
# UDOS-Software
# Systemhandbuch

**EAW** *electronic*

## P8000

*transcribed in ~16h by O. Lehmann during 2006-06-30 and 2007-02-20*
*Version 1.4 (2007-10-27)*

Diese Dokumentation wurde von einem Kollektiv des Kombinates

VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"

erarbeitet.

 Im Interesse einer staendigen Weiterentwicklung werden die Nutzer gebeten, dem Herausgeber Hinweise zur Verbesserung mitzuteilen.

Herausgeber:
Kombinat
VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"
Hoffmannstrasse 15-26
BERLIN
1193

WAE/03-0100-01

Ausgabe: 12/86

Die vorliegende Dokumentation unterliegt nicht dem Aenderungsdienst.

Spezielle Hinweise zum aktuellen Stand der Softwarepakete befinden sich in README-Dateien auf den entsprechenden Vertriebsdisketten.

In folgenden Unterlagen werden weitere Hinweise zur Arbeit mit dem Betriebssystem UDOS gegeben:

P8000 Dokumentation:

- Einfuehrung in die Software des Geraetesystems P8000

- UDOS-Software Dienstprogramme

- UDOS-Software Mikroprozessorsoftware

- UDOS-Software Programmiersprachen

Allgemeine Literaturempfehlung:

- Classen, L.; Oefler, U.: Wissensspeicher Mikrorechnerprogrammierung. Fachbuchreihe Technische Informatik. 1. Auflage. Berlin: VEB Verlag Technik 1986

- Betriebssystem UDOS 1526 Anwenderdokumentation. VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt 1983

Inhaltsverzeichnis

# 1. Einfuehrung

Dieses Handbuch beschreibt das UDOS-Betriebssystem mit seinen Systemeigenschaften. UDOS (Universelles Disketten-Operationssystem) ist ein komfortables, Floppy-Disk-orientiertes Betriebssystem fuer eine U880-Rechnerkonfiguration, die durch den 8-Bit-Teil des Programmier- und Entwicklungssystems P8000 realisiert wird.
Unter UDOS werden die Daten in Form von Dateien auf der Diskette abgelegt. Auch das Betriebssystem selbst ist mit allen seinen Komponenten auf Diskette gespeichert, wobei der Kern des Systems beim Start in den Speicher geladen wird und dort fuer die Arbeit zur Verfuegung steht. Neben der unter UDOS moeglichen Dateiarbeit, bei der sich der Anwender um die Anordnung der Daten auf der Diskette nicht kuemmern muss, bietet UDOS eine Reihe weiterer Vorteile wie wahlfreie Zuordnung von Ein/Ausgabe-Datenstroemen, automatische Speicherplatzverwaltung und einen umfangreichen Kommandosatz. Weiterhin ist unter der Steuerung von UDOS Entwicklungssoftware fuer die in der DDR produzierten Mikroprozessorfamilien U880, U881/U882/U883 und U8000 verfuegbar. Daneben existieren UDOS-Softwarepakete mit Compilern fuer hoehere Programmiersprachen. Die Struktur von UDOS macht es moeglich, jederzeit neue Ein/Ausgabe-Geraete einzubinden, sowie neue Kommandos zu kreieren und in UDOS einzufuegen.
Die UDOS-Implementation auf dem P8000 unterstuetzt die Arbeit mit 5 1/4" Disketten in 40-Spur-, 80-Spur-Laufwerken (einseitig) und 80-Spur-Laufwerken (doppelseitig). Die kleinste physische Einheit auf einer Diskette ist ein Sektor mit 256 Byte. UDOS verwaltet einen Speicherbereich von 64 KByte.
Im Einzelnen werden in dieser Dokumentation der Ladevorgang, das Operationssystem OS, die UDOS-Kommandos, das Dateiverwaltungssystem NODS und die Realisierung von Ein/Ausgabe-Zugriffen beschrieben

# 2. Struktur des Betriebssystems

## 2.1. Start des Systems

Das Betriebssystem UDOS besteht aus dem 4 KByte U880-Softwaremonitor SMON, dem 6 Kbyte Operationssystem OS, dem 8 KByte Dateiverwaltungssystem NDOS und externen UDOS-Standardkommandos. Ausser dem 4 KByte U880-Softwaremonitor, der im EPROM-Firmwarespeicher abgelegt ist, befinden sich alle Programmteile als Dateien auf der Systemdiskette.
Nach dem Einschalten des Geraetes oder nach Betaetigung der Reset-Taste meldet sich der U880-Softwaremonitor mit dem Promptzeichen '>'. Der im EPROM-Firmwarespeicher abgelegte U880-Softwaremonitor wurde in die untersten 4 KByte des 64 KByte Speichers geladen. Mit dem Umladen wird vom U880-Softwaremonitor ein Parametervektor zum einlesen der eigentlichen UDOS-Urladeroutine von der Systemdiskette aufgebaut. Es gibt damit zwei Moeglichkeiten fuer das Laden des Betriebssystems:

- Betaetigung von Return (Wagenruecklauf), ist nur nach Reset moeglich
- Eingabe des Debug-Kommandos 'O' (Buchstabe O)

Ist im Laufwerk 0 keine Systemdiskette eingelegt, so meldet sich der U880-Softwaremonitor mit der Ausschrift:

INSERT SYSTEMDISK

Die eingelegte Diskette ist keine Systemdiskette und ist gegen eine solche auszutauschen. Ist eine Systemdiskette eingelegt, wird von Spur 0, Sektor 0 der Diskette im Laufwerk 0 ein 80H Byte langes Urladeprogramm ausgelesen und gestartet. Dieses Programmm laedt ein 300H Byte langes Programmm von Spur 16, Sektoren 7,8,9 und startet es. von diesem Programm wird dann das Suchen von OS und NDOS auf der Diskette, das Uebernehmen dieser Programme uin den Speicher und der Start des OS durchgefuehrt. Diese Aktionen sind nur dann ordnungsgemaess durchfuehrbar, wenn sich die Systemdiskette im Laufwerk 0 befindet. Fuer den Systemstart ist unter UDOS kein anderes Laufwerk nutzbar. Das gilt auch fuer die aus dem U880-Softwaremonitor aufrufbaren Kommandos GET und SAVE.
Der Aufruf des OS kann den Aufruf anderer Programme einschliessen, da die Moeglichkeit der Implementierung einer Kommandofolge (in der Datei OS.INIT enthalten und aenderbar) vorgesehen ist. Ausserdem wird beim Start des OS der verfuegbare Speicherbereich getestet und eine Fehlermeldung generiert, falls die letzte belegbare RAM-Zelle nicht gleichzeitig das Ende eines 4 KByte Bereiches ist.
UDOS meldet sich nach dem Start des OS und der Abarbeitung der OS.INIT mit folgender Ausschrift:

```
P8000 UDOS2.2 (C)ZFT/KEAW
Datum der Systemdiskette
%
```

Das Promptzeichen des Betriebssystems ist das Zeichen '%'. Mit jedem '%' zeigt es seine Bereitschaft zur Uebernahme eines Kommandos an.

