2	! -	! Block belegen
	! Medium	! -	! -	! 3	! Block	! Block frei-
	!	!	!	!	!	! geben
Ausgang	! -	! -	! xx/43	! 0/FF	! -	! 1
	! -	! -	! xx	! 0/FF	! Block	! 2
	! -	! -	! xx	! -	! #	! 3

A = 0 : kein Fehler  
 A = FF : Fehler, siehe (IX+5)  
 (IX+5) = 43 : Directory-Record ist bereits belegt  
 sonst : siehe KOS-Fehlermeldungen  
 (alle Zahlen hexadezimal)

## b) Mailbox

Master und KOBUS-Slaves können Speicherbereiche des Masters über \$PUB/\$SLAV lesen und schreiben. Blöcke von 128 Bytes werden in/aus einem von acht 'Registern' übertragen. Im ersten Byte eines lokalen Puffers wird die 'Mailbox-Registeradresse' spezifiziert. Anschließend daran folgen 128 Byte, die gesendet oder empfangen werden.

89 !	(IX+0) !	A !	HL !	Funktion
<hr/>				
! RMB0...3	! 8	! Puffer-	!	Auslesen eines der acht Mailbox-Register.
! (log.	!	! zeiger	!	Im ersten Byte des Puffers (Pufferzeiger
! Kanal-	!	!	!	+ 0) steht die 'Registeradresse'. Der Puf-
! nummer)	!	!	!	fer wird mit dem Inhalt des 'Registers'
!	!	!	!	gefüllt (Pufferzeiger +1 bis Puffer-
!	!	!	!	zeiger +128).
<hr/>				
! RMB0...3	! 9	! Puffer-	!	Schreiben in eines der acht Mailbox-
!	!	! zeiger	!	Register. Pufferzeiger + 0 enthält die
!	!	!	!	'Registeradresse' 0...7. Dann folgen
!	!	!	!	128 Bytes, die gesendet werden.
<hr/>				

## c) Record Locking

KOS unterstützt das automatische Sperren (Locking) von einzelnen Sätzen (Records) von 128 Byte Länge einer Datei, um Konflikte zu vermeiden, die beim gleichzeitigen Zugriff auf denselben Datensatz durch mehrere Benutzer entstehen. Dieses Verfahren garantiert aus der Sicht des Anwenderprogramms die Kompatibilität zwischen Ein- und Mehrplatz-Systemen, sofern das Anwenderprogramm die Satzlänge 128 Byte verwendet und die folgenden Hinweise in der Anwendungsprogrammierung beachtet werden.

**Verfahren**

Der Automatismus des 'Record Locking' tritt nur bei Dateien in Kraft, die mit dem Property Bit 'R' entsprechend gekennzeichnet sind (siehe DEFP-Kommando).

Liest der Benutzer #X einen Satz einer solchen Datei, so bleibt dieser Satz für alle anderen Benutzer solange gesperrt, bis Benutzer #X entweder einen anderen Satz liest, oder den gesperrten Satz zurückschreibt.

KOS wartet normalerweise, bis ein gesperrter Satz verfügbar ist. Ein Anwenderprogramm hat jedoch die Möglichkeit über Bit 4 von (IX+0) das Betriebssystem zu veranlassen, im Falle eines gesperrten Satzes sofort in das aufrufende Programm zurückzukehren. In einem solchen Fall enthält (IX+5) den Fehlercode 8F (hexadezimal).

Bit 4 von (IX+0) wird so interpretiert:

- 0 : KOS wartet bis ein Satz verfügbar ist
- 1 : KOS kehrt sofort in das Anwenderprogramm zurück

Dieses Verfahren gilt für folgende Funktionen:

Funktionsnummer	Funktion
67H	Read File Sequential
68H	Write File Sequential
77H	Read File Random
78H	Write File Random

## 5.4 Allgemeine Systemfunktionen

Unter der Rubrik 'Allgemeine Systemfunktionen' sind Dienste allgemeiner Art, sowie Funktionen zur Speicher- und Taskverwaltung enthalten.

Der IX-Vektor umfaßt grundsätzlich 8 Byte:

(IX+0)	RMB0...3: 0
	RMB4: funktionsabhängig (*)
	RMB5...7: 0
(IX+1)	Funktionscode: 00H ... 3FH
(IX+2)	nicht verwendet (*)
(IX+3)	nicht verwendet (*)
(IX+4)	nicht verwendet (*)
(IX+5)	Return Code
(IX+6/7)	optionale Rückkehradresse im Fehlerfall

(\*) Diese Eingangsparameter müssen immer den Wert Null enthalten, falls sie die aufgerufene Funktion nicht auswertet. Dies sichert Aufwärtskompatibilität zu eventuellen zukünftigen Erweiterungen von KOS.

Derzeit sind die Funktionen 01H bis 08H implementiert. Bei allen anderen Funktionscodes im Bereich zwischen 0 und 3FH antwortet KOS mit dem 'Return Code' 81H.

### Tabelle der Allgemeinen Systemfunktionen

Request Code: Funktion

01H:	KOS Command Interpreter
02H:	FCB Generator
03H:	Print Standard Error Message
04H:	Memory Manager
05H:	Return KOS Table Pointers
06H:	Activate Task
07H:	Change Task Status
08H:	Transfer Parameter to Task

## KOS Command Interpreter

## Funktion 01H

---

Eingang:        (ix+0) - 0  
                 HL     - Zeiger auf Kommando-String

Ausgang:        -

Funktion zum Aufruf des Kommando-Interpreters von KOS.  
Registerpaar <HL> zeigt auf eine Folge von ASCII-Zeichen, welche  
mit 0 (binär!) abgeschlossen sein muß.

Der Kommando-String kann mehrere KOS-Kommandos, getrennt durch  
das Semicolon enthalten.

f01.example:

```
ld (ix+0),0
ld (ix+1),1
ld hl,cmd.string ;define string pointer
rst 8
ret
```

cmd.string:

```
defm 'STATUS;IL P=*;MAP'
defb 0
```

Mögliche Fehlermeldung des Betriebssystems unter (ix+5):

89H - Es war kein Speicher für den Stack mehr frei. Funktion 01H  
ist reentrant. Sie belegt pro Aufruf mindestens 3 Speicher-  
segmente von je 128 Byte.

## FCB Generator

## Funktion 02H

Eingang:        (ix+0) - 0  
                  HL     - Zeiger auf Speicherbereich für DSB  
                  DE     - Zeiger auf ASCII-String

Ausgang:        DE     - Zeiger auf den nächsten Parameter  
                               des ASCII-Strings

Funktion zur Aufbereitung einer beliebigen Zeichenkette zu einem Dateispezifikationsblock (File Control Block).

Diese Funktion ist besonders geeignet für Programme, die Dateinamen über Eingaben erhalten und in DSB's umwandeln müssen.

Beim Aufruf der Funktion zeigt <DE> auf die Zeichenkette, <HL> definiert einen Speicherbereich von 16 Byte, wo der DSB aufgebaut wird. Nur der erste Parameter einer aus eventuell mehreren Parametern bestehenden Zeichenkette wird verarbeitet. Vor und nach einem Parameter dürfen beliebig viele Trennzeichen (Blank, Tab) stehen.

Folgendes Beispiel soll die Arbeitsweise von Funktion 02H verdeutlichen:

Eingang: <DE> ---> /wdir/1:filename.typ parameter

Ausgang: <HL> ---> Byte 0 1 2 3 4 5 6 7 8 9 A B C D E F  
                               s f i l e n a m e t y p o w w o

mit s: DSB-Statusbyte

ww: Code des spezifizierten Working  
      Directory's /wdir/

<DE> ---> zeigt auf das erste Zeichen von  
          'parameter' oder das Ende des Strings, falls kein  
          zweiter Parameter 'parameter' vorhanden ist.

## a) Das DSB-Statusbyte 's'

Das DSB-Statusbyte in Stelle 0 des aufgebauten DSB liefert nähere Informationen über die Art der verarbeiteten Zeichenkette:

D7	D6	D5	D4	D3	D2	D1	D0	Bedeutung
-----	-----	-----	-----	-----	0	0	0	keine Mediennummer angegeben
-----	-----	-----	-----	-----	0	0	I	Medium 0
-----	-----	-----	-----	-----	0	I	0	Medium 1
-----	-----	-----	-----	-----	0	I	I	Medium 2
-----	-----	-----	-----	-----	I	0	0	Medium 3
-----	-----	-----	-----	-----	I	0	I	Medium 4
-----	-----	-----	-----	-----	I	I	0	Medium 5
-----	-----	-----	-----	-----	I	I	I	Mediennummer > 5
-----	-----	0	0	-----	-----	-----	-----	kein Name/Typ angegeben
-----	-----	0	I	-----	-----	-----	-----	eindeutiger Name/Typ
-----	-----	I	0	-----	-----	-----	-----	mehrdeutiger Name/Typ (?,*)
-----	-----	I	I	-----	-----	-----	-----	unzulässige Zeichen in Name/Typ
--	0	0	-----	-----	-----	-----	-----	nicht verwendet
0	-----	-----	-----	-----	-----	-----	-----	'normaler' Name
I	-----	-----	-----	-----	-----	-----	-----	Treiberkennzeichnung (dem Parameter war ein \$- Zeichen vorangestellt)

Hinweis: Die Bitstellen D3 bis D7 müssen vor dem Aufruf einer Dateiverwaltungsfunktion ausgeblendet werden.

## b) Working Directory Code 'ww'

Das zwischen Schrägstrichen optional spezifizierte Working Directory (maximal 15 beliebige Zeichen) wird in einen 2 Byte Code umgerechnet und in den Bytestellen 13 und 14 abgelegt. Dort steht 0, falls kein Working Directory angegeben wurde.

Im folgenden Beispiel wird zunächst über die E/A-Funktion 85H (String input) ein String eingelesen und anschließend zu einem DSB aufbereitet. Danach wird die Datei eröffnet.

f02.example:

```

ld hl,string.buffer      ;define input buffer
ld (ix+1),85h            ;define string length
ld a,max.length          ;system call 85H returns:
rst 8                    ;A: string length
                           ;HL: pointer to first string
                           ;element

and a
jr z,f02.example         ;try again, just a 'RETURN'
                           ;was entered

ld (ix+1), 2
ex de,hl                 ;de ---> string pointer
ld hl,fcf.buffer         ;hl ---> buffer for fcf
rst 8                    ;system call 02H
ld a,(hl)                ;read status byte
and 7                    ;here you may check the
ld (hl),a                ;statusbyte for a correct
                           ;syntax
ld (ix+1), 62h           ;Open File
set 5,(ix+0)             ;only if you specify a
ld a,(fcf.buffer+13)     ;working directory
ld (ix+8),a
ld a,(fcf.buffer+14)
ld (ix+9),a
rst 8                    ;system call 62H
ret                      ;returns <A>=0 if file found

```

```

string.buffer:
  defs max.length+3

```

```

fcf.buffer:
  defs 16

```

Funktion 02H meldet immer 0 unter (ix+5).



## Print Standard Error Message

## Funktion 03H

Eingang:        (ix+0) - 0  
                  A        - Error Code

Ausgang:        -

Funktion zur Ausgabe einer Standard Fehlermeldung entsprechend des Fehlercodes in Register <A>.

Damit ist Anwender- und Systemprogrammen die Möglichkeit gegeben, einheitliche Fehlermeldungen zu verwenden. Der Wertebereich von <A> liegt zwischen 0 und 0FH entsprechend der Fehlercodes 80H bis 8FH unter (ix+5). <A> darf auch den Fehlercode selbst enthalten, da Funktion 03H das höherwertige Bit ausblendet.

Funktion 03H ruft den Ausgabekanal 0-3 auf, der normalerweise dem Treiber \$KSM (KOS System Messages) zugeordnet ist. Dieser Treiber druckt anschließend die eigentliche Fehlermeldung aus. Es gilt folgende Zuordnung:

<A>:        Fehlermeldung  
 -----

80H:	'unzulässiger Funktionsparameter'
81H:	'unzulässiger Funktionsparameter'
82H:	'E/A-Kanal nicht bereit'
83H:	'E/A-Kanal nicht aktiviert'
84H:	'Übertragungsfehler'
85H:	'Übertragungsfehler'
86H:	'Medium ist mechanisch schreibgeschützt'
87H:	'Software Timeout'; wird z.B. von \$WINx erzeugt, falls kein Controller angeschlossen ist
88H:	'Medium belegt'
89H:	'Speicher bereits belegt'
8AH:	'Zu viele Dateien offen (>16)'
8BH:	'Directory Formatfehler'
8CH:	'Datei schreibgeschützt'
8DH:	'Datei löschgeschützt'
8EH:	'Medium inkonsistent'
8FH:	'Satz gesperrt'

## Memory Manager

## Funktion 04H

Eingang: (ix+0) - 0  
 A - Sekundärer Funktionscode (SFC)  
     1: Speicher belegen (allocate)  
     2: Speicher freigeben (deallocate)  
     3: Speicher suchen und belegen  
 HL - Segment Adresse, falls SFC=0 oder 1  
 DE - Anzahl der Segmente

Ausgang: C - Carry Flag  
     0: Speicher wurde belegt  
     1: Speicher war bereits belegt  
 HL - Segment Anfangsadresse, falls SFC=3

Funktion zum Aufruf der Speicherverwaltung von KOS. Diese teilt den Anwenderspeicher (Bank 0/64 kByte) in 16x32 Segmente zu je 128 Byte ein.

In Abhängigkeit des sekundären Funktionscodes in <A> verzweigt die Speicherverwaltung zu folgenden Aktivitäten:

- <A> = 1: Belegung von <DE> Segmenten ab der Adresse <HL>
- <A> = 2: Freigabe von <DE> Segmenten ab der Adresse <HL>
- <A> = 3: Belegung von <DE> Segmenten im höchstmöglichen Speicherbereich. Nach Rückkehr zeigt <HL> auf die Adresse des ersten belegten Segments.

Speicher, der bereits belegt ist, kann nicht erneut belegt werden. Funktion 04H antwortet mit dem Fehlercode 89H unter (ix+5), falls sich ein Belegungskonflikt ergibt. In diesem Fall ist auch das Carry-Flag der CPU gesetzt.

f04.example:

```
ld (ix+0),0
ld (ix+1),4
ld hl,1000h
ld de,16
ld a,1                ;allocate 16 segments
rst 8                ;beginning at adress 1000h
jp c,error           ;if allocation not possible
ld de,10             ;search and allocate
ld a,3               ;10 segments in the highest
rst 8                ;possible memory area
jp c,error           ;if allocation not possible
ret
```

## Return KOS Table Pointers

Funktion 05H

---

Eingang: (ix+0) - 0

Ausgang: HL - Zeiger auf eine 'Pointertabelle'

Diese Funktion liefert einen Zeiger auf folgende Pointertabelle (jeweils 2 Byte) des Betriebssystems

- a) Zeiger auf den Eingabepuffer des KOS Kommandointerpreters (x)
- b) Zeiger auf die Tabelle der Speicherverwaltung; umfaßt 64 Byte
- c) Zeiger auf die Tabelle der Taskverwaltung; umfaßt 10 x 16 Byte.

Funktion 05H liefert immer 0 unter (ix+5).

(x) Dieser Wert hat in KOS6 keine Gültigkeit

**Activate Task****Funktion 06H**

Eingang:        (ix+0) - 0  
                 HL        - TCB (Task Control Block)

Ausgang:        A        - Tasknummer

Funktion zur Aktivierung einer Hintergrund Task. Der 'Task Scheduler' von KOS arbeitet im 20ms Zeitraster. Bis zu 10 Tasks können gleichzeitig aktiv sein. Beim Versuch, mehr als 10 Tasks zu aktivieren, antwortet Funktion 6 mit dem Return Code 89H unter (ix+5).

Eine Task ist eindeutig beschrieben durch den Task Control Block (TCB). Dieser besteht aus 16 Byte, die bei der Aktivierung einer Task in die Tabelle des 'Task Scheduler' geschrieben werden. Die Bedeutung der einzelnen Bytes ist wie folgt:

Byte 0:        Statusbyte

Bit 0 - 0: Kein gültiger Tabelleneintrag  
          1: gültiger Tabelleneintrag

Bit 1 - 1: Task ist in Bearbeitung

Bit 2 - 0: Task wird nur einmal ausgeführt (Monotask)  
          1: Task wird periodisch ausgeführt (Autotask)

Bit 3 - 0: nicht verwendet

Bit 4 - 0:        "        "

Bit 5 - 0:        "        "

Bit 6 - 0:        "        "

Bit 7 - 1: Task ist temporär nicht aktiv.

Byte 1:        AMS (Allocated Memory Segments)

Bestimmt die Anzahl der von der Task belegten Speichersegmente ab der Adresse TLA (Bytes 14/15). Bei der Deaktivierung einer Task werden ab Adresse <TLA> <AMS> Segmente freigegeben.

Byte 2/3:       PCNT (Preset Counter)

Dieser 16 Bit Wert bestimmt die Anzahl der 20ms Perioden, nach der eine Task ausgeführt wird. Demzufolge kann eine Task im Abstand von nx20ms mit n=1...65536 ausgeführt werden.

Ist <PCNT> beispielsweise 50, so wird die entsprechende Task im Sekundentakt angestoßen (50x20ms=1sec).

- Byte 4/5: DCNT (Down Counter)  
Der 'Task Scheduler' verwendet diese beiden Speicherstellen als Rückwärtszähler, beginnend vom Wert PCNT. Eine Task wird nach dem Erreichen des Zählerstandes 0 ausgeführt. Anschließend erhält DCNT den Wert von PCNT, falls Bit 2 des Task Statusbytes (Byte 0) gesetzt ist. Andernfalls wird die Task automatisch deaktiviert.
- Byte 6/7: TEP (Task Entry Point)  
Einsprungpunkt einer Task
- Byte 8-13: TNF (Task Name Field)  
Name einer Task, bestehend aus 6 ASCII-Zeichen mit Blanks auf den nicht besetzten Stellen.
- Byte 14/15: TLA (Task Load Address)  
Dies ist gewöhnlich die Startadresse des Programmes, das die eigentliche Task definiert. TLA wird zur Speicherverwaltung benötigt und bestimmt die Anfangsadresse der AMS-Speichersegmente, die bei der Deaktivierung einer Task automatisch deallokiert werden.

f06.example:

```
- comment #
    This program allows to count the execution time of up to
    256 different tasks, subroutines etc. In order to identify
    any particular task a number has to be written into the
    memory location 40h.
    #

    which.task equ 40h

    ld hl,0
    ld (which.task),hl
    ld ix,kos.vector
    ld hl,tcb1      ;task control block
    rst 8
    ld (ix+5),44h   ;memory remains allocated
    ret

nrofms equ (end.tcnt-entry.point)/128+1

kos.vector:
    defb 0,6,0,0,0,0,0,0

tcb1:  defb 5      ;task status byte
        defb nrofms ;number of allocated memory segments
        defw 1     ;preset counter
        defw 1     ;down counter
        defw tep   ;task entry point
        defm 'Tcount'
        -
        defw entry.point
```

tep:

```

and a
jp nz,set.counters
ld a,(which.task)
ld e,a
ld d,0
ld hl,base.counter
add hl,de
add hl,de
ld a,(hl)          ;low byte of counter
add a,1
daa
ld (hl),a
inc hl
ld a,0
adc a,(hl)
daa
ld (hl),a
ret

```

```

set.counters          ;called via function 8
ld hl,base.counter
ld de,base.counter+1
ld bc,256*2
ld (hl),0
ldir
ret

```

-sall

```

base.counter:
rept 256
defw 0
endm

```

```

end.tcmt:             defb 0ffh

```

#### Hinweis:

1. Ist das Zeitintervall einer Auto-TASK größer als 255 Zeiteinheiten (a 20 ms), also größer als 5,1 sec., so kann beim Einsprung in diese Task nicht mehr unterschieden werden, ob es sich um einen zyklischen Start oder um einen Parameterstart handelt, d.h. es kann nicht mehr zwischen 'Parameterübergabe' und 'Taskausführung' unterschieden werden, da das Register 'A' zu diesem Zweck den Wert '0' oder '1' haben müßte, wogegen das Register 'A' immer einen Wert <0> enthält, wenn der PCNT-Wert von 255 überschritten wird. Dies hat zur Folge, daß die Unterprogramme für den zyklischen Betrieb nicht mehr durchlaufen werden, sondern nur noch die Unterprogramme für den Parameterstart. Die vorstehende Restriktion betrifft KOS 5.44/5.54.
2. Bei Autotasks, welche länger als 20 ms laufen, wird der Abstand zwischen zwei zyklischen Läufen um die Laufzeit der Task verlängert, d.h. bei Tasks mit Ausführungszeiten > 20 ms findet eine Zeitverschiebung statt, da der Zähler DCNT erst nach Beendigung der Task neu geladen wird. Diese Restriktion betrifft KOS 5.44/5.54.

**Change Task Status****Funktion 07H**

Eingang: (ix+0) - RMB4

0:	A	- Tasknummer (0...9)
	L	- Neues Statusbyte
1:	A	- Neues Statusbyte
	HL	- Taskname

Ausgang: -

Funktion zur Änderung des Statusbytes einer Task. Dies ist sowohl über die Tasknummer (RMB4=0) als auch den Tasknamen (RMB4=1) möglich. Funktion 7 antwortet mit dem Fehlercode 82H, falls die spezifizierte Task nicht aktiviert war. Unter anderem dient diese Funktion zur temporären oder endgültigen Deaktivierung einer Task.

## a) temporäre Deaktivierung

RMB4=0    <A> = 0..9 (Tasknummer)  
           <L> = 85H bzw. 81H

RMB4=1    <A> = 85H bzw. 81H  
           <HL> = Taskname

## b) endgültige Deaktivierung

RMB4=0    <A> = 0...9 (Tasknummer)  
           <L> = 0

RMB4=1    <A> = 0  
           <HL> = Taskname

Der Taskname besteht hier aus 6 Zeichen, nicht besetzte Stellen sind mit Blank (20H) aufgefüllt.

**Hinweis:**

Für die Funktion 7 (Change Task Status) gelten folgende Einschränkungen:

- a) Der Aufruf mit Reg. <A> als Tasknummer (RMB4=0) führt nicht zum Fehlercode 82H, falls die spezifizierte Task nicht aktiv war. Stattdessen wird in der Task Liste eine Task undefinierten Namens geführt.  
Diese Einschränkung gilt für KOS5.44/5.54.
- b) Funktion 7 mit RMB4=1 (A = Statusbyte, HL ---> Taskname) ist in KOS 5.xx nicht implementiert.

**Transfer Parameter to Task****Funktion 08H**

Eingang: (ix+0) - 0  
HL - Zeiger auf den G-stelligen Tasknamen  
DE - Parameter

Ausgang: -

Funktion zur Übertragung beliebiger Parameter an eine Task. Die Task ist durch ihren Namen in <HL> bestimmt.

Die Interpretation des zu übertragenden Parameters ist abhängig von der jeweiligen Task. So kann in <DE> neben 8 oder 16 Bit Werten auch ein Zeiger auf größere Informationseinheiten übertragen werden.

KOS verwendet für diese Funktion den Task Entry point (TEP), der auch für den 'Task Scheduler' als Einsprungpunkt dient. Zur Unterscheidung dient das Register <A>:

A = 0: Einsprung durch den 'Task Scheduler'  
A = 1: Einsprung durch Funktion 8

Funktion 8 antwortet mit dem Fehlercode 82H, falls die spezifizierte Task nicht aktiviert ist.

Der Taskname besteht hier aus 6 Zeichen, nicht besetzte Stellen sind mit Blank (20H) aufgefüllt.

**Hinweis:**

Beim Einsprung in eine Task über 'TEP' enthält das Register <A> immer einen Wert ungleich 0, falls die Zeitkonstante der Task größer 255 x 20 ms (=5.1s) ist. Dadurch kann zwischen der Taskausführung (<A>=0) und Funktion 8 (<A>=1) nicht mehr unterschieden werden. Diese Einschränkung trifft zu für KOS 5.44/5.54.



## Multitasking in KOS

KOS bietet die Möglichkeit, bis zu 10 Hintergrund-Tasks zu verwalten und quasi gleichzeitig zu einem Vordergrundprogramm zu bearbeiten. Hintergrund-Tasks nutzen die freien Systemkapazitäten aus.

Das Laden einer Task hängt von der gewählten Organisation ab.

Tasks, die aus einem **einzigem** Programmodul ohne Externbeziehungen bestehen, können als OBJ-Dateien abgelegt werden und durch das RLOAD-Kommando geladen werden. Am Programmstart muß ein Sprungbefehl zur Task-Initialisierungsroutine stehen, die folgende Aktivitäten beinhaltet:

- **ACTIVATE TASK** (KOS-Funktion 06H)  
Der Task Control Block muß die Adresse des ersten belegten Segmenten und deren Anzahl enthalten, siehe Beispiel unter KOS-Funktion 06H.
- Einmalige Operationen, wie z.B. Eröffnen von TASK-spezifischen Dateien, Gerätezuweisungen etc.,
- Rückkehr zu KOS mit  $(IX+5)=44H$  zur Reservierung des belegten Speichers.

Größere Tasks, die **aus mehr als einem Modul** gelinkt werden, müssen auf einer festen Adresse beginnen, die möglichst hoch sein sollte. Ein typischer LINK-Aufruf ist z.B.:

```
LINK APPLIC/N,INIT/P:8000,TASK1,TASK2/E:INIT<---
```

Diese Task wird geladen und gestartet durch

```
APPLIC<---
```

Mit dieser Methode können Tasks auch in höheren Programmiersprachen erstellt werden, **sofern keine Run-time-Module** angesprochen werden. Dazu muß die Initialisierungsroutine in Assembler erstellt werden. Die Task selbst wird als Prozedur geschrieben, z.B. in Fortran "PROCEDURE BGTASK". Diese Prozedur ist ein EXTERNAL zur Initialisierungsroutine. Tasks, die in höheren Programmiersprachen erstellt werden, können **nicht** mit der KOS-Funktion 08H Parameter erhalten. Assembler-Tasks dagegen können Parameter übernehmen und ihrerseits Unterprogramme aus z.B. FORTRAN aufrufen:

```
TASK:
    CP 0                ; Parameter da?
    JR z,TASK1          ; nein, zur normalen Task
    EX DE,HL            ; Parameteradresse ins HL
    CALL PARAM          ; FORTRAN-Auswertung
    RET                ; Ende
TASK1:
    CALL BGTASK          ; normale Background-Task
    RET
```

Die Prozedur PARAM ist mit genau einem Parameter zu deklarieren, der dem KOS-Parameter entspricht, z.B. einem Integerfeld:

```
PROCEDURE PARAM (IPARAM)  
  DIMENSION IPARAM (10)
```

Zu beachtende Randbedingungen bei Tasks sind:

- \$KEY ist nicht zu verwenden (keine Reentranz).
- Busy-Situationen von Treibern und KOS-Funktionen müssen berücksichtigt werden (Error 42H)
- Die bei Multi-Tasking üblichen Vorsichtsmaßnahmen müssen beachtet werden. Insbesondere gilt dies für den Zugriff auf alle Betriebsmittel, wie Medien, Dateien, Peripheriegeräte. Programme, die gleiche Betriebsmittel verwenden, müssen aufeinander abgestimmt werden und sich synchronisieren.

1

2

3

4

1

2

3

4

## **Kontron PSI - KOS-Utilities**

### **Technische Beschreibung**

Version: 4.33/5.44/5.54/6.05

Stand: Dezember 1984

Dieser Teil des Handbuches beschreibt die mit Kontron PSI-Systemen ausgelieferten KOS-Hilfsprogramme (= "Utilities") für den Systemprogrammierer: das Grafikpaket und Basistreiber. Beispielprogramme und die Beschreibung der Hardwareprogramme schließen sich an.



**Inhaltverzeichnis**

1. Übersicht und Inhalt der Utility-Diskette
2. Kontron PSI-Grafikpaket
  - 2.1 Einführung
  - 2.2 Programmieren mit dem Kontron PSI-Grafikpaket
  - 2.3 Einbau der Grafik-Moduln in Anwenderprogramme
    - 2.3.1 Grafik in ASSEMBLER-Programmen
    - 2.3.2 Grafik in FORTRAN- und BASIC80 Programmen
    - 2.3.3 Grafikerweiterungen unter KOS6
3. Ein-/Ausgabetreiber
  - 3.1 Aufbau von Ein-/Ausgabetreibern
    - 3.1.1 Basisroutinen des logischen Ein-/Ausgabetreibers LIOD
    - 3.1.2 Filtermodule xxx.FIL
    - 3.1.3 Hardware-Schnittstellenmodul HOD
    - 3.1.4 Assemblieren und Linken eines Treiberprogramms
  - 3.2 Konfigurierte Drucker-Treiber
    - 3.2.1 Treiber für OKI Microline
    - 3.2.2 Treiber für DAISY-M45
    - 3.2.3 Treiber für Olympia ESW103
    - 3.2.4 Universelle Druckertreiber
  - 3.3 Allgemeine serielle Treiber
  - 3.4 Allgemeiner paralleler Treiber
  - 3.5 Virtuelles Medium (\$VMED)
  - 3.6 Medientreiber \$BMD1, \$BMD2
  - 3.7 Ausgabetreiber \$MED0
4. Treiber und Formatierprogramme
  - 4.1 Programme und Treiber für Diskettenstationen
    - 4.1.1 Umsetztreiber für KOS 3.2-formatierte Disketten (KOS 4/5)
    - 4.1.2 Treiber für integrierte Diskettenlaufwerke (KOS 4/5)
    - 4.1.3 Kompatibilität von KOS4/5-Disketten zu KOS6-Disketten
    - 4.1.4 Treiber und Formatprogramm für externe 8"-Diskettenlaufwerke
  - 4.2 Treiber und Formatprogramm für externe Plattenspeicher
  - 4.3 Harddisk-Formatierprogramm FORMATW
  - 4.4 Harddisk-Formatierprogramm WFD
  - 4.5 Wechselplattenformatierungsprogramm FORMATZ
  - 4.6 Formatieren von Fest- und Wechselplatten
5. Softkey-Utility
6. Umsetztreiber für CP/M-Aufrufe \$CPM
7. Beispielprogramme
  - 7.1 INFO-Kommando
  - 7.2 BASIC-Demo-Programme
  - 7.3 Hintergrund-Programm TIME
8. Testprogramme
  - 8.1 Test der Peripherie-Bausteine
  - 8.2 Speichertest
  - 8.3 Laufwerk/Diskettentest
  - 8.4 PROM-residenter Test-Debugger
    - 8.4.1 Aufrufen und Verlassen des Test-Debuggers
    - 8.4.2 Speichertest
    - 8.4.3 Floppy-Disk Laufwerktest
  - 8.5 Messungen von Programmlaufzeit: TCNT/PCNT (KOS6)
9. Muster eines Ausgabe-Treibers
  - 9.1 Benutzer-spezifische Anpassung eines seriellen Treibers
10. Hilfsprogramme
  - 10.1 CONED
  - 10.2 WSEDT
  - 10.3 PT





## 1. Übersicht

Die KOS-Utility-Diskette enthält folgende Programme:

LIOD	SRC	Logischer E/A-Treiber
HOD	SRC	Hardware-Schnittstellen Modul
PSIX	SRC	Serieller Treiber, Kanal A/B
PSIA	OBJ	
PSIB	OBJ	
SIOX	SRC	Serieller Treiber, Kanal A/B
SIOA	OBJ	
SIOB	OBJ	
SIO	SRC	Serieller Treiber (KOS6), Kanal A,B,C,D
SIO	OBJ	
OKI80	SRC	Treiber für OKI Microline 80
OKI82	SRC	Treiber für OKI Microline 82A/83A
OKI84	SRC	Treiber für OKI Microline 84/92/93
OKI84	FIL	Filter für OKI Microline 84/92/93
OKI84G	FIL	Grafikfilter für OKI Microline 84/92/93
Oxyz	OBJ	Konfigurierte OKI-Microline-Treiber
DAISY45	SRC	Treiber für Daisy M45 Drucker
DAISY	FIL	Filter für Daisy M45 Drucker
D45x	OBJ	Konfigurierte Daisy M45 Treiber
OL103	SRC	Treiber für Olympia ESW103
OL103	FIL	Filter für Olympia ESW103
OLYP	OBJ	Konfigurierter Olympia ESW 103 Treiber
UP	SRC	Allgemeiner Druckertreiber
UP	FIL	mit Filtermodul
UPX	OBJ	
PIO	SRC	Allgemeiner paralleler Treiber
PIO	OBJ	
GRAP	OBJ	
GRAPHB/H	OBJ	Grafikpaket für normalauflösenden
GRAPHV/H	OBJ	und hochauflösenden Bildschirm (H)
GRAPHA/H	OBJ	
GRAPHD/H	SRC	Grafik-Beispielprogramm
GRAPHD/H	OBJ	
GRAPHD/H	COM	
GRALINK/H	CMD	
GRAP82	OBJ	Grafik für Kontron PSI82 (nur KOS5)
GRAPHB82	OBJ	
KDM	OBJ	Debug Monitor
KDM1	OBJ	
KDM2	OBJ	
KDMSG	OBJ	
KDMLINK	CMD	
TIME	SRC	Beispielprogramm TIME
INFO	SRC	Beispielprogramm INFO
INFMSG	SRC	
INFO	OBJ	
INFMSG	OBJ	
MAGIC	BAS	BASIC-Beispielprogramme
PIANO	BAS	(nur KOS5)
TONLEIT	DAT	(nur KOS5)
MEMTEST	COM	Hardwaretestprogramme (KOS5)
PSITEST	COM	
TMED	COM	Medientestprogramm
TCNT	OBJ	Programme zur Messung von
PCNT	COM	Ausführungszeiten (nur KOS6)
PT	COM	Terminal-Programm



## 2. KOS-Grafikpaket

### 2.1 Einführung

Das Grafik-Paket bietet die Möglichkeit

- Punkte
- Vektoren
- Alphanumerische Zeichen

im Grafik-Mode darzustellen. Die Auflösung beträgt 512 Spalten x 256 Zeilen bzw. 1024 x 400 bei hochauflösenden Monitoren (H-Systeme).

Das Grafik-Paket besteht aus 3 Moduln in relokativem Object-Code. Diese können durch LINK mit OBJ-Moduln aus ASSEMBLER, FORTRAN, BASIC-Compiler BASCOM und BASIC-Interpreter MBASIC verknüpft werden. Für den BASIC-Interpreter MBASIC sind die Grafikmoduln passend zu linkern. Die Parameterübergabe entspricht den FORTRAN-Konventionen. Jeder Parameter ist 2 Byte lang. Die Grafik-Funktionen werden wie Unterprogramme aufgerufen. Sie sind vorher der Sprache entsprechend zu deklarieren.

In Kontron BASIC sind spezielle Grafik-Befehle implementiert; das Grafik-Paket ist hierfür zusammen mit dem Schnittstellenmodul BG.OBJ in dem Treiber GRAP.OBJ enthalten. Die Aktivierung erfolgt durch folgenden Aufruf (Beispiel):

```
für KOS 4.33:  IODC $GRAP=ACTIVE
               BASIC "LOAD MAGIC<RUN"
```

```
für KOS 5.x:   RLOAD GRAPTASK
               IODC $GRAP=ACTIVE
               BASIC "LOAD MAGIC<RUN"
```

Beim Betriebssystem KOS 5.x sorgt die Task 'GRAPTASK' dafür, daß die Grafikmoduln auch auf den Adressen 8000H...C000H (Bildwiederholtspeicher) ablauffähig sind. Daher ist diese Task für Grafikausgaben zu aktivieren. Sie selbst muß außerhalb des Bildwiederholtspeichers liegen.

**Hinweis:** Bei Kontron PSI82 sind der Grafiktreiber \$GRAP82 und das Grafikmodul GRAPHB82 zu verwenden.

Am einfachsten werden die Kontron PSI80-Grafikprogramme GRAP.OBJ und GRAPHB.OBJ durch das Systemkommando REN mit Zusatz '80' versehen, bei GRAP82.OBJ und GRAPHB82.OBJ wird der Zusatz '82' entfernt. Danach gilt die folgender Beschreibung auch für Kontron PSI82-Systeme.

```
für KOS6.x:   BASIC "LOAD MAGIC<RUN"
```

Der Grafiktreiber ist bei KOS6 bereits im Betriebssystem aktiviert. Die für den hochauflösenden Bildschirm (H-Systeme) benötigten Graphiktreiber und Graphikmodule sind im Graphikpaket unter KOS V6.05 enthalten, Kennzeichen ist der angefügte Buchstabe "H".