## 2.2. System-Initilaisierung

Beim Start des Systems als Teil des Urladevorgangs erfolgen bestimmte Initialisierungen, die zum Betrieb des Rechners erforderlich sind:

- Zuweisung des Terminal-Betriebsprogramms CON
- Zuweisung von NDOS als Master-Device zur Identifizierung der verfuegbaren Diskettenlaufwerke
- Bestimmung des zur Verfuegung stehenden Speicherbereiches durch Schreiben und Lesen eines Zeichens ueber den gesamten Bereich, falls die letzte gueltige Adresse nicht eine 4 KByte Grenze ist, wird eine Warnung ausgegeben
- Schutz der Speicherbereiche (d.h. Eintragung in den Speicherbelegungsplan), die durch SMON, NDOS, OS und CON belegt sind

Weiterhin werden beim Initialisierungsvorgang bestimmte Systemkonstanten geladen, die fuer den Kontakt mit dem Terminal notwendig sind:

- Zuweisung der Kodierung 7FH fuer das LINDEL-Zeichen (=line delete), loescht eine vorherige Zeile in ihrer Gesamtheit
- Zuweisung der Kodierung 08H fuer das CHRDEL-Zeichen (=character delete), loescht das vorhergehende Zeichen
- Zuweisung des NULLCT-Zeichens (=null count) mit NULLCT=1, gibt die Anzahl von Leerzeichen an, die nach einem Wagenruecklauf ausgesandt werden
- Setzen des EXTINI-Bits von SYSFLG (Bit 2), bewirkt die Durchfuehrung des Initialisierungskommandos DO 0/OS.INIT, damit wird die Kommandodatei OS.INIT von der Diskette in Laufwerk 0 geladen und gestartet

Initialisiert werden ausserdem folgende Werte:

- das Interruptregister I wird auf den Basiswert 0FH gesetzt
- der Stackpointer wird mit dem Wert D000H geladen
- die Restart-Adressen 0 und 38H sind vom Anwender nicht verfuegbar, Restart 38H wird vom System als Unterbrechungspunktroutine zur Verfuegung gestellt

## 2.3. Kommandostruktur

Nach dem Start, der Initialisierung und der Meldung erwartet das System die Kommandoeingabe. In UDOS sind zwei Arten von Kommandos verfuegbar:

- Interne Kommandos
  Es stehen 11 interne Kommandos zur Verfuegung. Diese

werden als Teil des Operationssystem OS mit dem Aufruf von OS in den Speicher geladen und sind von dort aufrufbar.

-Externe Kommandos
Externe Kommandos sind alle Prozedurdateien. Im Rahmen des Betriebssystems werden dem Anwender 30 externe Kommandos zur Verfuegung gestellt. Der Anwender kann aber beliebig viele Kommandos neu kreieren und problemlos in UDOS einbinden.

Wird ein externes Kommando aufgerufen, so durchsucht das Betriebssystem alle vorhandenen Directory-Dateien nach dem Prozedurnamen. Die Anzahl der externen Kommandos ist nur von der Zahl der Prozedurdateien auf den momentan aktiven Diskettenlaufwerken abhaengig.

## 2.4. Ein/Ausgabe-Zuweisungen

Die Ein/Ausgabe-Struktur von UDOS macht die Programmentwicklung von den vorhandenen Ein/Ausgabe-Geraeten unabhaengig. Folgende Unterscheidungen sind vorgenommen:

.logische Funktion (logical unit)
.physische Einheit (pyhsical unit)
.Ein/Ausgabe-Treiber (I/O-device-handler)

Logische Funktion:
Als logische Funktion sind Ein/Ausgabe-Funktionen fuer bestimmte Aktivitaeten gekennzeichnet. In UDOS sind drei Funktionen bereits definiert:

-Terminaleingabe (console input), mit der Bezeichnung CONIN
-Terminalausgabe (console output), mit der Bezeichnung CONOUT
-Druckerausgabe, mit der Bezeichnung SYSLST

Physische Einheit:
Als physische Einheit werden die Ein/Ausgabe-Geraete (Terminals, Drucker, etc.) mit den zugehoerigen Schnittstellen bezeichnet.

Ein/Ausgabe-Treiber:
Damit wird die zur Bedienung der physischen Einheit noetige Software bezeichnet. Bevor mit einem Treiber gearbeitet werden kann, muss er in den Speicher geladen und dort vor Ueberschreiben durch andere Programme geschuetzt sein. Das erfolgt durch Aktivierung und Initialisierung auf UDOS-Kommandoebene.

Allgemein gilt, dass Ein/Ausgabe-Zugriffe mit Hilfe eines Standardvektorformats erfolgen. Dieser Paramtervektor, der in Abschn. 6.2. genau beschrieben ist, gibt die Datentransferadresse, Datenblocklaenge, den Abschlusskode und optional einen zweiten Parametervektor an.

# 3. UDOS-Operationssystem OS

## 3.1. Arbeit mit Dateinamen

In UDOS bestehen die Dateinamen aus drei Teilen:

a) dem Namen des Ein/Ausgabe-Treibers
b) der Nummer des Diskettenlaufwerks
c) dem eigentlichen Dateinamen

Der Dateiname:

Der UDOS Dateiname besteht aus 1 bis 32 Zeichen, das erste Zeichen muss ein Buchstabe sein. Die uebrigen 31 Zeichen koennen aus Buchstaben oder Ziffern, Fragezeichen, Punkt oder Unterstreichung bestehen. Gross- und Kleinbuchstaben werden als verschiedene Zeichen interpretiert. Steht ein Punkt ('.') innerhalb eines Dateinamens, so werden die nachfolgenden Zeichen als Namenserweiterung bezeichnet. Erweiterungen koennen in den verschiedensten Anwendungen nuetzlich sein. So ist es z.B. notwendig, die verschiedenen Dateien, die mit Assembler und Linker aus einem einzigen Quellprogramm (name.S) erzeugt werden, zu unterscheiden.

Die Nummer des Diskettenlaufwerks:
Die Floppy-Disk-Laufwerke sind durch eine Ziffer zwischen 0 und 3 gekennzeichnet. Im P8000 hat das untere Floppy-Disk-Laufwerk die Nummer 0 und das obere die Nummer 1. Das Zeichen '*' fuer die Angabe der Laufwerknummer bewirkt den Suchlauf 1,2,3,0.

Der Name des Ein/Ausgabe-Treibers:
Treibernamen sind im Prinzip Dateinamen. Beim Aufruf wird das Zeichen '$' vor den Namen gesetzt. Dieses Zeichen wirkt als Begrenzungszeichen und ist kein Bestandteil des Namens selbst. Das gewuenschte Treiberprogramm muss im System entweder bei der Initialisierung oder mit ACTIVATE (s. Abschn. 4.2.) aktiviert werden. Innerhalb des Betriebssystems UDOS stehen nach dem Einschalten fuenf Treiberprogramme zur Verfuegung:

NDOS - Dateisystem und Diskettenorganisation
CON - Terminaltreiber
NULL - Nulltreiber
PCON - Terminaltreiber im U880-Softwaremonitor
FLOPPY - Diskettentreiber im U880-Softwaremonitor

Beider Handhabung von Treibernamen ist zu beachten:

- zur Trennung des Treibernamens von der Nummer des Floppy-Disk-Laufwerks dient das Zeichen ':'
- zur Trennung des Treibernamens und/oder der Laufwerksnummer vom Dateinamen dient das Zeichen '/'
- fehlt die Angabe des Treibers im Dateinamen, verwendet das System den so genannten Master, urspruenglicher Master ist NDOS, kann aber durch das Kommando MASTER (s. Abschn. 4.29.) neu definiert werden

Beispiele:

| Kommando-eingabe | Beschreibung |
|---|---|
| CAT | Das Programm CAT wird auf allen Disketten aller aktiven Laufwerken gesucht und wenn vorhanden, geladen und gestartet. |
| 1/CAT | Das Programm CAT wird nur auf der Diskette in Laufwerk 1 gesucht, geladen und gestartet. |
| :1/CAT | Entspricht der Eingabe von 1/CAT. |
| $ABDOS/CAT | Das Programm CAT wird auf den Disketten aller aktiven Laufwerken, aber nur mit Hilfe von ABDOS gesucht und wenn vorhanden, geladen und gestartet. |
| $ABDOS:1/CAT | Das Programm CAT wird auf der Diskette in Laufwerk 0 und nur mit der Hilfe von ABDOS gesucht und wenn vorhanden, geladen und gestartet. |

## 3.2. Speicherorganisation

Das UDOS-Operationssystem OS enthaelt ein Speicherorganisationsprogramm, welches die Belegung des 64 KByte Speichers in Bereichen zu 80H Byte prueft. Der Status des gesamten Speichers ist in einer Bit-Liste festgehalten, wobei jedes Bit zu einem bestimmten 80H Byte Speichersegment zugewiesen ist. Dabei bedeutet Bit=1 ein belegtes Segment und Bit=0 ein freies Segment. Der Speicherraum der Systemprogramme wird vom Speicherorganisationsprogramm belegt gesetzt. Unterprogrammaufrufe zum Einsprungspunkt MEMMGR (Adresse siehe Anhang B) erlauben Belegung, Freigabe oder Feststellung der Belegung bestimmter Speicherbereiche. Die Nutzung von MEMMGR aus Anwenderprogrammen heraus ist in Anhang C beschrieben. Speicherbelegungsoperationen sind auch auf Kommandoebene moeglich (siehe ALLOCATE, DEALLOCATE, DISPLAY).