## 2.2 Programmieren mit dem Kontron PSI Grafik-Paket

Die Grafik-Software gliedert sich in Grafik-Basis-Software (Modul GRAPHB bzw. GRAPHBH) und Grafik-Aufbau-Software (Moduln GRAPHV, GRAPH A bzw. GRAPHVH, GRAPHAH). Zum Betrieb der Grafik-Aufbau-Software ist die Grafik-Basis-Software erforderlich.

### Initialisierungen (GRAPHB bzw. GRAPHBH)

**Hinweis:** für Kontron PSI82-Systeme ist das Modul GRAPHB82 zu verwenden.

#### INITGR

Funktion: Initialisiert die graphische Betriebsart  
Löscht den Bildspeicher

#### INITAL

Funktion: Initialisiert die alphanumerische Betriebsart  
Löscht den Bildspeicher

#### Wichtig:

Mehrfaches Aufrufen des gleichen Initialisierungsprogramms (INITGR, INITAL) ist nicht zulässig!

#### CLEARV

Funktion: Löscht den Bildspeicher in graphischer Betriebsart

### Punkt-Manipulationen (GRAPHB bzw. GRAPHBH)

Parameterübergabe: Adresse von x in HL  
Adresse von y in DE  
x (0...511); y (0...255)  
x (0...1023); y (0...399) bei H-Systemen

#### SETXY

Funktion: Setzt den durch die Koordinaten x,y  
bestimmten Punkt

#### RESXY

Funktion: Löscht den durch die Koordinaten x,y  
bestimmten Punkt

#### INVXY

Funktion: Invertiert den durch die Koordinaten x,y  
bestimmten Punkt

#### TESTXY

Funktion: liefert in A den Wert 0, wenn Punkt dunkel  
und einen Wert ungleich 0, wenn Punkt hell

**Plotten (GRAPHV bzw. GRAPHVH)****PLOT**

Parameterübergabe: Adresse von x in HL  
Adresse von y in DE  
Adresse von PEN in BC

Funktion: Schreibt Vektor bzw. Rechteck vom letzten Plot-Punkt zu dem Punkt mit den Koordinaten x,y.

Für 'PEN' gilt:

- P = 0: Punkt anfahren ohne Zeichnen
- P = 1: Vektor zeichnen
- P = 2: Vektor löschen
- P = 3: Vektor invertieren
- P = 4: Rechteckrahmen zeichnen
- P = 5: Rechteckrahmen löschen
- P = 6: Rechteckrahmen invertieren
- P = 7: Rechteckfläche zeichnen
- P = 8: Rechteckfläche löschen
- P = 9: Rechteckfläche invertieren

Es ist einzuhalten:  $x < 512$  und  $y < 256$  bei normalauflösenden Systemen  
 $x < 1024$  und  $y < 400$  bei H-Systemen

**Alphanumerische Zeichen (GRAPHA bzw. GRAPHAH)****SYMBOL**

Parameterübergabe: Adresse von x in HL ( $x < 512$ , bzw.  $x < 1024$  bei H-Systemen)  
Adresse von y in DE ( $y < 256$ , bzw.  $y < 400$  bei H-Systemen)  
Zeiger auf Adressenliste in BC

Die Adressenliste enthält:

Adresse von FAKTOR  
Adresse von TEXT  
Adresse von RICHTG  
Adresse von LÄNGE

FAKTOR (1...15); RICHTG (0,90,180,270) Grad

Funktion: TEXT ist die Anfangsadresse eines Strings mit der Länge LÄNGE. Dieser wird um den Faktor FAKTOR vergrößert in der Richtung RICHTG auf den Bildschirm geschrieben. Der Punkt mit den Koordinaten x,y ist der linke untere Eckpunkt des 1. Zeichens.

## 2.3 Einbau der Grafik-Moduln in Anwenderprogramme

### 2.3.1 Grafik in ASSEMBLER-Programmen

Zur Anwendung sind 2 Schritte erforderlich

- Bereitstellung der Parameter
- Aufruf der Funktion

Beispiel: Zeichnen eines Vektors von (100,100) bis (200,200)

EXTERNAL INITGR, INITIAL, PLOT

TEST:

```

CALL INITGR
LD HL,100      ; Parameter laden
LD (X1),HL
LD (Y1),HL
LD HL,200
LD (X2),HL
LD (Y2),HL

LD HL,X1      ;PLOT 100,100,0
LD DE,Y1
LD BC,PENUP
CALL PLOT

LD HL,X2      ;PLOT 200,200,1
LD DE,Y2
LD BC,PENDOWN
CALL PLOT

CALL INITIAL
RET

```

```

PENUP:  DEFW    0
PENDOWN:DEFW    1

```

```

X1:     DEFS 2      ;PARAMETER
Y1:     DEFS 2
X2:     DEFS 2
Y2:     DEFS 2

```

END TEST

Siehe auch das Beispielprogramm GRAPHD bzw. GRAPHDH. Darin wird die Verwendung jeder Grafik-Funktion gezeigt.

Beispiel zum Übersetzen und Binden eines Grafik-Programms:

```

EDIT GRAPHD.SRC
ASM =GRAPHD
LINK GRAPHD/N,GRAPHB,GRAPHV,GRAPHA,GRAPHD/E

```

### 2.3.2 Grafik in FORTRAN- und BASIC80-Programmen

Grafik-Funktionsaufrufe werden in FORTRAN80 und BASIC80 (beide Sprachen: Microsoft) wie externe Unterprogramme behandelt. Die Reihenfolge der Parameter entspricht der bei 'Parameterübergabe' beschriebenen. Diese Parameterrückgabe übernimmt bei BASIC80 jeweils ein zusätzliches Modul BASGRA.OBJ bzw. MBGRAP.OBJ. Dieses Modul wird hinzugelinkt. Die GRAPTASK muß bei KOS5 aktiviert sein.

#### BASIC80-Compiler BASCOM:

Ein Programm für die Einbindung der Grafiksoftware in BASIC80-Compiler verwendet die Aufrufe entsprechend:

```
100 X1% = 100
200 Y1% = 100
300 PEN% = 0
400 CALL INITGR
500 CALL PLOT (X1%, Y1%, PEN%)
600 CALL INITAL
```

#### BASIC80-Interpreter MBASIC:

Bei BASIC80-Interpreter wird die Kopplung über eine Sprungtabelle auf die externen Unterprogramme durchgeführt. Die externen Unterprogramme sind i.a. auf feste Adressen gelinkt.

Alle Grafik-Funktionen des Kontron KOS Grafikpaketes sind von MBASIC aus ansprechbar.

Dazu dient das Grafikmodul MBGRAP.COM. Dieses wird von der Diskette "BASIC80-Interpreter" in den Speicher geladen, aber nicht sofort ausgeführt. Dies erreicht man durch das Kommando

```
MBGRAP,<---
```

Die Grafikfunktionen sind dann über eine Sprungtabelle am Anfang dieses Moduls ansprechbar.

Die Adressierung der Sprungtabelle ist aus dem Programm 'MBDEMO.BAS' auf der MBASIC Diskette zu ersehen.

Das Grafikmodul MBGRAP kann durch das auf der MBASIC-Diskette enthaltene Kommando

```
DO MBGLINK adr<---
```

auf eine bestimmte Adresse gelinkt werden. Die Voreinstellung ist 0000H; wird das Programm auf eine andere Adresse gelinkt, muß dies im BASIC Programm entsprechend berücksichtigt werden (siehe DEMO). Außerdem sind dazu die Module 'GRAPHB', 'GRAPHV' und 'GRAPHH' (bzw. GRAPHBH, GRAPHVH und GRAPHAH für H-Systeme) von der UTILITY Diskette auf die BASIC Diskette zu kopieren.

Die Grafiktask GRAPTASK muß bei KOS 5.x ebenfalls geladen sein.

```
RLOAD GRAPTASK<---
```

**FORTRAN80-Compiler:****Beispiel:**

C  
 C DEMO program:     FORTRAN80-Compiler + Kontron PSI Graphics  
 C  
 C FORTRAN programs may use all functions of the Kontron PSI Graphic  
 C package.  
 C  
 C You have to 'LINK' the corresponding graphic modules with the  
 C DEMO program 'FORGRA.OBJ'.  
 C

```
EXTERNAL INITAL,INITGR,PLOT,SETXY,SYMBOL
INTEGER X1,X2,X3,X4,X5,X6,X7,X8
INTEGER Y1,Y2,Y3,Y4,Y5,Y6,Y7,Y8
INTEGER FAKT,LAEN,RICHT,PEN
REAL*8 STR1,STR2,STR3,STR4,STR5,STR6,STR7
DATA STR1,STR2,STR3/'KONTRON ',' PSI^80 ',' GRAPHIK'/
DATA STR4,STR5,STR6/'+FORTRAN',' 80 ','COMPILER'/
DATA STR7/' DEMO '/
```

C

```
CALL INITGR
FAKT=2
RICHT=1
LAEN=8
X1=20
Y1=180
X2=170
Y2=180
X3=300
Y3=180
X4=5
Y4=5
X5=505
Y5=250
PEN=0
CALL PLOT(X4,Y4,PEN)
PEN=4
CALL PLOT(X5,Y5,PEN)
CALL SYMBOL(X1,Y1,FAKT,STR1,RICHT,LAEN)
CALL SYMBOL(X2,Y2,FAKT,STR2,RICHT,LAEN)
CALL SYMBOL(X3,Y3,FAKT,STR3,RICHT,LAEN)
FAKT=2
X1=20
Y1=150
X2=170
Y2=150
X3=250
Y3=150
X4=150
Y4=50
CALL SYMBOL(X1,Y1,FAKT,STR4,RICHT,LAEN)
CALL SYMBOL(X2,Y2,FAKT,STR5,RICHT,LAEN)
FAKT=1
CALL SYMBOL(X3,Y3,FAKT,STR6,RICHT,LAEN)
FAKT=3
CALL SYMBOL(X4,Y4,FAKT,STR7,RICHT,LAEN)
X1=20
Y1=40
X2=490
Y2=120
```



```

PEN=0
CALL PLOT(X1,Y1,PEN)
PEN=9
CALL PLOT(X2,Y2,PEN)
X1=10
DO 100 I=10,490,5
X1=I
Y1=140
CALL SETXY(X1,Y1)
100 CONTINUE
DO 200 I=10,490,5
X1=I
Y1=240
CALL SETXY(X1,Y1)
200 CONTINUE
DO 300 I=140,240,5
X1=10
Y1=I
CALL SETXY(X1,Y1)
300 CONTINUE
DO 600 I=140,240,5
X1=490
Y1=I
CALL SETXY(X1,Y1)
600 CONTINUE
DO 800 I=1,6900
DO 700 I=1,10900
700 A=10*100
800 B=10*100
X1=3
Y1=3
PEN=0
CALL PLOT(X1,Y1,PEN)
X1=508
Y1=252
PEN=9
CALL PLOT(X1,Y1,PEN)
DO 400 I=1,6900
DO 350 I=1,10900
350 A=10*100
400 B=10*100
CALL INITAL
END
```

### 2.3.3 Grafikerweiterungen unter KOS6.05

#### Normalauflösende Systeme:

Bei KOS6.05 unterstützt \$GRAP 48kByte Bildwiederholtspeicher. Damit können drei voneinander unabhängige Bilder (je 512 x 256 Bildpunkte) in den Segmenten 1,2,3 gespeichert werden. Segment 0 behält den alphanumerischen Inhalt.

Das Umschalten der Grafiksegmente erfolgt von "OPEN \$GRAP" durch entsprechendes Einstellen eines Schalters in Speicherzelle 3CH (60 dezimal):

<pre>&lt;3CH&gt; = 0nH</pre>	<pre>Anwahl des Grafiksegmentes n (n = 1...3), und Löschen des Segmentes bei "OPEN \$GRAP"</pre>
<pre>&lt;3CH&gt; = 8nH</pre>	<pre>Anwahl des Grafiksegmentes n (n = 1...3), kein Löschen des Segmentes bei "OPEN \$GRAP"</pre>

#### Hinweis:

Diese Graphikerweiterung gilt nicht für H-Systeme. Hier wird die Speicherzelle 3CH zum Feststellen der Hintergrundfarben verwendet.

In den folgenden Beispielen ist vorausgesetzt, daß die Grafiksegmente bereits Bildinformationen enthalten.

#### Beispiel in BASIC:

```

10 POKE 60, 128+1
20 OPEN #9: '$GRAP'
30 GOSUB 1000
40 POKE 60, 128+2
60 OPEN #9: '$GRAP'
70 GOSUB 1000
80 POKE 60, 128+3
90 OPEN #9: '$GRAP'
100 GOSUB 1000
110 GOTO 10
1000 REM 'DELAY LOOP'
1010 FOR I=1 to 100
1020 NEXT I
1030 RETURN

```

#### Beispiel in Assembler:

<pre>ld a,128+1 ld (grap.cmd),a call INITGR call delay ld a,128+2 ld (grap.cmd),a call INITGR call delay . . ret</pre>	<pre>;image 1, do not clear memory ;transfer command ;'open' graphic mode ;wait a little bit ;image 2, do not clear memory ;transfer command ;open graphic mode</pre>
<pre>external INITGR grap.cmd equ 3CH</pre>	<pre>;defined in GRAPHB.OBJ</pre>

**Grafikerweiterungen unter KOS 6.05 für hochauflösende Systeme  
(Kontron PSI980 H und Kontron PSI9 CH):**

Der Bildwiederholtspeicher eines 980H/9CH-Systems besteht aus 3 Bänken zu je 64 kbyte.

- Bank 0 : Zeichen-Information im Alphamodus oder  
Punkt-Information im Graphikmodus
- Bank 1 : Attribute (Hintergrund, Vordergrund, Blinken,  
Unterstreichen) im Alphamodus oder Punkt-Information  
im Graphikmodus
- Bank 2 : keine Funktion im Alpha-Modus  
Punkt-Information im Graphik-Modus

Die Videobaugruppe TCB/IOV-2 bietet, abhängig von der gewünschten Auflösung, verschiedene Graphikmöglichkeiten bzw. Betriebsarten an. Bei einer Auflösung von 1024 x 400 Bildpunkten ist jede der drei Videobänke in der Lage, ein komplettes Bild aufzunehmen. Verzichtet man bei Graphik auf die Graustufen (Farbe), so bietet sich die Video-Bank 2 als Graphik-Speicher an, da beim Umschalten zwischen Alpha- und Graphik-Modus dann sowohl die Alpha- als auch die Graphik-Information erhalten bleibt. Diese Betriebsart ist in KOS V6.05 realisiert. Wählt man den Interlace-Modus, so erhält man eine Auflösung von 1024 x 800 Bildpunkten. Der sogenannte "Interlace-Mode" ermöglicht die Verdoppelung der Information auf dem Bildschirm dadurch, daß die Zwischenzeilen ausgenützt werden. Dies hat jedoch zur Folge, daß die Bildwiederholffrequenz halbiert wird (ca. 38.5 Hz.); das Bild ist nicht mehr flackerfrei. Ferner benötigt man für ein Bild zwei Video-Bänke (0 und 1):

1024 x 400 = 409600 Punkte = 51200 Bytes (= C800H)

1024 x 800 = 819200 Punkte = 102400 Bytes (= 19000H)

Über MA11 (Video-Controller-Adresse) und SEL.MODE (Bit 3 des Statusregisters 1) wird die Graphik-Betriebsart gewählt. Folgende Programmierung ist erforderlich:

**Non-Interlace-Mode**

Statusregister!	Video-Controller		Betriebsart
1	(*) Start	Scroll-Adresse!	
0000 0xxxH	1024 x 400	0000H	!Graphik mit 8 Graustufen !alle Video-Bänke werden !verwendet !
0000 0xxxH	1024 x 400	0800H	!Graphik ohne Graustufen !Bank 2 ist Graphikspeicher !(Standard-Einstellung) !
0000 1xxxH	1024 x 400	0000H	!Graphik ohne Graustufen !Bank 0 ist Graphik-Speicher !
0000 1xxxH	1024 x 400	0800H	!Graphik ohne Graustufen !Bank 1 ist Graphik-Speicher !
<b>Interlace-Mode</b>			
0000 1xxxH	1024 x 800	0000H	!Graphik ohne Graustufen !Videobank 0 und 1 verwendet !
(xxx=Bildrand- helligkeit)			!

(\*) siehe Tabelle "Initialisierung des CRT-Controllers MC 6845"  
nächste Seite.

### Programmierung des Bildrands

Über das Statusregister.1 kann der Bildrand, d.h. der noch sichtbare, aber nicht mehr mit Zeichen beschreibbare Teil des Bildschirms, in einer von acht Graufstufen programmiert werden.

Statusregister 1 (38H schreiben, 39H lesen)	Bit	2	1	0	Bildrand
		0	0	0	Stufe 0 (schwarz)
		0	0	1	Stufe 1
		0	1	0	Stufe 2
		0	1	1	Stufe 3
		1	0	0	Stufe 4
		1	0	1	Stufe 5
		1	1	0	Stufe 6
		1	1	1	Stufe 7 (weiß)

Die folgende Tabelle zeigt die Initialisierung des CRT-Controllers MC 6845 für verschiedene Betriebsarten:

Register	8x10	13x16	1024x400	1024x800	Bedeutung
0	44	27	44	44	Horizontal Total
1	33	20	32	32	Horizontal Displayed
2	36	22	36	36	Hsync Position
3	7	5	7	7	Hsync Width
4	45	27	57	57	Vertical Total
5	9	13	0	0	Vertical Adjust
6	40	25	50	50	Vertical Displayed
7	42	26	52	52	Vsync Position
8	0	0	0	1	Interlace Mode
9	9	15	7	7	Scan Lines per Row
10	60H	60H	32	32	Cursor Start
11	9	15	0	0	Cursor End
12	0	0	x	x	Start Address High ) scroll
13	0	0	x	x	Start Address Low ) adress
14	0	0	-(*)	-(*)	Cursor Address High
15	0	0	-(*)	-(*)	Cursor Address Low

(\*) Cursor nicht angezeigt im Graphikmodus

(x) scroll address abhängig von Betriebsart

Alle Video-Bänke sind als I/O-Ports ansprechbar, die Adresse innerhalb einer 64 K Bank muß jeweils vorher in das Video-Adress-Latch programmiert werden.

38H : Status-Register 1 (Video) schreiben  
 39H : Status-Register 1 (Video) lesen

40H : Video Adress Latch (High Byte)  
 41H : Video Adress Latch (Low Byte)

30H : Video Bank 0  
 31H : Video Bank 0 mit Autoinkrement  
 32H : Video Bank 0 mit Autodekrement

34H : Video Bank 1  
 35H : Video Bank 1 mit Autoinkrement

36H : Video Bank 2  
 37H : Video Bank 2 mit Autoinkrement

18H : CRT-Controller MC6845 Address  
 19H : CRT-Controller MC6845 Data, 15 Register (Tabelle 0 ... 13,  
 14 ... 15 = scroll)

Falls fortlaufend in den Bildspeicher geschrieben wird, empfiehlt sich die Verwendung der Autoinkrement-Adresse, da in diesem Fall nur am anfang eine Bildspeicheradresse in das Video-Adress-Latch geladen werden muß.

**Hinweis:** Das Video-Adress-Latch kann nicht gelesen werden!

Im Gegensatz zur TCB/IOV, dem Vorläufer der TCB/IOV-2, werden die momentan eingestellten Attribute nicht automatisch mitgeschrieben, wenn ein Zeichen in den Bildspeicher eingetragen wird. Es muß also jedesmal, wenn ein Zeichen in Video-Bank 0 geschrieben wird, in Bank 1 das entsprechende Attribut-Byte eingetragen werden. Dafür sind die Attribute aber voll lesbar, man kann also jederzeit die komplette Bild-Information aus den Video-Bänken 0 und 1 auslesen.

**Zuordnung Video-Bank <---> Bildpunkte bei Interlace-Mode:**

Adresse	Bit 7 6 5 4 3 2 1 0	...	Bit 7 6 5 4 3 2 1 0
	(x,y)=(0,799)		(x,y)=(1023,799)
0000...007F	* * * * *	. . .	* * * * *
0080...00FF	* * * * *	. . .	* * * * *
. . .			
	video-bank 0		
FF80...FFFF	* * * * *	. . .	* * * * *
0000...007F	* * * * *	. . .	(x,y)=(1023,287)
. . .			
	video-bank 1		
8F80...8FFF	* * * * *		* * * * *
	(x,y)=(0,0)		(x,y)=(1023,0)

### 3. Ein-/Ausgabe-Treiber

Die seriellen Treiber sprechen entweder die Serienschnittstelle A (PSIA, SIOA) oder die Serienschnittstelle B (PSIB, SIOB) an. Die Programme liegen sowohl als Quellprogramm (Typ '.SRC') als auch als relocativ übersetzte Programme (Typ '.OBJ') vor. Unter KOS6 steht ein zusätzlicher Basistreiber (SIO) zur Verfügung, der auf alle seriellen Schnittstellen anwendbar ist. Ein Basis-Treiber für die Parallelschnittstelle ist PIO.OBJ.

Basistreiber sind in sich funktionale Programme zur Bedienung von Schnittstellen. Sie können als Gerüst für Kundenseitige Gerätespezifische Anpassungen dienen.

Ebenfalls im Quelltext enthalten sind Treiber für eine Reihe von seriellen und parallelen Druckern.

#### 3.1 Ein-/Ausgabetreiber-Aufbau

Ein-/Ausgabetreiber sind in der Regel Programme zur Ansteuerung einer spezifischen Hardwareschnittstelle, an die ein Gerät angeschlossen ist. Unter KOS können jedoch auch andere Funktionen als Treiber realisiert werden und so bei Bedarf durch Aktivierung (IODC- bzw. RLOAD-Kommando) zum residenten Teil des Betriebssystems dazugebunden werden. Ein Beispiel ist der Umsetzmodul für CP/M-Aufrufe \$CPM.

KOS zeichnet sich dadurch aus, daß es bis zu 20 verschiedene E/A-Treiber verwalten kann. Jedem E/A-Treiber wird hierbei eine logische Kanalnummer zugeordnet, über die dieser von der E/A-Verwaltung aus adressierbar ist. Diese Zuordnung ist beliebig; die E/A-Verwaltung ist transparent. Deswegen können grundsätzlich alle Ein-/Ausgaben auf weitgehend frei wählbare Kanäle gelenkt werden, ohne daß Änderungen in den Anwendungsprogrammen notwendig werden.

E/A-Treiber residieren für gewöhnlich über längere Zeit im Arbeitsspeicher des Computers, beanspruchen also Speicherplatz, der damit anderen Programmen nicht mehr zur Verfügung stehen kann. Um für Programme einen möglichst großen zusammenhängenden Speicherbereich bereitzustellen ist es zweckmäßig, E/A-Treiber möglichst weit nach oben zu laden.

Besteht der assemblierte Treiber vom Typ 'OBJ' nur aus einem Modul und enthält weder Externals noch Globals, übernimmt diese Aufgabe das IODC-Kommando; anderenfalls muß der Treiber mit dem 'LINK'-Kommando freien Speicherbereich zugewiesen werden, das Laden dieses Treibers mit dem Typ 'IOD' übernimmt dann wieder das IODC-Kommando.

Für anwendungsspezifische Treiber stellt die Utility-Diskette folgende Module zur Verfügung:

LIOD.SRC	Logischer E/A-Treiber, KOS-Schnittstelle
xxx.FIL	Gerätespezifische Filter, Anpassung an die Eigenschaften der Geräte xxx
HOD.SRC	Hardware-Ausgabetreiber mit Moduln für: <ul style="list-style-type: none"><li>- Parallelschnittstelle unter Interruptsteuerung</li><li>- serielle Schnittstelle unter Interruptsteuerung</li><li>- Parallelschnittstelle</li><li>- serielle Schnittstelle</li></ul>

Aus diesen Modulen werden die Peripherietreiber mit spezifischen Erweiterungen aufgebaut. Die Module werden dazu in ein Treibergerüst eingebunden (INCLUDE-Anweisung, siehe Muster unter Kapitel 7).

Das Konzept der Aufteilung eines Treibers in LIOD, FILTER und physikalische Schnittstellensteuerung hat folgende Vorteile:

- klare Treiberstruktur
- Trennung zwischen logischer und von Hardware-abhängiger Programmebene und
- Zeitersparnis bei der Entwicklung von Treibern.

### 3.1.1 Basisroutinen des logischen Ein-/Ausgabetreibers LIOD

#### ISTATUS, OSTATUS

Diese Routinen geben mit dem Status des Z-Flag an, ob ein Zeichen übertragen werden kann. In diesem Fall enthält dann der Akku einen Wert ungleich Null.

#### INPUT, OUTPUT

Der Inhalt des Akkus wird ein- oder ausgegeben. Falls nötig, wird gewartet, bis die Ein-/Ausgabe durchgeführt werden kann.

#### INIT.IOD

Hier wird die Ein-/Ausgabeeinheit initialisiert.

#### INIT.MSG

Nach dem Initialisieren wird die bei INIT.MSG angegebene Zeichenkette ausgegeben. INIT.MSG muß mit einem Byte mit Inhalt '0' enden.

#### OPEN.INPUT, OPEN.OUTPUT

Öffnen der Ein-/Ausgabeeinheit. Diese Routinen leiten den Beginn einer Übertragung ein.

#### CLOSE.INPUT, CLOSE.OUTPUT

Schließen der Ein-/Ausgabeeinheit. Hiermit werden Aktivitäten zum Beenden einer Übertragung ausgelöst (z.B. Seitenvorschub bei Druckern).



**Einstellmöglichkeiten des LIOD**

Der logische Ein-/Ausgabetreiber kann mit den folgenden Parametern, die eine bedingte Assemblierung bewirken, den jeweiligen Anforderungen angepaßt werden:

**KOS5**

Wenn dieser Parameter existiert und zu 1 gesetzt ist, wird ein zu KOS5 und KOS6 passender Treiber erzeugt.

Der für KOS5 eingestellte Treiber ist auch zu KOS6 kompatibel. Da KOS6 nur über den ersten Einsprung mit dem Treiber verkehrt, kann das Initialisieren, öffnen und schließen des Treibers auch mit der KOS-Funktion IOC-Call (8ch und 8dh) erfolgen.

**input.driver, output.driver**

Beide Parameter müssen existieren. Eine '1' bedeutet, daß ein Ein- oder Ausgabetreiber oder, wenn beide Parameter gleich 1 sind, ein bidirektionaler Treiber erzeugt wird.

**pstat.info**

Wenn dieses Symbol existiert, entsteht eine Zeichenkette zur Treiberidentifikation. Diese Zeichenkette steht am Anfang des Treibers nach den Sprungbefehlen und kann z.B. mit dem Kommando PSTAT ausgegeben werden. Sie wird mit dem Macro 'id.text' außerhalb von LIOD definiert.

**direct.87**

Dieses Symbol hat nur für KOS6 eine Bedeutung. In Treibern für KOS5 sollte es nicht existieren.

**3.1.2 Filtermodul xxx.FIL**

Der Modul xxx.FIL ist gerätespezifisch. Bei Druckern wird durch dieses Modul das Konzept des "virtuellen Druckers" realisiert. Aufgrund von Kontrollzeichen, die dem Treiber geschickt werden, sind alle häufig verwendeten Druckfunktionen steuerbar. Der Benutzer braucht sich nicht um druckerspezifische Steuersequenzen zu kümmern.

Da durch diesen gerätespezifischen Filter (=Umsetzer) die Kontrollzeichen im Text auf verschiedenen Druckern gleiche Wirkung haben ist es möglich, Textdateien ohne Änderung auf unterschiedlichen Druckern auszugeben. Beispielsweise kann ein Text während der Korrekturphase auf einem schnellen Matrixdrucker ausgegeben werden, während der endgültige Druck auf einem Typenraddrucker erfolgt. Druckfunktionen wie Fettdruck, Unterstreichung, Hoch- und Tiefstellen von Textstücken werden dabei in beiden Fällen gleichartig ausgeführt. Bitte beachten Sie jedoch, daß aufgrund unzureichender bzw. verschiedener Normierungen Sonderzeichen wie <, >, \$ etc. bei Druckern auch mit anderen Zeichen belegt sein können. Einzelheiten gibt die Beschreibung des jeweiligen Druckers.

Da die verwendeten Kontrollzeichen den in "WordStar" definierten entsprechen, können auch WordStar-Dateien unter KOS ausgedruckt werden. Daneben ist es möglich, unter direkter WordStar-Steuerung zu drucken. Der Treiber schaltet dabei automatisch in den Transparent-Modus. Dies erfolgt durch CTRL-P.

Druckfunktionen, die nicht mit den definierten Kontrollzeichen ansprechbar sind, können unter Umständen auch im Filtermodus durch ESC-Sequenzen oder Kontrollzeichen erreicht werden. Die Filter sind transparent für ESC-Sequenzen, die nur druckbare ASCII-Zeichen (Code 20 bis 7E) enthalten.

### 3.1.3 Hardware-Schnittstellenmodul HOD

Die Module LIOD.SRC und HOD.SRC werden im allgemeinen unverändert übernommen. Bei seriellen Treibern ist die Einstellung der Schnittstelleneigenschaften (Baudrate, SIO A, SIO B, etc.) über Software-Schalter möglich. Die entsprechenden Schalter sind im Hauptprogramm-Quellcode enthalten.

Es sind lediglich die gewünschten Schalter EQU-Anweisungen auf "1" bzw. "0" zu stellen. Dies geschieht durch Editieren. Der Aufruf dazu lautet z.B.

```
EDIT OKI84.SRC<---
```

Anschließend werden die EQU-Anweisungen aufgesucht und passend gesetzt, danach wird das Modul zurückgeschrieben und neu assembliert.

### 3.1.4 Assemblieren und Linken eines Treiberprogramms

Nach einem Editiervorgang wird das Programm assembliert durch Aufruf des Assemblers, z.B. durch das Kommando "ASM PSIA =PSIX<---".

Besteht der Treiber nur aus einem einzigen Modul ohne Externals und Globals (wie in diesem Beispiel), muß er nicht auf eine bestimmte Adresse gelinkt werden, sondern das IODC-Kommando verschiebt den Treiber automatisch auf den freien Speicherbereich, der direkt unterhalb des Betriebssystems noch frei ist.

Soll ein Treiber auf eine definierte Adresse, z.B. A000, gelegt werden, muß mit dem 'LINK'-Kommando gearbeitet werden. Der Linkerlauf erzeugt eine Datei des Typs 'IOD' durch das Kommando

```
LINK PSIA.IOD/N,PSIA/P:A000/E<---
```

Treiber werden mit dem IODC-Kommando aktiviert:

```
IODC $PSIA=ACTIVE<---
```

Existieren von einem Treiber beide Typen (OBJ und IOD), so verwendet das 'IODC'-Kommando die Datei mit dem Typ OBJ.

IODC ruft die Routine 'INIT' des E/A-Treiberprogramms auf und schützt den Speicher ab der Startadresse, siehe MAP-Kommando. \$PSIA ist nun aktiviert, hat allerdings noch keine Kanalnummer zugewiesen. Dies ist möglich mit dem Kommando

```
IODC 0-n=$PSIA<---
```

"n" definiert die Kanalnummer und liegt zwischen 0 und 9 (n=2 ist CP/M-Ausgabekanal).

Ist hierbei z.B. n = 1, so gelangen ab sofort alle KOS-Ausgaben über den Treiber \$PSIA auf die Serienschnittstelle SIOA.

### 3.2 Drucker-Treiber

Für Drucker aus dem Lieferprogramm stehen ausgearbeitete spoolfähige Treiber im Quelltext xxx.SRC zur Verfügung. Sie bestehen aus dem Steuerteil xxx.SRC, der ein Filtermodul xxx.FIL und die Module HOD.SRC und LIOD.SRC einbindet.

#### Implementierte Druckfunktionen in xxx.FIL

Kontroll- zeichen	! Wirkung ! KOS-gesteuert	! Wirkung ! WordStar-gesteuert
CTRL-B	* ! Schattenschrift	! Dreifachanschlag (Boldface)
CTRL-D	* ! Doppelanschlag	! Doppelanschlag (Double strike)
CTRL-S	* ! Unterstreichung	! Unterstreichung (Underscore)
CTRL-X	* ! Durchstreichung	! Durchstreichung (Strike out)
	! (nicht OKI)	!
CTRL-V	* ! Tiefstellung	! Tiefstellung (Subscript)
CTRL-T	* ! Hochstellung	! Hochstellung (Superscript)
CTRL-A	* ! Enger Buchstabenabstand	! Enger Buchstabenabstand
	!	! (Alternative character pitch:
	!	! 12CPI)
CTRL-N	* ! Normaler Buchstaben-	! Normaler Buchstabenabstand
	! abstand	! (Standard character pitch:
	!	! 10CPI)
CTRL-R	* ! Doppelter Buchstaben-	! Doppelter Buchstabenabstand
	! abstand	!
CTRL-G	*** ! Sonderzeichen 7FH drucken	! --
	! (nicht OKI)	!
CTRL-F	*** ! Sonderzeichen 20FH drucken	! --
	! (nicht OKI)	!
CTRL-O	! Code 20 an Drucker =	! Code 20 an Drucker = Leerzeichen
	! Leerzeichen	! (Non break space)
CTRL-Q	* ! Schattenschrift	! --
	! (wie CTRL-B und CTRL-D)	!
CTRL-P	** ! Transparent-Modus EIN	! Transparent-Modus EIN
	! (nur für spez. Anwendg.)	! (automatisch)
CTRL-E	! Korrespondenzqualität	! Korrespondenzqualität EIN
	! EIN (nur OKI)	!
CTRL-W	! Korrespondenzqualität	! Korrespondenzqualität AUS
	! AUS (nur OKI)	!
CTRL-C	*** ! --	! --
CTRL-K	*** ! --	! --
CTRL-Y	*** ! --	! --

\* EIN/AUS-Funktionen (Flip-Flop).

\*\* CTRL-P wird von WordStar automatisch vor dem Drucken gesendet. Es darf nicht im Text stehen. Ausschalten des Transparent-Modus geschieht durch Drucken unter KOS bzw. durch den Aufruf des "N"-Kommandos ("N \$Treiber").

\*\*\* reserviert.

### Auto-Linefeed-Funktion

Die Auto-Linefeed-Funktion wird beim Initialisieren des Treibers ein- bzw. ausgeschaltet. Sie gleicht die unterschiedliche Handhabung der Zeichenende-Codefolge aus.

Im EIN-Zustand hat sie folgende Wirkung:

ankommende Zeichenfolge	! wird gemacht zu
CR + *	! CR + LF + *
CR + LF + *	! CR + LF + *
LF + CR + *	! CR + LF + *
LF + *	! LF + *

\* steht für beliebiges Zeichen

### 3.2.1 Treiber für Drucker OKI-Microline

Dateien: OKI80.SRC für MIC80  
 OKI82.SRC für MIC82A/83A  
 OKI84.SRC für MIC84/92/93

Die Treiber können auf die gewünschte Konfiguration durch Editierung von Software-Schaltern im Quellcode zugeschnitten werden.

Die Quelldatei OKI84.SRC enthält einen Softwareschalter für OKI Microline 84-Serien vor bzw. nach Mai 1983. Dieser Schalter muß bei OKI Microline 92/93 und bei Microline 84 Serien nach Mai 1983 zu Null gesetzt sein.

Standardeinstellung ist 1200 Baud für den seriellen Betrieb; änderbar im Quellprogramm.

Aus den Quelldateien wurden die Treiber 'Oxyz.OBJ' durch Assemblierung erzeugt. Dabei gilt folgende Zuordnung:

O x y z.OBJ

```

* * *
* * * * * Druckertyp: 0 = MIC 80
* *                2 = MIC 82/83
* *                4 = MIC84/92/93
* *
* * * * * Anschluß: P = parallel
*                  A = seriell Kanal SIOA
*                  B = seriell Kanal SIOB
* * * * * Funktion: A = Alphamodus
                  G = Grafikbetrieb

```

OKI.BAS enthält ein BASIC-Testprogramm für die graphische Ausgabe über einen Treiber \$OGxx.