## 3.3. Kommandoeingabe und Durchfuehrung

Nach der Meldung des Betriebssystems mit der Ausgabe von '%' erwartet es die Eingabe des Bedieners. Alle dann eingegebenen Zeichen (maximal 205) einschliesslich des ersten Wagenruecklaufzeichens (Return) gelangen in den Kommandokettenpufferspeicher. Mit dem Zeichen ';' werden die einzelnen Kommandos abgeschlossen, jedoch erst ausgefuehrt, nachdem das erste Zeichen Wagenruecklauf (Return) eingegeben wurde. Das Zeichen ';' muss nicht nach dem letzten Kommando eingegeben werden, sondern dient nur als Trennzeichen zwischen mehreren Kommandos auf einer Zeile. Kommandos koennen in der Kommandokette von optionalen Parametern durch die Zeichen '(', Pause (13H), ')', Horizontaltabulator (09H) und ',' getrennt werden.
Soll das eingegebene einzelne Kommando mit seiner optionalen Parameterliste oder die gesamte Kommandokette als Echo an das Terminal geliefert werden, damit der Anwender eine Kontrolle ueber die vorherige Eingabe hat, muss das Komman-

do VERBOSE (s. Abschn. 4.40.) eingegeben werden, da implizit BRIEF-Mode (s. Abschn. 4.4.) eingestellt ist.
Nach der Eingabe des Kommandos wird der Inhalt des Eingabe-Pufferspeichers zuerst mit der Liste der internen Kommandos verglichen. Es wird nur soweit geprueft, wie zur Unterscheidung des einen Kommandos von einem anderen noetig ist. Ist keine Unterscheidung moeglich, wird das erste entsprechende Kommando gewaehlt. So unterscheidet die Eingabe von 'D' nicht zwischen den internen Kommandos DEBUG und DEALLOCATE, sondern es wird nur aufgrund der Reihenfolge DEBUG gewaehlt. Die Eingaben von 'D', 'DE', 'DEB', 'DEBU' oder 'DEBUG' rufen somit alle den U880-Softwaremonitor auf. Handelte es sich bei der Eingabe nicht um ein internes Kommando, wird die Kommandokette als Dateiname interpretiert und auf den Disketten gesucht. Ist die Datei gefunden, wird untersucht, ob es eine Prozedurdatei ist und ob der benoetigte Speicherbereich noch nicht belegt ist. Die im Deskriptor der Datei eingetragenen Werte fuer die untere und obere Adresse werden mit dem Speicherorganisationsprogramm geprueft. Die Datei wird nur dann geladen und gestartet, wenn es sich um eine Datei vom Typ P (Prozedur) handelt und der zulaessige Speicherbereich nicht belegt ist. Im anderen Fall erfolgen Fehlermeldungen.
Die sofortige Durchfuehrung einer Prozedur wird durch die Begrenzung der Kommandoeingabe mit ',' verhindert. Auf diese Art koennen mehrere Prozeduren geladen und an einer beliebigen Stelle gestartet werden. Es koennen beispielsweise Anwenderprogramm und Debugger gleichzeitig geladen werden, so dass ein Test des Anwenderprogramms durch Setzen von Unterbrechungspunkten u.ae. moeglich wird.
Ist das Kommando geladen, beginnt das Operationssystem einen Stack-Bereich zu lokalisieren. Die Groesse dieses Bereiches ist im Deskriptor der Prozedur enthalten und wurde mit LINK oder IMAGE zugewiesen. Zur Festlegung des Stack-Bereiches fuer diese Prozedur wird zuerst der Speicherbereich oberhalb des geladenen Programms und anschliessend zwischen der Adresse 0 und der unteren Adresse des Programms untersucht. Ist in beiden Bereichen kein ausreichender Speicherplatz feststellbar, wird das Kommando nicht gestartet.
Der vom geladenen Kommando belegte Speicherbereich wird vom Operationssystem reserviert und geschuetzt. Beim Ruecksprung zum OS wird der Bereich wieder freigegeben. Wurde die Prozedur nur geladen (durch Eingabe von ','), bleibt der Bereich geschuetzt, bis er mit DEALLOCATE oder RELEASE (s. Abschn. 4.13., Abschn. 4.32.) freigegeben wird.
Tritt waehrend der Abarbeitung eines Kommandos ein Fehler auf, generiert das Operationssystem eine Fehlermeldung zum Terminal. Wenn vorhanden, wird anschliessend das naechste Kommando bearbeitet.
Jedes vom Terminal startbare Systemkommando oder Anwenderprogramm kann durch ein Programm aufgerufen werden. Dazu muss UDOs gemeinsam mit der notwendigen Kommandokette aktiviert werden. In dieser Weise erfolgt auch die Verkettung mehrerer Kommandos untereinander als auch die Implementation komplexer Overlay-Strukturen

# 4. UDOS-Kommandos

## 4.1. Uebersicht

Die folgenden beschriebenen Prozeduren stehen dem Anwender als UDOS-Kommandos zur Verfuegung. Es handelt sich hierbei um 11 interne und 30 externe Kommandos. Jeder der folgenden Abschnitte gehoert zu einem Kommando. Fuer alle Abschnitte sind diese Einteilung und Festlegungen gueltig:

- -Name des Kommandos, Kennzeichnung ob internes Kommando
- -Darstellung der Syntax
  Dabei gilt:
  .die gross geschriebenen Buchstaben sind fuer die Eingabe unbedingt notwendig
  .optionale Teile einer Parameterliste sind in eckige Klammern eingeschlossen, sie koennen mehrfach wiederholt oder weggelassen werden
  .das Zeichen '|' gibt an, dass einer und nur einer der so angegebenen Werte oder Parameter verwendet werden kann
  .interne Kommandos koennen mit Kleinbuchstaben angegeben werden
- -Beschreibung des Kommandos, d.h. allgemeine Erklaerung und Angabe seiner Option
- -Angabe eines Beispiels, es gilt, das Speicheradressen und einer Speicheradresse zugewiesene Konstante hexadezimale Zahlen sind
- -Ein/Ausgabe-Benutzung (E/A-Benutzung), es werden die logischen Funktionen angegeben, die das Kommando einsetzt

## 4.2. ACTIVATE

ACTIVATE $Treibername [Adresse]

Die angegebene Prozedur wird in die Liste der aktiven Geraete (ADT, active device table) eingetragen. Der Treibername kann anschliessend als zusaetzliche Spezifikation in Dateinamen verwendet werden.
Ist die optionale Adresse nicht angegeben, wird die Datei geladen. Voraussetzung ist, dass es sich um eine Datei mit den festgelegten Eigenschaften eines Treibers (Type=Procedure, Subtype=1) handelt und das Programm keine geschuetzten Speicherbereiche belegt. Die Zuweisung des durch die Ladung belegten Speicherbereiches wird bis zur Entaktivierung des Treibers durch das Kommando DEACTIVATE (s. Abschn. 4.12.) aufrechterhalten.
Ist die optionale Adresse angegeben, wird das Treiberprogramm auf dieser Adresse gestartet, Die Groesse des belegten Speichers steht in diesem Fall nicht fest und wird auf 0 gesetzt.
Innerhalb des Kommandos erfolgt grundsaetzlich ein Initialisierungsaufruf zum Treiber, damit er fuer spaetere E/A-Aufrufe aktiviert ist.

Beispiel:
%ACTIVATE $LP
Mit diesem Aufruf wird der Druckertreiber aktiviert. Es wird ein Initialisierungs-Request ausgesandt und der Drucker initialisiert.