Für die Drucker OKI Microline 80/82/83/84/92/93 entstehen serielle und parallele Treiber durch Editieren und Assemblieren der jeweiligen Quelldatei.

OKIxx.FIL (Text) und OKIxxG.FIL (Hardcopy, Grafik, nur für OKI84/92/93 unter KOS6) stehen zur Verfügung.

Für die Funktion des Filters OKIxx.FIL gilt die in 3.1.2 enthaltene Tabelle über die Auswirkung von Kontrollzeichen.

Wenn Korrespondenzqualität eingeschaltet ist, sind folgende Funktionen nicht verwendbar:

Schattenschrift (CTRL-B, CTRL-Q)  
Doppelanschlag (CTRL-D)

Für ältere OKI Microline 84 Drucker (vor Mai '83) gelten außerdem folgende Einschränkungen.

Innerhalb einer Zeile dürfen nicht kombiniert werden:

Schattenschrift (CTRL-B, CTRL-Q)  
Doppelanschlag (CTRL-D)  
Hochstellung (CTRL-T)  
Tiefstellung (CTRL-V)

In einem Text, der in Korrespondenzqualität gedruckt wird, dürfen keine TAB-Zeichen (Code 09h, CTRL-I) stehen.

Beim Erstellen von Dateien unter WordStar sind die Kontrollzeichen CTRL-A, CTRL-N und CTRL-R nur bedingt verwendbar. CTRL-N und CTRL-R sollten immer am Anfang einer Zeile stehen. Texte, die mit engem Zeichenabstand gedruckt werden sollen (CTRL-A), müssen bündig am linken Rand beginnen, d.h. es muß vorher der Punktbefehl ".PO 0" eingegeben werden.

Die Rückschaltung von CTRL-R (Doppelter Buchstabenabstand) auf den vorherigen Zeichenabstand muß am Anfang einer Zeile erfolgen und aus einer Kombination von CTRL-A und CTRL-N bestehen:

CTRL-A, CTRL-N ---> normaler Zeichenabstand  
CTRL-N, CTRL-A ---) enger Zeichenabstand

## WordStar-Modifikation für OKI84/92/93:

Die zu verwendende Version des WordStar ist gegenüber der Standardversion "WSU.COM" an einigen Stellen geändert. Die Änderung erfolgt im INSTALL-Programm des WordStar. Die Beschreibung von "INSTALL" gehört zur Option "WordStar".

## Zu beachten ist:

Bei "Teletype-like"-Druckern, wie dem OKI Microline 84, können einige Punktbefehle in WORDSTAR nicht benutzt werden, so z.B.:

.CW für die Schreibschrittweite

.LH für den Zeilenabstand

"Microjustification" ist ebenfalls nicht möglich.

## Geänderte ("patched") Speicherstellen:

Marke	! Speicherstelle	! Inhalt	
PSCRLF:	! 0696	! 03	
	! 0697	! 0D	
	! 0698	! 0A	
	! 0699	! 0A	
PSCR:	! 06A1	! 01	
	! 06A2	! 0D	
PSHALF:	! 06A8	! 02	
	! 06A9	! 0D	
	! 06AA	! 0A	
PALT:	! 06B5	! 01	
	! 06B6	! 1C	(bei älteren OKI Microline 84: 1D)
PSTD:	! 06BA	! 01	
	! 06BB	! 1E	
USR2:	! 06CE	! 02	
	! 06CF	! 1B	
	! 06D0	! 30	
USR3:	! 06D3	! 02	
	! 06D4	! 1B	
	! 06D5	! 31	
USR4:	! 06D8	! 01	
	! 06D9	! 1F	
PSINIT:	! 06E7	! 06	
	! 06E8	! 10	
	! 06E9	! 1B	
	! 06EA	! 25	
	! 06EB	! 39	
	! 06EC	! 0C	
	! 06ED	! 0D	
PSFINI:	! 06F8	! 02	
	! 06F9	! 1B	
	! 06FA	! 36	

**Filter OKI84G.FIL für Vollgraphik-Ausgabe bei KOS6-Systemen**

Dieser Filter erzeugt ein 1:1-Abbild des Bildschirms im Grafik-Modus auf dem Drucker. Die Auflösung ist dabei mit der Bildschirm-Auflösung identisch (256 x 512 Bildpunkte). Bei der Grafik von KOS6, die in verschiedenen Videobänken arbeiten kann, wird immer das gerade auf dem Bildschirm sichtbare Bild gedruckt. Das erzeugte Bild hat die Größe 18 x 18 cm, so daß die Wiedergabe auf DIN A4-Papier möglich ist. Der Treiber erwartet ein beliebiges ASCII-Zeichen (außer CR oder LF) als Startsignal und druckt dann selbsttätig den gesamten Inhalt des Bildschirms aus. Der Druckvorgang dauert 2 - 3 Minuten. Während dieser Zeit ist der Rechner blockiert. Abbruch ist mit ESC-Taste möglich. Es ist nötig, vor dem Druckvorgang einen "OPEN"-Aufruf an den Treiber zu richten und die Ausgabe mit "CLOSE" abzuschließen (über Systemfunktion 8DH; in BASIC mit OPEN #7: "\$OGxx" und CLOSE #7:).

**Hinweis:**

Der OKI84 Graphik-Treiber in Verbindung mit H-Systemen unterstützt automatisch die hochauflösende Graphik (400 x 1024 Bildpunkte), d.h. die volle Druckerbreite eines Microline 84 bzw. Microline 93 Druckers wird ausgenutzt. Die Verwendung eines Microline 92 mit halber Breite (DIN A4) ist für hochauflösende Graphik nicht vorgesehen.

**Filter OKI84G.FIL für Vollgraphik-Ausgabe bei KOS5-Systemen**

Dieser Filter funktioniert wie der vorstehend beschriebene Grafik-Filter, wobei ein Schalter für KOS5 gesetzt ist. Er verwendet zum Lesen des Bildspeichers die GRAPTASK. Es ist bei KOS 5.xx zu beachten, daß sowohl GRAPTASK als auch der Druckertreiber außerhalb des Adreßbereichs des BildwiederholSpeichers (8000H..BFFFH) liegen müssen.

**Betrieb der Drucker OKI Microline**

Der Drucker sollte grundsätzlich mit Traktor verwendet werden. Dabei können die Andruckrollen auf dem Lochrand des Papiers aufliegen. Falls gewisse Druckfunktionen dabei nicht zufriedenstellend arbeiten (Schattenschrift, Hochstellung), müssen die Andruckrollen geöffnet werden.

Schalterstellungen und Kabelbelegungen sind im Source-Listing (Utility-Diskette) und in der zugehörigen Informationsdatei PRINTER.INF dokumentiert.

### 3.2.2 Treiber für DAISY M45-Drucker

Für diesen Schönschreibdrucker werden die geeigneten Treiber aus den Modulen DAISY45.SRC und DAISY.FIL durch Editieren und Assemblieren von DAISY45.SRC erzeugt.

#### Einzelblatteinzug

Während der Initialisierung des Treibers wird die Frage gestellt, ob ein Einzelblatteinzug verwendet wird. Wird dies bejaht, so muß angegeben werden, ob breites oder schmales Druckerpapier verwendet wird.

Damit wird der linke Textrand festgelegt:

breites Papier: Kopfposition 10

schmales Papier: Kopfposition 30

Bei Betrieb des Einzelblatteinzugs ist außerdem eine Auto-Formfeed-Funktion eingeschaltet. Falls ein Text innerhalb von 68 Zeilen kein Formfeed aufweist, wird dieses automatisch eingefügt. Damit ist es möglich, unformatierte Texte auf Einzelblatt auszudrucken, ohne daß Zeilen verloren gehen.

Die Papierlängeneinstellung am Einzelblatteinzug und am Drucker müssen übereinstimmen und 2 Zoll über dem tatsächlichen Wert liegen, d.h. bei DIN A4-Papier muß am Einzelblatteinzug "14" und am Drucker an Schalter 2S Stellung "E" eingestellt sein.

#### Tabulator-Einstellung

Bei der Initialisierung werden dem Drucker horizontale Tabulatormarken in Abständen von je acht Zeichen programmiert.

#### Transparenz für ESC-Sequenzen

Auch im Filter-Modus ist der Treiber für ESC-Sequenzen, die nur aus "ESC" und druckbaren ASCII-Zeichen (Code 20 bis 7E) bestehen, durchlässig. Damit lassen sich vom Anwender auch über den Funktionsumfang des "Virtuellen Druckers" hinausgehende Druckersteuerungen realisieren, ohne auf dessen Funktionen verzichten zu müssen.



**WORDSTAR-Modifikationen**

Die zu verwendende Version des WORDSTAR "WS.COM" ist gegenüber dem Standard-WORDSTAR "WSU.COM" folgendermaßen geändert:

- a) anstelle von "Teletype-linke printer" ist die Installation für "DIABLO" 1610/1620 printer" gewählt
- b) geänderte ("patched") Speicherstellen:

Speicherstelle	Inhalt
(PSINIT:) 06E7	! 0C
06E8	! 10
06E9	! 0D
06EA	! 1B
06EB	! 34
06EC	! 1B
06ED	! 42
06EE	! 1B
06EF	! 1E
06F0	! 09
06F1	! 1B
06F2	! 1F
06F3	! 0D

Beim Drucken unter WORDSTAR spielt es keine Rolle, ob Auto-Linefeed ein- oder ausgeschaltet ist.

**Zulässige Typenräder für DAISY-Drucker**

Da die verfügbaren deutschen Typenräder nicht der deutschen Referenzversion des ASCII-Codes entsprechen, wird ein Steuerprom des Druckers geliefert, welcher per Steckbrücke umschaltbar ist, je nachdem, ob deutsche oder amerikanische bzw. französische Typenräder verwendet werden.

Bei Brückenstellung "deutsch" sind folgende Typenräder verwendbar:

GERMAN PICA 10A	(Nr. B8154)	*
GERMAN COURIER 10	(Nr. B8156-01)	*
GERMAN ELITE 12	(Nr. B8157)	*

Bei Stellung "US" werden folgende Typenräder empfohlen:

PICA 10	(Nr. B8101-01)	*
PICA 10	(Nr. B8101-01E)	
ELITE 12	(Nr. B8102-01)	*
PRESTIGE ELITE 12	(Nr. B8301-01)	*
LETTER GOTHIC 12	(Nr. B8401-01)	
THEME 10PT	(Nr. B8501-01)	
FRENCH PICA 10	(Nr. B03411-01)	
FRENCH PRESTIGE K10	(Nr. B8131)	
BRITISH PICA 10	(Nr. B8139)	
UK COURIER 10	(Nr. B8140)	
UK ELITE 12	(Nr. B03413-01)	

Mit \* gekennzeichnete Typenräder sind Vorzugstypen.

Die erwähnte Steckbrücke befindet sich im Gerät hinter den Schaltern und Leuchtdioden der Frontblende und ist nach Abnehmen der vorderen Gehäuseabdeckung zugänglich. Hier sind sechs Kontaktstifte sichtbar, von denen jeweils zwei senkrecht übereinander liegende eine Brücke bilden. Für die Umschaltung bei Typenradwechsel ist nur Brücke 1 umzustecken (rechts, von vorne gesehen).

Stellung: "deutsch" - Brücke geschlossen  
          "US"       - Brücke offen

Die Brücken 2 und 3 müssen offen sein.

### 3.2.3 Treiber für Olympia ESW103

Das Modul OL103.SRC bildet das Treibergerüst mit der Assembler-Schalter-Tabelle und den "Include"-Anweisungen für die einzelnen Module.

Zusammen mit dem Filter OL103.FIL und den genannten Standard-Modulen HOD.SRC und LIOD.SRC entsteht daraus ein Druckertreiber für die OLYMPIA-Schreibmaschine ESW 103, der die WORDSTAR-Kontrollzeichen auch beim Drucken unter KOS-Steuerung unterstützt.

**Zu beachten ist:**

Bei WORDSTAR-Installationen für "Teletype-linker-printers" können einige Punktbefehle in WORDSTAR nicht benutzt werden, so z.B.:

    .CW für die Schreibrschrittweite

    .LH für den Zeilenabstand

"Microjustification" ist ebenfalls nicht möglich.

## WORDSTAR-Modifikationen

Die zu verwendende Version des WORDSTAR "WSOL.COM" ist gegenüber dem Standard-WORDSTAR "WSU.COM" in folgenden Speicherstellen geändert ("patched"):

Speicherstelle	Inhalt
(POSMTH:) 0690	01
(BLDSTR:) 0691	03
(PSCR:) 06A1	01
(PALT:) 06B5	03
06B6	1B
06B7	0D
06B8	05
(PSTD:) 06BA	03
06BB	1B
06BC	0D
06BD	06
(ROLUP:) 06BF	02
06C0	1B
06C1	07
(ROLDOW:) 06C4	02
06C5	1B
06C6	03
(USR1:) 06C9	04
06CA	1B
06CB	10
06CC	1B
06CD	11
(USR4:) 06D8	02
06D9	1B
06DA	14
(PSINIT:) 06E7	09
06E8	10
06E9	0D
06EA	1B
06EB	15
06EC	1B
06ED	0D
06EE	06
06EF	1B
06F0	19

Beim Drucken unter WORDSTAR spielt es keine Rolle, ob Auto-Linefeed ein- oder ausgeschaltet ist.

### 3.2.4. Universelle Druckertreiber

#### Universeller Druckertreiber "UP"

Das Treibergerüst UP.SRC bietet die Möglichkeit, einen Treiber ohne Filter-Modul zu erzeugen. Dieser Treiber ist völlig transparent für alle 8-bit Codes von 00 (hex) bis FF (hex). Er ist wie die anderen Druckertreiber einstellbar auf die Schnittstellen SIOA, SIOB und CENTRONICS (parallel) und kann "polling"- oder interruptgesteuert betrieben werden.

Er ist gedacht für Druckerausgabe ohne Steuerfunktionen bzw. für Anwenderprogramme, welche die Steuerung des Druckers selbst durchführen (z.B. WORDSTAR).

Falls im UP.SRC der Schalter "filter" auf "1" gesetzt ist, wird das Modul UP.FIL beim Assemblieren eingebunden.

Dieser einfach Filter bewirkt lediglich folgendes:

- a) In der INIT-Routine wird die Frage, ob "Auto-Linefeed" gewünscht wird gestellt.
- b) Falls die INIT-Frage mit "Y" beantwortet wurde, wird ein intelligenter Linefeed-Check durchgeführt und nur wenn "Carriage return" ohne "Linefeed" vorkommt, wird ein "Linefeed" eingefügt.

Bei negativer Beantwortung der Frage wird der Datenstrom nicht beeinflusst.

- c) In der OPEN-Routine wird ein "Carriage return" gesendet.
- d) In der CLOSE-Routine werden "Formfeed" und "Carriage return" gesendet.

### 3.3 Allgemeine serielle Treiber

#### PSIA/PSIB

Der Treiber dient zur Datenübertragung über die seriellen Schnittstellen A und B; insbesondere ist er dafür ausgelegt, zwei Kontron PSI-Systeme oder ein Kontron PSI-System und einen anderen Rechner zu koppeln. Ab KOS5.44/5.54/6.0 sind diese Treiber spoolfähig.

In beiden Systemen aktiviert man die Treiber z. B. mit dem Kommando 'IODC \$PSIx=ACTIVE'<--- x=A,B.

Zuerst muß der empfangende Rechner mit dem Kommando

```
COPY $PSIx datei.typ<---
```

gestartet werden. Erst dann darf der sendende Rechner seine Daten übertragen; das Kommando lautet:

```
COPY datei.typ $PSIx<---
```

Die Übertragung erfolgt vollduplex. Ist der empfangende Rechner nicht bereit (da er z.B. gerade einen Teil der Daten auf Floppy Disk abspeichert), überträgt er an den sendenden das Zeichen 19H (CTRL-Y). Der sendende Rechner wartet so lange, bis er das Zeichen 16H (CTRL-V) bekommt, und fährt dann mit der Übertragung fort.

Bei Dateiende überträgt der sendende Rechner die Zeichenfolge 04H,04H (CTRL-D,CTRL-D); dies interpretiert der Treiber auf der Empfängerseite als EOF (End of file) und schließt die Datei. Muß der sendende Rechner das Zeichen 04H übertragen (was nur bei Dateien des Typs '.COM' vorkommen kann, nicht aber bei ASCII-Dateien), sendet er die Zeichenfolge 04H,FFH. Der empfangende Rechner entfernt dann das Zeichen FFH, und trägt in die Datei nur das Zeichen 04H ein.

Bei der Kopplung mit anderen Rechnern muß dessen Serienkanaltreiber diese Software-Synchronisation nachbilden, oder es muß der Treiber PSIA/PSIB angepaßt werden.

Die Treiber arbeiten mit:

9600 Baud	kein Parity
2 Stopbits	CTS nicht abfragen

Diese Parameter können im Quellcode des Treibers geändert werden. Nach der Quellcode-Änderung wird der Treiber (xxxx.OBJ) durch Assemblieren erzeugt.

Belegung des Kabels:

Kontron PSI-1		Kontron PSI-2
2	-----	3
3	-----	2
7	-----	7

**SIOA/SIOB**

Allgemeiner serieller Treiber für die seriellen Schnittstellen A und B. Der bidirektionale Treiber überträgt alle Zeichen und ist spoolfähig (siehe 'SPOOL'-Kommando).

Der Treiber arbeitet mit:

1200 Baud (KOS4/5), 9600 Baud (KOS6)  
2 Stopbits  
kein Parity  
CTS nicht abfragen

Diese Parameter können im Quellcode durch Editierung geändert werden. Danach wird die geänderte Quellcodedatei assembliert.

**SIO (nur KOS6)**

Allgemeiner serieller Treiber für alle seriellen Schnittstellen der Kontron PSI9xx-Systeme.

Bei der Aktivierung wird die Kanalwahl interaktiv festgelegt.

**Verwendung des Treibers in KOS-Systemkommandos**

Beim Datentransfer mittels des COPY-Kommandos werden folgende Aktivitäten ausgeführt:

1. Aufruf der OPEN - Routine
2. Datenübertragung
3. Aufruf der CLOSE - Routine

**Hinweis:** Das COPY-Kommando ordnet einem E/A-Treiber automatisch die Kanalnummer 5 zu. Eine vorherige Zuweisung mit dem IODC-Kommando ist deshalb nicht erforderlich.

Soll die Datei mit dem SPOOL-Kommando ausgedruckt werden, wird zuerst die Drucker-Task aktiviert (s. PTASK-Kommando, Systemkommandos) durch das Kommando "RLOAD PTASK<---".

Anschließend wird der Spool-Auftrag z.B. mit dem Kommando erteilt:

SPOOL KOS.INF \$SIOA<---.

### 3.4 Allgemeiner paralleler Treiber PIO

Allgemeiner paralleler Ausgabe-Treiber für die parallele Schnittstelle (Centronics). Der Treiber überträgt alle Zeichen; er ist spoolfähig (siehe 'SPOOL'-Kommando).

### 3.5 Virtueller Medientreiber unter KOS (\$VMED)

Der Medien-Treiber \$VMED verwendet einen Halbleiterspeicherbereich als virtuelles Medium. Dieses Medium ist bezüglich seines Verhaltens vollkommen identisch zu Medien im herkömmlichen Sinne des Wortes (Floppy Disk etc.).

Unter KOS4/5 ist die Kapazität des Mediums \$VMED in Schritten von 4 kByte von 8 k bis 32 kByte einstellbar durch Rückfrage am Bildschirm. \$VMED bietet extrem kurze Zugriffszeiten und ist deshalb immer dann besonders zu empfehlen, wenn Programme oder Daten häufig benötigt werden.

Beispiel für die Aktivierung von \$VMED:

```
IODC $VMED=ACTIVE M-2=$VMED LIST<---  
N 2<---  
M 2<---
```

Dieses Beispiel aktiviert den Treiber \$VMED und ordnet diesem die Mediennummer 2 zu. Selbstverständlich belegt \$VMED Speicherplatz (siehe MAP-Kommandos). Je nach Anwendungsfall sollte dem Medium \$VMED deshalb so wenig Speicher wie möglich zugewiesen werden. Die Größe des zugewiesenen Speichers ist bei der Initialisierung im IODC-Kommando definierbar. Zu beachten ist außerdem, daß \$VMED nur für temporäre Dateien geeignet ist. Jeder Reset beendet zwangsläufig die Existenz des Mediums und damit der dort gespeicherten Dateien.

**Hinweis:** Für einen erweiterten virtuellen Medienspeicher können bis zu 3 Baugruppen in der Reihe ECB/D256 in den Einschubrahmen der Kontron PSI80/82-Systeme eingesetzt und unter dem Treiber \$Dxxx betrieben werden.

Unter KOS6 ist \$VMED bereits standardmäßig aktiviert und greift auf die Speicherbänke 2 und 3 zu. Die Deaktivierung erfolgt durch KOSGEN.

Die Kapazität dieses Mediums beträgt 128 kB und kann durch Einsatz von ECB/D256-Baugruppen bis zu 1 MB erweitert werden. \$VMED erkennt selbständig die zur Verfügung stehende Speichergröße.

### 3.6 Medientreiber \$BMD1 und \$BMD2 für die ECB-Baugruppen ECB/BM, ECB/BM1 und ECB/BM2 in Kontron PSI9xx-Systemen.

Mit den Baugruppen ECB/BM, ECB/BM1 und ECB/BM2 steht ein Magnetblasenspeichersystem zur Verfügung, das sich in Schritten von 128 kByte bis zu 1 MByte Speicherkapazität ausbauen läßt.

Durch die Medientreiber \$BMD1 und \$BMD2 können unter KOS beliebige Magnetblasenspeicher-Konfigurationen bis 1 MByte Speicherkapazität als ein Medium angesprochen werden. Während vom Treiber \$BMD1 immer nur ein Speicherbaustein selektiert wird, selektiert der Treiber \$BMD2 immer 2 Magnetblasenspeicher gleichzeitig, wodurch eine höhere Transfargeschwindigkeit erzielt wird. Bei einer Konfiguration mit ungerader Anzahl von Magnetblasenspeichern wird bei Verwendung des Treibers \$BMD2 der letzte Magnetblasenspeicher nicht benutzt.

**Hinweis:** Bei Verwendung des Treibers \$BMD2 muß die Bubblespeichermedien-Konfiguration aus mindestens zwei Speicherbausteinen bestehen.

**Aktivierung:**

z.B.: IODC \$BMD1=ACTIVE M-3=\$BMD1<---

Dieses Kommando weist dem Treiber \$BMD1 die Mediennummer 3 zu. Anschließend ist vor dem erstmaligen Gebrauch des Mediums ein Filesystem anzulegen. Dies geschieht durch das Kommando MAKEFS.

**Hinweis:** Durch das Kommando MAKEFS werden alle auf dem entsprechenden Medium bereits vorhandenen Dateieinträge gelöscht.

Die beiden Medientreiber \$BMD1 und \$BMD2 benutzen die Portadresse 70H, die auf der ECB/BM-Baugruppe durch den fünffach-DIL-Schalter bereits im Werk wie folgt eingestellt wurde:

Schalter:	1	2	3	4	5
Stellung:	on	off	off	off	on

Schalterstellung: on = 0, off = 1.

Weitere Informationen zum Einsatz von ECB/BM-Baugruppen finden Sie in der Technischen Beschreibung 'ECB-Speicher-Baugruppen'.



### 3.7 Der Ausgabetreiber \$MEDO

Dieser Ausgabetreiber kann einem beliebigen Ausgabe-Kanal (außer 0-0) zugewiesen werden. Der Treiber schreibt die Ausgaben eines Programms (oder KOS-DIALOGS) auf eine Datei, deren Name beim Eröffnen von \$MEDO angegeben wird. Außerdem bietet \$MEDO einige zusätzliche Funktionen, wie Auto-Linefeed einfügen, Linefeed entfernen oder große Dateien aufspalten (z.B. zum Bearbeiten großer systemfremder Dateien mit Kontron Editor EDIT).

Anwendung von \$MEDO:

- a) Umwandeln einer CP/M-kompatiblen Datei in eine KOS-kompatible Datei (z.B. Linefeed entfernen).
- b) Umwandeln einer KOS-Datei in eine CP/M-kompatible Datei (z.B. nach jedem 'Carriage Return' ein 'Linefeed' einfügen).
- c) Löschen (ignorieren) von Nicht-ASCII-Zeichen.
- d) Spalten einer großen Datei in mehrere 32 KB große Teile.
- e) Deutsche Umlaute ersetzen (z.B. ä --> ae).
- f) Erzeugen einer Source-Datei aus einer ASM-PRN-Datei.

Um alle diese Funktionen ausführen zu können, verfügt \$MEDO über Softwareschalter, die beim Eröffnen des Treibers entsprechend gesetzt werden können (Benutzerführung, Voreinstellung = OFFh). Bei jedem Aufruf des Treibers müssen diese Schalter neu gesetzt werden. Wird eine der Fragen zum Setzen oder Rücksetzen eines Schalters mit "ESC" beantwortet, so bleiben die folgenden Schalter unverändert.

#### Beispiele:

- a) IODC \$MEDO=ACTIVE 0-1=\$MEDO  
Eröffnen des Treibers und Ausgabekanal 1 (0-1) zuweisen.  
Bildschirm Ausgaben werden auf einer Datei mitgeschrieben. \$MEDO, sowie die Protokoll-Datei bleiben geöffnet bis das KOS-interne Kommando "N \$MEDO" gegeben wird.
- b) Zur Konvertierung einer großen CP/M-kompatiblen Datei (z.B. 200 KB) in mehrere KOS-kompatible Dateien ist folgendes Kommando einzugeben:

COPY CPMFILE.SRC \$MEDO

COPY eröffnet \$MEDO, wobei der gewünschte Dateiname, z.B. KOSFILE.001, angegeben wird. Beim ersten "new-line character" (<CR>) nach 32 KByte fragt \$MEDO, ob die Datei hier geteilt werden soll. Falls "nein" geantwortet wird, wird die Frage beim nächsten <CR> erneut gestellt.

- c) Generieren einer Assembler Source-Datei aus einer Listing-Datei:

COPY FILE.SRC \$MEDO

Dazu muß der Softwareschalter "Generate SRC file..." beim Eröffnen von \$MEDO gesetzt sein.

**Hinweis:** \$MEDO entfernt nicht die Liste der Labels am Ende der Listing Datei. Dies kann mit Hilfe des Kontron Editors EDIT geschehen.

#### 4. Treiber und Formatierprogramme

##### 4.1 Programme und Treiber für Diskettenstationen

Zusätzlich zu den ins Betriebssystem integrierten Treibern \$DSK0/1 stehen besonders für KOS5 zusätzlich frei aktivierbare Treiber zur Verfügung. Die Treibernamen sind 4-stellig:

Stelle 1:            S = Standard (8")-Laufwerk  
                  M = Mini        (5")-Laufwerk

Stelle 2:            I = Integriertes Laufwerk  
                  X = Externes Laufwerk

Stelle 3:            S = Single-density  
                  D = Double-Density

Stelle 4:            S = Single-sided  
                  D = Double-sided

3 + 4:            CP = CP/M-Format

Beispiel:           SXSS Standard- (8")-Laufwerk  
                              extern  
                              single density  
                              single sided

##### Aktivierung der Medien-Treiberprogramme

Die Treiberprogramme prüfen in der Initialisierungsphase, ob sie in der Soft- und Hardware-Umgebung arbeiten können. Falls nicht, wird eine Fehlermeldung ausgegeben und eine nicht erlaubte Treiberidentifikation an das IODC-Programm übergeben; somit wird die Aktivierung des Treibers verhindert.

Die Aktivierung der Treiber erfolgt, wie unter KOS üblich, mittels des IODC-Kommandos; ebenso die Zuteilung einer logischen Mediennummer. Dabei gilt: gerade Mediennummern für Laufwerke mit gerader physikalischer Laufwerksadresse, ungerade für Laufwerke mit ungerader physikalischer Laufwerksadresse.

##### Zuordnung der Laufwerknummern zu den Mediennummern:

- a) integriertes Diskettenlaufwerk
  - gerade Mediennummer        ---> physikalische Laufwerknummer 0 (rechts bzw. oben)
  - ungerade Mediennummer      ---> physikalische Laufwerknummer 1 (links bzw. unten)
- b) externes Diskettenlaufwerk
  - gerade Mediennummer        ---> physikalische Laufwerknummer 2
- c) integrierte Festplatte    ---> physikalische Plattennummer 0
- d) integrierte Wechselplatte ---> physikalische Plattennummer 1 bzw. 2. Festplatte

#### 4.1.1 Umsetztreiber für KOS 3.2-formatierte Disketten (nur KOS4/5)

Der Übergang von KOS3.2-formatierten Disketten auf KOS 4/5-Format wird durch den Treiber MICP.OBJ geleistet.

Bei seiner Aktivierung durch z.B.

```
IODC $MICP=ACTIVE M-1=$MICP<---
```

fragt der Treiber nochmals nach dem Laufwerk, das die KOS3.2-kompatible Diskette aufnehmen soll, i.a. ist dies Laufwerk 1. Durch MOVE können dann die gewünschten Dateien auf eine KOS 4/5-kompatible Diskette transferiert werden:

```
MOVE,<---      (Diskettenwechsel in Laufwerk 0)
N $MICP<---
X 1:* J<---
```

Hinweise:

- Programme sind neu zu übersetzen
- \$MICP kann nur lesen in Blöcken < 16 KB
- keine KOS3.2-Systemprogramme auf KOS 4/5-Disketten bringen!
- bei Diskettenwechsel "N \$MICP<---" geben

Die Deaktivierung von \$MICP und die Wiederzuweisung des KOS 4/5-Disktreibers \$DSK1 erfolgt durch

```
IODC $MICP=DEACTIVE M-1=$DSK1<---
```

#### 4.1.2 Treiber für interne Diskettenlaufwerke (KOS5)

Im Lieferumfang zum Betriebssystem KOS5 sind Diskettentreiber für die unterschiedlichen Kontron PSI80/82-KOS-Diskettenformate enthalten:

MISS.OBJ	Treiber für Mini-Laufwerke, integriert, single density, single sided
MIDS.OBJ	Treiber für Mini-Laufwerke, integriert, double density, single sided
MIDD.OBJ	Treiber für Mini-Laufwerke, integriert, double density, double sided

Durch Aktivieren eines Treibers auf einem (freien) Medienkanal Nr. 0...5 wird ein Laufwerk dem Treiber entsprechend behandelt, sofern dies technisch möglich ist:

**System:                      zulässige Treiber:**

Kontron PSI80/M2              MICP.OBJ, MISS.OBJ  
                              /S2

Kontron PSI80D/xx            MICP.OBJ, MISS.OBJ, MIDS.OBJ  
"            PSI82D/xx

Kontron PSI80Q/xx           MICP.OBJ, MISS.OBJ, MIDS.OBJ, MIDD.OBJ

Unter MIDS.OBJ kann die erste Seite (308 kByte) einer unter MIDD.OBJ beschriebenen Diskette gelesen und auch beschrieben werden. Kopieren durch COPYM2 ist ebenfalls möglich.

**Beispiel zur Treiberaktivierung:**

```
IODC $MISS=ACTIVE M-3=$MISS<---  
(ungerade Laufwerksnummern = linkes Diskettenlaufwerk)  
IL 3:*<---  
MOVE 3:* 0:<---  
IODC $MISS=DEACTIVE
```

**Hinweis:** CPM-Treiber müssen nach jedem Diskettenwechsel im zugehörigen Laufwerk neu initialisiert werden (mit N \$MICP<---) und können ausschließlich lesen. Auch CPM-Formatierung ist nicht möglich.

**4.1.3 Kompatibilität von KOS4/5-Disketten zu KOS6-Disketten**

Bei Kontron PSI9xx-Systemen wird standardmäßig vom doppelseitigen Betrieb mit doppelter Schreibdichte ausgegangen. Disketten sind wegen der unterschiedlichen Spurdichte zwischen Kontron PSI80/82-Systemen und PSI9xx-Systemen nicht kompatibel. Der Transfer erfolgt über Kontron KOBUS oder über die seriellen Treiber \$PSIA/B oder über optionale Softwarepakete, z.B. MOVE-IT jeweils unter Rechnerkopplung.

**4.1.4 Treiber und Formatprogramm für externe 8"-Diskettenlaufwerke****Treiber**

Der Treiber SXSS.OBJ dient zum Betrieb der externen 8"-Floppy Disk-Station (Bestellbezeichnung EFD-SD). Dieser Treiber ist bei KOS6 standardmäßig auf der Utility-Diskette verfügbar, bei KOS5 ist er zusammen mit dem FORMXSS-Programm Teil der Option EFD-SD.

Der Treiber \$SXCP dient nur zum Einlesen von CPM-Dateien von Disketten, die im Standard CP/M-Format A1 (Lifeboat) im Format single density/single sided beschrieben sind.

CPM-Treiber müssen nach jedem Diskettenwechsel im zugehörigen Laufwerk neu initialisiert werden (mit N \$SXCP<---) und können ausschließlich lesen. Auch CPM-Formatierung ist nicht möglich.

**Beispiel zur Aktivierung:**

IODC \$SXSS=ACTIVE M-4=\$SXSS<---

**8"-Betrieb + Grafik bei KOS5.xx:**

KOS.INI:        RLOAD GRAPTASK  
              IODC \$SXSS=ACTIVE M-4=\$SXSS  
              X \$GRAP=ACTIVE

Die Interrupt-Service-Routinen des 8"-Treibers müssen außerhalb des Bildwiederhol-speichers liegen. Deshalb ist er sofort nach der Graptask zu aktivieren (gilt für KOS 5.44/5.54).

**Formatieren von 8"-Disketten**

Das Formatieren von 8"-Disketten erfolgt auf externen Laufwerken mit dem Programm FORMATF (KOS6) bzw. FORMXSS (KOS5).

KOS5: Die Ausführung des FORMXSS-Programms erfordert nicht das 'Aktivsein' des Treibers \$SXSS, FORMXSS spricht die physikalische Laufadresse an.

KOS6: FORMATF arbeitet auf einem logischen Medienkanal.

**Besondere Hinweise**

Man steckt die Disketten mit der Beschriftung nach oben bzw. links und ihrem Spurbtastschlitz voraus in die Laufwerke, bis sie leicht einrasten. Danach schließt man die Laufwerk-tür durch leichtes gewaltloses Drücken nach links. Wird die Laufwerk-tür zu schnell geschlossen, kann sich die Diskette evtl. nicht zentrieren.

Beachten Sie bitte, daß bei 8"-Disketten der mechanische Schreibschutz bei offener Randaussparung einsetzt, anders als bei Mini-Floppy Disk Laufwerken. Verwenden Sie das der Betriebsart entsprechende Diskettenmaterial (unterschiedliche Lochkennungen in der Diskette!). Die Treiberprogramme setzen "soft-sektorierte" Disketten voraus.

Für den kundenseitigen Anschluß von Laufwerken an Kontron PSI-Systeme wird keine Gewähr über die Funktionalität übernommen. Technische Unterstützung bei Anschlußproblemen ist nicht möglich.

#### 4.2 Treiber- u. Format-Programm für externe Plattenspeicher

Die externe Box wird mit einem 50-pol. Flachbandkabel mit einem 50-pol. Anschluß der Platine ECB/SASI (Nr. 1029) bzw., bei Kontron PSI908/916 ohne integrierte Festplatten, mit der SASI-Schnittstelle der KDT6 verbunden. Die Platine ECB/SASI kann auch in KDT5, KDT6 und TCB/Z80 basierenden Maschinen betrieben werden. Voraussetzung ist natürlich ein ECB-Bus-Rahmen.

#### I/O-Adresse und Interrupt-Vektor

Die I/O-Adresse der ECB/SASI ist 78h für alle Systeme. Die Schalterstellung ist dazu wie folgt:

Schalter	!	1	2	3	4	5	6
Zustand	!	zu	offen	offen	offen	offen	zu

Für die Interrupt-Vektoren des DMA-Bausteins sind die Adressen xx78h bis xx7Fh reserviert, wobei xx dem Inhalt des I-Registers entspricht.

#### Mediendefinition

Die Mediengröße ist beschränkt auf maximal 64 KB.

KOS5: Falls sich in der Box zwei Winchester-Laufwerke befinden, so können durch geeignete Treiberwahl beide Laufwerke zu einem Medium zusammengefaßt werden oder als zwei getrennte Medien verwendet werden.