E/A-Benutzung:
Funktion 0: Treiberhantierung
Funktion 2: Fehlermeldungen

## 4.3. ALLOCATE (internes Kommando)

Allocate untere Adresse obere Adresse Blockgroesse

Ein Speicherblock von angegebener Laenge (in Blockgroesse, aufgerundet auf ein Vielfaches von 80H) soll im durch untere und obere Adresse festgelegten Bereich reserviert und damit fuer die Systemarbeit schreibgeschuetzt werden. Die Suche beginnt bei der unteren Adresse und der erste noch freie Block angegebener Groesse wird im Speicherbelegungsplan als belegt eingetragen.
Wird kein freier Block angegebener Groesse gefunden, erfolgt die Meldung: INSUFFICIENT MEMORY.

Beispiel:
%A 6000 6800 278
Durch dieses Kommando wird ein freier Speicherblock der Groesse 27FH im Bereich 6000H bis 6800H gesucht und im Speicherbelegungsplan als belegt eingetragen.

E/A-Benutzung:
keine

## 4.4. BRIEF (internes Kommando)

Brief

Es erfolgt die Umschaltung des Terminals auf BRIEF-Mode. Ein eingegebenes Kommando wird nicht zum Terminal zurueckgemeldet.

Beispiel:
%B
Das Kommando bewirkt die Umschaltung in den BRIEF-Mode.

E/A-Benutzung:
keine

## 4.5. CAT

CAT [String] [T=Type] [P=Properties] [D=Drive] [F=Format] [L=Listingdisposition] [DATE rel Datum] [CDATE rel Datum]

Es werden alle in den Directorys vorhandenen Dateien ausgedruckt, die die angegebenen Bedingungen erfuellen. Sind keine Angaben gemacht, werden die nicht geheimen Dateien aus jedem Directory und jedem aktiven Laufwerk ausgedruckt. Werden gleiche optionale Angaben mehrfach benutzt, ist immer die jeweils letzte gueltig.

String:
Vollstaendig oder teilweise angegebene Dateinamen oder durch Leerzeichen voneinander getrennte Dateinamen. In nur teilweise angegebenen Dateinammen kann fuer beliebige Zeichenketten das Zeichen '*' an beliebiger Stelle eingesetzt werden. So bedeutet die Angabe TE*, dass alle mit TE beginnenden Dateinamen auszudrucken sind.

Type:
Bei Angabe von T=Type werden nur Dateien des angegebenen Types ausgegeben. Fuer Type sind A (ASCII), P (Procedure), B (Binary) und D (Directory) zulaessig.

Properties:
Es werden nur die Dateien mit den angegebenen Eigenschaften ausgedruckt. Dabei gilt:

W schreibgeschuetzt (write protected)
E loeschgeschuetzt (erase protected)
L unveraenderbare Eigenschaften (properties locked)
R wahlfrei (random)
S geheim (secret)
F auf jeden Fall ladbar, auch in geschuetzten Speicherbereich (force memory allocation)
& alle Dateien

Drive:
Entsprechend des angegebenne Wertes werden nur Dateien dieses Laufwerks ausgegeben. Fuer Drive kann einer der Werte zwischen 0 und 3 stehen. Ohne Angabe wird ueber alle Laufwerke gesucht.

Format:
Die Angabe F=L (langes Format) bewirkt die Ausgabe der Dateinamen und aller ihrer Eigenschaften. Ohne Angabe wird immer im kurzen Format ausgegeben, welches nur die Angabe des Dateinamens und der Laufwerksnummer beinhaltet. Folgende Dateieigenschaften werden im langen Format ausgegeben:
Name, Laufwerksnummer, Dateitype, Satzanzahl, Satzlaenge, Dateieigenschaften, Startadresse, Datum der Erstellung, Datum der letzten Aenderung.
Ausserdem wird angegeben:
Anzahl der untersuchten Dateien; Anzahl der ausgegebenen Dateien; Anzahl der Sektoren, die von den ausgegebenen Dateien belegt sind.

Listingdisposition:
Hiermit wird das Geraet angegeben, auf das die Ausgabe erfolgen soll. Ueblicherweise erfolgt die Ausgabe auf das Terminal, kann aber zu jedem Treiber oder jeder Datei erfolgen. Die Angabe L=1/CATLIST1 liefert die Ausgabe zur

Datei CATLIST1 auf Laufwerk 1 und Master-Treiber

DATE | CDATE rel Datum:
Mit dieser Angabe werden nur Dateien ausgegeben, die die Bedingung zwischen DATE oder CDATE und dem angegebenen Datum erfuellen- CDATE bedeutet Vergleich mit DATE OF CREATION, DATE vergleicht mit DATE OF LAST MODIFICATION. Sind Datumziffern nicht vorhanden , werden sie rechts mit * aufgefuellt. Fuer rel kann eine der Relationen =, >, <, >=, <= oder <> eingetragen werden. Das anzugebende Datum besteht aus maximal 6 Ziffern oder *.

Beispiel:
%CAT F=L P=&
Das Kommando bewirkt die Ausgabe aller Dateien aller Laufwerke im langen Format.
%CAT *.S CDATE < 860101
Dieses Kommando gibt alle Dateien aus, die mit .S enden und deren Erstellungsdatum vor dem 1.Januar 1986 liegt.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
Funktion 3: automatische Zuweisung zur Ausgabe
Funktion 4: Directorys
Funktion 5: Dateien, die im Directory angegeben sind
Funktion 6: nicht automatische Zuweisung zur Ausgabe

## 4.6. CLOSE (internes Kommando)

Close Unit | *

Fuer die mit Unit als Hexadezimalzahl angegebene logische Einheit wird ein CLOSE-E/A-Aufruf erzeugt. Steht an Stelle von Unit ein * wird der CLOSE-E/A-Aufruf fuer saemtliche logischen Einheiten erzeugt. Fehlermeldungen werden vom Kommando ignoriert.

Beispiel:
%C 4
Es wird ein CLOSE-E/A-Aufruf fuer Funktion 4 erzeugt.
%C *
Es wird ein CLOSE-E/A-Aufruf fuer saemtliche Funktionen erzeugt.

## 4.7. COMPARE

COMPARE datei1 datei2

Die Inhalte der angegebenen Dateien werden verglichen. Ein Vergleich der Deskriptoren beider Dateien wird nicht durchgefuehrt. Bei Gleichheit der Dateien erfolgt keine Meldung.

Bei jedem Bytefehler wird folgende Fehlermeldung ausgegeben (Beispiel):

```
File 1: Byte 0284 Record 0005= C8
File 2: Byte 0284 Record 0005= A7
```

Die Eingabe von 'ESC' beendet das Kommando vorzeitig.

Beispiel:
%COMPARE BLUME1 BLUME2
%
Beide Dateien werden gelesen und verglichen. Da keine Rueckmeldung erfolgt, sind die Dateien identisch.
%COMPARE WALD1 WALD2
%I/O ERROR C9 ON UNIT 6
Die Datei WALD1 ist kuerzer als die Datei WALD2. Bis zum Ende von WALD1 sind beide Dateien gleich.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
Funktion 6: datei1 Eingabe
Funktion 7: datei2 Eingabe

## 4.8. COPY

COPY datei1 datei2 ([RL=Rekordlaenge] [T=Type] [A | U | O])

Die datei1 wird mit einem BINARY-READ-Request gelesen, der Inhalt wird als datei2 mit einem BINARY-WRITE-Request auf Diskette abgelegt. Dabei koennen datei1 und datei2 Treibernamen oder voll qualifizierte Dateinamen sein.
Mit Ausnahme des Erstellungsdatums werden alle Dateiattribute von datei1 in datei2 uebernommen, wobei die optional fuer die Rekordlaenge und den Dateityp einzugebenden Werte die aus datei1 in datei2 uebernommenen Werte ueberschreiben. Dabei sind fuer die Rekordlaenge 80H, 100H, 200H, 400H, 800H, 1000H und fuer den Dateityp D (directory), B (binary), A (ascii) und P (procedure) angebbar. Die Optionen A (append), U (update) und O (output) spezifizieren den Open-Aufruf fuer datei2 (s. Abschn. 5.4.4.).