KOS6: Jedes Laufwerk wird einem Medium zugewiesen.

#### Definition der Treibernamen

KOS5: Der allgemeine Treibername lautet WIN2.OBJ. Es wird durch Kopieren eines ausgewählten Treibers erzeugt.

Die mit der Option WINSExx.-8 gelieferten Treiber sind nach dem Schema WkkkdmT.OBJ definiert:

kkk =	Kapazität im MByte je Box
d = 1	; ein Laufwerk je Box
d = 2	; zwei Laufwerke je Box
m = S	; alle Laufwerke zu einem Medium zusammengefaßt
m = D	; zwei Laufwerke sind zwei Medienkanälen zugeordnet
	; gerade Mediennummer zu Laufwerk 0 und
	; ungerade Mediennummer zu Laufwerk 1
t = C	; Datenübertragung über CPU
t = D	; Datenübertragung über DMA (nicht Standard)

Beispiel: In der Box sind zwei Laufwerke je 10 MByte eingebaut, beide Laufwerke sollen zu einem Medium zusammengefaßt werden, die Datenübertragung soll über CPU laufen. Der zugehörige Treibername ist W0202SC.OBJ. Der lauffähige Treiber wird durch Umbenennung erzeugt.

**Aufruf:** COPY 1:W0202SC.OBJ 0:WIN2.OBJ<--- kopiert  
von der Treiberdiskette in Medium 1 einen lauffähigen  
Treiber nach Medium 0. Medium 0 enthält die  
Systemdiskette.

**Aktivieren:** IODC \$WIN2=ACTIVE M-2=\$WIN2

**KOS6:** Die benötigten Treiber sind bereits im System KOS6 integriert.  
Das entsprechende Modul ist KOSA.SYS.  
Da KOS standardmäßig nur die Module KOSO.SYS bis KOS8.SYS lädt,  
wird für den Fall eines Anschlusses von externen Winchester-  
Laufwerken das Modul KOSO.SYS rekonfiguriert, indem das Ladebit  
für KOSA.SYS gesetzt wird.

**Aufruf:** SYSGEN KOSO<---

Es erscheint unter dem Cursor die Zeile "Segment Mask". Diese Maske  
bestimmt, welche Module zusätzlich geladen werden, beginnend mit  
KOS9.SYS. Demnach muß die zweite Ziffer von links eine "1" sein. Wenn  
dies der Fall ist, wird SYSGEN mit "K" wieder verlassen, andernfalls  
kann mit dem "S"-Kommando die Maske verändert werden. Mit "RETURN" wird  
weitergeschoben. Ist die Maske in richtiger Weise verändert, so wird die  
Information dauerhaft auf das Medium zurückgeschrieben: Eingabe: "W"  
(Write) und anschließend "Y" (Yes). KOSA.SYS muß insgesamt innerhalb  
der ersten 32 Dateien am Anfang des Lademediums vorhanden sein.

Mit dem KOS-Systemkommando KOSGEN kann die Art der Massenspeicher  
spezifiziert werden.

### Formatierung der Laufwerke

Im allgemeinen ist eine physikalische Formatierung der Laufwerke nur  
einmal erforderlich. Dies wird im Werk erledigt. Eine Formatierung ist  
nur dann notwendig, wenn Sektorkennungen zerstört sind. Ansonsten  
genügen TMED (siehe 6.3) als Bitmuster-Generator für BBR, BBR dient  
zum Ersetzen fehlerhafter Blöcke und MAKEFS zur logischen Formatierung  
(siehe Systemkommandos, TB-D).

**Achtung:** Durch die Formatierung werden alle zuvor abgespeicherten Daten  
gelöscht!

Der Name des Programms zur physikalischen Formatierung von internen  
und externen Platten-Laufwerken ist FORMATW (KOS5 und KOS6/DTC-  
Controller) bzw. WFD (KOS6/Adaptec-Controller).

Die wichtigsten I/O-Adressen werden im FORMATW anhand einer Auswahl  
eingestellt (interner SASI-Bus = 38h; externer ECB/SASI-Bus = 78h).

Bei zwei Laufwerken muß FORMATW zweimal gestartet werden. Der  
Formatierungsvorgang dauert einige Minuten je Laufwerk.

**Achtung:** Die Laufwerknummer (nicht Mediennummer) bezieht sich auf die  
festverdrahtete Adresse der beiden Laufwerke. Diese Adresse  
wird bei der Fertigung der Box festgelegt. Die Laufwerknummer  
ist durch FORMATW nicht feststellbar! FORMATW fragt nach der  
physikalischen Laufwerknummer. Die Mediennummer ist irrelevant.

### Fehlerhafte Blöcke ersetzen

Dies ist nur notwendig nach einer Formatierung oder später im Gebrauch, falls weitere Blöcke defekt werden sollten.

Hierzu muß ein Treiber aktiviert werden, der jedem Laufwerk eine separate Mediennummer zuordnet. Anschließend wird das Programm BBR auf jedes einzelne Laufwerk angewandt. Die Laufwerke sind nun für eine Verwendung in geeigneter Weise zu initialisieren, z.B. bei KOS muß ein Inhaltsverzeichnis angelegt werden.

### Inhaltsverzeichnis anlegen

Bei Verwendung unter dem Betriebssystem KOS muß ein Inhaltsverzeichnis angelegt werden.

**Achtung:** Das Anlegen eines Inhaltsverzeichnisses löscht alle zuvor abgespeicherten Daten logisch.

Ein neues Inhaltsverzeichnis wird auch dann angelegt, wenn ein Medium vollständig gelöscht werden soll.

Aufruf des Programms lautet: MAKEFS<---

Das Programm fragt nach der Mediennummer und nach dem Directorynamen.

Außerdem fragt MAKEFS nach der Anzahl der Dateieinträge. Ein Wert von 500 ... 1000 ist für normale Anwendungen geeignet. Sollten die definierten Dateieinträge später nicht ausreichen, verlängert KOS das Inhaltsverzeichnis automatisch. Eine Verlängerung sollte aber aus Gründen einer verlängerten Suchzeit vermieden werden.



#### 4.3 FORMATW Physik. Formatierung der Fest- und Wechselplatte (für alle Festplatten mit mindestens 10 MB Kapazität)

FORMATW dient zur physikalischen Formatierung von Fest- und Wechselplatten mit DTC (Kontron PSI9xx) oder XEBEC (Kontron PSI80/82-W10) Controller. Für Kontron PSI80/82-W5 mit HC-Controller steht das Programm FORMATW5 zur Verfügung. Kontron PSI9xx Systeme mit Adaptec Controller werden mit dem nachfolgend beschriebenen Programm WFD (Vormals FORMATA) formatiert (siehe auch 'KOSGEN' Kommando/KOS Systemkommandos, Konfigurierung von KOS5.SYS).

FORMATW beschreibt alle Sektoren einer internen oder externen Fest- oder Wechselplatte mit einem einheitlichen Bitmuster. Dieser Vorgang dauert einige Minuten pro Laufwerk. Es wird kein Inhaltsverzeichnis angelegt.

FORMATW ermittelt interaktiv Typ (Kapazität, Fest/Wechselplatte) und die Medienkanalnummer eines Laufwerks. Voraussetzung für FORMATW ist, daß der passende Medientreiber \$WINO etc. aktiviert ist.

Nach FORMATW sind die Kommandos 'BBR' (nur KOS6) und 'MAKEFS' (siehe dort) auszuführen, um ein KOS Medium zu initialisieren.

Für Kontron PSI80/82-W5-Systeme ist entsprechend das Formatierungsprogramm FORMATW5 zu verwenden.

FORMATW5 fragt nach der physikalischen Adresse des Festplattenlaufwerks, hier ist '0' anzuwählen. Vor dem Beginn des Formatierlaufes wird noch einmal rückgefragt.

**Hinweis:** Das physikalische Formatieren der Festplatte ist benutzerseitig i.a. nicht notwendig. Fehlermeldungen während des Formatierens deuten auf Hardware-Probleme hin, die im Service behoben werden sollten. Durch das Programm CHMED kann die Lesbarkeit aller Sektoren der Festplatte überprüft werden.

#### 4.4 WFD - Winchester Format und Debug Utility / Rev. 6

WFD arbeitet auf der untersten Treiberebene d.h. nicht über logische KOS Medienkanäle. Die Laufwerknummern entsprechen somit den physikalischen Adressen (0 oder 1).

WFD überprüft zunächst, ob Controller und Drives angeschlossen (On Line) sind und gibt das Ergebnis am Monitor aus.

**Beispiel:** Internal SASI Bus, Drive 0: On Line  
Internal SASI Bus, Drive 1: Off Line

Danach wird eine Liste aller WFD-Funktionen ausgegeben:

- S - Select Drive (0, 1)
- C - Change Controller Address and Bus Type
- L - Move Heads to Landing Zone (Stop Drive)
- R - Reinitialize Drive (Start Drive)
- P - Print Disk Capacity
- V - Verify Disk Blocks
- T - Check Controller Type
- F - Format Winchester Disk
- M - Mode Select (Define Drive Characteristic)
- D - Display Media Defect List
- H - Help (Print Command List)
- E - Error Code Messages
- K - Return to KOS

**WFD - Kommando**

Die einzelnen Funktionen von WFD werden kurz erläutert. In jedem Fall ist nach dem Promptzeichen WFD: nur ein einzelner Buchstabe einzugeben.

**S - Select Drive (0,1)**

Drive 0 oder 1 (physikalisch) ist auszuwählen. Als Voreinstellung gilt: Drive 0 ist selektiert.

**C - Change Controller Address and Bus Type**

Die SASI-Adresse des angeschlossenen Controllers ist zu wählen. SASI kennt acht Adressen (1, 2, 4, 8, 16, 32, 64, 128). Voreinstellung ist 1, was der Adresse des System-internen Controllers entspricht. Externe Controller sind auf 2 eingestellt.

Die C-Funktion bietet darüber hinaus die Möglichkeit, den Bus von intern (TCB/SASI-Bus) auf extern (ECB/SASI-Bus) zu schalten.

**L - Move Heads to Landing Zone (Stop Drive)**

Führt die Schreib-/Leseköpfe des selektierten Laufwerks auf die Lande-Position (siehe auch F-Format).

**R - Reinitialize Drive (Start Drive)**

Initialisiert Laufwerk und Controller. Der Adaptec Controller liest bei dieser Gelegenheit die Drive-spezifischen Parameter von Track 0 des selektierten Drives (siehe auch F-Format).

**P - Print Disk Capacity**

Die Kapazität des selektierten Drives in 512 Byte Blocks wird als dezimaler Wert ausgegeben. Bei Drives ohne Media Defects errechnet sich die Kapazität nach folgender Formel:

$$\text{Capacity} = \text{Heads} \times \text{Cylinder} \times 17$$

**V - Verify Disk Blocks**

Die Lesbarkeit aller Blöcke des selektierten Drives wird überprüft. Eventuelle Fehler werden im 'Cylinder-Head-Byte' Format aufgelistet. Verify Fehler sollten nach dem Formatieren nicht mehr auftreten, es ist jedoch insbesondere bei Drives größerer Kapazität nicht ausgeschlossen. Ein Verify Fehler wird bereits dann gemeldet, wenn fehlerhafte Datenbits vom Controller korrigiert wurden.

**T - Check Controller Type**

Gibt aus, welcher Controller angeschlossen ist:

Adaptec ACB 4000 oder DTC 510A

**Achtung:** WFD ist nur für den Adaptec Controller ACB 4000 geeignet.

**F - Format Winchester Disk**

Formatiert den selektierten Drive und ist deshalb mit besonderer Vorsicht anzuwenden. Interaktiv müssen eine Reihe von Drive-spezifischen Parametern eingegeben werden.

	M3012	M4020	A3046	Syquest
Interleave Faktor:	9	9	9	9
Number of Cylinders:	612	480	635	306
Number of Heads:	2	4	7	2
Write Precompensation:	0	0	321	128
Landing Zone:	44	42	8	0
Seek Mode:	2	2	2	2

Danach ist letztmals Gelegenheit, das Formatieren der Platte zu verhindern (Format really? mit 'n' beantworten).

Der Formatiervorgang selbst besteht aus mehreren Phasen:

- Phase 1: Formatieren
- Phase 2: Verify
- Phase 3: Reformatieren

Phase 3 ist nur notwendig, wenn Phase 2 Verify Fehler ergab. Am Ende von Phase 3 kann erneut ein Verify Durchlauf mit anschließendem Reformatieren gestartet werden. Es wird empfohlen, den Vorgang rekursiv zu wiederholen, bis weniger als 2 neue Verify Fehler pro 10 MByte Disk Kapazität auftreten.

**Hinweis:** Nach dem Formatieren 'BBR' und 'MAKEFS' anwenden.

**M - Mode Select (Define Drive Characteristic)**

Der Adaptec Controller ACB 4000 speichert die Drive-spezifischen Parameter (Kopf/Zylinderzahl etc.) auf Track 0/Cylinder 0 einer Winchester Platte. Ist diese Information nicht mehr lesbar (Error Code 28), so können mit Hilfe der M-Funktion die Drive Parameter definiert werden. Die Daten sind dann eventuell noch lesbar. Eine neue Formatierung der Platte wird dringend empfohlen.

**Hinweis:** Drive-spezifische Parameter siehe F (Format)-Kommando.

**D - Display Media Defect List**

Gibt die von der Verify Funktion ermittelte Defect Liste in sortierter Form aus.

**H - Help**

Gibt die Liste aller WFD-Funktionen aus.

**E - Error Code Messages**

Gibt alle möglichen 'Error Codes' mit dazugehörigen Klartexten aus.

**K - Return to KOS**

Kehrt zum Betriebssystem KOS zurück.

**4.5 Wechselplattenformatierprogramm FORMATZ**

'FORMATZ' dient zum physikalischen Formatieren von externen 10 MB IOMega Wechselplattenkassetten. 'FORMATZ' setzt einen aktivierten zugeordneten Treiber voraus, d. h. 'FORMATZ' arbeitet über logische Medienkanäle. Nach Aktivierung und Zuweisung des Treibers (KOS5, z.B. 'IODC \$IMEG=ACTIVE M-3=\$IMEG') bzw. nach Eröffnen des Treibers (KOS6, z.B. O \$IOM0) kann 'FORMATZ' aufgerufen werden, wenn beide Leuchtdioden am Wechselplattenlaufwerk aufleuchten. 'FORMATZ' erfragt beim Aufruf den logischen Medienkanal des Treibers sowie einen 'Interleave-Faktor', welcher mit "4" angegeben wird, für einen Interleaving-Faktor von 16.

**4.6 Allgemeine Hinweise zum Formatieren von Fest- und Wechselplatten****KOS5:**

Nach dem Formatieren mit 'FORMATW', 'FORMATW5' oder 'FORMATZ' wird das Directory mit 'MAKEFS' angelegt.

**KOS6:**

Nach dem Formatieren mit 'FORMATW', 'WFD' oder 'FORMATZ' ist folgendermaßen vorzugehen: \_ \_ \_

1. Programm 'TMED -R' aufrufen, um ein einheitliches Muster auf das Medium zu schreiben (z.B. 'E5').  
Achtung: eine falsch angegebene Mediennummer führt zur Löschung des angewählten Mediums!
2. Programm 'BBR' starten, um defekte Blöcke auszutauschen, dies sollte mit Mustervergleich durchgeführt werden.
3. Programm 'MAKEFS' aufrufen, um ein Directory auf dem Medium anzulegen. Je nach Verwendungszweck ist die Anzahl der Dateien anzugeben. Reicht dieser später nicht aus, so verlängert KOS das Dateisystem automatisch, was jedoch wegen längerer Suchzeiten vermieden werden sollte.  
Achtung: eine falsch angegebene Mediennummer führt zur Löschung des angewählten Mediums!

## 5. SOFTKEY-Utility

'SOFTKEY' ist ein interaktives Programm, mit dessen Hilfe die Zuordnung von Zeichenketten zu Tasten (typischerweise Funktionstasten) verwaltet wird. Das Programm ist ablauffähig auf allen Maschinen der Reihe Kontron PSI, unabhängig von der eingesetzten Tastatur.

Aufruf:           SOFTKEY<---

### Anwendung:

'SOFTKEY' führt Menü-gesteuert folgende Funktionen aus:

1. Auflistung der aktuellen Tastenzuordnung
2. Laden einer 'Zuordnungsdatei' mit der Möglichkeit, diese Zuordnung zu aktivieren
3. Deaktivieren der aktuellen Tastenzuordnung
4. Rückkehr zu KOS
5. Nur bei KOS4/5: Aktivieren von \$FKEY

### Zuordnungsdatei:

Die Zuordnungsdatei wird mit dem Editor 'EDIT' erstellt. Ihr Name ist beliebig, für den Typ gilt als Voreinstellung 'KEY' (z.B.: KOS.KEY).

Die Zuordnungsdatei beschreibt pro Zeile die Zuordnung einer Taste zu einer Zeichenkette. Letztere darf 1...32 Zeichen enthalten.

Folgendes Format muß eingehalten werden:

Fx=Zeichenkette für Funktionstaste  
Fy=Zeichenkette für Funktionstaste

Es bedeuten:

Fx, Fy:           beliebige Funktionstaste  
Zeichenkette...: beliebige Zeichenkette aus 1...32 Zeichen

Ein eventuelles 'Return' am Ende der Zeichenfolge wird durch 'senkrechter Pfeil' (KOS6) bzw. '<' (KOS4/5) gekennzeichnet.

Ist die Zuordnung einmal aktiviert (Menü-Funktion #2), so wird bei jedem Tastendruck Fx die zugeordnete Zeichenkette in den Tastatur-Eingabepuffer übertragen.

**Anwendung unter KOS:**

Zur Speicherung der Zuordnungsdatei wird bei KOS6 kein Anwenderspeicher benötigt, da \$KEY bereits die erforderlichen Pufferbereiche enthält. Es steht etwa 1 kByte zur Verfügung. Eine Datei 'xxx.KEY' darf also nicht länger als 8 Records (= 1 kByte) sein.

'SOFTKEY' kommuniziert mit \$KEY (Kanal I-1).

Die Wirksamkeit der Tastenzuordnung kann freigegeben bzw. gesperrt werden, was auch durch den Kommandointerpreter von KOS möglich ist:

```
nn/KOS:0 $KEY<---- ;Freigabe der Zuordnung
nn/KOS:C $KEY<---- ;Sperrern der Zuordnung
```

**Hinweis:** Eine 'gesperrte' Zuordnung bleibt in der Menü-Funktion #1 von 'SOFTKEY' weiterhin sichtbar. Die Auflistung stellt Funktionstasten (Code > 80H) als F1...F3 dar; Control-Tasten (Codes > 20H) werden mit einem Pfeil gekennzeichnet.

**Standard-Funktionstastenbelegung von KOS6.05**

Die Ebene 1 der Funktionstasten F1...F13 ist nach dem Kaltstart von KOS6.05 folgendermaßen belegt:

F1	-	SOFTKEY	F8	-	RLOAD TIME
F2	-	STATUS	F9	-	SETTIME
F3	-	IODC LIST	F10	-	SETDATE
F4	-	IODC \$CPM=ACTIVE	F11	-	MAP
F5	-	WDIR	F12	-	CHMED
F6	-	IL P=*	F13	-	BASIC
F7	-	IL			

Diese Zuordnung kann durch entsprechende Konfiguration von KOS7.SYS (siehe Kommando 'KOSGEN') unterdrückt werden.

**Anwendung unter KOS4/5**

Ein Treiber \$FKEY kommuniziert zwischen Zuordnungsdatei xxx.KEY und dem Tastaturtreiber \$KEY. \$KEY kann unabhängig von SOFTKEY aktiviert und deaktiviert werden.

## 6. Umsetztreiber für CP/M-Aufrufe \$CPM

\$CPM ist notwendig für alle Programme, die Systemaufrufe über CP/M-'CALLS' durchführen, also z.B. FORTRAN, MBASIC, BASCOM, alle mit diesen Übersetzern erzeugten Programme, und für WORDSTAR, SuperCalc etc.

Bei fehlender Aktivierung wird das aufgerufene Programm abgebrochen.

Näheres zur Thematik "CP/M-Programme unter KOS" siehe Abschnitt Konzepte und Grundlagen, TB-A. Die Quelldatei CPM.SRC gibt weitere Detailinformationen. Sie ist Teil der UTILITY-Diskette.

CP/M-kompatible (CP/M-Versionen 1.4 und 2.2) Programme verwenden als Systemaufruf die Instruktion 'Call 5'. Über das Umsetzprogramm CPM.OBJ werden diese Funktionen in KOS-kompatible Aufrufe (RST8) transformiert. Dieses Programm ist als Treiber organisiert. Der CP/M-Translator \$CPM Version 2.97 ist zum einen separat verfügbar (Aktivierung mit "IODC \$CPM=ACTIVE") und zum anderen im Betriebssystemmodul KOSB.SYS (nur KOS6) eingebunden. Er benötigt als KOSB.SYS den geringeren Platz von 256 Bytes (100H Bytes) im Anwenderspeicher und braucht nicht mittels IODC-Kommando aktiviert werden.

Die Aktivierung erfolgt automatisch beim Laden von KOS V6.05, vorausgesetzt, daß das KOSB.SYS-Flag in KOS0.SYS gesetzt ist. Das Setzen und Rücksetzen des KOSB.SYS-Flags erfolgt durch die KOS Kommandofolge (siehe Techn.Beschreibung):

KOSGEN KOS0<---

Vor der ersten Verwendung des CP/M-Translators muß dieser eröffnet werden mit dem Systemkommando "öffne Treiber":

O \$CPM<---

Die Frage nach "debug mode" bei eventuellem mehrfachen Eröffnen wird mit N(ein) beantwortet. Ist der CP/M Translator nicht geöffnet, erscheint die Systemmeldung "\$CPM aktiv?". Die verwendete Translator-Version kann festgestellt werden durch Neuinitialisierung von \$CPM mit dem Systemkommando:

N \$CPM<---

Wir empfehlen in Zukunft nur noch mit CP/M-Translatermodul Version V2.97 (oder später) zu arbeiten.

Ab der Version 2.93 sorgt der Translator selbst dafür, daß Anwenderdateien geschlossen werden.

Bei früheren \$CPM-Versionen wurde eine Utility CLOSEX angeboten, mit deren Hilfe Dateien von Anwenderprogrammen aus selektiv geschlossen werden konnten. Dies ist ab \$CPM 2.93 nicht mehr erforderlich.

CLOSEX-Aufrufe müssen beim Arbeiten ab CP/M-Translator Version 2.93 aus den Anwenderprogrammen entfernt werden, da sie ansonsten zu einem undefinierten Systemverhalten führen können.

Einige CPM-Programme (z.B.: COBOL und MTPLUS) reservieren nur einen relativ kleinen Stackbereich. Dies kann eventuell zu Konflikten mit dem Multi-Tasking von KOS führen. Näheres zu diesem Thema entnehmen Sie bitte dem Abschnitt 7 CP/M 2.2 Kompatibilität in der KOS Technischen Beschreibung TB-A.

**Programmbeschreibung: \$CPM, Version 2.97**

\$CPM setzt CP/m 2.2 kompatible Systemaufrufe (Call 0005) in KOS kompatible Systemaufrufe (RST 8) um. \$CPM belegt 2 kByte (16 Segmente) des Anwenderspeichers. Folgende CP/M-Systemaufrufe sind implementiert:

**A: Simple Device Operations**

00:	System Reset	-
01:	Console Input	(KOS Kanal I-1/0-1)
02:	Console Output	(KOS Kanal 0-1)
03:	Reader Input	(KOS Kanal I-2)
04:	Punch Output	(KOS Kanal 0-4)
05:	List Output	(KOS Kanal 0-2)
06:	Direct Console I/O	(KOS Kanal I-1/0-1)
07:	Get I/O Byte	-
08:	Set I/O Byte	-
09:	Print String	(KOS Kanal 0-1)
10:	Read Console Buffer	(KOS Kanal I-1/0-1)
11:	Get Console Status	(KOS Kanal I-1)
12:	Return Version Number	-

**B: FDOS Operations****KOS-Funktion**


---

13:	Reset Disk System	-
14:	Select Disk	61H
15:	Open File	42H
16:	Close file	63H/6DH
17:	Search for First	64H
18:	Search for Next	65H
19:	Delete File	66H
20:	Read Sequential	77H
21:	Write Sequential	68H/78H
22:	Make File	49H
23:	Rename File	4AH
24:	Return Log-in Vector	-
25:	Return Current Disk	-
26:	Set DMA Address	-
27:	Get ADDR (Alloc)	-
28:	Write Protect Disk (*)	-
29:	Get Read Only Vector	-
30:	Set File Attributes (*)	-
31:	Get ADDR (Disk Parms)	-
32:	Set/Get User Code (*)	-
33:	Read Random	77H
34:	Write Random	78H
35:	Compute File Size	79H
36:	Set Random Record	-
37:	Reset Drive (*)	-
40:	Write Random with Zero File	78H

---

(\*) keine Wirkung unter KOS



**Offene Dateien:**

CP/M Programme können theoretisch beliebig viele Dateien gleichzeitig offen halten. Ebenso besteht keine Notwendigkeit, Dateien, die nur gelesen wurden, wieder zu schließen.

KOS hingegen erlaubt 16 offene Dateien und fordert das Schließen einer jeden geöffneten Datei, unabhängig davon, ob nur gelesen oder auch geschrieben wurde. Die daraus resultierende CP/M-Inkompatibilität wird von \$CPM 2.97 abgefangen. Die Folge ist:

- a) CP/M-Programme dürfen beliebig viele Dateien eröffnen
- b) ein Schließen von Dateien ist nur nach Schreibvorgängen notwendig
- c) Offen gelassene Dateien werden durch einen 'Warmstart' (JP 0 - normaler Abschluß eines CP/M-Programms) automatisch geschlossen. CP/M-Programme können somit keine offenen Dateien zurücklassen.

**KOS E/A-Fehlermeldungen**

Bei Zugriffsfehlern auf Speichermedien erzeugt \$CPM die Fehlermeldung:

**\$CPM-KOS I/O-Error: nn**

'nn' ist ein Fehlercode im Bereich zwischen 82 und 86. Bedeutung siehe Technische Beschreibung, Abschnitt "Betriebssystem KOS". Folgende Benutzereingaben sind möglich:

ESC bricht das Programm ab und kehrt zu KOS zurück

<CR> kehrt in das Anwenderprogramm zurück mit dem Fehlercode FFH in Register A.

## 7. Beispielprogramme auf der Utility-Diskette

### 7.1 INFO-Kommando (Ausgabe von ASCII-Zeichen)

Dateien:       INFO.SRC  
              INFMSG.SRC  
              INFO.OBJ  
              INFMSG.OBJ

**INFO** ist ein Beispiel eines Assembler-Programms:

- Kopf und Ende eines Assembler-Programms
- Schnittstelle zu KOS (Systemaufrufe)  
  Datei eröffnen, Record lesen  
  Zeichenausgabe auf Bildschirm usw.

INFMSG ist die Text-Datei für INFO. Alle Meldungen des Kommandos INFO sind in dieser separaten Datei abgespeichert, um Änderungen, z.B. für Fremdsprachen, zu erleichtern.

Durch Editieren der Datei INFO.SRC kann das Programm INFO in eine deutsche oder englische Version umgestellt werden.

### 7.2 BASIC-Programme

Dateien:       MAGIC.BAS  
              PIANO.BAS               (nur bei KOS 5.x)  
              TONLEIT.DAT           (nur bei KOS 5.x)

Dies sind Beispiele für BASIC-Programme. Insbesondere demonstriert **MAGIC** die leistungsfähige Grafik (Grafiksystem ist zu aktivieren!) und **PIANO** die akustische Ausgabe über den Lautsprecher (nur KOS 5.x).

Aufrufe:       BASIC "LOAD MAGIC<RUN"<---  
              BASIC "LOAD PIANO<RUN"<---

### 7.3 Hintergrund-Programme TIME

Dateien:           TIME.SRC     (KOS4/5/6)  
                  SETTIME     (KOS6)  
                  SETDATE     (KOS6)

#### KOS4/5: TIME und zugehörige Dateien

Beispiel für ein Programm, das als TASK im Hintergrund abläuft (siehe auch Systemkommando 'TASK').

Das Programm TIME zählt in der Task 'CLOCK' die im 20ms-Rhythmus laufenden Intervalle und bringt jede Sekunde die Anzeige der Uhrzeit auf den Bildschirm. Dazu dient die Task 'DISPLY'.

Das Kommando "RLOAD TIME<---" aktiviert diese beiden Tasks, die von da an im Hintergrund, gesteuert durch Interrupt, ablaufen. Die Deaktivierung der Tasks erfolgt durch das Systemprogramm TASK.

**Hinweis:** Die Genauigkeit der Zeitmessung durch TIME entspricht den Toleranzen des Videoszillators. Die Abweichungen liegen unter 1 %. Ein Abgleich kann gerätespezifisch in Software erfolgen.

#### KOS6: Unterstützung der 'Real Time Clock

Ab KOS6.04 werden beim Kaltstart Uhrzeit und Datum aus dem 'Real Time Clock'-Baustein der Computer Hardware ausgelesen, wobei automatisch die zugrunde liegende Hardwarekonfiguration verschiedener Systeme (PSI980 mit TCB/IOV, PSI908 mit KDT6) ermittelt wird. Uhrzeit und Datum werden in die dafür reservierten Systemspeicherstellen übertragen (Datum: OE/OF, Uhrzeit 1C/1D/1E), wo sie auf Wunsch im Sekundentakt verändert werden können. Dem Anwender stehen somit Datum und Uhrzeit in hardwareunabhängiger Form zur Verfügung.

Drei Utility-Programme stehen zur Verfügung:

SETTIME   -   Setzen der Uhrzeit  
SETDATE   -   Setzen des Datums  
TIME       -   Tasks zur permanenten Anzeige der Uhrzeit auf dem  
              Bildschirm, sowie zum periodischen Auslesen des RTC-  
              Bausteins. Beide Tasks können unabhängig voneinander  
              deaktiviert werden (Systemkommando TASK).

**Aufrufe:**   nn/KOS: SETTIME<---  
              nn/KOS: SETDATE<---  
              nn/KOS: RLOAD TIME<---

Die Programme SETDATE und SETTIME erlauben benutzerführend das Setzen von Datum und Zeit. Das Datum der Uhrenbausteine ist unabhängig vom Datum eines Mediums, das über das Systemkommando DATE verändert werden kann.

## 8. Testprogramme

### 8.1 Test der Peripherie-Bausteine (nur KOS4/5)

Aufruf: PSITEST<---

Dieses Testprogramm für KOS4/5 überprüft die wichtigsten Funktionen der hochintegrierten Peripherie-Bausteine. Bausteintyp, Betriebsart, Portadresse und Fehlerart werden in einer übersichtlichen Tabelle angezeigt. Eingeschriebenes und ausgelesenes Datenwort wird nur im Fehlerfall angezeigt.

Die Frage 'Testloop im Fehlerfall' ist immer mit 'N' zu beantworten (nur für Service bestimmt).

**Achtung:** Der PIO-Baustein kann nur erfolgreich geprüft werden, wenn er auch vorhanden ist. Eine Fehlermeldung des PIO kann durch eine externe Beschaltung bedingt sein.

**Warnung:** PSITEST überschreibt den letzten Sektor der letzten Spur (Sektor 16, Spur 76) auf Laufwerk 0.

### 8.2 Speichertest (nur KOS4/5)

Aufruf: MEMTEST<---

Hier handelt es sich um Testprogramme für den Arbeitsspeicher des Kontron PSI-Computers. Nach dem Aufruf laufen sie in einer endlosen Schleife und zählen jeden Schleifendurchlauf. Eventuell aufgetretene Fehler werden angezeigt.

Abbruch des Programms nur durch ESC (Escape).

### 8.3 Laufwerk/Disketten-Test TMED

Mit dem Programm TMED kann auf das Medium ein vom Anwender vorgegebenes Datenmuster geschrieben und beim Lesen geprüft werden. Es ist anwendbar für Disketten und Plattenstationen.

TMED liest bzw. schreibt physikalische Sätze (Records). Mit "Record" ist ein Datensatz mit 128 Bytes gemeint. Ein "Block" ist ein physikalisch speicherbarer Datensatz, der das 2 hoch n-fache eines Records sein muß. Bei Festplattenspeichern zum Beispiel beträgt die Länge eines Blocks 512 Bytes.

**Anwendung:**

Um die Richtigkeit der in einen Massenspeicher geschriebenen Daten prüfen zu können, ermöglicht TMED, ein eigenes Datenmuster festzulegen und auf das Medium zu schreiben. Das Datenmuster wird aus einem einzigen, frei wählbaren Byte gebildet. Dann können alle Records des Mediums gelesen und mit einem Bezugs-Record verglichen werden. Der Bezugs-Record wird aus dem vom Anwender festgelegten Datenmuster gebildet oder von TMED selbst vom Medium gelesen.

Voraussetzung für das Vergleichen der gelesenen Daten mit dem Bezugs-Record ist, daß alle Records des Mediums den gleichen Inhalt haben. Hier muß mit TMED erst ein gleichbleibendes Datenmuster auf das Medium geschrieben werden. Damit kann TMED auch die Vorformatierung für das BBR-Kommando (s. Systemkommandos, TB-B) übernehmen.

**Handhabung:**

Die von TMED benötigten Daten werden im Dialog eingegeben. Vor dem eigentlichen Schreibvorgang wird noch einmal zur Sicherheit eine Bestätigung des Anwenders eingeholt. TMED kann mit der ESC-Taste jederzeit abgebrochen werden. Eine Ausnahme hiervon ist die Schreibphase, in der die ESC-Taste den sofortigen Übergang in die Lesephase bewirkt.

Aufruf: TMED<---  
TMED -R<---

Die "-R"-Option ermöglicht das Testen (Schreiben und Lesen) des Reservebereichs von Medien mit "Bad Block Handling".

**Achtung:** Beim Schreiben auf den Reservebereich geht eine möglicherweise dort vorhandene "Bad Block"-Information verloren! In diesem Fall muß nach der Schreibphase der Medientreiber neu initialisiert werden (mit "N \$xxxx"). Eine Gewährleistung für das Beschreiben der Festplatte mit einem einheitlichen Muster mittels TMED -R<--- ist nur nach den folgenden Kommandos gewährleistet:

TMED -R<---  
N \$xxxx<---  
TMED -R<---

**Programmablauf:**

Als erstes verlangt TMED die Mediennummer. Danach wird gefragt, ob ein Datenmuster auf das Medium geschrieben werden soll. Wenn ja, muß dieses Datenmuster durch Eingeben eines Bytes angegeben werden.

Als nächstes wird der Prüfmodus festgelegt. Mit ihm wird bestimmt, ob TMED nach dem Erkennen eines Fehlers anhält und mit dem Anwender einen kurzen Dialog führt und ob jeder gelesene Record mit dem Bezugs-Record verglichen werden soll. Der Bezugsrecord wird, falls der Anwender kein eigenes Datenmuster schreiben läßt, von TMED selbst ermittelt.

In der darauf folgenden Programmausführung hat der Anwender vor dem ersten Schreibzugriff zu bestätigen, ob tatsächlich geschrieben werden soll. Beim Schreiben werden alle auf dem Medium enthaltenen Daten gelöscht.

Im Fehlerdialog wird, falls ein Datenvergleich durchgeführt worden ist, zunächst die Art des Fehlers angezeigt. Dies ist entweder die Meldung "Zugriffsfehler", wenn der Medientreiber einen Fehler gemeldet hat, oder "Datenfehler", falls die gelesenen Daten nicht mit dem Bezugs-Record übereinstimmen. Sodann kann der Anwender entscheiden, ob der Block noch einmal gelesen oder ob das Programm fortgesetzt werden soll.

Ist ein Record als fehlerhaft erkannt worden, wird das Programm mit dem ersten Record des nächsten Blocks fortgesetzt: d.h., der Rest des fehlerhaften Blocks wird nicht überprüft.

**Hinweise:** Eine Überprüfung von Disketten findet auch durch das Kommando FORMAT P V statt, siehe TB-B, Systemkommandos.

TMED und FORMAT P V stellen bei Verwendung einwandfreier Disketten die Funktion des Laufwerks sicher. Treten mit einer bestimmten Diskette in einem einwandfreien Laufwerk Fehler auf, so handelt es sich um Bit-Fehler auf der Diskette.

## 8.4 Promresidenter Testdebugger

Seit 1982 sind Kontron PSI-Systeme mit einem promresidenten Testdebugger ausgerüstet. Dieser befindet sich in den 3 (Kontron PSI80) bzw. 2 (Kontron PSI9xx) EPROMS, die auch den Urlader enthalten.

Der Testdebugger ist für die Überprüfung von Anlagen durch den Kontron-Service vorgesehen. Der promresidente Testdebugger gehört nicht zum spezifizierten Lieferumfang, Änderungen im Funktionsumfang bleiben vorbehalten.

In Anlagen, in denen der promresidente Testdebugger implementiert ist, können die folgenden Funktionen verwendet werden.

### 8.4.1 Aufruf und Verlassen des promresidenten Testdebuggers

Der Testdebugger kann nach einem RESET mit CTRL-K aufgerufen werden. Am Bildschirm erscheint der Test:

```
TESTDEBUGGER VERSION : (Versionsnummer) (Datum)
TD>
```

Es können nun Kommandos eingegeben werden, deren Syntax im wesentlichen der Syntax der KDM-Kommandos (siehe TB-F) entspricht.

Die ESC-Taste bewirkt einen sofortigen Abbruch des Programms und einen Warmstart. Eine Ausnahme bilden die Floppy-Disk Zugriffe, bei denen die ESC-Funktion unwirksam ist.

**Hinweis:** Da die ESC-Funktion einen sofortigen Abbruch des Programms zur Folge hat, können immer dann Probleme auftreten, wenn eine Interrupt-Service-Routine unterbrochen wird.

Der promresidente Testdebugger wird verlassen durch Eingabe von "K<---" oder durch RESET.

### 8.4.2 Speichertest

Für den Speichertest stehen zwei Kommandos zur Verfügung. MT und MX. Mit MT kann Speicher nur innerhalb einer Bank getestet werden. MX testet in allen Bänken (nur Kontron PSI9xx).

MT = Speichertest innerhalb einer Speicherbank

FORMAT 1: MT startadresse länge<---  
oder : MT startadresse - endadresse<---  
FORMAT 2: MT startadresse länge (anzahl loops)<---  
oder : MT startadresse - endadresse (anzahl loops)<---

Beispiel 1: MT 6000 1000<---  
oder : MT 6000-6FFF<---  
Beispiel 2: MT 6000 1000 33<---  
oder : MT 6000-6FFF 33<---

Der Speicherbereich von Adresse 6000H bis 6FFFH wird getestet. Die Angabe der Anzahl der Loops ist optional (Beispiel 2). Der Test läuft in Beispiel 1 solange, bis ein Abbruch mit <ESC> erfolgt. In Beispiel 2 wird der Speicher insgesamt 33 mal entsprechend der Anzahl der eingegebenen Loops durchgeführt.

Das Testergebnis erscheint am Bildschirm entweder als:

"NO ERROR DETECTED !"

oder als "MEMORY-ERROR AT ADDRESS"

unter Angabe der fehlerhaften Speicherstelle und deren IST- und SOLL-Wert.

**Hinweis:** Der Speichertest überschreibt den Speicherinhalt und darf nicht im Speicherbereich von 4000H bis 41FFH durchgeführt werden, da dies der Arbeitsspeicher des Testdebuggers ist.

MX = Speichertest über alle Bänke (nur Kontron PSI9xx).

FORMAT 1: MX startadresse länge<---  
oder : MX startadresse - endadresse<---  
FORMAT 2: MX startadresse länge (anzahl loops)<---  
oder : MX startadresse - endadresse (anzahl loops)<---

Beispiel: MX 4200-FFFF 2<---

Der Bereich von 4200-FFFF wird getestet, wobei im Bereich 8000..FFFF über alle Bänke getestet wird.

Mit MX kann sowohl die Funktionsfähigkeit des Memory Mappers als auch die aller Speicherbänke getestet werden. Der Ablauf dabei ist folgendermaßen:

- a) Die Bänke 1..3 werden mit einem bestimmten Datermuster gefüllt (1. Hälfte Bank 1 mit 02. 2. Hälfte mit 03 usw.).
- b) In Bank 0 wird der angegebene Bereich getestet (genau wie bei MT-Kommando).
- c) Falls Test in Bank 0 erfolgreich, wird geprüft ob die Datermuster in den folgenden Bänken unverändert geblieben sind. Falls nicht, erfolgt in der entsprechenden Bank eine Fehlermeldung: ERROR IN BANK X (X = 1..3).  
Der Mapper bleibt in diesem Fall auf diese Bank programmiert, so daß die betreffende Bank sofort überprüft werden kann.
- d) Falls kein Fehler auftritt, wird die erste Hälfte von Bank 1 in den Bereich 8000..FFFF gemappt (entspricht dem Kommando MA 2).
- e) Der angegebene Bereich wird getestet.
- f) Das Datermuster in den folgenden Bänken wird geprüft.
- g) Falls kein Fehler, wird die zweite Hälfte von Bank 1 nach 8000..FFFF gemappt usw.

Während des Tests wird ausgegeben, welche Bank gerade getestet wird, bei erfolgreichem Ablauf ergibt sich (pro Durchlauf) der Ausdruck:

MEMORY TEST BANK 0 1 1 2 2 3 3 NO ERROR DETECTED !

Da die Bänke 1..3 in zwei Hälften getestet werden, erscheint die Bank-Nummer zweimal im Ausdruck.

Im Bereich 0000..7FFF liegt immer die 1. Hälfte von Bank 0, dies bedeutet, daß bei dem oben angegebenen Beispiel der Bereich 4200-7FFF von Bank 0 mehrfach (pro Durchlauf 7 mal) getestet wird.

Beispiele für mögliche Fehlermeldungen:

MEMORY TEST BANK 0 ERROR IN BANK 1

Bank 0 wurde erfolgreich getestet, in Bank 1 wurde das am Anfang eingetragene Datermuster (02/03) nicht gefunden. Es liegt entweder ein Speicherfehler in Bank 1 vor, oder die Mapper-Logik weist Fehler auf.

MEMORY TEST BANK 0 ERROR IN BANK 3

Bank 0 erfolgreich getestet, in Bank 3 wurde das eingetragene Datermuster (06/07) nicht gefunden. Mapper vermutlich in Ordnung, wahrscheinlich Speicherfehler in Bank 3.

Falls nach einer Fehlermeldung oder nach <ESC> abgebrochen wird, bleibt der Memory Mapper in dem Zustand, in dem er zuletzt war.

Bei normalen Ende (ohne Fehler) wird der Mapper wieder auf Bank 0 initialisiert.



### 8.4.3 Floppy Disk-Laufwerkstest (nicht bei 'H'-Systemen)

Der promresidente Testdebugger enthält auf Adresse 1800H ein Floppy-Disk-Testprogramm. Dieses Programm ist interaktiv und kann mit "J 1800<---" angesprungen werden.

Der Test führt nur Lesezugriffe aus, er kann also mit jeder Diskette durchgeführt werden. Der Ablauf sieht in etwa folgendermaßen aus:

Eingabe im Testdebugger: J 1800<---

DISKTEST VERSION 1.51 VOM 30.11.1982 (derzeit aktuelle Version)

DRIVE ID? 1<---

VERSUCHE: 5<--- (Eingabe dezimal, 2-stellig)

LOOPS: 2<--- (Eingabe hexadezimal, 2-stellig)

MODE 1/2: 2<---

Es erfolgt nun ein Test in Laufwerk 0, 5" Mini, double density, kein DMA. Dabei wird Spur für Spur gelesen, jeweils maximal 5 Versuche, dabei auftretende Hard- oder Soft-Errors werden aufgelistet. Der Rest wird in dem oben angegebenen Beispiel zweimal durchlaufen.

Bei MODE 2 erfolgt die Darstellung der Fehler in einem sehr anschaulichen Histoprogramm, das in etwa folgendes Aussehen hat:

```

!   Softerrors
!
!                                     X
!                                     X
!                                     X
!
!IO---0---1---0---2---0---3---0---4---0-->
!                                     X   Spur
!                                     X
!
!   Harderrors

```

Bei dem Test traten somit 2 Harderrors in Spur 38 und 3 Softerrors in Spur 42 auf.

Über den angezeigten SCALE FACTOR wird der Wert eines Tabelleneintrags (X) variiert:

```

SCALE FACTOR 0010 : X bedeutet: 1 Hard- oder Softerror
SCALE FACTOR 0020 : X bedeutet: 2 Hard- oder Softerrors
SCALE FACTOR 0040 : X bedeutet: 4 Hard- oder Softerrors

```

Bei MODE 1 erfolgt eine Fehlermeldung im folgenden Format:

PROBLEM AT TRACK XX WITH .....

Der Test kann mit W angehalten und mit jeder anderen Taste abgebrochen werden.

Wird bei "VERSUCHE" keine Zahl eingegeben, so werden 99 Versuche gemacht, wird bei "LOOPS" keine Zahl eingegeben, so werden 9999 Durchläufe ausgeführt.

Bei der Frage nach der Laufwerksidentifikation (DRIVE-ID) sind folgende Eingaben möglich:

DRIVE-ID	Bedeutung	Laufwerk	
1	5" Mini	0	double density ohne DMA
2	5" Mini	1	double density ohne DMA
3	5" Mini	0	single density ohne DMA
4	5" Mini	1	single density ohne DMA
9	5" Winchester		
A	5" Mini	0	double density mit DMA
B	5" Mini	1	double density mit DMA
C	5" Mini	0	single density mit DMA
D	5" Mini	1	single density mit DMA

Bei double-sided Laufwerken kann durch eine führende "1" bei der Drive-ID die Seite 2 angesprochen werden.

Beispiel: 12" bedeutet Seite 2 des zweiten Laufwerks in double density-Betrieb ohne DMA.

Im allgemeinen gilt die Zuordnung: Laufwerk 0 = rechts oder oben  
Laufwerk 1 = links oder unten

### 8.5 Messung von Programmlaufzeiten (KOS6)

KOS6 stellt eine Utility zur Messung von Programmlaufzeiten durch die Task TCNT.OBJ und das Anzeigeprogramm PCNT.COM zur Verfügung. Diese Funktionen sind sowohl von der KOS-Ebene aus, wie auch aus Anwenderprogrammen heraus aufrufbar.

#### Funktion der Task TCNT.OBJ

Diese Task zählt die Laufzeit von Aktivitäten in Einheiten von 20ms. TCNT verwaltet 256 Softwareregister, die den Aktivitäten #0 bis #255 zugeordnet werden können. TCNT wird durch "RLOAD TCNT<---" aktiviert.

TCNT empfängt Kommandos und Daten über die Speicherzellen 2BH/2CH. 2BH enthält die Nummer n der Aktivität, 2CH das zugehörige Kommando:

- 0: STOP
- 1: Alle Zähler zurücksetzen (RESET)
- 2: Zähler für Aktivität n
- 3: Löschen Zähler für Aktivität n

Der Zähler #255 hat eine Sonderrolle, er läuft kontinuierlich von RESET bis STOP.

#### Anwendung unter KOS6:

Durch das C-Kommando werden Zählvorgänge eingeleitet und beendet (n=0...9, alle weiteren Zähler unter Programm ansprechen):

- C n G<--- Starte Zähler n neu (GO)
- C n H<--- Stop Zähler n (HALT)
- C n R<--- RESET alle Zähler

Die gemessenen Werte werden durch das PCNT-Kommando angezeigt.

Beispiel:

```
C 1 GO;ASM =FILE;C 1 HALT;PCNT<---  
C 2 G;ASM =FILE/L;C 2 H;PCNT<---
```

Anwendung aus Assembler Programmen heraus:

a) Start counting

```
start:    ld a,1           ;reset all counters  
          ld (2ch),a  
          call sync       ;wait until operation is completed  
          ld a,n  
          ld (2bh),a      ;define process number #n  
          ld a,2          ;start counting command  
          ld (2ch),a  
          ret  
  
sync:     ld hl,1fh        ;this location is incremented  
          ld a,(hl)        ;by the KOS task scheduler  
          inc a            ;this loop guarantees that the  
                          ;reset command has been recognized  
                          ;by TCNT  
  
sync.loop: cp (hl)  
           jr nz,sync.loop  
           ret
```

b) Stop counting

```
stop.cnt: ld a,0          ;stop command  
          ld (2ch),a  
          ret
```

## 9. Muster eines Ausgabe-Treibers

```
;Name of the program:          UP.SRC
;Date of creation:            28.06.1983
;Date of last modification:   -
;Current release number:      1.0
```

---

```
.comment#
```

This is a universal printer driver.  
 It is generally interrupt driven and uses hardware handshake.  
 By setting the switch 'interrupt' to '0' you may  
 generate a non-interrupt version (e.g. for KOS5 systems  
 if the driver should be located in the video memory area).  
 By setting of assembler switches it can be specified  
 for the serial or the parallel interface.  
 If the switch 'filter' is set to '1', a simple filter module  
 is included, which only inserts "auto linefeeds" if wanted.  
 A "carriage return" in the OPEN routine and "carriage return"+  
 "formfeed" in the CLOSE routine are sent to the printer.

---

Choose wanted driver configuration by setting assembler switches:

---

```
#
interrupt      equ 1      ;'1' generates an interrupt controlled
                        ; driver

serial         equ 0      ;'1' generates a serial driver

parallel       equ 1      ;'1' generates s parallel driver

kos5           equ 0      ;'1' generates a driver for KOS5.xx

filter         equ 1      ;'1' includes the filter module
```

```
;If "serial" has been set to '1', the following switches have
; to be set:
```

```
SIOA           equ 0      ; SIOA or SIOB may be used
SIOB           equ 1      ; (on KDT5, KDT6 or TCB/Z80)
```

```
;Define baudrate here:
```

```
bd300          equ 0
bd600          equ 0
bd1200         equ 0
bd2400         equ 0
bd4800         equ 0
bd9600         equ 1
```

;Note:

;-----

;The serial driver for KOS5 must not be used for KOS6  
;because of the different baudrate programming.

;If "parallel" has been set to '1', the following switch has  
; to be set to 1, if an iprime pulse is wanted, else to 0:

iprime            equ 0    ;'1' if an iprime pulse during init routine  
                         ; is wanted

page

.comment#

Connections:

-----

serial:

-----

Printer -        PSI

-----

Pin            Pin

3            3

\*\*           4        ;\*\* = 4 for DAISY 45  
                 ;       11 for OKI Microline

7            7

parallel:

-----

Uses the CENTRONICS interface on KDT5, KDT6 or TCB/Z80.

Printer -		PSI(Rev.6) - PSI(Rev.5)	
Pin	Signal	Pin	Pin
1	strobe	1	12
2	data1	2	19
3	data2	3	17
4	data3	4	18
5	data4	5	16
6	data5	6	4
7	data6	7	3
8	data7	8	2
9	data8	9	15
11	busy	11	21
31	iprime	31	11
14,16,		14,16,	1,6,9,14
19...30	gnd	19...30	

#

page  
 .comment#

# CONDITIONING OF LIOD

Please define here these conditional switches:

```
input.driver      :for input or bid. drivers
output.driver     :for output or bid. drivers
pstat.info        :if PSTAT format is desired (*)
direct.87         :if function 87h is implemented
                  :in your driver (*)
```

(\*) Those Switches are optional. If they are not defined they are considered to be 0!

```
-----#
input.driver      equ 0
output.driver     equ 1
pstat.info        equ 1
direct.87         equ 1
```

page

```
id.text macro
defm 'UP'
if serial
if SIOA
defm 'A'
else
defm 'B'
endif
if interrupt
defm 'I'
endif
ifndef kos5
defm ' 1.0(//6.0)'
else
iff kos5
defm ' 1.0(//6.0)'
else
defm ' 1.0(/5.3/)'
endif
endif
else
defm 'P'
if interrupt
defm 'I'
endif
ifndef kos5
defm ' 1.0(//6.0)'
else
iff kos5
defm ' 1.0(//6.0)'
else
iff graphic
defm ' 1.0(/5.3/6.0)'
else
defm ' 1.0(/5.3/)'
endif
endif
endif
endif
defm '-28.06.1983'
endm
```

page

```
;Here we include our logical I/O-driver LIOD.SRC  
;-----
```

```
include LIOD.SRC
```

```
;Here we include the FILTER module  
;-----
```

```
if filter  
include UP.FIL  
endif
```

subttl Hardware dependant routines

page

```
;Here we include the hardware I/O-driver HIOD  
;-----
```

```
include HOD.SRC
```

page

```
INIT.MSG:  
defb CR,LF,LF  
defm "Universal Printer Driver"  
defm " - 28.06.1983"  
defb CR,LF  
if serial  
defm "serial"  
if SIOA  
defm " (SIOA"  
else  
defm " (SIOB"  
endif  
if bd300  
defm "/300 Bd)"  
endif  
if bd600  
defm "/600 Bd)"  
endif  
if bd1200  
defm "/1200 Bd)"  
endif  
if bd2400  
defm "/2400 Bd)"  
endif  
if bd4800  
defm "/4800 Bd)"  
endif  
bd9600  
defm "/9600 Bd)"  
endif  
else  
defm "parallel"  
endif  
if interrupt
```

```

defm " - interrupt controlled"
if kos5
defb CR,LF,LF
defm "Attention:"
defb CR,LF
defm "Driver has to be located outside memory area"
defm " 8000 to BFFF (Video Memory)!"
endif
endif
defb CR,LF,0

```

```

title driver: UP - UP - UP

```

```

end

```

### 9.1 Benutzerspezifische Anpassung eines seriellen Treibers

Eine Benutzer-spezifische Anpassung eines seriellen Treibers an die vorgegebene Hardware kann vom Anwender im vorliegenden Beispiel des UP-Treibers durch Eintragen der gewünschten Änderungen in UP.SRC, UP.FIL, LIOD.SRC und HOD.SRC vorgenommen werden. Das Assemblieren des UP.SRC erfordert das Vorhandensein dieser Quelltexte auf dem Mastermedium. Aufruf Beispiel: ASM UPx =UP<--- x=A,B,P. Anpassungen an vorgegebene Hardware, wie Drucker oder ähnliche periphere Geräte, sollen im Beispiel des UP.SRC und HOD.SRC aufgezeigt werden. Im UP.SRC können Einstellungen wie Anpassung an gewünschtes Betriebssystem (KOS5/KOS6), Interruptsteuerung, gewünschte Schnittstelle (seriell A, seriell B, parallel) Filter und Baudrate durch Setzen oder Rücksetzen der zugehörigen Equates vorgenommen werden. Sollen weitere Anpassungen eines seriellen Treibers durch erneutes Festlegen von Stopbits, Parity, Handshake, Bits/Character, Transmitter enable, Receiver enable etc. erfolgen, so kann dies durch Ändern der SIO-Programmierung im HOD.SRC erreicht werden.

Vor dem Assemblieren des Treiberquelltextes (UP.SRC) empfiehlt es sich, den geänderten HOD.SRC zur Unterscheidung umzubenennen und den neuen Namen im UP.SRC einzutragen, da HOD.SRC, LIOD.SRC und UP.FIL beim Assemblieren des UP.SRC automatisch eingebunden werden.

Im folgenden soll anhand von Beispielen aufgezeigt werden, wie die SIO-Programmierung über HOD durch Festlegen von Parametern den Benutzerwünschen angepaßt werden kann.

#### 1. Stopbits, Parity:

sioclk equ	**** 11xxb	2 Stopbits
sioclk equ	**** 10xxb	1 1/2 Stopbits
sioclk equ	**** 01xxb	1 Stopbit
sioclk equ	**** yy00b	no parity
sioclk equ	**** yy01b	odd parity
sioclk equ	**** yy11b	even parity

Dabei ist \*\*\*\* = 0100 für 600 Baud bis 9600 Baud und \*\*\*\* = 1000 für 300 Baud, ferner ist xx die Parity und yy die Stopbitzahl. Der sioclk-Wert wird durch HOD ins SIO-Register 4 eingetragen.



## 2. Handshake, Bits/Character, Transmitter/Receiver enable:

Diese Parameter werden durch Eintragen der gewünschten Werte in der sio.table festgelegt.

Der Inhalt von Register 3 bestimmt, ob 7 Bits/Character oder 8 Bits/Character empfangen werden sollen und ob RTS abgefragt werden soll (Hardware Handshake).

11x0 000yb	8 Bits/Character
01x0 000yb	7 Bits/Character
zzx0 0000b	Receiver disable (keine Daten empfangen nur Ausgabe-Treiber)
zzx0 0001b	Receiver enable
zz00 000yb	RTS nicht abfragen
zz10 000yb	RTS abfragen (auto enables)

Dabei ist x der RTS-Parameter, y der Receiver-on/off-Parameter und zz die Bitzahl/Character.

Der Inhalt von Register 5 bestimmt, ob 7 Bits/Character oder 8 Bits/Character gesendet werden sollen.

1110 x000b	8 Bits/Character
1010 x000b	7 Bits/Character
1yy0 0000b	Transmitter disable (keine Daten senden nur Eingabe-Treiber)
1yy0 1000b	Transmitter enable

Dabei ist x der Transmitter-on/off-Parameter und yy die Bitzahl/Character.

## 10. Hilfsprogramme

### 10.1 CONED

Aufruf: CONED /wdir1/mn1:name1.typ1 /wdir2/mn2:name2.typ2<---

Voreinstellung:

mn1 (Quelle)	= Mastermedium
mn2 (Ziel)	= mn1 (Quelle)
name2 (Ziel)	= name1 (Quelle)
typ1 (Quelle)	= .SRC
typ2 (Ziel)	= .BAS
wdir1	= aktuelles wdir
wdir2	= aktuelles wdir

CONED wandelt Dateien im KOS-EDIT-Format um in ein KOS-CP/M-typisches Format, CR (Code 0dh) wird gewandelt in CR-LF (Codefolge 0dh 0ah). Die KOS-EDIT-Dateiendekennung 0dh 0ffh 0dh 1ah 0ffh....0ffh wird gewandelt in die KOS-CP/M-Dateiendekennung 0dh 0ah 1ah 0ffh...0ffh.

Anwendungsbeispiele:

- Wandlung von KOS-Editor-Dateien in Dateien, welche zu WordStar (MicroPro) Format-kompatibel sind.
- Wandlung von Basic-Programmen, welche unter dem KOS-Editor EDIT erstellt wurden in solche, welche zum MBASIC (MicroSoft) Interpreter Format-kompatibel sind. Basic-Programme, welche unter dem KOS-Editor erstellt wurden, müssen mittels CONED vor dem Compilieren mittels BASCOM (MicroSoft) konvertiert werden. Dabei hat CONED als Voreinstellung den Dateityp .SRC bei der Eingabedatei und den Dateityp .BAS bei der Ausgabedatei.

### 10.2 WSEDIT

WSEDIT konvertiert unter WordStar (MicroPro) erstellte Dateien in solche, welche unter dem KOS-Editor EDIT bearbeitet werden können. Die Konvertierung erfolgt unter englischer Benutzerführung!

Zuerst wird die zur konvertierende Datei erfragt:

Enter source file name.... (mn:) name1.typ

dann die anzulegende Zieldatei:

Enter destinationfile name... (mn:) name2.typ

Anschließend erfolgt die Frage nach dem Löschen von WordStar Punktkommandos:

Delete dot commands (Y/N)?...

Nach der Konvertierung erscheint die Frage nach einer weiteren Konvertierung:

Transfer complete,  
Do you wish to copy another file (Y/N)?...

### 10.3 PT (nur KOS6)

Aufruf: PT<---

Das Programm PT (PSI Terminal) ermöglicht die Verwendung eines Kontron PSI9xx/9xxx Computer Systems als einfaches Terminal. Die Verbindung erfolgt über eine serielle Schnittstelle, wobei die Anschlüsse 2 (Receive Data), 3 (Send Data) und 7 (Signal Ground) der gewünschten seriellen Schnittstelle des Kontron PSI Systems mit den entsprechenden Anschlüssen des zu verbindenen Systems verdrahtet werden müssen.

Das Programm PT empfängt Zeichen über Kanal I-1 (Standard: \$KEY/Tastatur) und schickt diese auf Kanal O-5, parallel dazu werden Zeichen über Kanal I-5 empfangen und über Kanal O-1 (Standard: \$MON/Monitor) ausgegeben.

Den Kanälen I-5/O-5 können beliebige Ein-/Ausgabe-Treiber zugeordnet werden, es ist empfehlenswert, interruptgesteuerte Treiber zu verwenden.

Beispiel: IODC \$INTx=ACTIVE I-5=\$INTx O-5=\$INTx<--- x=A oder B  
PT<---

Wird über Kanal I-1 (Standard: \$KEY/Tastatur) das Zeichen CTRL-K (Code Obh) empfangen, so kehrt PT zu KOS zurück, d.h. PT kann über die Tastatur mit CTRL-K abgebrochen werden, falls keine Verbindung zum externen System zustande kommt oder das Programm beendet werden soll.

PT verwendet folgende Systemaufrufe:

- 82h - Input Channel Status (SFC=1)
- 86h - Character/Byte Output.

## **Kontron KOS Assembler, Linker, Crossreference-Generator**

### **BESCHREIBUNG**

Stand: Juli 1984

Version: 4.33/5.44/5.54/6.05

Copyright by Kontron ELEKTRONIK, Eching/München

Die folgende Beschreibung gilt der Handhabung des relocativen Makroassemblers ASM.COM der Kontron PSI-Systeme unter dem Betriebssystem KOS. Auskunft über den Z80-Befehlssatz gibt die Z80-Befehlssatz-Beschreibung.

Daran anschließend finden Sie die Beschreibung der Handhabung des Kontron PSI Linkers LINK unter dem Betriebssystem KOS. LINK erzeugt aus relocativen Objektdateien ablauffähige Programme. LINK arbeitet speicher-orientiert. Die maximal erzeugbare Programmsegmentgröße ist durch den während des Linkens verfügbaren Speicherraum gegeben.

Der Crossreference-Generator CROSS dokumentiert die zwischen den Modulen bestehenden Querbeziehungen.



## I N H A L T

- 1. Assembler der Kontron PSI-Systeme
  - 1.1 ASM Kommandosyntax
  - 1.2 Dateiformat für ASM-Quellprogramme, Symbole und Konstante Werte
  - 1.3 Arithmetisch/logische Ausdrücke
  - 1.4 Pseudooperationen
    - 1.4.1 Datendefinition durch DEFB, DEFW, DEFM
    - 1.4.2 Speicherdefinition
    - 1.4.3 Symboldefinition durch EQU, DEFL
    - 1.4.4 Symboldeklaration, GLOBAL, EXTERNAL
    - 1.4.5 Referenzadrezähler, ORG-Anweisung, Segmentierung
    - 1.4.6 Makros von ASM: REPT, IRP, IRPC, EXITM, LOCAL
    - 1.4.7 Allgemeine Assemblersteuerung, COMMENT, PRINTX, RADIX
    - 1.4.8 Bedingte Assemblierung
    - 1.4.9 Include Anweisung
    - 1.4.10 Moduldokumentation, TITLE, SUBTTL, END
    - 1.4.11 Listingsteuerung, LIT, XLIT, CREF, XCREF, LALL, SALL, XALL
  - 1.5 ASM-Fehlermeldungen
  - 1.6 ASM-Listing Format
  - 1.7 Assembler/Linker - Kommandodateien
- 2. Linker für Kontron PSI-Systeme
  - 2.1 Übersicht LINK
  - 2.2 Kommandoaufruf LINK
  - 2.3 LINK-Fehlermeldungen
  - 2.4 LINK-Beispiele
- 3. Crossreference-Generator



## 1. Assembler der Kontron PSI-Systeme

Der Kontron KOS-Assembler ist auf jeder KOS-Systemkommando-Diskette unter dem Namen ASM.COM abgelegt. Der Aufruf des Assemblers erfolgt von KOS aus, wobei das Parameterfeld des Kommandos gewöhnlich den Namen der Quelldatei enthält. Zusätzliche optionale Parameter bestimmen den Namen der vom ASM erzeugten Listing- und Objektdateien. Der Assemblerlauf kann jederzeit durch Eingabe von <ESC> abgebrochen werden.

### 1.1 ASM-Kommandosyntax

Das allgemeine Format des Assembleraufrufs ist: (die Taste 'RETURN' wird durch <CR> oder '<--->' symbolisiert)

ASM mn:objdatei,mn:lstdatei =mn:quelldatei.typ<CR>

Hierbei bedeuten:

mn:objdatei - Name und Typ der vom ASM auf dem Medium mn erzeugten Objektdatei.

Voreinstellungen:

mn = Mastermedium  
typ = OBJ

mn:lstdatei - Name und Typ der vom ASM auf dem Medium mn erzeugten Listingdatei.

Voreinstellungen:

mn = Mastermedium  
typ = PRN

mn:quelldatei - Name und Typ des zu übersetzenden Quellprogramms auf dem Medium mn.

Voreinstellungen:

mn = Mastermedium  
typ = SRC (Source)

Die Angabe von Working Directories ist nicht möglich, alle Dateien sind im aktuellen wdir.

Die Reihenfolge der Parameter ist bindend, d.h.: der erste Parameter links des Gleichheitszeichens kennzeichnet immer die Objektdatei, der zweite immer die Listingdatei. Als zusätzliche Trennzeichen sind Leerzeichen in beliebiger Anzahl möglich.



Alle Parameter auf der linken Seite des Gleichheitszeichens sind optional. Fehlen sie, so gilt als Voreinstellung:

- eine Objektdatei mit dem Namen des Quellprogramms und dem Typ 'OBJ' wird auf dem Mastermedium erzeugt
- eine Listingdatei wird nicht erzeugt.

#### Beispiele für ASM-Aufrufe:

ASM 1:TEST.ABC,1:TEST.LST =0:TEST<---

Assembliert die Datei TEST.SRC von Medium 0 und erzeugt auf Medium 1 die Dateien TEST.ABC (Objektdatei) und TEST.LST (Listingdatei).

ASM TEST, =TEST<---

Assembliert die Datei TEST.SRC auf dem Mastermedium und erzeugt dort die Objektdatei TEST.OBJ.  
Eine Listingdatei wird nicht erzeugt.

Fehlt links des Gleichheitszeichens jeglicher Parameter, so wird als Sonderfall des ASM-Aufrufs eine Objektdatei erzeugt, also:

ASM =TEST<---     oder  
ASM ,=TEST<---

Assembliert die Datei TEST.SRC und erzeugt die Objektdatei TEST.OBJ. Falls keine Mediennummer für die Quelldatei angegeben ist, wird die automatische Dateisuchsequenz von KOS gestartet. Zu beachten ist, daß Listing- und Objektdateien, wenn nicht anders spezifiziert, immer auf dem Mastermedium erzeugt werden.

**ASM-Aufruf mit Listingausgabe**

Die Ausgabe des Listings auf den Sichtschirm oder ein beliebiges anderes Peripheriegerät des Kontron PSI-Computers kann bereits während des Assembliervorganges erfolgen. In diesem Fall wird keine Listingdatei auf Diskette erzeugt.

**Beispiele:**

ASM TEST,MON: =TEST<---

Assembliert die Datei TEST.SRC und erzeugt die Objektdatei TEST.OBJ. Das Listing wird auf den E/A-Treiber \$MON (Sichtschirm) des Kontron PSI-Computers ausgegeben.

ASM TEST,LST: =TEST<---

Wie oben, das Listing wird allerdings auf den Ausgabekanal 0-2 gegeben. Dieser kann beispielsweise einem Drucker zugeordnet werden. Treten Assemblerfehler auf, so werden diese in jedem Fall auch auf den Sichtschirm (Kanal 0-1) ausgegeben.

**ASM-Aufruf ohne Parameterangabe**

Wird der Assembler ohne Parameter aufgerufen, so antwortet ASM mit dem Prompt-Zeichen 'ASM:' und ist daraufhin zur Aufnahme von Kommandos bereit (Kommando Modus). Diese Kommandos sind identisch mit dem Parameterfeld des Assembleraufrufs (siehe Abschnitt 1.1).

**Beispiel:**

ASM<---

ASM:=TEST           Assembliert die Datei TEST.SRC und erzeugt die  
Objektdatei TEST.OBJ

ASM kehrt daraufhin nicht selbst ins Betriebssystem zurück, sondern erwartet nach Ausgabe der Prompt-Zeichen weitere Kommandos. Die Rückkehr ins Betriebssystem kann durch Eingabe des Kommandos 'KOS' veranlaßt werden. ASM verwendet Kanal I-1 für Eingaben.

**ASM-Funktionsschalter**

ASM verarbeitet eine Reihe von Zusatzparameter, die eine Schalterfunktion auslösen, d.h. diese Parameter schalten eine bestimmte Funktion des ASM ein und gleichzeitig eine jeweils voreingestellte aus.

Zusatzparameter werden unmittelbar nach dem Namen der Quelldatei getrennt durch einen Schrägstrich (/) angegeben.

**Möglich sind:**

- /O     Alle numerischen Werte im Listing werden im oktalen Zahlensystem ausgegeben
- /H     Alle numerischen Werte im Listing werden im hexadezimalen Zahlensystem ausgegeben (dies ist Voreinstellung)
- /L     Eine Listingdatei soll erzeugt werden
- /C     Eine für das Programm 'CROSS' vorbereitete Listingdatei soll erzeugt werden
- /I     Quellcode ist in Intel 8080-Mnemonic geschrieben
- /Z     Quellcode ist in ZILOG Z80-Mnemonic geschrieben. Dies ist die Voreinstellung von ASM.

**Beispiele:**

ASM =TEST/C<---

Assembliert die Datei TEST.SRC und erzeugt eine für das Programm CROSS vorbereitete Listingdatei mit dem Namen TEST.CRF. Diese Datei kann anschließend mit dem Programm CROSS zu einer Symbolreferenzdatei verarbeitet werden (siehe CROSS-Beschreibung).

ASM =1:TEST1/L<---

Assembliert die Datei TEST1.SRC von Medium 1 und erzeugt Listing- und Objektdatei (TEST1.OBJ bzw. TEST1.PRN) auf demselben Medium.

## 1.2 Dateiformat für ASM-Quellprogramme, Symbole und konstante Werte

ASM verarbeitet KOS-kompatible Quelldateien (Sourcefiles), die gewöhnlich mit dem PSI-Editor (EDIT) erstellt wurden. Folgende Regeln sind bei der Erstellung von Quellprogrammen zu beachten:

- a) die maximale Zeichenanzahl pro Zeile beträgt 132 Zeichen (für den ASM, bei EDIT gilt max. 71).
- b) Klein- oder Großschreibung ist möglich, wird aber nicht zur Unterscheidung von Symbolen verwendet.
- c) Deutsche Umlaute können nur in Kommentaren und Zeichenfolgen verwendet werden.

### Format von Quellprogrammzeilen

Jedes Quellprogramm besteht aus einer Reihe von Zeilen, deren Inhalt folgendem allgemeinen Format entspricht:

Marke: Operator Argument ;Kommentar

Eine solche Zeile (bis einschließlich des Arguments) wird im englischen Sprachgebrauch als 'Statement' bezeichnet. Alle vier Bestandteile einer solchen Anweisungszeile sind in der Regel optional.

#### Definition:

Ein Statement ist im allgemeinen Sinn eine Anweisung für das Übersetzerprogramm (hier ASM), die entweder selbst zum erzeugten Code beiträgt, oder die Funktion des Übersetzers beeinflusst (Pseudooperationscode).

Es ist nicht notwendig, daß Anweisungen in Spalte 1 beginnen. Zur Verbesserung der Lesbarkeit können Leerzeichen (ASCII-Code: 20H) und/oder Tabulationszeichen (ASCII-Code: 09H) verwendet werden. Diese beiden Zeichen sind überall innerhalb einer Quelltextzeile als Trennzeichen zugelassen.

Marken (engl.: Labels) stehen immer zu Beginn einer Quelltextzeile, vorausgehende Trennzeichen nicht berücksichtigt. Einer Marke muß unmittelbar ein Doppelpunkt folgen.