Beispiel:
%COPY 1/ADATEN 2/BDATEN (O RL=200)
Die Datei ADATEN von der Diskette in Laufwerk 1 wird als Datei BDATEN mit der Rekordlaenge von 200H auf die Diskette in Laufwerk 2 abgelegt. Falls die Datei BDATEN bereits existierte, wird sie ueberschrieben.
%COPY TESTDRUCK $LP
Die Datei TESTDRUCK wird automatisch auf den Disketten in allen aktiven Laufwerken gesucht und an den Druckertreiber LP ausgegeben.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
Funktion 6: Quelldatei
Funktion 7: Zieldatei

## 4.9. COPY.DISK

Das Programm kopiert die Diskette, die sich in dem als Quelle angegebenen Laufwerk befindet auf die Diskette, die sich in dem als Ziel angegebenen Laufwerk befindet. Die Abfrage von Quellen- (source) und Ziellaufwerk (destination) erfolgt dialoggesteuert. Das Kopieren wird nur auf physisch gleiche Disketten vorgenommen. Das physische Format wird aus der Zelle FDCONF ermittelt. Nach der Eingabe von Quelle und Ziel erscheint die Ausschrift:

DRIVES READY?

Alle Eingaben ausser 'Y' (YES) fuehren zum Abbruch des Kommandos. Ein Vergleich der Disketten wird nicht durchgefuehrt, aber ein Kontrollesen der beschriebenen Diskette. Sind ungleiche physische Diskettenformate vorhanden, wird die Fehlermeldung 'NOT THE SAME FORMAT, USE SETFD' ausgegeben und das Kommando abgebrochen.

Beispiel:
%COPY.DISK
SOURCE:1
DESTINATION:2
DRIVES READY??(Y/N):Y
Der Inhalt der Diskette 1 wird auf die Diskette 2 kopiert.

E/A-Benutzung:
Funktion 1: Terminal-Interaktion
Funktion 2: Terminal-Interaktion

## 4.10. COPYSD

COPYSD dateiname

Es wird eine UDOS-Datei auf einem Diskettenlaufwerk von einer Diskette auf eine andere Diskette kopiert. Die Quellen und die Zieldiskette muessen so viele Male eingelegt werden, bis die Datei gaenzlich uebertragen ist. fuer den Anwender erfolgt auf dem Terminal eine der beiden folgenden Meldungen:

INSERT SOURCE DISK.
TYPE ANY KEY TO CONTINUE, ESCAPE TO ABORT:
INSERT DESTINATION DISK
TYPE ANY KEY TO CONTINUE, ESCAPE TO ABORT:

Bis zur Fertigstellung der Kopie sollte der entsprechende Aufruf befolgt werden. Mit dem ESC-Zeichen kann abgebrochen werden, ansonsten bewirkt jedes andere Zeichen die weitere Uebertragung. Auf der Zieldiskette erhaelt die Datei den gleichen Namen. Ihre Eigenschaften sind mit denen der Quelldatei identisch.

Beispiel:
%COPYSD DATEN.E
die Datei DATEN.E wird mit allen ihren Eigenschaften von der Quellen. zur Zieldiskette kopiert.

E/A-Benutzung
Funktion 1: Terminal
Funktion 2: Terminal
Funktion 6: NDOS

## 4.11. DATE

DATE [JJMMTT]

Das Programm liest das in der Datei DAY befindliche Datum, traegt es in das Datumsfeld ein und gibt es auf dem Terminal aus. Mit der optionalen Angabe eines Datums wird dieses Datum in die Datei DAY und in das Datumsfeld eingetragen. Das Datumsfeld dient zur Information ueber "Datum der Erstellung" oder "Datum der letzten Aenderung". Fuer die Zahlen JJMMTT sind Jahr, Monat und Tag anzugeben.
Das Kommando DATE ist Teil der externen UDOS-Initialisierungs-Kommandodatei OS.INIT. Fuer eine ordnungsgemaesse Funktion von DATE darf die Systemdiskette nicht schreibgeschuetzt sein, da das angegebene Datum in die Datei DAY eingetragen wird.

Beispiel:
%DATE
Thursday, January 1, 1987
Zeigt das in der Datei DAY befindliche Systemdatum und traegt es in das Datumsfeld ein.

E/A-Benutzung:
Funktion 2: Ausgabe

## 4.12. DEACTIVATE

DEACTIVATE $Treibername

Aus der Liste ADT (active device table) wird der Treibername geloescht und fuer UDOS unauffindbar gemacht. Fuer alle logischen Funktionen, die zum deaktivierten Treiber gehoerten, wird ein CLOSE-E/A-Aufruf und fuer den Treiber selbst wird ein DEACTIVATE-E/A-Aufruf durchgefuehrt. Der vom Treiber belegte Speicherplatz wird freigegeben.

Beispiel:
%DEACTIVATE $TR2
Der Treiber TR2 wird aus der ADT gestrichen. Fuer alle logischen Funktionen, die TR2 zugeordnet sind, wird ein CLOSE-Request erzeugt. Ausserdem wird zu TR2 ein DEACTIVATE-Request gesandt und der fuer TR2 belegte Speicherbereich wird freigegeben.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
Ausserdem alle mit dem Treiber verbundenen funktionen.

## 4.13. DEALLOCATE (internes Kommando)

DEAllocate Blockadresse Blocklaenge

Der Speicherblock angegebener Laenge wird ab in Blockadresse eingetragenem Wert als frei im Speicherbelegungsplan eingetragen. War der betroffene Speicherbereich vorher nicht belegt, erfolgt die Meldung:

MEMORY PROTECTION

Es ist zu beachten, dass die Blockadresse ein Vielfaches von 80H ist (im anderen Fall wird abgerundet), die Blockgroesse wird auf ein Vielfaches von 80H aufgerundet.

Beispiel:
%DEA 8145 23A7
Ab Adresse 8100H wird ein 2400H Byte Block freigegeben.

E/A-Benutzung:
keine

## 4.14. DEBUG (internes Kommando)

Debug

Mit diesem Kommando wird der U880-Softwaremonitor aufgerufen (siehe P8000 Dokumentationsband Einfuehrung in die Software des Geraetesystems P8000, Beschreibung des U880-Softwaremonitors). Das Kommando 'Q' des U880-Softwaremonitors realisiert den direkten Ruecksprung ins Operationssystem OS von UDOS, waehrend durch das Kommando 'O' ein Neuladen des OS und eine Neuinitialisierung des Systems vorgenommen wird.

Beispiel:
%D
>B 7800
>Q
%
Mit der Eingabe von 'D' erfolgte der Uebergang in den U880-Softwaremonitor. Nach dem Setzen des Breakpoints wird durch das Kommando 'Q' in das OS zurueckgegangen.

E/A-Benutzung: keine

## 4.15. DEFINE

DEFINE log. Funktion Dateiname | log. Funktion Treibername | log. Funktion * | * [A] | [O] | [U] | [I] | [NF] | [NO]

Eine logische Funktion (dargestellt durch eine Zahl zwischen 0 und 20) wird mit einem momentan aktiven Treiber verknuepft oder die Funktion wird entsprechend des Originalwertes wiederhergestellt (bei Systeminitialisierung). Wenn die Funktion bereits definiert war, wird ein CLOSE-Aufruf erzeugt. Es kann der Funktion optional ein Dateiname zugeordnet werden, fuer die auch ASSIGN- und OPEN-Aufrufe geniert werden koennen.

log. Funktion Dateiname
Hiermit wird die Funktion dem Master-Treiber oder, wen der Dateiname qualifiziert ist, einem bestimmten Treiber zugeordnet. Dem Treiber wird ein ASSIGN-Aufruf gesandt. Der Dateiname uebernimmt dabei die Rolle eines Parameters, gefolgt von einem OPEN-Aufruf (automatische Zuordnung ist OPEN FOR UPDATE).

log. Funktion Treibername
Die Funktion wird mit dem Treibernamen verknuepft. Dazu muss der Treiber aktiviert sein. Es werden keine anderen E/A-Aufrufe erzeugt.

log. Funktion *
Die Funktion wird wieder mit dem automatisch bei der Initialisierung zugeordneten Wert verknuepft.