Nach der Marke steht im allgemeinen Fall der Operator. Hierbei handelt es sich entweder um

- einen MACRO-Aufruf
- einen Operationscode (Z80-Mnemonic)
- eine Pseudooperation oder
- einen arithmetisch/logischen Ausdruck

Arithmetisch/logische Ausdrücke im Operatorfeld betrachtet der Assembler als DEFB-Pseudooperation, also:

```
LABEL1: 24 - 5
          entspricht
LABEL1: DEFB 24-5
```

Kommentare dienen zur Dokumentation des Quellprogramms. Sie beginnen immer mit einem Strichpunkt (Semikolon) und enden mit Carriage Return (RETURN-Taste). Eigene Kommentarzeilen (ohne Anweisungen) sind zulässig (siehe auch Abschnitt - COMMENT Pseudooperation).

## Symbole

Symbole sind alphanumerische Zeichenketten von im Prinzip beliebiger Länge. Signifikant und vom Programm CROSS verwendet sind allerdings nur die ersten sechzehn Zeichen eines Symbols. Namen von Externals/Globals sind beschränkt auf 7 Zeichen.

Folgende Zeichen dürfen verwendet werden:

A-z    alle Groß- und Kleinbuchstaben  
0-9    alle Ziffern

und die Zeichen

'\$', '?' und '.'

Das erste Zeichen eines Symbols muß ein Buchstabe sein. Es sei nochmals darauf hingewiesen, daß Kleinbuchstaben in Großbuchstaben umgewandelt werden und deshalb nicht zur Differenzierung von Symbolen dienen:

SYMBOL = Symbol

## Numerische Konstante

Numerische Konstante sind Zahlenwerte zu einer beliebigen Basis im Bereich von 2 (Binärsystem) bis 16 (Hexadezimalsystem). Das dezimale Zahlensystem ist Voreinstellung. Eine Änderung der Basis ist jederzeit durch die Pseudooperation RADIX möglich. Wird die Basis 10 überschritten, so kennzeichnen die Buchstaben A bis F die der 9 folgenden 'Ziffern'. Ist die erste Stelle eines numerischen Wertes größer 9 (z.B. A oder F), so muß eine Null vorangestellt werden, da der Assembler sonst eine Marke erkennt.

Numerische Größen werden als vorzeichenlose 16 Bit Größen entsprechend der aktuellen Zahlenbasis berechnet. Die aktuelle Zahlenbasis bleibt unberücksichtigt, falls einer der folgenden Ausdrücke auftritt:

nnnnB	-	Binär
nnnnD	-	Dezimal
nnnnO	-	Oktal
nnnnQ	-	Oktal
nnnnH	-	Hexadezimal
X'nnnn'	-	Hexadezimal

Bei Überschreitung von 2 Bytes (16 Bit) tragen nur die niederwertigen 16 Bit zum Resultat bei.

## Zeichenketten

Zeichenketten (engl. Strings) bestehen aus Null oder mehreren ASCII-Zeichen, begrenzt durch Hochkommas und gekennzeichnet durch die Pseudooperation 'DEFM' (Define Message). Zweimaliges Auftreten der Begrenzungszeichen speichert dieses selbst ab.

**Beispiel:** Der Ausdruck

```
DEFM 'Ich schreibe 'ASM'-Programme'
```

speichert folgende Zeichenkette ab:

```
Ich schreibe 'ASM'-Programme
```

### 1.3 Arithmetisch/logische Ausdrücke

Arithmetisch/logische Ausdrücke dienen zur Berechnung von numerischen Größen während des Übersetzerlaufs.

Dies hat nichts zu tun mit dem Befehlsvorrat des Prozessors selbst.

Folgende Operatoren sind zulässig (Reihenfolge entspricht der Wertigkeit); ihre Verwendung ist auch aus den ausführlichen Beispielen im Anhang zu ersehen.

```

NUL
LOW,HIGH
*, /, MOD, SHR, SHL
- (Unary Minus)
+, -
EQ, NE, LT, LE, GT, GE
NOT
AND
OR, XOR

```

Zur Veränderung der Wertigkeit dienen runde Klammern. Mit Ausnahme von +, -, \*, / müssen alle Operatoren mit mindestens einem Leerzeichen (Blank) vom zugehörigen Operanden getrennt sein. Die folgenden Beispiele verwenden den Wert 'CONST' = 1122H.

#### Erläuterung der Operatoren:

NUL	<p>Nulloperator</p> <p>LD A,NUL ----&gt; LD A,OFFH</p>
LOW, HIGH	<p>Isoliert das nieder-(LOW) oder höherwertige (HIGH) Byte einer 16 Bit Größe. Ist diese relokativ, so ergibt sich ein zur Adresse 0 relativer Wert.</p> <p>LD A,LOW CONST ----&gt; LD A,22H</p> <p>LD B,HIGH CONST ----&gt; LD B,11H</p>
*, /	<p>16 Bit Ganzzahl Multiplikation bzw. Division</p> <p>LD HL,CONST*2 ----&gt; LD HL,2244H</p>
MOD	<p>Modulo Operator</p>
SHR, SHL	<p>Schiebeoperatoren (Shift right/left)</p> <p>LD HL,CONST SHL 2 ----&gt; LD HL,4488H</p>
+, -	<p>16 Bit Ganzzahl Addition bzw. Subtraktion</p> <p>LD HL,CONST-22H ----&gt; LD HL,1100H</p>

**Vergleichsoperatoren:**

EQ: Equal                - gleich  
NQ: Not equal          - ungleich  
LT: Less than          - kleiner  
LE: Less equal         - kleiner gleich  
GT: Greater than       - größer  
GE: Greater equal      - größer gleich

**Boole'sche Operatoren:**

NOT    Invertierung  
      LD HL,NOT CONST ----> LD HL,0EEDDH

AND    Und Verknüpfung  
      LD HL,CONST AND OFF00H ----> LD HL,1100H

OR     Oder Verknüpfung  
      LD HL,CONST OR OFFH ----> LD HL,11FFH

XOR    Exklusiv Oder Verknüpfung  
      LD HL,CONST XOR OFFH ----> LD HL,11DDH



## 1.4 Pseudooperationen

Pseudooperationen erzeugen keinen ausführbaren Maschinencode. Sie dienen zur Kontrolle der Codeerzeugung und zur Definition von Daten- und Speicherbereichen.

ASM verarbeitet folgende Pseudooperationen:

a) Datendefinition:	DEFB, DEFW, DEFM
b) Speicherdefinition:	DEFS
c) Symboldefinition:	EQU, DEFL
d) Symboldeklaration:	GLOBAL, EXTERNAL, LOCAL
e) Bedingungsdefinition:	IF.....ELSE...ENDIF
f) Makrodefinition:	MACRO...ENDM
g) Referenzadrezzähler- steuerung:	ORG
h) allgemeine ASM- steuerung:	TITLE etc.

Das Format der Pseudooperationen entspricht dem allgemeinen Format einer Quelltextzeile.

Marke: Pseudooperation Argument ;Kommentar

Abweichungen von der Regel sind bei der entsprechenden Pseudooperation beschrieben.

### 1.4.1 Datendefinition durch DEFB, DEFW, DEFM

Definition von 8/16 Bit-Größen und Zeichenketten.

Die Einrichtung der Speicherzellen für diese Daten erfolgt ab dem Stand des Referenzadrezzählers.

Diese Gruppe umfaßt drei Pseudooperationen. Ihr Format entspricht dem oben erwähnten Basisformat.

- DEFB: Definiere Byte
- DEFW: Definiere Wort
- DEFM: Definiere Zeichenkette

**DEFB - Definiere Byte****Syntax:**

(Marke:) DEFB Argument1 (,Argument2,...,Argumentn) (;Kommentar))

Als Argument ist zulässig:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein intern definiertes Symbol
- ein extern definiertes Symbol: DEFB EXTERNAL Name.1,...

Mehrere Argumente können, getrennt durch Kommas, hintereinandergestellt werden. Das Ergebnis eines arithmetisch/logischen Ausdrucks darf in diesem Fall 255 (=FFH) nicht überschreiten, andernfalls wird der Fehlercode A generiert.

Beispiele: DEFB 10H, 11H, 12H	Speicher: 10
DEFB 10H+2, 10H*2	11
	12
	12
	20

**DEFW - Definiere Wort****Syntax:**

(Marke:) DEFW Argument 1 (,Argument2,...,Argumentn) (;Kommentar)

Als Argument ist zulässig:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein Symbol jeglicher Klassifikation (intern oder extern definiert).

Die DEFW-Pseudooperation entspricht der DEFB-Pseudooperation. Als Ergebnis wird allerdings immer eine 16 Bit-Größe erzeugt. Als Argument dürfen deshalb auch extern definierte Symbole (EXTERNAL's) verwendet werden (siehe 4.4.2 über die Verwendung von EXTERNAL's in arithmetisch/logischen Ausdrücken).

Beispiele: DEFW 1234H	Speicher: 34 (LSB)
DEFW 2x4, 100 4H-5	12 (MSB)
	08
	00
	FF
	0F

**DEFM - Definiere Zeichenkette (Message)****Syntax:**

(Marke:) DEFM 'Zeichenfolge'            (;Kommentar)

Diese Pseudooperation erzeugt den ASCII-Code der innerhalb der Hochkommas stehenden Zeichen. Soll das Hochkomma selbst als Zeichen auftreten, so muß dies pro Vorkommen durch zwei aufeinanderfolgende Hochkommas gekennzeichnet werden. Neben dem einfachen Hochkomma kann auch das zweifache verwendet werden. Unterschiedliche Hochkommas vor und nach 'Zeichenfolge' sind allerdings nicht zugelassen.

Die Ablage der ASCII-Zeichen erfolgt ab dem momentanen Stand des Referenzadrezählers, wobei das höherwertige Bit eines jeden Bytes zu Null gesetzt ist.

Beispiel: DEFM '012ABC'

Speicher: 30  
          31  
          32  
          41  
          42  
          43

**1.4.2 Speicherreservierung****Syntax:**

(Marke:) DEFS Argument            (;Kommentar)

Als Argument ist erlaubt:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein internes Symbol

Das DEFS-Statement reserviert ab dem Stand des Referanzadrezählers einen Speicherbereich der Größe 'Argument'. Das Argument bestimmt die Anzahl der Bytes. Alle in einer DEFS-Pseudooperation verwendeten Symbole müssen vor ihrer erstmaligen Verwendung definiert sein, andernfalls wird der Fehlercode V generiert.

Beispiel: NBYTE    EQU    100H  
          DEFS    NBYTE

;Reserviert 100H Bytes

### 1.4.3 Symboldefinition durch EQU, DEFL

Zur Zuweisung von beliebigen Werten für Symbole im allgemeinen Sinn dienen die Pseudooperationen:

- EQU: Wertzuweisung mit festem Wert
- DEFL: Wertzuweisung mit änderbarem Wert

Das allgemeine Format ist:

Name      Opcode      Argument      (;Kommentar)

Name und Argument sind erforderlich.

#### EQU - Wertzuweisung

Syntax:

Name EQU Argument      (;Kommentar)

Als Argument ist zulässig:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein intern definiertes Symbol

Die EQU - Anweisung ordnet dem Symbol 'Name' den Wert 'Argument' zu. Das Argument kann im Programm mit dem vereinbarten symbolischen Namen verwendet werden. Mehrmalige Wertzuweisung für ein und denselben Namen ist mittels der EQU-Anweisung nicht zulässig (Fehler: M)

Hinweis: ASM ist ein 2 Pass-Assembler. Dies bedeutet unter anderem, daß Namen (und Symbole) an beliebiger Stelle eines Quellprogrammes definiert sein können. Eine Definition vor ihrer erstmaligen Anwendung ist gewöhnlich nicht erforderlich.

#### DEFL - Wertzuweisung

Syntax:

Name DEFL Argument      (;Kommentar)

Als Argument ist zulässig:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein intern definiertes Symbol

Die Anweisung DEFL (Define Label) entspricht der EQU-Anweisung (4.3.1) mit einer Ausnahme: mehrmalige Wertzuweisung für einen Namen während eines Programmes ist zulässig.

Beispiel:   ZAHL DEFL 10  
          LD A, ZAHL                   ;---> LD A,10  
          ZAHL DEFL 20  
          LD A, ZAHL                   ;---> LD A,20

#### 1.4.4 Symboldeklaration, GLOBAL, EXTERNAL

Symbole haben grundsätzlich einen von vier Gültigkeitsbereichen:

**INTERN:** ein Symbol ist nur innerhalb eines Programmes (eines Moduls) gültig. Dies ist Voreinstellung und Bedarf keines besonderen Hinweises.

**GLOBAL:** ein Symbol hat in allen durch einen Linkerlauf zusammenge- bundenen Modulen Gültigkeit. Dies ermöglicht intermodulare Symbol- referenzen und ist Voraussetzung für eine modulare Programmstruktur.

**EXTERN:** ein Symbol ist nicht dort definiert, wo es verwendet wird, sondern in einem anderen Modul. Der Assembler ordnet einem extern definierten Symbol keinen Wert zu, sondern reserviert nur den Platz dafür. Die eigentliche Wertzuweisung erledigt der Linker. Extern definierte Symbole müssen in einem anderen Modul als GLOBAL deklariert sein.

**LOCAL:** ein Symbol gilt nur innerhalb eines Makros (siehe Abschnitt: Makros).

#### GLOBAL - Symboldeklaration

##### Syntax:

GLOBAL Name1 (,Name2,.....,Namen)            (;Kommentar)

Hier darf keine Marke zu Beginn der Zeile stehen. Die Symbole 'Name1....Namen' sind innerhalb des Moduls, wo sie definiert werden, und in anderen Modulen (wo sie als EXTERNAL deklariert sind) zugänglich.

Alle in einem Programmmodul als GLOBAL deklarierten Symbole müssen dort definiert sein, ansonsten resultiert der Fehler U (undefiniertes Symbol). Ein Fehler M (mehrfache Symboldefinition) wird erzeugt, falls der Name mit demjenigen eines EXTERNAL's oder eines Moduls übereinstimmt.

**EXTERNAL - Symboldeklaration****Syntax:**

EXTERNAL Name1 (,Name2,.....,NameN)                    (;Kommentar)

Eine Marke zu Beginn der Quelltextzeile ist unzulässig. Die Symbole 'Name1.....NameN' sind nicht innerhalb des Moduls definiert, wo sie angewandt werden. ASM erzeugt den Fehler M (mehrfache Symboldefinition), falls einer der externen Namen mit dem Namen eines internen Symbols übereinstimmt.

**Achtung:**

Die Anzahl der Zeichen für die als EXTERNAL deklarierten Symbole ist auf 6 beschränkt.

EXTERNAL's sind in allen arithmetischen Ausdrücken verwendbar. Bei der Division darf nur der Dividend ein EXTERNAL sein.

Beispiel:    BYTE EXTERNAL EXT1, EXT2

LD A,EXT1-EXT2  
LD B,EXT1\*EXT2  
LD D,EXT1/2

16 Bit EXTERNAL's sind auch in Byte Operation (z.B. LD A,EX16) zugelassen. In diesem Fall wird das höherwertige Halbbyte des 16 Bit EXTERNAL's ignoriert.

### 1.4.5 Referenzadrezähler, ORG-Anweisung, Segmentierung

Der vom Assembler geführte Referenzadrezähler bestimmt die Speicheradresse, auf der ein erzeugter Code abgelegt wird. Das \$-Zeichen kennzeichnet in einem Quellprogramm den momentanen Stand dieses Zählers.

Beispiel: aktueller Stand sei 1000H, dann gilt

LD HL, \$+25H = LD HL, 1027H

Auf der Adresse 1000H steht der Maschinencode für LD HL (=21H).

ASM ist ein relokativer Assembler. Dies bedeutet unter anderem, daß die Startadresse eines Moduls in der Regel erst nach dem Übersetzerlauf festgelegt wird (LINK-Kommando). Wenn nicht anders angegeben, assembliert ASM relativ zur Adresse Null.

### ORG - Definition des Referenzadrezählers

Der Wert des Referenzadrezählers kann durch folgende Pseudooperation auf einen beliebigen Wert gesetzt werden:

#### Syntax:

(Marke:) ORG Argument            (;Kommentar)

Als Argument ist zulässig:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein intern definiertes Symbol

Die ORG-Pseudooperation setzt den Referenzadrezähler des Assemblers auf den durch das Argument bestimmten Wert. Jegliches Symbol im Argument muß zuvor definiert sein, andernfalls wird der Fehler U (undefiniertes Symbol) generiert.

Beispiel:

```
STARTA EQU 100H
STARTB EQU 200H
ORG STARTA      ; Programmteil beginnt ab
                  ; 100H

ORG STARTB      ; Programmteil beginnt ab
                  ; 200H
```

Die Marke vor dem ORG-Statement wird von dieser noch nicht beeinflusst.

### Programmsegmentierung

ASM gestattet die Segmentierung von Programmteilen oder ganzen Programmen in absolute, Code-relative und Daten-relative Speicherbereiche. Hierzu dienen die Pseudooperationen:

ASEG : absolutes Speichersegment  
CSEG : Code-relatives Speichersegment  
DSEG : Daten-relatives Speichersegment

Als Wirkung setzen alle drei Pseudo's den Referenzadrezähler auf den Wert des letzten entsprechenden Segments, bis eine ORG-Anweisung nach einem ASEG-, CSEG- oder DSEG-Pseudo den Wert des Referenzadrezählers ändert. Die Voreinstellung ist Null.

Enthält ein Modul keines der drei Pseudo's ASEG, CSEG oder DSEG, so gilt als Voreinstellung für ASM die CSEG-Bedingung. Dies bedeutet, daß die Assemblierung des gesamten Moduls relativ zum Anfangswert des Referenzadrezählers (Voreinstellung 0) erfolgt.



#### 1.4.6 Makros von ASM: REPT, IRP, IRPC, EXITM, LOCAL

Makros erlauben dem Anwender die Definition eigener Anweisungen. Ein Makro in ASM besteht gewöhnlich aus einer Reihe von Statements (Z80-Mnemonics oder Pseudooperationen), die bei jedem Makroaufruf automatisch in das Quellprogramm eingebunden werden.

Die Funktion von Makros ist vergleichbar mit der Funktion von Unterprogrammen, mit dem Unterschied, daß für Makros bei jedem Aufruf der entsprechende Code erzeugt wird. Dies erfordert zwar mehr Platz im Programmspeicher, verbessert aber die dynamischen Eigenschaften eines Programms. Ein weiterer Vorteil von Makros liegt darin, daß Parameter auf einfache Weise beim Makroaufruf übergeben werden können.

Neben der Makro-Pseudooperation unterstützt ASM drei Blockwiederholungsmakros, mit Hilfe deren Blöcke von Statements beliebig oft wiederholt werden können. Alle Makro-Pseudooperationen müssen mit dem Pseudobefehl 'ENDM' (für End Makro) abgeschlossen sein.

#### Makrodefinition

##### Syntax:

```
Name  MACRO PREP1,PREP2,...,PREPn      (;Kommentar)
      .
      .      ; beliebige Reihe von Statements
      .      ; Makrokörper
      ENDM
```

Alle Statements zwischen MACRO und ENDM werden als 'Makrokörper' bezeichnet und bei jedem Aufruf mit den entsprechenden Parametern in das Programm eingebunden. Die Parameterrepräsentanten (PREP1...PREPn) in der Makrodefinition sind frei wählbare Symbole, auf die innerhalb des Makrokörpers bezuggenommen werden kann. Beim Aufruf des Makros erfolgt die Parameterzuordnung entsprechend der Reihenfolge der Parameterrepräsentanten bei der Makrodefinition.

**Beispiel:**

16 Bit Addition zweier Zahlen: ZAHL1 + ZAHL2

## a) Makrodefinition:

```
ADD16 MACRO ZAHL1,ZAHL2
      LD HL,ZAHL1      ;
      LD DE,ZAHL2      ; Makrokörper
      ADD HL,DE        ;
      ENDM
```

## b) Makroaufruf:

Zum Aufruf dieses Makros müssen nur Name und die beiden Parameter angegeben werden. Als Parameter ist zulässig:

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck
- ein beliebiges Symbol

```
ADD16 5,2
```

Dieser Makroaufruf führt zur automatischen Einbindung des Makrokörpers mit folgendem Aussehen:

```
LD HL,5
LD DE,2
ADD HL,DE
```

Die Anzahl der Parameter beim Makroaufruf muß nicht notwendigerweise mit der Anzahl der Parameterrepräsentanten bei der Makrodefinition übereinstimmen. Sind mehr Parameter als Parameterrepräsentanten vorhanden, so wird der Rest ignoriert, sind weniger angegeben, so wird für die restlichen Parameter der Wert Null angenommen.

**Blockwiederholungsmakros**

Solche Makro's wiederholen den Makrokörper n-mal (entsprechend eines Arguments n).

**REPT - einfache Blockwiederholung****Syntax:**

REPT Argument            (;Kommentar)

Der Makrokörper (alle Statements zwischen REPT und ENDM) wird 'Argument-mal' wiederholt.

Beispiel: Erzeugung des ASCII-Codes der Ziffern 0 bis 4

```
NAME  DEFL 30H
      REPT 5      ; fünfmalige Wiederholung
      DEFB NAME
NAME  DEFL NAME+1 ; erzeugt 30H ... 34H
      ENDM
```

**IRP - unbegrenzte Blockwiederholung****Syntax:**

IRP PREP, <Argument1,...,Argumentn>            (;Kommentar)

Die Anzahl der Argumente bestimmt, wie oft der Block zwischen IRP und ENDM wiederholt wird. Das n-te Argument wird bei der n-ten Wiederholung anstelle des Parameterrepräsentanten (PREP) eingesetzt. Die Argumente müssen mit dreieckigen Klammern (< - >) eingeschlossen sein.

Beispiel:        IRP X,<30H,31H,32H,33H,34H>  
                 DEFB X  
                 ENDM

Erzeugt denselben Code wie das Beispiel im vorherigen Abschnitt. Nach der Makro-Expansion ergibt sich somit:

```
DEFB 30H
DEFB 31H
DEFB 32H
DEFB 33H
DEFB 34H
```

**IRPC - unbegrenzte Zeichenwiederholung****Syntax:**

IRPC PREP, <Zeichenfolge>            (;Kommentar)

Die Anzahl der Zeichen in der Zeichenfolge bestimmt die Anzahl der Wiederholungen des Blocks zwischen IRPC und ENDM. Der Parameterrepräsentant (PREP) innerhalb des Blocks wird bei der n-ten Wiederholung durch das n-te Zeichen der Zeichenfolge ersetzt.

Beispiel:            IRPC X,<01234>  
                      DEFB X  
                      ENDM

Führt zur Ablage der Werte 0 bis 4  
ab dem Stand des Referenzadrezählers.

**EXITM - Verlassen eines Makro's****Syntax:**

EXITM                (;Kommentar)

Marke oder Argument dürfen hier nicht auftreten. Die EXITM-Pseudooperation bricht einen REPT-, IRP-, IRPC- oder MACRO-Aufruf ab. Der Makrokörper wird nach einer EXITM-Anweisung nicht weiter übersetzt.

Ist der Block, der EXITM enthält, verschachtelt, so wird der äußere Block weiter übersetzt.

Beispiel:            NAME        DEFL 0  
                              REPT 100                    ;erzeugt die Hexa-  
                              IF NAME EQ 10H            ;dezimalzahlen  
                              EXITM                    ;von 0 bis 10H  
                              ENDIF  
                              DEFB NAME  
                      NAME        DEFL NAME+1  
                              ENDM

**LOCAL - Symboldeklaration****Syntax:**

LOCAL OP1 (,OP2,...,OPn)            (;Kommentar)

Diese Pseudooperation ist nur innerhalb einer Makrodefinition zulässig. Sie muß dort vor allen anderen Statements stehen.

LOCAL ist immer dann erforderlich, wenn in der Makrodefinition Marken (als Sprungziele oder Unterprogramme) verwendet werden.

In diesem Fall muß bei jedem Aufruf dieses Makros ein eigenes Symbol generiert werden. ASM generiert automatisch Marken der Form: ..nnnn (mit nnnn = 0000 bis FFFF), und inkrementiert diese bei jedem Aufruf des Makro's. Mehrfach definierte Symbole werden somit vermieden.

Beispiel:            Makro zur n-maligen Inkrementierung  
                      eines Wertes

```

        INCN MACRO N,WERT
            LOCAL JR
            LD A,WERT
            LD B,N
JR      INC A
        DJNZ JR
        ENDM

```

Bei einem angenommenen 1. Aufruf INCN 4,20H sieht die Makroexpansion folgendermaßen aus:

```

                LD A,20H
                LD B,4
..0000      INC A
                DJNZ ..0000

```

Bei einem eventuellen zweiten Aufruf wird das Symbol ..0001 generiert, usw.

#### 1.4.7 Allgemeine Assemblersteuerung, COMMENT, PRINTX, RADIX

Pseudooperationen zur allgemeinen ASM-Steuerung erfüllen eine der folgenden Funktionen:

- Kennzeichnung eines längeren Kommentars im Quellprogramm
- Ausdruck eines Textes während des ASM-Laufes
- Änderung der Zahlenbasis für numerische Werte

Alle Pseudooperationen dieser Gruppe müssen in Spalte 1 mit einem Punkt (.) beginnen.

##### COMMENT - Kommentarkennzeichnung

###### Syntax:

.COMMENT /beliebige Anzahl von Zeichen  
- beliebig viele Zeilen  
...../

Als Begrenzungszeichen (delimiter) dient das erste Nicht-Blank nach der COMMENT-Pseudooperation. Derselbe Delimiter muß am Ende des Kommentarfeldes stehen.

##### PRINTX - Textausdruck

###### Syntax:

.PRINTX /Zeichenfolge/                    (;Kommentar)

Die Zeichenfolge innerhalb beliebiger Begrenzungszeichen (siehe Abschnitt oben) wird während des ASM-Laufes ausgegeben. Dies ist ein äußerst nützliches Hilfsmittel, um den Fortgang des ASM-Laufes bei längeren Programmen anzuzeigen. Der Ausdruck erfolgt in beiden Durchläufen des Assemblers.

Die Ausgabe der Zeichenfolge nur im Pass1 bzw. nur im Pass 2 ist durch die Verwendung der Pseudooperationen IF1 bzw. IF2 möglich.

Beispiel:            .PRINTX /Testpunkt1/            ;Ausdruck in beiden  
                     oder                                ;Durchlaufen  
                     IF1  
                     .PRINTX /Testpunkt1/            ;und nur in Pass1  
                     ENDIF

## RADIX - Änderung der Zahlenbasis

### Syntax:

.RADIX Argument            (;Kommentar)

Als Argument ist zulässig:

- ein numerischer Wert von 2 bis 16 (immer dezimal)
- ein intern definiertes Symbol (vom Wert 2 bis 16)

Die RADIX-Pseudooperation ändert die Zahlenbasis numerischer Werte (siehe dazu auch 2.3.). Numerische Werte zur Zahlenbasis 2 bis einschließlich 16 sind möglich. Als Voreinstellung gilt das dezimale System.

Beispiel:            LD HL,255  
                      .RADIX 16  
                      LD HL,OFF  
                      .RADIX 10  
                      LD HL, OFFH

Alle drei obigen LD-Befehle sind identisch.

Ist die Zahlenbasis 16 eingeschaltet, so sind die Ausdrücke

nnnnD    für dezimale Zahlen, bzw.  
nnnnB    für binäre Zahlen

nicht verwendbar, da die Zeichen D und B von den hexadezimalen Zeichen D und B nicht zu unterscheiden sind.

## Quellcode-Mnemonic

### Syntax:

.8080                Umschalten auf Intel 8080 Quellcode-Mnemonic  
.Z80                 Umschalten auf ZILOG Z80 Quellcode-Mnemonic

Diese Umschaltung kann zur Assemblerlaufzeit erfolgen. Die Voreinstellung ist .Z80 bzw. durch /I bestimmt.

### 1.4.8 Bedingte Assemblierung

ASM bietet verschiedene Pseudooperationen zur bedingten Assemblierung von einzelnen Statements oder auch größeren Programmteilen. Bei allen Bedingungs-Pseudooperationen bestimmt der Bool'sche Wert eines Arguments (TRUE oder FALSE) ob der entsprechende Quelltextteil übersetzt wird oder nicht. Folgende Pseudo's sind implementiert:

- IF Argument            Wie IFT-Pseudo
- IFT Argument           Wahr (TRUE), falls der Wert des Arguments nicht Null ist  
(IF TRUE-Bedingung)
- IFF Argument           Wahr (TRUE), falls der Wert des Arguments gleich Null ist  
(IF FALSE-Bedingung)
- IF1/IF2                Wahr (TRUE), falls ASM-PASS1 bzw. PASS2. Hier ist kein Argument zulässig.
- IFDEF Symbol           Wahr (TRUE), falls das Symbol definiert oder als EXTERNAL deklariert ist  
(IF DEFINED-Bedingung)
- IFNDEF Symbol          Wahr (TRUE), falls das Symbol nicht definiert oder nicht als EXTERNAL deklariert ist.  
(IF NOT DEFINED-Bedingung)
- IFB <Argument>        Wahr (TRUE), falls der Wert des Arguments gleich 'Blank' ist. Die dreieckigen Klammern sind erforderlich.  
(IF BLANK-Bedingung)
- IFNB <Argument>        Wahr (TRUE), falls der Wert des Arguments ungleich 'Blank' ist. Die dreieckigen Klammern sind erforderlich.  
(IF NOT BLANK-Bedingung)

IFB und IFNB dienen dazu, um festzustellen, ob Parameterrepräsentanten (siehe Makro's) vorhanden sind oder nicht.



Die allgemeine Syntax der Bedingungs-Pseudooperationen ist:

```
IFxx (Argument)
.
.
ELSE
.
.
ENDIF
```

IF-Pseudooperationen können beliebig oft ineinander verschachtelt werden. Alle Argumente müssen während PASS1 des Assemblerlaufs bekannt sein.

In der IF/IFT- bzw. IFF-Pseudooperation darf als Argument

- ein numerischer Wert
- ein arithmetisch/logischer Ausdruck oder
- ein intern definiertes Symbol

verwendet werden. Alle Symbole müssen zuvor definiert und absolut sein.

In jeder IF-Pseudooperation kann optional das ELSE-Pseudo verwendet werden. Damit ist die Übersetzung eines alternativen Quelltextteils möglich, falls die IF-Bedingung nicht zutrifft. Innerhalb einer IF-Bedingung ist nur ein ELSE-Pseudo erlaubt.

Der Abschluß einer jeden IF-Bedingung muß mit der ENDIF-Pseudooperation erfolgen. Der Fehlercode C wird generiert, falls IF-Statements nicht ordnungsgemäß abgeschlossen sind.

Anstelle des IF-Pseudos kann auch das Pseudo 'COND' (Condition) verwendet werden. In diesem Fall muß statt 'ENDIF' das Pseudo 'ENDC' verwendet werden.

### 1.4.9 Include Anweisung

**Syntax:**

```
INCLUDE mn:quelldatei.typ      (;Kommentar)
```

Die INCLUDE-Anweisung ermöglicht das Einbinden einer gesamten Quelldatei während der Übersetzung. Die Angaben von 'mn' und 'typ' sind optional. Als Voreinstellung gilt:

```
mn = Mastermedium  
typ = SRC
```

Die Datei 'quelldatei' wird an der Stelle eingebunden, wo das INCLUDE-Statement auftritt. Verschachtelte INCLUDE-Anweisungen sind nicht gestattet. Es ist zu beachten, daß die eingebundene Datei nicht mit einem END-Statement abgeschlossen ist, da ASM sonst die Übersetzung abbricht und weitere Befehle der Datei, in der das INCLUDE-Statement auftrat, nicht mehr übersetzt. Im Listing erscheint das Zeichen 'C' zwischen dem assemblierten Code und dem Quelltext in jeder Zeile einer eingebundenen Datei. Das INCLUDE-Statement kann folgende Fehlermeldungen verursachen:

- O - Verschachtelte INCLUDE-Anweisung
- V - Spezifizierte Datei existiert nicht. In diesem Fall wird die INCLUDE-Anweisung ignoriert.

#### 1.4.10 Moduldokumentation, TITLE, SUBTTL, END

Zur Benennung von Modulen stehen drei Pseudooperationen zur Verfügung. Hierzu kommt in dieser Gruppe das END-Statement, das in jedem Fall zum Abschluß eines ASM-Programms notwendig ist.

##### Definition des Modulnamens

Syntax:  
NAME 'Modulname'

Die Pseudooperation NAME definiert den Namen eines Moduls zu 'Modulname'. Nur die ersten sechs Zeichen in 'Modulname' sind signifikant. Ein Modulname kann auch durch die TITLE-Pseudo Operation festgelegt werden (siehe 4.9.2). Fehlen beide Pseudo's, so wird dem Modul automatisch der Name der Quelldatei zugeordnet.

##### TITLE - Listingüberschrift

Syntax:  
TITLE Text

TITLE spezifiziert den Titel der in jeder Kopfzeile eines Listings aufgeführt wird. Die ersten sechs Zeichen von 'Text' werden als Modulname verwendet, bis eine NAME-Pseudooperation erkannt wird (siehe auch 4.9.1)

##### SUBTTL - Untertitel

Syntax:  
SUBTTL Text

SUBTTL spezifiziert den Untertitel, der in der ersten Zeile nach der Kopfzeile (TITLE-Pseudo) aufgeführt wird. 'Text' darf maximal 60 Zeichen enthalten. Die Anzahl von Untertiteln innerhalb eines Programms ist unbeschränkt.

##### END - Quelldateiende

Syntax:  
END (Argument)

Das END-Statement ist zur Kennzeichnung des Endes eines Quellprogramms notwendig.

Als Argument ist ein Label oder ein numerischer Wert zulässig. Im zweiten Fall bestimmt 'Argument' die Startadresse für das Programm (von LINK verarbeitet).

#### 1.4.11 Listingsteuerung, LIST, XLIST, CREF, XCREF, LALL, SALL, XALL

Zum An- und Abschalten von Listingteilen, Symbolreferenzen und Makroexpansionen stehen jeweils Pseudooperationen zur Verfügung:

- LIST/XLIST      für Listing allgemein
- CREF/XCREF      für Symbolreferenz
- LALL/SALL/XALL für Makroexpansionen

Alle hier erwähnten Pseudooperationen müssen mit einem Punkt beginnen. Marken und Argumente in der Quelltextzeile sind nicht zugelassen. Daneben kann mit der Pseudooperation PAGE die Anzahl der Zeilen einer Listingseite festgelegt werden.

#### Seitenformat

Syntax:

PAGE (Argument)

PAGE bewirkt den Beginn einer neuen Seite, der Assembler fügt an dieser Stelle ein Formfeed-Zeichen ein. Das Argument, falls vorhanden, bestimmt die Anzahl der Zeilen pro Seite. Der Wert des Arguments muß im Bereich 10 bis 255 liegen, als Voreinstellung gilt 50 (alle Werte dezimal). Der Beginn einer neuen Seite kann auch durch das ASM-Kommando

\*EJECT (oder ein CTRL-L im Quelltext)

erreicht werden (der Stern muß in der ersten Spalte einer Zeile stehen - es genügt \*E).

#### LIST/XLIST - allgemeine Listingsteuerung

Syntax:

.LIST (bzw. .XLIST)

XLIST unterdrückt die Ausgabe einer Listingdatei, LIST gibt sie frei. Voreinstellung ist LIST. Beide Pseudooperationen haben keinen Einfluß, wenn kein Listing erzeugt wird.

**CREF/XCREF - Symbolreferenzsteuerung****Syntax:**

**.CREF** (bzw. **.XCREF**)

**XCREf** unterdrückt die Ausgabe einer Cross Referenzdatei, bis das Pseudo **CREF** auftritt (letzteres ist Voreinstellung). Ist die Option **C** beim **ASM**-Aufruf nicht angegeben (Abschnitt 1.2), so haben **CREF/XCREf** keine Wirkung.

**LALL/SALL/XALL - Makroexpansionssteuerung****Syntax:**

**.LALL** (bzw. **.SALL** oder **.XALL**)

Diese Pseudooperationen steuern die Ausgabe der Expansionen bei **MACRO**-, **REPT**-, **IRP**- und **IRPC**-Pseudo's.

- **.LALL** listet den vollständigen Makrotext für alle Expansionen.
- **.SALL** listet nur den Objektcode eines Makros, nicht aber den dazugehörigen Text.
- **.XALL** ist Voreinstellung.  
Eine Makroexpansion erscheint nur dann im Listing, wenn Code erzeugt wird.

## 1.5 ASM-Fehlermeldungen

Fehlermeldungen von ASM sind durch einen einbuchstabigen Hinweis in Spalte 1 der Listingdatei gekennzeichnet. Die Ausgabe der entsprechenden Zeile auf den Sichtschirm erfolgt in jedem Fall, auch dann, wenn keine Listingdatei erzeugt wird. Nachstehend die Liste der möglichen Fehlercodes.

- A     - Argumentfehler  
      Das Argument eines beliebigen Statements ist nicht im korrekten Format oder außerhalb des zulässigen Bereichs.  
      Beispiele:     .RADIX 1  
                      LD A,1234H  
                      JR 100H
- C     - Syntaxfehler bei IF-Statements  
      Dieser Fehler weist auf einen der drei folgenden Fälle hin:
- a) ELSE ohne IF
  - b) ENDIF ohne IF
  - c) ELSE/IF-Anzahl nicht übereinstimmend
- D     - Doppelt definiertes Symbol  
      Referenz auf ein Symbol, das mehrfach definiert ist.
- E     - Unzulässige Verwendung eines EXTERNAL's  
      Die Verwendung des EXTERNAL's in der fehlerhaften Zeile ist unzulässig.  
  
      Beispiel:     LD A,EXTNAM    (EXTNAM sei ein EXTERNAL)
- M     - Mehrfach definiertes Symbol  
      Definition eines Symbols, das bereits definiert war.
- N     - Numerik Fehler  
      Kennzeichnet in der Regel ein unzulässiges Digit innerhalb einer Zahl.  
  
      Beispiele:    LD A,0112B  
                      .RADIX 10  
                      LD A,OF

- O - Opcode unbekannt oder allgemeiner Syntaxfehler.  
Tritt bei folgenden Fällen auf:
- a) ENDM oder LOCAL außerhalb eines Makros
  - b) EQU, DEFL oder MACRO ohne Namen
  - c) Syntaxfehler in einem Opcode (z.B.: LD A,)
  - d) Syntaxfehler allgemein (fehlende Hochkommas etc.)
- P - Nichtübereinstimmung von Symbolwerten, oder Verwendung von internen Namen (z.B.: PEEK, POKE, MAX, C, Z etc.) in PASS1 und PASS2
- Q - Warnung (engl. Query)  
Weist gewöhnlich auf eine nicht ordnungsgemäß abgeschlossene Zeile hin.  
Beispiel: LD A,20,  
  
Dies ist nur eine Warnung, die Zeile wird richtig übersetzt.
- R - Relokation  
Unzulässige Verwendung relokativer Symbole in Ausdrücken.
- U - undefiniertes Symbol  
Kennzeichnet die Referenz auf ein nicht definiertes Symbol.  
(In bestimmten Fällen führt ein undefiniertes Symbol im PASS1 zu einem Fehler V und im PASS2 zu einem Fehler U)
- V - undefiniertes Argument  
Das Argument bestimmter Pseudooperationen ist im PASS1 nicht definiert, wird aber vor dem erstmaligen Aufruf benötigt. Wird das Symbol später definiert, so wird im PASS2 kein U-Fehler erzeugt.

Am Ende eines Assemblerlaufes erscheint grundsätzlich der Ausdruck der Fehleranzahl. Fehlt das END-Statement, so wird dies durch folgende Warnung angezeigt: (in beiden ASM-Läufen)

- END Statement nicht gefunden
- Datei nicht vorhanden
- Medium voll oder KOS-E/A-Fehler

**Ursache:** Das Abspeichern von Object- und/oder Listingdatei ist nicht möglich, da entweder das Medium (Diskette) voll ist, oder ein E/A-Fehler beim Datentransfer mit dem Medium aufgetreten ist, oder die Datei schon vorhanden ist, mit den Properties P=WE.

**Hinweise:**

- a) die Zahlerwerte 08 und 09 werden nur in der Form 08H, 09H oder 08D, 09D (dezimal) oder einfach 8,9 akzeptiert
- b) bei mehr als 4 Operanden in arithmetisch/logischen Ausdrücken wird der Fehler 'E' erzeugt, obwohl die Berechnung richtig ausgeführt wird.

## 1.6 ASM-Listing Format

Auf jeder Seite eines von ASM erzeugten Listings haben die beiden ersten Zeilen folgende Form:

(TITLE Text) PSI80/ASM                      Seite x(-y)  
(SUBTTL Text)

Hierbei bedeuten:

- a) 'TITLE Text' ist der Text, der durch eine Pseudooperation TITLE vereinbart wurde.
- b) x ist die Hauptseitennummer, die nur dann inkrementiert wird, wenn im Quelltext ein Formfeed-Zeichen erkannt wird. Beim Ausdruck der Symboltabelle ist x = S.
- c) y ist die eigentliche Seitennummer, die immer dann erhöht wird, wenn ein PAGE-Pseudo auftritt oder die aktuelle Seitengröße überschritten wird. ASM fügt nach jeder Seite ein Formfeed-Zeichen ein.
- d) 'SUBTTL Text' ist der Text, der durch eine Pseudooperation SUBTTL vereinbart wurde.

Nach diesen beiden Zeilen folgt in jedem Fall eine Leerzeile. Anschließend erscheint die erste Zeile des übersetzten Quellprogramms in folgendem Format:

(Crf#) (Fehlercode) Loc#m xx xxxx .... Quelltext

Die Symbolreferenznummer (Cross referenz - cref) erscheint am Anfang einer Listingzeile, falls beim Aufruf des Assemblers der Funktionsschalter C angegeben wurde. War dies nicht der Fall, so ist der Einzeichen Fehlercode das erste Zeichen einer Zeile. Dem Fehlercode folgt unmittelbar ein Leerzeichen, danach der Wert des Referenzadrezählers. Dies ist eine vier- oder sechsstelligen Zahl (hexadezimal oder oktal, H- oder O-Option). Es folgen drei Leerzeichen, sowie der erzeugte Code. Die Darstellung von 2 Byte Werten erfolgt in der umgekehrten Reihenfolge als ihre Abspeicherung, nämlich zuerst das höherwertige und dann das niederwertige Byte. Der Rest der Zeile erhält den Quelltext, so wie er eingegeben wurde.

Die letzte Seite eines Assembler Listings enthält eine Auflistung aller verwendeten Symbole in alphabetischer Reihenfolge. Nach jedem Symbol steht der Wert des Symbols, sowie ein Zeichen zur Identifikation von einem der folgenden Fälle:

- U - undefiniertes Symbol
- \* - Externes Symbol (EXTERNAL)
- I - Globales Symbol (GLOBAL)
- ' - Relokatives Symbol

Diese vier Identifikationszeichen kennzeichnen auch den Wert des Referenzadrezählers und des Opcodes in einem Listing.



### 1.7 Assembler/Linker Kommandodateien

Eine Kommandodatei zum Assemblieren und Linken kann mit EDIT erstellt werden, z.B. unter dem Namen ASMLINK.CMD.

Die Kommandodatei enthält folgende Kommandosequenz:

```
ASM #1,#1 =#1
LINK #1/N,#1/P:100/E
IL #1.*
```

Beim Aufruf der Kommandodatei wird anstelle von #1 der Ausdruck 'mn:quelldateiname' eingesetzt. Ein Typ darf nicht angegeben werden. Der Aufruf lautet dann z.B.:

```
DO ASMLINK mn:quelldateiname<CR>
```

Folgende Dateien werden generiert:

```
mn:quelldateiname.OBJ
mn:quelldateiname.PRN
mn:quelldateiname.COM
```

Beispiel:

Die Datei TEST.SRC auf dem Mastermedium soll assembliert und gelinkt werden. Der entsprechende Aufruf lautet:

```
DO ASMLINK TEST<---
```

Zur Ausführung kommen dann die Kommandos:

```
ASM TEST,TEST =TEST
LINK TEST/N,TEST/P:100/E
IL TEST.*
```

## 2. Linker für Kontron PSI-Systeme

Die primäre Funktion eines solchen Programms besteht darin, mehrere relocative Module zu einem ablauffähigen Maschinenprogramm zusammenzubinden und intermodulare Symbolreferenzen mit den dazugehörigen Adressen zu versehen. Diese relocativen Module können sowohl von Assembler, als auch von Compilern (z.B. FORTRAN80) stammen.

Schließlich legt LINK die Startadresse des erzeugten Programms oder einzelner Module fest.

### 2.1 Übersicht LINK

Das Linker/Lader-Programm der Kontron PSI-Computer ist unter dem Namen LINK auf der KOS-Systemkommandodiskette abgelegt.

Ein Linkerlauf ist in jedem Fall erforderlich, auch dann, wenn nur ein Modul bearbeitet wird. Die Objektdatei des Assemblers selbst ist kein ablauffähiges Maschinenprogramm, sondern enthält eine Reihe von Zusatzinformationen für den Linker.

Für Programmodule ohne Externbeziehungen kann das Dienstprogramm RLOAD als Linker/Lader verwendet werden (Beschreibung siehe Teil "Systemkommandos").

## 2.2. Kommandoaufruf LINK

Beim Aufruf des Linkers enthält das Parameterfeld des Kommandos neben dem Namen des zu erzeugenden Maschinenprogramms (COM-Datei) im allgemeinen Fall die Namen der Objektdateien, die zu einem ablauffähigen Maschinenprogramm zusammengebunden werden sollen. Es können beliebig viele Dateien zusammengebunden werden, solange die Kommandozeile nicht mehr als 80 Zeichen enthält.

Ein LINK-Lauf kann jederzeit durch <ESC> abgebrochen werden.

### Allgemeine Kommandosyntax

Das allgemeine Format des Linker-Aufrufs ist:

LINK mn:comdatei/N,mn:objektdatei/P:stadr,...,mn:objektdatei/E

Es bedeuten:

comdatei	Name des zu erzeugenden Maschinenprogramms Voreinst.: mn = Mastermedium typ = COM
objektdatei	Name der Objektdatei(en). Zwischen einzelnen Dateinamen steht ein Komma als Trennzeichen.  Voreinst.: mn = Mastermedium typ = OBJ
stadr	Startadresse des Programms. 'stadr' kann für jedes Objektmodul separat angegeben werden, ist aber grundsätzlich optional. Fehlt 'stadr', so gilt als Voreinstellung für das erste Modul: 'stadr' = 100H und für alle anderen Module: 'stadr' = Endadresse des vorhergehenden Moduls.

### LINK-Aufruf ohne Parameterangabe

Wird LINK ohne Parameter aufgerufen, so antwortet LINK mit den Promptzeichen 'LINK:' und ist daraufhin zur Aufnahme von Kommandos bereit (Kommandomodus). Diese Kommandos sind identisch mit dem Parameterfeld des Assembleraufrufs (siehe 1.1).

Beispiel:

```
LINK<---  
LINK:TEST/N,TEST/P:100/E
```

Linkt die Datei TEST.OBJ auf die Adresse 100H und speichert das Programm unter dem Namen TEST.COM auf Diskette ab. Ist der Funktionsschalter /E nicht angegeben, so kehrt LINK nicht in das Betriebssystem zurück, sondern wartet auf neue Eingaben. Die Rückkehr zu KOS kann mit dem Kommando 'KOS' veranlaßt werden.

## LINK-Funktionsschalter

Funktionsschalter haben ähnlich wie beim Assembler eine Schaltfunktion. Jedem Funktionsschalter muß das Zeichen /(Slash) vorausgehen. Sie sind sowohl im Zusammenhang mit Dateinamen, als auch davon isoliert (im Link-Kommandomodus) möglich. Folgende Funktionen sind implementiert:

Funktionsschalter /R - Reset:

Bringt das Linker/Lader Programm in seinen ursprünglichen Zustand. Die Reset-Funktion ist nur im Kommandomodus sinnvoll und immer dann von Nutzen, wenn irrtümlich eine falsche Datei geladen wurde. /R wirkt sich sofort nach dem Erkennen innerhalb eines Kommandostrings aus.

Funktionsschalter /E oder /E:Name - Exit:

Führt zum Verlassen von LINK und Rückkehr in das Betriebssystem KOS. Wird die optionale Form /E:Name angegeben, so bestimmt das Symbol 'Name' die Startadresse des erzeugten Programms. 'Name' muß ein GLOBAL in einem der zusammengebundenen Module sein.

LINK ermöglicht das direkte Abspeichern von Programmen mit beliebigen Startadressen auf eine Datei.

Die Startadresse ist entweder die Startadresse des Hauptprogramms oder die Adresse der Marke, die beim Linker-Aufruf mit /E:<Marke> angegeben wird. Das Hauptprogramm ist das erste Modul, das im Quelltext hinter der Assembler-Anweisung 'END' als Argument die Startadresse oder Startmarke stehen hat.

Ist die Startadresse nicht gleich Programmbeginn (= kleinste Adresse des Programms), so setzt der Linker eine Sprunganweisung (3 Bytes) vor den Programmbeginn. In diesem Fall wird zusätzlich die Adresse des Programmbeginns auf dem Bildschirm ausgegeben. Ist diese Adresse kleiner 100H, wird eine Warnung ausgegeben, die keinen Einfluß auf das Linken hat.

Beim Abspeichern wird die Startadresse, das erste freie Byte nach dem Programm und die Anzahl der Speichersegmente angegeben. Die Anzahl der Speichersegmente (= Records mit 128 Bytes) gibt den Speicherbereich an, der allokiert wird. Diese Zahl wird zusätzlich hexadezimal angegeben, gekennzeichnet mit einem 'h'. Der Funktionsschalter /P:<adr> gibt nur die Anfangsadresse des Moduls an und ist nicht mit der Startadresse zu verwechseln.

**Funktionsschalter /N - Name:**

Definiert den Dateinamen, unter dem das erzeugte Programm auf Diskette abgelegt wird.

**Beispiel:**

LINK TEST/N,TEST1,TEST2/E<--

Bindet die Module TEST1.OBJ und TEST2.OBJ. LINK erzeugt daraus die Datei TEST.COM.

**Funktionsschalter /P:adr - Programmanfangsadresse:**

Dieses LINK-Kommando ermöglicht das Setzen der Anfangsadresse des Moduls. /P hat keinen Einfluß auf bereits geladene Programm. Der Wert 'adr' bestimmt die Anfangsadresse des spezifizierten Moduls. Er ist als hexadezimaler Zahlenwert anzugeben. Ist die /P-Option überhaupt nicht, oder nicht für das erste Modul gegeben, so gilt als voreingestellte Anfangsadresse für das erste Modul der Wert 100H. Alle weiteren Module werden unmittelbar an das Ende des vorhergehenden Moduls angebunden.

## 2.3 LINK-Meldungen

LINK 5.3 verwendet den KOS Ausgabekanal 0-1 für Fehler- und sonstige Meldungen. Grundsätzlich wird der Name der Object-Datei, die momentan in Bearbeitung ist, ausgedruckt, mit dem Hinweis auf den Status der Bearbeitung. Folgender Status ist möglich:

Statusmeldungen:

Status: ok	--> kein Fehler aufgetreten
Objectformat unzulässig	--> Die spezifizierte Datei entspricht nicht den Vorschriften für relokative Objektdateien
Speicherüberlauf	--> Der zur Verfügung stehende Speicherbereich reicht nicht aus, um das Programm zu linken.
nicht vorhanden	--> Die spezifizierte Datei konnte nicht gefunden werden

Weitere Fehlermeldungen sind:

- Eingabefehler	--> Das eingegebene Kommando konnte nicht verstanden werden
- Globals(s) mehrfach definiert	--> Globals sind mehrfach definiert in den zusammengeordneten Modulen
- Medium voll oder KOS E/A-Fehler	--> Der Versuch, das erzeugte Maschinenprogramm auf ein Medium (Diskette etc.) abzuspeichern ist nicht gelungen

Start Symbol - <Name> - undefiniert

Das in der /E: Option angegebene Symbol <Name> ist nicht definiert.

Überschneidender Programm/Daten Bereich

Programm-(Code-) und Datensegmente überschneiden sich.

Neben diesen Fehlermeldungen sind folgende Warnungen möglich:

Mehrf. def. GLOBAL <Name>

Das GLOBAL <Name> wurde mehrfach definiert.

Überlagernde Programm-/Datenbereiche

Eine /D- oder /P-Option überschreibt bereits geladene Programme.

Speicherende überschritten

Das Programm geht über OFFFFH hinaus (nur bei /N möglich).  
Das Linken wird abgebrochen.

Warnung: Programmbeginn ist unter 100H!  
Diese Warnung hat keinen Einfluß auf das Linken.

## 2.4 LINK-Beispiele

### Linken eines einzelnen Programms

#### a) LINK TEST/N,TEST/P:100/E<---

Linkt das Programm TEST.OBJ auf Adresse 100H und legt es unter dem Namen TEST.COM auf dem Mastermedium ab.

Das Programm TEST.SRC hat das folgende allgemeine Aussehen:

START:

```
.  
.   
.   
END START
```

#### b) LINK TEST/N,TEST/P:5000/E

Programmbeginn und Startadresse ist 5000H. In das Directory der Datei wurde 5000H als Startadresse eingetragen.

#### c) LINK TEST/N,TEST/P:5000/E:START

Programmbeginn ist 4FFDH, dort ist ein Sprungbefehl auf die Adresse der Marke 'START' eingetragen, das Modul TEST beginnt bei 5000H. Das Directory der Datei zeigt den Programmbeginn mit 4FFDH an.

#### d) LINK TEST/N,TEST/E

Eine Startmarke oder Startadresse innerhalb des Programms wird erwartet.

Ist die Startadresse gleich dem Anfang des Moduls, ist kein Sprungbefehl nötig. Programmbeginn ist dann 103H und wird in das Directory eingetragen.

Wurde eine Startadresse innerhalb des Moduls definiert, wird auf 100H der Sprungbefehl auf die Startadresse eingetragen. Programmbeginn ist dann 100H, auch im Directory. Das Modul hat seinen Anfang auf 103H.

#### e) Beispiel b) - d) ohne 'TEST/N,'

Das jeweils erzeugte Programm wird nicht auf eine Datei gespeichert, sondern im Speicher auf die richtige Adresse (wenn möglich) zwischen 100H und MEMTOP geladen.

Eine evtl. notwendige Sprunganweisung auf die Startadresse des Programms wird immer zuerst auf 100H geschrieben und dann wird das Programm auf die verlangte Anfangsadresse geschrieben. Bei /P:100 wird der Sprungbefehl überschrieben!

#### f) LINK KDM/N,KDM/P:9000/E,KDM1,KDM2,KDMMSG

Das KDM (Debugging Module) wird auf die Adresse 9000H gelinkt und als Datei gespeichert.



**Linken mehrerer Module**

g) LINK MAIN/N,MAIN/P:100/E,SUB1,SUB2,SUB3/P:2000,SUB4<--

Linkt ab Adresse 100H die Programme MAIN. SUB1 und SUB2, sowie ab Adresse 2000H die Module SUB3 und SUB4. MAIN ist das Hauptprogramm. Die übrigen Programme SUB1 bis SUB4 sind untergeordnete Module. Das entstehende Programm hat den Namen MAIN.COM.

### 3. Crossreference Generator

Das Programm 'CROSS' dient zur Erzeugung einer Symbolreferenzliste (crossreference). 'CROSS' verarbeitet Listingdateien des Assemblers, falls dieser mit dem Funktionsschalter /C aufgerufen wurde. Das Resultat des Programms 'CROSS' ist eine spezielle Listingdatei (Typ:PRN) mit Zeilennummern und einer Liste aller verwendeten Symbole, einschließlich ihrer Werte und den Zeilennummern ihres Vorkommens.

#### Kommandoaufruf

Das Kommando 'CROSS' wird von KOS aus aufgerufen und erwartet als Parameter den Namen einer speziellen Listingdatei 'crfdatei' vom Typ CRF (vom Assembler erzeugt).

CROSS =mn:crfdatei

Es gelten folgende Voreinstellungen:

mn = Mastermedium; fehlt die Angabe der Mediumnummer, so wird die Datei automatisch auf allen angeschlossenen Medien gesucht

typ = CRF

CROSS erzeugt eine Datei des Namens crfdat.PRN auf dem Mastermedium. Wie bereits erwähnt, setzt 'CROSS' voraus, daß die Datei crfdat durch folgenden Assembleraufruf erzeugt wird:

ASM =mn:crfdat/C<---

## **KDM - Kontron Debugging Module**

### **Technische Beschreibung**

Version 4.33/5.44/5.54/6.05

Juli 1984

**KDM - Kontron Debugging Module - ist ein wichtiges Hilfsmittel zum Test von Programmen in Z80-Assembler-Sprache. Es ermöglicht den protokollierenden Ablauf von Programmen bis herunter zur Mikroprozessorebene. Das zu testende Programm braucht dazu nicht verändert zu werden, es wird in seiner ablauffähigen Form als '.COM'-Datei zusammen mit KDM geladen und unter KDM gestartet.**

**Diese Beschreibung setzt genaue Kenntnisse von Betriebssystem KOS und Assemblerprogrammierung voraus.**



## Inhaltsverzeichnis

1. KDM Funktionsübersicht
2. Aufruf und Bedienung von KDM
  - 2.1 Kommandoaufbau
  - 2.2 Eingabekorrektur
  - 2.3 Eingabefehler
  - 2.4 Kommandoabbruch
3. Beschreibung der Kommandos
  - 3.1 B-Kommando: Speicherbelegungsplan ausgeben
  - 3.2 D-Kommando: Anzeigen und Ändern von Speicherstellen,  
Anzeigen von Haltepunkten und Ports,  
Disassemblieren
  - 3.3 F-Kommando: Fülle-Kommando
  - 3.4 G-Kommando: Programmstart mit Retten der Register
  - 3.5 J-Kommando: Programmstart ohne Retten der Register
  - 3.6 K-Kommando: Rückkehr zu KOS
  - 3.7 L-Kommando: Lokalisieren von Bytefolgen
  - 3.8 N-Kommando: Einzelschritt im Anwenderprogramm (Next)
  - 3.9 S-Kommando: Haltepunkte, Ports und Speicherstellen setzen
  - 3.10 R-Kommando: Register anzeigen
  - 3.11 T-Kommando: Transfer von Bytefolgen
  - 3.12 X-Kommando: KOS Kommandomanager aufrufen



## 1. KDM: Funktionsübersicht

Unter KOS steht zum Austesten von Programmen der disk-residente Debug-Monitor KDM zur Verfügung.

Bei der Ausführung eines Programms unter KDM-Kontrolle (Haltepunkte, Einzelschritt etc.) ist darauf zu achten, daß Haltepunkte nur innerhalb des Anwenderprogramms gesetzt werden, nicht aber im Betriebssystembereich. Haltepunkte in Programmen mit graphischer Betriebsart des Sichtschirms sind nicht möglich. Der Anwender muß sicherstellen, daß beim Erkennen eines Haltepunktes der alphanumerische Modus eingeschaltet ist. Programmzähler und Stackpointer der virtuellen Anwender-CPU sind mit zulässigen Werten vorbesetzt (PC=Startadresse; SP innerhalb KDM). Damit kann das Programm mit 'G' gestartet werden. KDM bietet die Möglichkeit, 'Keyboard Breaks' zu generieren. Hierzu muß die Taste CTRL-K gedrückt werden. Auch unendliche Schleifen können hiermit verlassen werden. KDM zeigt nach dem Erkennen eines 'Keyboard Breaks' (wie beim normalen Haltepunkt) den Registersatz der CPU an.

## 2. Aufruf und Bedienung von KDM

Aufruf: KDM /wdir/mn:dateiname.typ<---  
Voreinst.: mn = Mastermedium  
typ = COM  
wdir = aktuelles wdir  
Dateispez.: EDA

### E/A-Kanäle:

Eingabe ---> Kanal E-1  
Status ---> Kanal E-1  
Ausgaben ---> Kanal A-1

### Funktion:

KDM lädt diese Datei automatisch in den Anwenderspeicherbereich entsprechend der Startadresse dieses Programms. Die Angabe von '/wdir/mn:dateiname.typ' ist optional. KDM gibt als Prompt-Zeichen "KDM:" aus.

### Beispiele:

#### KDM BASIC<---

Lädt die Datei BASIC.COM unter KDM-Kontrolle. BASIC kann anschließend durch das KDM-Kommando 'G' (oder 'J 100') ausgeführt werden.

Bei Angabe eines Dateinamens wird diese Datei zusammen mit KDM geladen und kann dann getestet werden. KDM selbst liegt im Speicherbereich ab Adresse 8000H. Das Programm kann mit der Kommandodatei KDMLINK.KMD (siehe Utility Diskette) auf eine andere Adresse gelinkt werden, z.B. 'DO KDMLINK 3000<---'.

#### KDM<---

Lädt und startet das Kommando 'KDM'. KDM meldet sich mit seinen Prompt-Zeichen "KDM:" und zeigt damit seine Bereitschaft an, Kommandos entgegen zu nehmen.



## 2.1 Aufbau eines Kommandos

Jede Kommandoeingabe (jedes Kommandofeld) besteht aus einem Identifikations- und einem Parameterfeld (ID-/P-Feld), sowie dazwischenliegenden Trennzeichen.

```
<---ID-Feld---><-----Parameterfeld----->
KOMMANDOAUFTRUF TZ P1 TZ P2 TZ...Pi TZ...Pn <CR>
```

Das ID-Feld muß mit einem Großbuchstaben (A bis Z) beginnen und reicht bis zum ersten Trennzeichen (TZ1). Da der Kommandointerpreter in der Regel nur einen, höchstens aber zwei Zeichen zur Identifikation eines Kommandos benötigt, bleiben eventuell vorhandene weitere Zeichen im ID-Feld bedeutungslos.

Als Trennzeichen sind zugelassen:

```
20H (Leerzeichen)
09H CNTR-I (Tabulation)
```

Ihre Anzahl zwischen den Parametern spielt keine Rolle. Das Parameterfeld ist für verschiedene Kommandos optional. Es enthält mit Ausnahme des Registerkommandos nur hexadezimale Zahlenwerte (0...9 und A..F). Die Eingabe des Parameterfeldes ist formatfrei, d.h. führende Nullen brauchen nicht mit eingegeben zu werden.

```
Beispiel:   Die Eingaben           9
                                         09
                                         009
                                         0009
```

sind gleichbedeutend und werden als vierstellige Hexadezimalzahl 0009 betrachtet.

Bei mehr als vier Zeichen werden lediglich die letzten vier berücksichtigt. Entsprechendes gilt auch dann, wenn KDM nur 2 hexadezimale Zeichen als Eingabe erwartet. Wie bereits erwähnt, können pro (logischer) Zeile mehrere Kommandos hintereinander eingegeben werden. Der Zeilenpuffer ist 256 Bytes groß, das entspricht mehr als drei Zeilen des PSI80-Sichtschirmes.

Zur Trennung zweier Kommandos dient das Semikolon (ASCII-Code 3BH). Der ASCII-Code 0DH (<CR>, Taste RETURN) schließt die Kommandozeile ab.

Es ergibt sich das folgende Format für die Kommandozeile:

```
<-----logische Zeile (256 Bytes max)----->
KOMMANDO1;KOMMANDO2;...;KOMMANDOi;...;KOMMANDOn<---
```

## 2.2 Eingabekorrektur

Eine Korrektur der laufenden Eingabe vor Abschluß der Zeile geschieht durch

- die Taste RUBOUT (ASCII-Code 7FH):  
Löschen des gesamten Zeilenpuffers
- die Tastenkombination CNTR-H  
oder die Taste 'Cursor links' (ASCII-Code 08H):  
Löschen des zuletzt eingegebenen Zeichens.

In beiden Fällen wird an den Ausgabetreiber pro zu löschendes Zeichen die Kombination 'Backspace-Blank-Backspace' übertragen.

Bei Löschen von am Bildschirm nicht als 1-stellige Zeichen sichtbaren Codes (TAB, Controlzeichen) stimmt die Darstellung am Bildschirm (Cursorposition) nicht mit dem Abbild des Eingabepuffers überein.

## 2.3 Eingabefehler

Die Fehlermeldungen zeigen einen der folgenden Zustände an:

- die Kommando-Identifikation ist unbekannt
- das Kommandofeld beginnt mit einem unzulässigen Zeichen
- das Parameterfeld enthält unzulässige Zeichen
- das Parameterfeld enthält nicht die geforderte Anzahl von Parametern.

Die Fehlermeldungen aus KDM sind in Klartext gehalten.

Grundsätzlich führt eine Fehlermeldung zum Abbruch des Kommandos. Weitere Kommandos einer logischen Zeile werden allerdings ordnungsgemäß abgearbeitet.

## 2.4 Kommandoabbruch

Kommandos, die einen längeren Ausdruck auf dem Sichtschirm bewirken, können jederzeit unterbrochen, wieder fortgesetzt oder abgebrochen werden. Dazu überprüft das Monitorprogramm während der Kommandoausführung periodisch, ob in der Zwischenzeit eine Taste gedrückt wurde.

Für KDM gelten die KOS-Konventionen:

- CNTR-S schaltet die Geschwindigkeit der Anzeige um
- ESC bricht das Kommando ab
- Jede Eingabe stoppt und startet die Kommandoausführung
- CNTR-P schaltet die 'Paging-Funktion' ein

### 3. Kommandobeschreibung

Für jedes Kommando folgt nach einer kurzen einleitenden Funktionsbeschreibung die Syntax von Identifikations- und Parameterfeld. Für die Taste 'RETURN' steht das Symbol '<---'.

Folgende Kommandos stehen zur Verfügung:

B	KOS Speicherbelegungsplan ausgeben
D adr	Anzeigen und Ändern von Speicherstellen
D adr len	Speicherbereich anzeigen
D adr1-adr2	Speicherbereich anzeigen
DA adr len	Speicherbereich disassemblieren
DA adr1-adr2	Speicherbereich disassemblieren
DH	Haltepunkt anzeigen
DP adr	Port mit Adresse adr anzeigen
DR lrn mn segm adr	Record lrn von Medium mn in Puffer ab adr anzeigen
DW	Window anzeigen
F adr n by	Speicher ab adr mit by füllen (n Stellen)
F adr1-adr2 by	Speicher ab adr1 bis adr2 mit by füllen
G (adr)	Anwenderprogramm ab adr ausführen
J adr	nach adr springen und Programm starten
K	Rückkehr zu KOS
L adr n data data..	ab adr bis adr+n Folge data.. lokalisieren
L adr1-adr2 data...	ab adr1 bis adr2 Folge data.. lokalisieren
N n	Einzelschritt im Anwenderprogramm (n mal)
SH adr	Haltepunkt auf adr setzen
SP adr by	Port adr auf Wert by setzen
SR lrn mn segm adr	Record lrn auf Medium mn mit Daten aus Puffer ab adr beschreiben
SW adr len	Window (Speicheranz. n. break) len Byte ab adr
SW adr1-adr2	Window von adr1 bis adr2 setzen
R	Register anzeigen
R rn	Anzeigen und Ändern von Register rn
T adr1 adr2	Übertragen von Bytes von adr1 nach adr2
X	KOS Kommandomanager aufrufen

### 3.1 Speicherbelegungsplan (Bitmap) der Speicherverwaltung

Format: B<---

Ausdruck des aktuellen Speicherbelegungsplans, ähnlich dem MAP-Kommando in KOS.

### 3.2 Darstellen von Speicher- und Port-Inhalten

Das Darstelle-Kommando (Display) dient zum Ausdruck von Speicherinhalten, Ports und der Haltepunkt-Adresse. Speicherstellen können sowohl einzeln (interaktiver 'Display and Alter Mode') als auch im Block (mit ASCII-Äquivalent) ausgegeben werden.

#### a) Darstellen und Ändern einzelner Speicherinhalte

Format: D adr<---

Ausdruck des Inhalts der Speicheradresse adr.

Die Angabe von adr ist optional;

In diesem Modus arbeitet das D-Kommando interaktiv, d.h. nach dem Ausdruck des Speicherinhalts kann der Benutzer wahlweise den Inhalt des angezeigten Speicherplatzes verändern und/oder auf den nächsten/vorhergehenden Speicherplatz weiterschalten. Der Abbruch einer derartigen 'Display and Alter Sequenz' erfolgt mit dem Zeichen 'Q' (Quittierung).

Ausgabeformat: adr BB

Fünf Funktionen können nun durch Eingabe folgender Zeichen (-folgen) veranlaßt werden:

<---	Weiterschalten und Ausdruck des Inhaltes der Speicherstelle adr+1
^<---	Weiterschalten und Ausdruck des Inhaltes der Speicherstelle adr-1
xx<---	Ersetzen des Inhaltes von adr durch den Wert xx mit anschließendem Weiterschalten auf adr+1
xx Q<---	wie oben, mit anschließendem Abbruch des Kommandos
Q<---	Abbruch des Kommandos

## b) D-Darstellen von Speicherbereichen

Format: D adr1-adr2<---  
 D adr1 n<---

mit adr1 Anfangsadresse  
 adr2 Endadresse  
 n Anzahl der Speicherstellen

Entspricht die Differenz  $\text{adr2} - \text{adr1}$  bzw. die Größe  $n$  keinem ganzzahligen Vielfachen von hexadezimal 10, so erfolgt automatisch die Aufrundung auf das nächste ganzzahlige Vielfache von 10H. Im Modus 'Bereichsausdruck' sind deshalb minimal 16 Speicherinhalte ab  $\text{adr1}$  ausdrückbar, was einer physikalischen Zeile auf dem Sichtschirm des PSI80 entspricht.

Der Ausdruck erfolgt in folgendem Format:

adr1 B0 B1 B2.....BE BF \*AOA1.....AF\*

Die Größen AO bis AF kennzeichnen die ASCII-Äquivalente der Bytes B0 bis BF. Nicht abdruckfähige ASCII-Codes erscheinen als Punkt (.).

Nur im Ausgabeformat unterscheidet sich das Disassembliere-Kommando:

DA adr1 n bzw. DA adr1-adr2

## c) Darstellen des Haltepunktes

Format: DH<---

Ausdruck der Adresse, auf der momentan ein Haltepunkt gesetzt ist. Diese Adresse hat den Wert 0, falls bisher kein Haltepunkt gesetzt war, bzw. ein ehemals gesetzter Haltepunkt gelöscht wurde.

Ausgabeformat: adr

## d) Darstellen von Port-Inhalten

Format: DP portadr<---

Ausdruck des Inhaltes der Ein-/Ausgabeadresse portadr. Die korrekte Programmierung der Ein-/Ausgabe-Bausteine ist den jeweiligen Handbüchern zu entnehmen. Der KOS5-Statusport, 1 C Adresse, kann nicht gelesen werden. Sein Inhalt wird von Byte 3 hex mitgeführt.

## e) Darstellen von logischen Sätzen (display record)

Format: DR lrn mn segm adr<---

mit: lrn     logische Satznummer (logical record number)  
       mn     Mediennummer  
       segm   Nummer des adressierten 8 MByte-Segments  
              eines Mediums  
       adr     Pufferadresse ab der der Satz abgelegt wird

Voreinst.: lrn = 0  
             mn = 0  
             segm = 0  
             adr wird von KDM dynamisch festgelegt

Ausgabe des Inhalts der logischen Satznummer lrn von Medium mn im Segment segm auf die Adresse adr.

Wird ein Parameter nicht angegeben, so gelten als Voreinstellung die Werte bei der letztmaligen Ausführung des DR-Kommandos. Das DR-Kommando liest den Satz in einen Speicherbereich ein. Dort sind die üblichen Modifikationen möglich; diese Änderungen können mit dem Kommando SR auf die Diskette zurückgeschrieben werden (siehe 3.9).

## f) Darstellen des aktuellen Speicherfensters (display Window)

Format: DW<---

Ausdruck von Anfangs- und Endadresse des aktuellen Speicherfensters (siehe 3.9).

## 3.3 Fülle-Kommando

Mit dem Fülle-Kommando (Fill) können beliebige Speicherbereiche mit einer Konstanten gefüllt werden.

Format: F adr1-adr2 by<---  
       F adr1 n by<---

mit adr1    Anfangsadresse  
       adr2    Endadresse  
       n       Anzahl  
       by       einzuschreibendes Byte (Hex.)

Vorbesetzung der Parameter ist 0.

Bei diesem Kommando ist darauf zu achten, daß die von KDM und KOS benötigten Speicherbereiche nicht zerstört werden.