*
Alle Funktionen werden wieder zu den Originaltreibern verknuepft. Die Terminalfunktion wird fuer 1,2 und 3 definiert, alle anderen wird der Master-Treiber zugeordnet.
Die Optional anzugebenden Werte A, C, U, I, NF, NO bedeuten, (s. Abschn. 5.4.4. fuer NDOS-Open-Funktion):

```
A  OPEN TYPE = APPEND
O  OPEN TYPE = OUTPUT
U  OPEN TYPE = UPDATE
I  OPEN TYPE = INPUT
NF OPEN TYPE = NEW FILE
NO kein OPEN-Aufruf
```

Beispiel:
$DEFINE SYSLST $LP
Der Treiber LP, der vorher definiert sein muss, wird als Druckerausgabefunktion (3) definiert. Die nachfolgenden E/A-Aufrufe fuer die logische Funktion 3 werden zu LOP gegeben.
%DEFINE 15 NEU1
Die Funktion 15 wird zum Master-Treiber verknuepft. ASSIGN - und OPEN-Aufrufe werden zur Funktion 15 gesandt, mit NEU1 als Parameter.
%DEFINE 7 $NEUDOS/NEU2 NO

Die Funktion 7 wird zum bereits aktivierten Treiber NEUDOS verknuepft. Es wird ein ASSIGN-Aufruf (aber kein OPEN-Aufruf) mit NEU2 als Parameter generiert.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
alle anderen: parameterabhaengig

## 4.16. DELETE

DELETE [String] [T=Type] [P=Properties] [D=Drive]
[Q=Frage] [DATE rel Datum] [CDATE rel Datum]

Das Kommando loescht die durch String vollstaendig oder teilweise anzugebenden Dateien auf den Disketten aller aktiven Laufwerke, fuer die die angegebenen Optionen zutreffen. Von diesen Dateien werden alle Datensaetze freigegeben und die Namen aus dem Directory entfernt.
Optionen koennen in beliebiger Reihenfolge und Anzahl angegeben werden. Ausser fuer String gilt fuer alle Optionen bei Mehrfacheingabe immer die letzte. Ohne Angabe von Optionen werden alle sichtbaren (non-secret) Dateien aller aktiven Laufwerke geloescht. Fuer QUERY gilt implizit YES, d.h., es wird folgende Meldung auf das Terminal ausgegeben:

DELETE Laufwerk/Dateiname (Y/N/A/Q)?

Als Eingabe wird ein Zeichen akzeptiert, dabei gilt:

- Y Ja, die Datei wird geloescht.
- N Nein, die Datei wird nicht geloescht.
- A Ja, die Datei wird geloescht, ebenso alle weiteren.
- Q Damit wird das Kommando abgebrochen, Rueckkehr in das Betriebssystem.

String
String sind voll oder teilweise angegebene Dateinamen, wodurch nur zutreffende Dateien aus den Directorys zum Loeschen angeboten werden. Zur teilweisen Angabe des Dateinamens wird das Zeichen '*' verwendet (siehe auch CAT-Kommando). Fuer String duerfen keine qualifizierten Dateinamen verwendet werden.

Q=Frage
Es wird der Abfragemodus fuer die Loeschoperationen gesetzt. Q=Y (implizit eingestellt) bewirkt die Anfrage vor jeder Loeschung. Mit Q=N wird die Anfrage unterdrueckt.
Achtung: Q=Y kann durch die Antwort A unterdrueckt werden.

Die Beschreibung der anderen Optionen siehe CAT-Kommando (Abschn. 4.5.).

Beispiele:
%DELETE T=P
Nach Abfrage werden alle Prozedurdateien geloescht.

%DELETE *.L Q=N P=&
Alle Dateien aktiver Laufwerke, die mit .L enden, werden ohne Anfrage geloescht.
%DELETE D=2 P=R ST*
Nach Anfrage werden alle Dateien auf Laufwerk 2 mit wahlfreiem Zugriff geloescht, die mit ST beginnen.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
Funktion 4: Directorys
Funktion 5: Dateien, in Directorys eingetragen

## 4.17. DISPLAY

DISPLAY

Der momentane Speicherbelegungsplan wird an das Terminal ausgegeben. Dabei gilt:

A - 80H Byte sind belegt
. - 80H Byte sind verfuegbar

E/A-Benutzung:
Funktion 2: Speicherausgabe

## 4.18. DO

DO Kommandodatei [Parameterliste]

Durch das Kommando wird eine Datei gestartet, die aus UDOS-Kommandos besteht. Diese Datei wird in einen dynamisch zugewiesenen Pufferspeicher geladen. Jede Kommandozeile (begrenzt mit Return) kann in Abhaengigkeit von eventuell vorhandenen Erweiterungssymbolen optional erweitert werden.

Parametererweiterung
Eine Parameterzeichenkette ist eine Gruppe von Symbolen, welche durch folgende Zeichen begrenzt werden kann: Space (20H), Komma (2CH), horizontaler Tabulator (09H), runde Klammer auf (28H), runde Klammer zu (29H), Semikolon (3BH), Return (0DH).
Bei jedem Auftreten der Zeichenkette #n erfolgt einfache Parametersubstitution. Dabei ist n eine ganze Zahl, kleiner oder gleich der Anzahl der aufgetretenen Parameter.
Ist n groesser als die Zahl der vorhandenen Parameter, entsteht COMMAND EXPANSION ERROR und der Prozess wird abgebrochen. Fuer jedes #n wird die n-te Parameterzeichenkette aus der Parameterliste substituiert. Es sind maximal 64 Parameter substituierbar, nicht angesprochene Parameter werden ignoriert.

Bedingte Erweiterung
Durch '[' und ']' erfolgt die bedingte Erweiterung einer Kommandozeile. Die Tiefe der bedingten Erweiterung vergroessert sich bei jedem Auftreten von '[' um eins. Der Kommandokettenteil wird bis zum entsprechenden Zeichen ']'

ignoriert, wenn die resultierende Erweiterungstiefe groesser als die Parameterzahl ist. Ist die resultierende Erweiterungstiefe kleiner als die Parameterzahl, wird das Zeichen '[' geloescht und die Erweiterung geht weiter. Jedes Auftreten von ']' erniedrigt die Erweiterungstiefe um Eins. So erweitert die Kommandokette STU[V]W zu STUVW, wenn genau ein Parameter angegeben ist. in jedem anderen Fall entsteht STUW.
Das Kommando DO kann sich selbst aufrufen (wenn die Properties von DO F enthalten), die Tiefe der Schachtelung ist nur vom verfuegbaren Speicherbereich begrenzt. So kann beispielsweise die Kommandodatei DAT.A das Kommando DO DAT.B und dieses seinerseits das Kommando DO DAT.C aufrufen.

Beispiele:
Die Kommandodatei LPRINT hat folgenden Inhalt:
```
ACTIVATE $LP;COPY #1 $LP;DEACTIVATE $LP
```
```
%DO LPRINT LISTE1
```
Folgende Kommandos werden ausgefuehrt:
```
ACTIVATE $LP
COPY LISTE1 $LP
DEACTIVATE $LP
```
Die Kommandodatei LPRINT hat folgenden Inhalt:
```
ACTIVATE $LP
[COPY #1 $LP[;COPY #2 $LP[;COPY #3 $LP]]]
DEACTIVATE $LP
```
```
%DO LPRINT LISTE1
```
Folgende Kommandos werden ausgefuehrt:
```
ACTIVATE $LP
COPY LISTE1 $LP
DEACTIVATE $LP
```
Da nur ein Parameter vorhanden ist, erfolgt die Parametersubstitution nur einmal und die gesamte Kommandozeile wird nach und einschliesslich des zweiten '[' ignoriert.
Die Kommandodatei MAKEP hat folgenden Inhalt:
```
EDIT #1.S[;ASM #1[;LINK $=9000 #1[:#1]]]
```
```
%DO MAKEP DATEI! A L X
```
Folgende Kommandos werden ausgefuehrt:
```
EDIT DATEI1.S; ASM DATEI1;LINK $=9000 DATEI1;DATEI1
```
Es sind vier Parameter angegeben, daher Erweiterung in vier Ebenen und damit Ausfuehrung aller vier Kommandos. Die Zeichen A, L und X sind nur symbolische Bezeichnungen, sie koennen jede andere beliebige Zeichenkette sein.
Folgendes Kommando wird eingegeben:
```
%DO MAKEP DATEI2 A
```
Folgende Kommandos werden ausgefuehrt:
```
EDIT DATEI2.S; ASM DATEI2
```
Dieses Kommando enthaelt zwei Parameter und damit wird die Erweiterung nur auf zwei Ebenen durchgefuehrt.

E/A-Benutzung:
Funktion 0: Kommandodatei
Funktion 2: Fehlermeldungen

## 4.19. DUMP

DUMP Dateiname [m] [n]

Bon der angegebenen Datei wird ein Speicherbelegungsbild an das Terminal ausgegeben. Jedes Byte der Datei wird zusammen mit den Adressen im hexadezimalen Kode angezeigt. Die Ausgabe druckbarer Bytes erfolgt zusaetzlich als ASCII-Zeichen, nicht druckbare Bytes sind als '.' dargestellt.
Die optionalen Angaben m und n geben den ersten und den letzten auszugebenden Datensatz an. Ohne optionale Angabe werden alle Datensaetze ausgegeben. Die Zahlenangabe muss dezimal erfolgen.
Die Ausgabe kann durch die Eingabe von CTRL-S und CTRL-Q angehalten und fortgesetzt werden. Durch die Eingabe von ESC wird die Ausgabe abgebrochen.

Beispiel:
%DUMP 1/TENNIS
Der Inhalt der Datei TENNIS auf Laufwerk 1 wird hexadezimal und als ASCII-Zeichen mit Angabe der Speicheradressen auf das Terminal ausgegeben.

E/A-Benutzung:
Funktion 2: Ausgabe
Funktion 4: auszugebende Datei

## 4.20. ECHO

ECHO Zeichenkette

Die angegebene Zeichenkette wird bis zum Begrenzungszeichen zum Terminal kopiert. Auf diese Art koennen Meldungen aus einer Kommandozeile zum Terminal gesandt werden.

Beispiel:
%COPY,;ECHO INSERT DISKETTES;I;X * #1 #2
Diese Kommandodatei kopiert Dateien von einer Diskette auf eine andere, ohne dass auf diesen Disketten das Kommando COPY enthalten sein muss.

E/A-Benutzung:
Funktion 2: Zeichenkettenausgabe

## 4.21. ERROR

ERROR Fehlerkode

Bei Eingabe des entsprechenden von UDOS oder vom Treiber zurueckgegebenen Fehlerkodes wird die Fehlermeldung in lesbarem Text auf das Terminal ausgegeben. Ist kein Fehlerkode angegeben, erfolgt der Ausdruck dieser Beschreibung.

Beispiel:
%ERROR D3
D3: DISK IS FULL

E/A-Benutzung:
Funktion 2: Ausgabe

## 4.22. ERRORS

ERRORS

Es werden saemtliche seit dem letzten Urladen aufgetretenen Diskettenfehler ausgegeben. die Ausgabe wird folgendermassen vorgenommen:

```
THE FOLLOWING RECOVERABLE ERRORS HAVE OCCURED
SINCE SYSTEM RESTART

     0000   SEEK ERRORS
     0000   SECTOR ADRESS ERRORS
     0000   DATA TRANSFER ERRORS
```

Die Zaehlung erfolgt hexadezimal. Die Fehlerbeschreibung enthaelt die Beschreibung des U880-Softwaremonitors im P8000 Dokumentationsband: Einfuehrung in die Software des Geraetesystems P8000.

E/A-Benutzung:
Funktion 2: Ausgabe

## 4.23. EXTRACT

EXTRACT Dateiname

Fuer die Datei mit dem angegebenen Dateinamen werden folgende Informationen ausgegeben:

- Record Count (Satzanzahl)
- Record Length (Satzlaenge)
- Bytes in last Record (Anzahl der Bytes im letzten Satz)

Fuer Dateien vom Typ P (procedure) wird zusaetzlich angegeben:

- Entry (Startadresse)
- Low Adresss (niedrigste benutzte Speicheradresse)
- High Adress (hoechste benutzte Speicheradresse)
- Segments (Adressen der Speichersegmente, die von der Datei belegt werden)
- Stack Size (Groesse des fuer das Programm zur Verfuegung gestellten Stack-Bereiches)

Beispiel:
%EXTRACT CAT
RECORD COUNT=0004 RECORD LENGTH=0400 BYTES IN LAST RECORD=0400
ENTRY=4000 LOW ADDRESS=4000 HIGH ADDRESS=521D STACK SIZE=0080
SEGMENTS:
4000 4FFF

E/A-Benutzung:
Funktion 0: Eingabe Prozedur
Funktion 2: Ausgabe, Fehlermeldungen

## 4.24. FORCE (internes Kommando)

Force Kommandokette

Es werden unabhaengig von der Belegung des Speichers alle Kommandodateien der Kommandokette geladen, obwohl Prozeduren normalerweise nur in als nicht belegt gekennzeichnete Speicherbereiche geladen werden. So wird FORCE beispielsweise zum Laden von rekursiven Programmen oder zum Laden von Overlay-Strukturen verwendet. ausser der Arbeit mit FORCE kann jeder Datei die Eigenschaft F (FORCE MEMORY ALLOCATION) zugeteilt werden.

Beispiel:
%F DELETE
Die Prozedur DELETE wird unabhaengig von der Speicherbelegung geladen.
%F DATEIK,; DATEIR
Die Prozedur DATEIK wird unabhaengig von der Speicherbelegung geladen, aber nicht ausgefuehrt. Die Prozedur DATEIR wird nur geladen und ausgefuehrt, wenn der benoetigte Speicherbereich zur Verfuegung steht.

E/A-Benutzung:
keine

## 4.25. FORMAT

FORMAT [S] [D=Drive] [ID=Diskettenname] [Q=Frage]

Die Diskette in dem mit Drive angegebenen Laufwerk wird entsprechend der Zelle FDCONF formatiert (s. Abschn. 4.37., Kommando SETFD). Folgende Diskettenformate sind einstellbar:

- 40 Spuren mit 16 Sektoren je 256 Bytes (einseitig)
- 80 Spuren mit 16 Sektoren je 256 Bytes (einseitig)
- 80 Spuren mit 32 Sektoren je 256 Bytes (doppelseitig)

Nach dem physischen Formatieren werden der Diskettenbelegungsplan (disk allocation map) und die Diskettennutzungsstatistik initialisiert. Weiterhin wird ein leeres Directo-

ry, welches nur die Angabe des Directorys selbst enthaelt, angelegt.
Von den zur Verfuegung stehenden Sektoren einer formatierten Diskette sind 9 Sektoren immer fest belegt, 7 Sektoren fuer das Directory mit seinen 5 Datensektoren und 2 Sektoren fuer den Diskettenbelegungsplan und die Diskettennutzungsstatistik. Gegenueber einer Anwenderdiskette sind auf einer Systemdiskette zusaetzlich 38 Sektoren belegt, davon 4 Sektoren fuer den UDOS-Urlader und 34 Sektoren fuer das U880-Softwaremonitor-GET/SAVE-Programmpaket.

S
Ist in der Kommandoeingabe die Option S angegeben, wird die Diskette als Systemdiskette formatiert. Damit ist die Diskette zusaetzlich mit dem UDOS-Urlader und dem GET/SAVE-Programmpaket belegt. Wird die Formatierung nicht in Laufwerk 0 ausgefuehrt, holt das Kommando die notwendigen Programmpakete von der Diskette in Laufwerk 0. Bei Formatierung in Laufwerk 0 selbst meldet sich das System mit der Ausschrift:

SYSTEMDISK IN DRIVE:

Danach ist die Systemdiskette in das angegebene Laufwerk einzulegen und die Taste Return zu druecken. Das Programmpaket wird in den Speicher gelesen und auf die zu formatierende Diskette geschrieben (wenn die Systemdiskette nicht im Laufwerk 0 eingelegt wurde). In diesem Fall meldet sich das System mit dem Promptzeichen. Im anderen Fall ist entsprechend Dialog die zu formatierende Diskette gegen die Systemdiskette auszutauschen. Ohne Angabe der Option S wird diese nicht zusaetzlich abgefragt.