### 3.4 Programmstart 'Gehe' (Goto)

Das G-Kommando ermöglicht den Start eines Anwenderprogramms bei beliebiger Adresse, bzw. die Fortsetzung eines Programms nach einem Haltepunkt. Vorher werden grundsätzlich die Register der Anwender-CPU rückgespeichert.

Format: G adr<---

Ist der Parameter adr spezifiziert, erfolgt ein Sprung auf die Adresse adr, andernfalls wird der momentane Stand des Anwender-PC (Programmzählers) als Startadresse adr verwendet. Soll also beispielsweise ein Programm nach einem Haltepunkt fortgesetzt werden, so genügt das Kommando G ohne Parameterangabe. Die ordnungsgemäße Weiterführung des Programms ist gewährleistet, da das Monitorprogramm zunächst überprüft, ob die Sprungadresse einen Haltepunkt enthält. Trifft dies zu, so wird der Haltepunkt vorübergehend deaktiviert und das Programm anschließend fortgesetzt. Ein Haltepunkt bleibt bestehen, bis er explizit durch das Kommando SH oder Neuinitialisierung gelöscht wird.

### 3.5 Jump-Kommando

Im Gegensatz zum G-Kommando führt das J-Kommando direkt zum Sprung auf eine angegebene Adresse, ohne daß die Register der Anwender-CPU rückgespeichert werden.

Format: J adr<---

Ist der Wert adr nicht spezifiziert, wird stattdessen 0 eingesetzt.

.

### 3.6 KOS-Kommando: Verlassen von KDM

Mit der Eingabe von

K<--- oder  
KOS<---

wird KDM verlassen.

### 3.7 Lokalisieren von Bytefolgen

Format: L adr n b1 b2...<---  
          L adr1-adr2 b1 b2...<---

Durch dieses Kommando können Bytefolgen im Speicher gesucht werden. Dabei kann der zu durchsuchende Speicherbereich und eine beliebig lange Folge (begrenzt durch die Länge einer Eingabezeile = 255 Zeichen) von Bytes in Hexadezimaldarstellung angegeben werden.

Die gefundene Folge wird in Form des D-Kommandos, beginnend 16 Byte vor dem 1. Byte der Folge ausgegeben. Bei erfolgloser Suche erfolgt keine Ausgabe.

### 3.8 Nächster Programmschritt

Ausführung von 1...255 Einzelschritten mit Ausdruck der CPU-Register nach jedem Schritt.

Format: N n<---

Die n ( $1 < n < FFH$ ) nächsten Programmschritte ab dem momentanen Stand des Anwender-PC werden ausgeführt; n = 1 falls nicht spezifiziert.

Nach jedem Schritt werden die CPU-Register gerettet und anschließend auf den Sichtschirm ausgegeben (Format und Reihenfolge siehe Registerkommando). Beim N-Kommando wird ein eventuell gesetzter Haltepunkt gelöscht.

Da das N-Kommando interruptgesteuert ist, darf der Z80-Maschinenbefehl 'LD I,A' nicht als Einzelschritt ausgeführt werden. Der Befehl DI (Disable Interrupt) wird vom N-Kommando ignoriert.



### 3.9 Setzen von Ports, Haltepunkten, Records und Speicherbereichen.

Mit diesem Kommando können Ports und Haltepunkte gesetzt, logische Sätze auf ein Medium geschrieben und Speicherbereiche (Window) festgelegt werden.

#### a) Setze Haltepunkt

Format: SH adr<---

Setzt einen Haltepunkt auf die Adresse adr, falls der Wert adr spezifiziert wurde. Andernfalls wird ein bisher aktiver Haltepunkt deaktiviert.

Das Programm KDM arbeitet mit Software-Haltepunkten. Das Prinzip besteht darin, den Maschinencode der Speicherzelle adr mit dem Code FFH (Restart 38H) zu ersetzen. Der Befehlscode FFH bewirkt einen Restart bei der Adresse 38H, womit schließlich nach der Rettung der CPU-Register ein definierter Rücksprung vom Anwenderprogramm nach KDM erfolgt. Da der Restart-Befehl einem Call-Befehl entspricht, muß der Stapelzeiger der 'Anwender-CPU' auf einen zulässigen Speicherbereich zeigen, um das Überschreiben bereits belegter Speicherzellen zu verhindern. Um dies sicherzustellen, sollte der Anwender dafür sorgen, daß entweder

- der Stapelzeiger im Anwenderprogramm selbst definiert wird oder
- der Stapelzeiger vor dem Start des Anwenderprogramms per Register-Setz-Kommando (R SP) auf den gewünschten Wert gesetzt wird.

Es ist zu beachten, daß Haltepunkte nur dann als solche erkannt werden können, wenn der Parameter adr auf das erste Byte eines Befehls zeigt. Um dies unmittelbar nach der Eingabe des Kommandos SH überprüfen zu können, wird der Speicherbereich adr-8 bis adr+7 vor und nach dem Einsetzen des Haltepunkts mit dem Format des D-Kommandos automatisch ausgedruckt. Läuft ein Programm auf einen Haltepunkt auf, so meldet sich das Monitorprogramm mit dem Ausdruck der Registerinhalte, sowie dem als 'Window' definierten Speicherbereich.

## b) Setze Port

Format: SP adr by<---

Das Byte by wird auf den Port mit der Adresse adr übertragen. Sind Parameter nicht spezifiziert, wird dafür der Wert 0 eingesetzt. Nach dem SP-Kommando wird automatisch das DP-Kommando ausgeführt.

## c) Setze logischen Satz (set record)

Schreibt einen logischen Satz auf ein aktives Medium.

Format: SR lrn mn segm adr<---

Voreinst.: lrn = 0  
mn = 0  
segm = 0  
adr = wird von KDM dynamisch festgelegt

mit	lrn	logische Satznummer (logical record number)
	mn	Mediennummer
	segm	Nummer des adressierten 8 MByte-Segments eines Mediums
	adr	Pufferadresse aus der der Satz geschrieben werden soll

Schreibt den logischen Satz lrn auf Medium mn in das Segment segm ab Adresse adr. In Verbindung mit dem DR-Kommando kann damit gezielt ein Satz auf einem Medium geändert werden. Um unabsichtliche Zerstörungen von Medien zu vermeiden verlangt das SR-Kommando nach dem Ausdruck der Daten eine Bestätigung vom Anwender. Erst dann erfolgt der Schreibvorgang.

**Achtung:** bei Systemen mit doppelter Schreibdicke erfolgt der Rückschreibvorgang erst mit dem nachfolgenden Lesen eines beliebigen Satzes endgültig.

## d) Setze Speicherfenster (set window)

Format: SW adr1-adr2<---  
SW adr1 n<---

Dieses Kommando bestimmt denjenigen Speicherbereich, der nach einem Haltepunkt automatisch im Format des D-Kommandos ausgedruckt wird. Das SW-Kommando eignet sich insbesondere zum Ausdruck von Vektorinhalten bei Systemaufrufen. Im Gegensatz zu früheren KDM-Versionen darf ein Haltepunkt auch auf die Adresse 8 (KOSCAL) gesetzt werden.

Das SW-Kommando kann durch die Eingabe von SW 0<--- rückgängig gemacht werden.

### 3.10 Register-Kommando

Ausdruck der Registerinhalte der in KDM 'virtuell' vorhandenen Anwender-CPU. Diese stehen in einem für KDM reservierten Speicherbereich. Beim Programmstarten mit dem G- oder N-Kommando (3.4 bzw. 3.8) werden alle CPU-Register auf die Werte der Anwender-CPU gesetzt. Entsprechend werden die Registerinhalte der Anwender-CPU beim Rücksprung in das Debug Module in den dafür vorhandenen Speicherbereich gerettet.

Format: R rn<---

Wird kein Registername rn spezifiziert, erfolgt der Ausdruck des gesamten Registersatzes der CPU in untenstehender Reihenfolge:

A F B C D E H L A' F' B' C' D' E' H' L' I IX IY PC SP

Die Angabe eines Registernamens führt in einen interaktiven Anzeigemodus (Display and Alter Mode), in dem einzelne Registerinhalte angezeigt und modifiziert werden können. Das Ausgabeformat entspricht dem D-Kommando (Abschnitt 3.2).

Der Benutzer hat wieder die Möglichkeit, den angezeigten Registerinhalt zu verändern und/oder auf das nächste Register weiterzuschalten. Es gelten die Konventionen des D-Kommandos. Ein Zurückschalten auf das vorhergehende Register ist allerdings nicht möglich. Die Reihenfolge der Register entspricht der Reihenfolge beim R-Kommando (ohne rn) von links nach rechts. Die Initialisierung von KDM setzt alle Register mit Ausnahme des I-Registers auf den Wert 0. Da das N-Kommando interruptgesteuert ist, darf das I-Register der Anwender-CPU gewöhnlich nicht verändert werden, es sei denn, der Anwender sperrt den Interrupt der CPU (Befehl: DI) und verzichtet auf die interruptgesteuerten Abläufe.

Interruptbasierende Anwenderprogramme sollten die im Schreib-/Lesespeicher liegende Interrupttabelle, wofür 100H Bytes reserviert sind, verwenden.

### 3.11 Transfer-Kommando

Das Transfer-Kommando verschiebt einen beliebigen Speicherbereich in einen anderen.

```
Format:      T adr1 adr2 n<---
Voreinst:    adr1  = 0
              adr2  = 0
              n     = 0
```

verschiebt n Bytes ab Adresse adr1 in den Speicherbereich ab adr2. Es gilt folgende Zuordnung:

```
adr1      -----> adr2
adr+1     -----> adr2+1
adr1+2    -----> adr2+n
```

Auch bei diesem Kommando ist zu beachten, daß die für den Anwender unerlaubten Speicherbereiche nicht überschrieben werden.

### 3.12 KOS-Kommandointerpreter aufrufen

```
Format: X<---
```

Dieses Kommando ruft den KOS-Kommandointerpreter auf. KDM zeigt die Bereitschaft zur Entgegennahme von externen Kommandos durch das Prompt-Zeichen ">" an. Nun kann der Aufruf zum Laden des diskresidenten Kommandos eingegeben werden. Zu beachten ist jedoch, daß der benötigte Speicherbereich frei sein muß. Nach der Ausführung des Kommandos erfolgt die Rückkehr zu KDM. Dies wird durch das Prompt-Zeichen "KDM:" bestätigt.

Beispiel:

```
KDM:X<---
>IL P=*<---
```

**A N H A N G**

—

—

—

—

**Tabelle der KOS Ein-/Ausgabe Funktionen**Request Code: Funktion  
-----

80H:	Return Media Driver Identification
81H:	Output Channel Status
82H:	Input Channel Status
83H:	KOS Output Control
84H:	Character/Byte Input
85H:	String Input
86H:	Character/Byte Output
87H:	String Output
88H:	Hex Output
89H:	Read/Write Mailbox
8AH:	Read Logical Record
8BH:	Write Logical Record
8CH:	Special Functions: Input Driver
8DH:	Special Functions: Output Driver
8EH:	Read Cursor Address
8FH:	Read Scroll Address
90H:	Set Cursor Address
91H:	Set Scroll Address
92H:	Return IOD-Control Table
93H:	Assign Logical Channel Number
9CH:	Special Functions: Media Read
9DH:	Special Functions: Media Write
A0H:	Read Logical Block
A1H:	Write Logical Block

## Tabelle der KOS-Returncodes

80H	wie 81H
81H	Eingangsparameter 'out of range' Ursache: logisch Beispiel: Sektor/Spurnummer überschritten, Funktion nicht implementiert
82H	E/A-Gerät nicht bereit ('online') Ursache: physikalisch Beispiel: Diskette nicht eingelegt
83H	E/A- oder Medientreiber nicht aktiviert Ursache: logisch
84H	Datenübertragung fehlerhaft durchgeführt Ursache: physikalisch Beispiel: CRC-Fehler (Cyclic Redundancy Check Error) bei Diskettenzugriffen
85H	Datenübertragung nicht durchgeführt Ursache: physikalisch Beispiel: Sektor/Spur nicht gefunden
86H	Datenübertragung nicht durchgeführt: Richtungskonflikt Ursache: physikalisch/logisch Beispiel: Diskette mechanisch schreibgeschützt, deshalb 'Read Only'-Richtung
87H	nicht definiert
88H	Medium (Massenspeicher) voll Ursache: logisch
89H	Betriebsmittel, wie Speicher, Rechenzeit für Tasks etc. nicht verfügbar Ursache: logisch
8AH	Maximale Anzahl der gleichzeitig offenen Dateien überschritten (zulässig sind 16) Ursache: logisch



8BH	<b>Formatfehler im Inhaltsverzeichnis eines Massenspeichers (Diskette etc.)</b> Ursache: logisch / physikalisch
8CH	<b>Dateizugriffskonflikt: Datei ist schreibgeschützt</b> Ursache: logisch
8DH	<b>Dateizugriffskonflikt: Datei ist löschgeschützt</b> Ursache: logisch
8EH	<b>Dateisystem ist inkonsistent</b> Ursache: zwei oder mehrere Dateien belegen ein und denselben physikalischen Bereich einer Diskette
8FH	<b>Zugriff auf den gewünschten Satz einer Datei derzeit durch einen anderen Benutzer blockiert (nur in KOBUS-Systemen möglich)</b>
40H	<b>Anfang einer Datenübertragung</b> Kennzeichnet das erste Byte einer beginnenden Übertragung und wird zur MODEM-Steuerung benötigt.
41H	<b>Ende einer Datenübertragung</b> Wird von allen Eingabetreibern mit der Übertragung des letzten Datenbytes erwartet und beispielsweise vom COPY-Kommando als Abbruch-Kriterium betrachtet.
42H	<b>Systemfunktion busy</b> Der 'System Call Entry Point' von KOS ist reentrant. Dies bedeutet, daß Systemaufrufe zu beliebigen Zeitpunkten von Tasks oder Interruptserviceroutinen unterbrochen werden können, die ihrerseits wieder Systemaufrufe durchführen. KOS antwortet mit dem Return Code 42H, falls die aufgerufene Funktion momentan nicht ausführbar ist, also 'Busy' ist. Dies kann nur bei Systemaufrufen innerhalb von 'Backgroundtasks' oder Interrupt Service Routinen vorkommen.
43H	<b>Datei bereits eröffnet</b> Eine bereits offene Datei wurde erneut eröffnet.
44H	<b>Speicher bleibt allokiert (belegt)</b> Kehrt ein Programm mit dem Wert 44H unter (ix+5) in das Betriebssystem zurück, so bleibt der für dieses Programm reservierte Speicher geschützt (allokiert).
45H	<b>Datei logisch nicht vorhanden</b> Rückmeldung der Systemfunktion 'Open File', falls eine Datei mit Benutzerkennzeichen logisch nicht vorhanden ist, da ein falsches Benutzerkennzeichen eingegeben wurde.

**Stichwortverzeichnis beider Handbücher**  
**(CROSS-Reference)**

STICHWORT	SEITE
A-Kommando .....	TB-7
Aktivierung der Medien-Treiberprogramme .....	TD-38
Allgemeine serielle Treiber .....	TD-33
Allgemeine Systemfunktionen .....	TC-78
Allgemeiner paralleler Treiber PIO .....	TD-35
ALOAD .....	BF-17
Arbeitsdisketten .....	BD-24
ASAVE .....	BF-17
ASCII-Code-Tabelle .....	BG-5
ASCII-Datei (Ausgabe) .....	BD-17
ASM-arithmetisch/logische Ausdrücke .....	TE-12
ASM-Aufruf mit Listingausgabe .....	TE-7
ASM-Aufruf ohne Parameterangabe .....	TE-7
ASM-bedingte Assemblierung .....	TE-29
ASM-Fehlermeldungen .....	TE-35
ASM-Funktionsschalter .....	TE-8
ASM-Kommandosyntax .....	TE-5
ASM-Listing Format .....	TE-37
ASM-Listingsteuerung .....	TE-33
ASM-Makrodefinition .....	TE-22
ASM-Moduldokumentation .....	TE-32
ASM-Operatoren .....	TE-12
ASM-Programmsegmentierung .....	TE-21
ASM-Pseudooperationen .....	TE-14
ASM-Quellcode-Mnemonik .....	TE-28
ASM-Quellprogramm-Dateiformat .....	TE-9
ASM-Steuerung .....	TE-27
ASM-Symbole .....	TE-10
ASM-Zeichenketten .....	TE-11
Assembler der Kontron PSI-Systeme .....	TE-5
Assemblieren und Linken eines Treiberprogramms .....	TD-22
Ausgabe einer ASCII-Datei .....	BD-18
Ausgabesteuerung auf Sichtschirm .....	TA-19
Ausgabetreiber \$MEDO .....	TD-37
Ausschalten .....	BC-7
AUTO .....	BF-18
Auto-Linefeed-Funktion .....	TD-24
BACKUP-Kommando .....	TB-13
Basic .....	BF-1
Basic-Anweisungen .....	BF-13
Basic-Anweisungen .....	BF-29
Basic-Aufruf .....	BF-5
Basic-Beispielprogramme .....	BF-93
Basic-Datentyp .....	BF-7
Basic-Fehlermeldungen .....	BF-91
Basic Funktionen .....	BF-15
Basic-Kommandos .....	BF-12
Basic-Kommandos .....	BF-16
Basic-Operatoren .....	BF-81
Basic-Programmerstellung .....	BF-10
Basic-Programmtest .....	BF-11
Basic-Stichwortverzeichnis .....	BF-95
Basis Ein-/Ausgabe Funktionen .....	TC-34
Basisroutinen des log. Ein-/Ausgabebetreibers LIOD .....	TD-20
BASKOS .....	BF-5
BBR-Kommando .....	TB-16
Beispielprogramme auf Utility-Diskette .....	TD-54

STICHWORT	SEITE
Benutzer-Identifikation .....	BD-15
Betriebssystem-Anwenderspeicher unter KOS4/5 .....	TC-16
Betriebssystem-Anwenderspeicher unter KOS6 .....	TC-13
Bubble Memory Treiber .....	TD-36
Buchsenbelegung .....	BD-22
C-Kommando .....	TB-7
C-Kommando .....	BD-11
CALL .....	BF-30
CEDIT-Kommando .....	TB-18
CHMED-Kommando .....	TB-17
CLEAR .....	BF-33
CLOSE .....	BF-34
COLOR-Kommando .....	TB-21
COMMENT .....	TE-27
consistent .....	BD-19
CONT .....	BF-19
COPY-Kommando .....	BD-14
COPY-Kommando .....	TB-22
COPY1-Kommando .....	TB-24
COPYM2-Kommando .....	BD-14
COPYM2-Kommando .....	TB-25
COPYWFD-Kommando .....	TB-27
CP/M Kompatibilität .....	TA-26
CPFILE-Kommando .....	TB-30
CREF/XCREF .....	TE-34
CROSS .....	TE-47
Crossreferencegenerator .....	TE-47
CS16D-Kommando .....	TB-29
CS10D-Kommando .....	TB-29
CS16E-Kommando .....	TB-29
CS10E-Kommando .....	TB-29
D-Kommando .....	TB-7
DATA .....	BF-35
DATE-Kommando .....	TB-31
Datei löschen .....	BD-15
Dateiadressen .....	TA-20
Dateiadressen .....	BD-8
Dateieigenschaften .....	BD-15
Dateieigenschaften .....	TB-32
Dateieigenschaften .....	BD-8
Dateieigenschaften .....	BD-10
Dateigruppen .....	BD-9
Dateilänge .....	BE-6
Dateiname .....	BE-6
Dateinamen .....	BD-8
Dateinamen .....	TA-20
Dateiorientierte Datenträger .....	BD-8
Dateispezifikationsblock (DSB) .....	TC-57
Dateityp .....	BE-6
Dateityp .....	TA-20
Dateitypen .....	BD-8
Dateiverwaltungsfunktionen .....	TC-55
Dateiverwaltungsfunktionen .....	TC-59
Dateiverwaltungsfunktionen .....	TC-76
DEFB .....	TE-15

STICHWORT	SEITE
DEFL .....	TE-17
DEFM .....	TE-16
DEFP-Kommando .....	BD-15
DEFP-Kommando .....	TB-32
DEFS .....	TE-16
DEFW .....	TE-15
DEL .....	BF-36
DEL-Kommando .....	BD-15
DEL-Kommando .....	TB-35
DELETE .....	BF-20
DIM .....	BF-37
Directory Format Fehler .....	TB-40
Diskette formatieren .....	BD-16
Disketten-Schreibschutz .....	BC-5
Diskettenkopieren .....	BD-14
Diskettenprobleme .....	BC-4
DO-Kommando .....	BD-16
DO-Kommando .....	TB-36
DUMP-Kommando .....	TB-38
E/A-Treiber .....	TC-18
E/A-Treiber Aktivierung .....	TB-46
E/A-Treiber Auflistung .....	TB-47
E/A-Treiber Deaktivierung .....	TB-46
EAK-Kommando .....	TB-45
ECB/KCP-Subsystem .....	TA-23
ECB/SASI .....	TD-42
Editor .....	BE-5
Editor-Abspeichern .....	BE-19
Editor-Aufruf .....	BE-6
Editor-Betriebsarten .....	BD-20
Editor-Cursororientierte Betriebsart .....	BE-17
Editor-Einführung .....	BD-20
Editor-Eingabe .....	BE-9
Editor-Einstellung des Bildschirmformates .....	BE-20
Editor-Fetch .....	BE-15
Editor-Groß-, Kleinschreibung .....	BE-16
Editor-Hilpfunktion .....	BD-21
Editor-Hilpfunktion .....	BE-7
Editor-Hilpfunktion .....	BE-16
Editor-Kommando-Übersicht .....	BE-8
Editor-Kommandos .....	BE-7
Editor-Kommandowiederholung .....	BE-16
Editor-Kommunikation mit externen Dateien .....	BE-14
Editor-modifizieren .....	BE-13
Editor-positionieren .....	BE-11
Editor-Save .....	BE-14
Editor-Statusanzeige .....	BE-7
Editor-Steuer- und Semigrafikzeichen .....	BE-20
Editor-Zeilen ersetzen .....	BE-10
Editor-Zeilenlöschen .....	BE-10
Ein-/Ausgabe .....	BD-11
Ein-/Ausgabe-Treiber .....	TD-19
Ein-/Ausgabe: logische Kanäle .....	TA-9
Ein-/Ausgabetreiber-Aktivierung .....	BD-16
Ein-/Ausgabetreiber-Aufbau .....	TD-19
Eindeutige Dateiadressen .....	TA-21
Eingabefehler .....	BD-6

## STICHWORT

## SEITE

Eingaben zur Ausgabesteuerung .....	BD-7
END .....	BF-38
END .....	TE-32
EQU .....	TE-17
ESCAPE-Sequenzen .....	TC-23
EXTIM .....	TE-25
EXTERNAL .....	TE-19

FETCH - Editor-Kommando .....	BF-22
FETCH .....	BF-21
File Control Block FCB .....	TC-57
Filter OKI84G.FIL bei KOS5-Systemen .....	TD-27
Filter OKI84G.FIL bei KOS6-Systemen .....	TD-27
Filtermodul xxx.FIL .....	TD-21
Floppy Disk-Laufwerkstest .....	TD-61
FOR...NEXT .....	BF-39
FORMAT .....	BD-16
FORMAT-Kommando .....	TB-39
Formatieren von 8"-Disketten .....	TD-41
Formatieren von Harddisks mit DTC-Controller .....	TD-45
Formatieren von Harddisks mit Adaptec-Controller .....	TD-45
Formatieren von IOMega Wechselplatten .....	TD-48
Formatierung von Plattenspeichern .....	TD-42
Formatierung von Plattenspeichern .....	TD-48
FORMATW-Kommando .....	TD-45
FORMATW5-Kommando .....	TD-45
FSCHECK-Kommando .....	TB-40
Funktionstastenbelegung von KOS6.05 .....	TD-50

Geheim-Dateien .....	BD-15
Generierung von KOS6 .....	TB-49
GET .....	BF-41
GLOBAL .....	TE-18
GOSUB .....	BF-42
GOTO .....	BF-44
Graphik in ASSEMBLER-Programmen .....	TD-10
Graphik in FORTRAN-BASIC80-Programmen .....	TD-11
Graphik-Alphanumerische Zeichen .....	TD-9
Graphik-Initialisierungen .....	TD-8
Graphik-Punkt-Manipulationen .....	TD-8
Graphikerweiterungen unter KOS6 .....	TD-14
Graphikerweiterungen unter KOS6.05 für H-Systeme .....	TD-15
Graphikpaket .....	TD-7

H-Kommando .....	TB-8
Halbleiterfloppy unter KOS6 .....	TC-15
Hardware-Schnittstellenmodul HOD .....	TD-22
HELP-Funktion .....	BD-7

I-Kommando .....	BD-11
I-Kommando .....	TB-8
I-Register .....	TC-17
IF...THEN .....	BF-45
IL-Kommando .....	BD-17
IL-Kommando .....	TB-42

STICHWORT	SEITE
Implementierte Druckfunktionen in xxx.FIL .....	TD-23
Inbetriebnahme .....	BC-3
Include-Anweisung .....	TE-31
inconsistent .....	BD-19
inconsistent .....	TB-41
INFO-Kommando .....	TB-43
INFO-Kommando .....	BD-17
Inhaltsausgabe im Langformat .....	BD-17
INISER-Kommando .....	TB-44
INPUT # .....	BF-48
INPUT .....	BF-46
Interrupttabelle .....	TC-17
INV .....	BF-49
INV .....	BF-73
IODC-Kommando .....	BD-16
IODC-Kommando .....	TB-45
IPCA .....	TA-9
IPCA .....	TC-14
IRP .....	TE-24
IRPC .....	TE-25
Kanalzuweisung .....	TB-47
KDM - Kommandoaufbau .....	TF-7
KDM .....	TF-5
KDM-Aufruf .....	TF-6
KDM-Bedienung .....	TF-6
KDM-Darstellen von Speicher- und Port-Inhalten .....	TF-10
KDM-Fehlermeldungen .....	TF-8
KDM-Fülle-Kommando .....	TF-12
KDM-Jump-Kommando .....	TF-13
KDM-Kommandoabbruch .....	TF-8
KDM-Kommandobeschreibung .....	TF-9
KDM-KOS-Kommandointerpreter aufrufen .....	TF-18
KDM-Lokalisieren von Bytefolgen .....	TF-14
KDM-Nächster Programmschritt .....	TF-14
KDM-Programmstart .....	TF-13
KDM-Register-Kommando .....	TF-17
KDM-Setzen von Haltepunkten .....	TF-15
KDM-Setzen von Ports .....	TF-15
KDM-Setzen von Records .....	TF-15
KDM-Setzen von Speicherbereichen .....	TF-15
KDM-Speicherbelegungsplan .....	TF-10
KDM-Transfer-Kommando .....	TF-18
KDM-Verlassen .....	TF-14
KOBUS .....	TA-22
KOBUS-Masterstation .....	TA-23
KOBUS-Peripherie .....	TA-25
KOBUS-Preprozessor .....	TA-23
KOBUS-Slavestation .....	TA-23
KOBUS-Terminalstation .....	TA-23
KOKOM-Kommando .....	TB-48
KOKOMS-Kommando .....	TB-48
Kommando-Eingabeformat (KOS) .....	TA-18
Kommandodatei .....	BD-16
Kommandoeingabe .....	BD-6
Kommandoeingabe .....	TA-16
Kommandozeile .....	BD-6
Kommandozeile .....	TA-18

STICHWORT	SEITE
Konfigurierung mit residenten E/A-Treiber bei KOS6 .....	TC-28
Konfigurierung von KOS .....	TC-26
Konfigurierung von KOS0.SYS .....	TB-50
Konfigurierung von KOS4.SYS .....	TB-51
Konfigurierung von KOS5.SYS .....	TB-52
Konfigurierung von KOS7.SYS .....	TB-54
Konfigurierung von KOS8.SYS .....	TB-57
Konfigurierung von KOSA.SYS .....	TB-58
Konfigurierung von KOSC.SYS .....	TB-58
Konfigurierung von KOSD.SYS .....	TB-58
Kopieren von Dateien .....	BD-18
KOS .....	BF-50
KOS-Aufbau .....	TA-6
KOS-diskresidente-Systemkommandos .....	BD-14
KOS-diskresidente Systemkommandos .....	TB-13
KOS-Graphikpaket .....	TD-7
KOS-IF .....	TC-18
KOS-IF .....	TC-28
KOS-interne Systemkommandos .....	TB-7
KOS-interne Systemkommandos .....	BD-11
KOS-Systemfunktionen .....	TA-9
KOS-Systemfunktionen .....	TC-29
KOS-Systemkommandos .....	TA-15
KOS-Systemkommandos .....	TB-1
KOS-Utilities .....	TD-3
KOS.INI .....	BD-6
KOSGEN-Kommando .....	TB-49
KPPLOAD KPP-Kommando .....	TB-59
LALL/SALL/XALL .....	TE-34
Laufwerk/Disketten-Test .....	TD-56
LET .....	BF-51
LINELEN .....	BF-52
LINK-Allgemeine Kommandosyntax .....	TE-40
LINK-Aufruf ohne Parameterangabe .....	TE-40
LINK-Funktionsschalter .....	TE-41
LINK-Meldungen .....	TE-43
Linker für Kontron PSI-Systeme .....	TE-39
LIST .....	BF-23
LIST/XLIST .....	TE-33
LOAD .....	BF-24
LOCAL .....	TE-26
Login .....	BD-6
Login .....	TA-14
Löschschutz .....	BD-15
M-Kommando .....	BD-12
M-Kommando .....	TB-9
MACRO .....	TE-22
Mailbox .....	TC-46
Mailbox .....	TC-76
MAKEFS-Kommando .....	TB-60
MAP-Kommando .....	TB-61
Mastermedium .....	BD-8
Medien .....	BD-8
Medienbelegung .....	BD-19
Medienkonzept .....	TA-10



# S T I C H W O R T V E R Z E I C H N I S

STICHWORT	SEITE
Mediennummer .....	BD-8
Medienorganisation .....	TA-12
Mehrdeutige Dateiadressen .....	TA-21
MEMTEST .....	TD-56
MOVE-Kommando .....	BD-18
MOVE-Kommando .....	TB-62
Multi-PSI-Systeme (Aufbau) .....	TA-22
Multi-User-Fähigkeit .....	TA-24
Multitasking .....	TC-91
Muster eines Ausgabe-Treibers .....	TD-64
N-Kommando .....	BD-12
N-Kommando .....	TB-9
NEW .....	BF-26
O-Kommando .....	TB-9
ON .....	BF-54
ON ERROR .....	BF-56
ON INTR .....	BF-57
OPEN .....	BF-58
ORG .....	TE-20
Organisation des Speichers .....	TC-9
Organisation des Speichers unter KOS4/5 .....	TC-16
Organisation des Speichers unter KOS6 .....	TC-13
OUT .....	BF-60
P-Kommando .....	BD-12
P-Kommando .....	TB-10
PAGE .....	TE-33
PCNT .....	TD-62
Peripherie-Bausteine (Test) .....	TD-56
Peripheriegeräte .....	BD-22
PLIST .....	BF-23
PLOT .....	BF-61
Plotten .....	TD-9
POKE .....	BF-63
PRINT # .....	BF-66
PRINT .....	BF-64
PRINT USING .....	BF-67
PRINT-Kommando .....	BD-18
PRINT-Kommando .....	TB-64
PRINTX .....	TE-27
Programmlaufzeiten .....	TD-62
Promresidenter Testdebugger .....	TD-58
Property-Flags .....	TB-33
PSITEST .....	TD-56
PSTAT-Kommando .....	TB-65
Public Files .....	TA-11
Public Dateien .....	BD-15
R-Kommando .....	TB-10
R-Property .....	TA-24
RADIX .....	TE-28
RANDOM .....	BF-69
READ .....	BF-70

STICHWORT	SEITE
Real Time Clock .....	TD-55
RECORD .....	BF-71
Record Locking .....	TC-77
Reinigungsdiskette .....	BC-4
REM .....	BF-72
REN-Kommando .....	BD-19
REN-Kommando .....	TB-66
RENUMBER .....	BF-27
REPT .....	TE-24
RES .....	BF-49
RES .....	BF-73
Reservierte Speicheradressen .....	TC-9
Residente E/A-Treiber .....	TC-20
RESTORE .....	BF-74
RESTORE .....	TB-28
RLOAD PTASK-Kommando .....	TB-68
RLOAD SELECT-Kommando .....	TB-69
RLOAD TIME-Kommando .....	TB-71
RLOAD-Kommando .....	TB-67
RUN .....	BF-28
Rückmeldecode .....	TC-32
S-Kommando .....	TB-11
SAVE .....	BF-24
SAVE .....	TB-27
Schreibgeschützt .....	BD-15
SET .....	BF-49
SET .....	BF-73
SETDATE-Kommando .....	TB-72
SETTIME-Kommando .....	TB-72
SM-Numerische Konstanten .....	TE-11
SOFTKEY-Utility .....	TD-49
Speicherbelegungspläne .....	TA-7
Speichertest .....	TD-56
Speichertest .....	TD-59
SPOOL-Kommando .....	TB-73
Spooling .....	TA-25
Spooling .....	TB-74
Start des Betriebssystems .....	BD-6
Start des Betriebssystems .....	TA-13
Statusinformation für H-Systeme .....	TC-49
STATUS-Kommando .....	BD-19
STATUS-Kommando .....	TB-75
Statusport .....	TC-12
Steuerzeichen für \$MON .....	TB-37
Steuerzeichen für die Kommandoedition .....	TA-17
Steuerzeichen für die Monitoranzeige .....	TA-16
Steuerzeichen für Treiber \$MON .....	TC-21
STOP .....	BF-76
STOP-Kommando .....	TB-76
STRING .....	BF-78
SUBTTL .....	TE-32
Syntaxausdruck (Help-Funktion) .....	TA-19
SYSGEN-Kommando .....	TB-77
Systemaufrufvektor .....	TC-30
Systemkommandos .....	BD-5
Systemstart .....	TA-14

STICHWORT	SEITE
Tabelle der KOS Ein-/Ausgabe Funktionen .....	TG-3
Tabelle der KOS-Returncodes .....	TG-4
TASK-Kommando .....	TB-78
TCNT .....	TD-62
Testprogramme .....	TD-56
TIME .....	TD-55
TITLE .....	TE-32
TMED .....	TD-56
TRACE .....	BF-80
TRACEOFF .....	BF-80
Treiber \$MON .....	TC-21
Treiber für DAISY M45-Drucker .....	TD-28
Treiber für Diskettenstationen .....	TD-38
Treiber für Drucker OKI-Microline .....	TD-24
Treiber für externe 8"-Diskettenlaufwerke .....	TD-40
Treiber für interne Diskettenlaufwerke .....	TD-39
Treiber für Olympia ESW103 .....	TD-30
Treiber-\$CPM .....	TD-51
Treiber-\$MICP .....	TD-39
Treiber-\$PIO .....	TD-35
Treiber-\$PSIA .....	TD-33
Treiber-\$PSIB .....	TD-33
Treiber-\$SIO .....	TD-34
Treiber-\$SIOA .....	TD-34
Treiber-\$SIOB .....	TD-34
Treiber-\$SXCP .....	TD-40
Treiber-\$SXSS .....	TD-40
Treiber-\$VMED .....	TD-35
Treiber-\$WIN2 .....	TD-42
Treiber-\$XMED .....	TB-25
Treiber-Programm für externe Plattenspeicher .....	TD-42
Treiberverwendung in KOS-Systemkommandos .....	TD-34
Umbenennung einer Datei .....	BD-19
Umsetztreiber für CP/M-Aufrufe .....	TD-51
Umsetztreiber für KOS 3.2 Disketten .....	TD-39
Universelle Druckertreiber .....	TD-32
Utility-Diskette .....	BD-11
Virtueller Medientreiber .....	TD-35
WDIR-Kommando .....	TB-79
WFD-Winchester Format und Debug Utility .....	TD-45
WordStar-Modifikation für OKI84/92/93 .....	TD-26
WordStar-Modifikationen für DAISY M45 .....	TD-29
WordStar-Modifikationen für Olympia ESW103 .....	TD-31
Working Directories .....	BD-10
Working Directory .....	TA-11
X-Kommando .....	BD-13
X-Kommando .....	TB-11
Y-Kommando .....	TB-12
Zeichensatz definieren .....	TB-18
Zuordnung der Laufwerknummern zu den Mediennummern .....	TD-38