D=Drive
Die in dem angegebenen Laufwerk befindliche Diskette wird formatiert. Fehlt die Angabe, erfolgt die Ausgabe:

DRIVE:

Als Antwort muss eine Zahl zwischen 0 und 3 angegeben werden.

ID=Diskettenname
Der Diskettenname kann maximal 24 Zeichen (ausser Return) enthalten. Der angegebene Name wird auf die Diskette geschrieben und von NDOS zur Bestimmung der Gueltigkeit des Belegungsplanes benutzt. Fehlt die optionale Angabe, erfolgt auf dem Terminal die Ausgabe:

DISK ID:

und es muss der gewuenschte Name eingetragen werden.

Q=Frage
Normalerweise erfolgt vor der Formatierung die Abfrage:

READY?

Implizit ist immer Q=Y eingetragen. Jede andere Antwort als 'Y' bricht das Kommando ab. Mit Q=N wird die Frage unterdrueckt.
Zur Beachtung:
Mit UDOS-1526 (Format von BC A5110, BC A5120, BC A5130) formatierte Disketten sind von ihrem physischen Format her nicht kompatibel zum UDOS-Format des P8000. Mit einem speziellen Treiber kann auf dem Buerocomputer eine Konvertierung beliebiger Dateien in beiden Richtungen vorgenommen werden. Auf dem P8000 ist die Bearbeitung von Disketten im Format von UDOS-1526 nicht moeglich.

Beispiel:
```
%FORMAT D=2 ID=USER1
READY?Y
```
Die in Laufwerk 2 eingelegte Diskette wird mit dem Namen USER1 als Nutzerdiskette formatiert.
```
%FORMAT S
DRIVE:1
DISK ID:UDOS.SYS
READY?A
%
```
Die in Laufwerk 1 befindliche Diskette wird nicht formatiert, da die Abfrage READY nicht mit Y beantwortet wurde.

E/A-Benutzung:
Funktion 1: Terminalbedienung
Funktion 2: Terminalbedienung

## 4.26. IMAGE

IMAGE Dateiname erster Speicherplatz letzter Speicherplatz [E=Eintrittspunkt] [RL=Satzlaenge] [ST=Stackgroesse]

Der Inhalt der angegebenen Speicherbereiche wird in die durch den Dateinamen spezifizierte Datei kopiert. Es entsteht eine Datei vom Typ P (Procedure, Subtype=0). Maximal 16 Segmente sind angebbar, wobei jedes Segment ab ganzzahliges Vielfaches der Satzlaenge abgespeichert wird. Damit entsteht die Moeglichkeit der Schaffung von Overlay-Strukturen in nicht durch IMAGE spezifizierten Speicherbereichen. Alle Angaben (ausser Dateiname) sind hexadezimal vorzunehmen. Bei fehlender Eingabe der optionalen Parameter wird fuer E=0 und fuer RL sowie fuer ST 100H eingetragen. Allgemein sind fuer RL und ST die Werte 80H, 100H, 200H, 400H und 1000H einsetzbar. Die niedrigsten und hoechsten Speicheradressen werden im Deskriptor der Datei eingetragen und beim Laden der Datei von UDOS zur Feststellung der Speicherbelegung genutzt.

Beispiel:
%IMAGE DATENAB A100 A276 B000 B157 F=100
Der Inhalt der Speicherbereiche A100H bis A276H und B000H bis B157H wird in die Datei DATENAB, bestehend aus 4 Datensaetzen, mit dem Eintrittspunkt A100H, der Rekordlaenge 100H und der Stackgroesse 100H kopiert.

E/A-Benutzung:
Funktion 0: Datei-E/A
Funktion 2: Fehlermeldungen

## 4.27. INITIALIZE (internes Kommando)

Initialize [$Treibername] [Parameterliste]

Zum Master-Treiber oder zum optional spezifizierten Treiber (welcher aktiviert sein muss) wird ein INITIALIZE-Aufruf gesandt. Die Zusatzparameteradresse des Vektors (SPV/supplemental parameter vektor address) zeigt auf das Trennzeichen nach dem Kommando oder, falls angegeben, auf das Trennzeichen nach dem Treibernamen.

Beispiel:
%I
Zum Master-Treiber, i.allg. NDOS, wird ein INITIALIZE-Aufruf gesandt.
%I $ASV 4800 KBd
Zum ASV-Treiber wird ein INITIALIZE-Aufruf mit einem Zeiger zum Zeichen Space unmittelbar vor 4800 KBd gesandt.

E/A-Benutzung:
Funktion 0: E/A-Aufruf

## 4.28. LADT

LADT

Das Kommando erzeugt eine Liste der augenblicklich aktiven Treiber, ihrer Einsprungadresse, ihrer Groesse und der mit ihnen verknuepften logischen Funktionen. MODUL ADDRESS 0000 zeigt an, dass die entsprechenden Treiber im U880-Softwaremonitor liegen.

E/A-Benutzung:
Funktion 3: Listenausgabe

## 4.29. MASTER

MASTER [$Treibername]

Ohne Angabe der Option Treibername wird der momentane Master-Treiber auf das Terminal ausgegeben. Im anderen Fall wird der angegebene Treiber zum Master-Treiber (muss vorher aktiviert sein) und damit zur automatisch zugewiesenen

Quelle fuer alle unqualifizierten Dateinamen. Damit besteht die Moeglichkeit, mit verschiedenen Dateisystemen zu arbeiten, ohne dass der Dateiname vom Anwender voll spezifiziert werden muss.

Beispiel:
```
%MASTER
NDOS IS THE MASTER
%MASTER $GRAFOS
```
Hiermit wird GRAFOS als automatisch zugewiesener Treiber spezifiziert.
```
%MASTER
GRAFOS IS THE MASTER
```

E/A-Benutzung:
Funktion 2: Fehlermeldungen

## 4.30. MOVE

MOVE [String] [T=Type] [P=Properties] [F=Format] [D=Zieltreiber] [S=Quelltenreiber] [L=Ausgabetreiber] [Q=Frage] [DATE rel Datum] [CDATE rel DATUM]

In dem Directory des Quellentreibers werden die Dateien mit String, die den angegebenen Optionen genuegen, gesucht. Alle zutreffenden Dateien werden vom Quellen- zum Zieltreiber kopiert. Ohne Angabe von Quellen- oder Zieltreiber gilt implizit als Quelle Laufwerk 0 und als Ziel Laufwerk 1.

String
Voll oder teilweise qualifizierte Dateinamen (siehe CAT-Kommando).

T=Type
Es werden nur Dateien des angegebenen Typs kopiert (siehe CAT-Kommando).

P=Properties
Es werden nur Dateien mit den angegebenen Eigenschaften verwendet (siehe CAT-Kommmando).

D=Zieltreiber
Als Zieltreiber sind jeder momentan aktive Treiber und jedes Laufwerk angebbar. Die ausgewaehlte Datei wird zu diesem Treiber kopiert.

S=Quellentreiber
Der Quellentreiber definiert die Quelle, von der die Dateien entnommen werden. Es ist jeder momentan aktive Treiber und jedes Laufwerk angebbar.

F=Format
Fuer den Ausdruck des Uebertragungsprotokolls wird langes (F=L) oder kurzes (F=S) Format spezifiziert (siehe auch CAT-Kommando). Ohne Angabe ist immer kurzes Format vereinbart.

L=Ausgabetreiber (listing disposition)
Der Ausdruck des Uebergabeprotokolls kann jedem Treiber oder jeder Datei uebergeben werden. Mit L=$CON wird die Ausgabe zum Terminal geliefert (implizit vereinbart). L=0/INHALT lenkt die Ausgabe zur Datei INHALT auf Laufwerk 0. Durch Pufferung der Ausgabe werden mehrere Zeichen jeweils als Datenblock transferiert. Anhalten oder Abbrechen siehe CAT-Kommando.

Q=Frage
Den Kriterien entsprechend wird mit Q=Y der selektive Transfer von Dateien erlaubt, ohne Angabe ist Q=N vereinbart. Bei Angabe von Q=Y erfolgt die Frage:

MOVE voll spezifizierter Dateiname TO voll spezifizierter Dateiname (Y/N/A/Q)?

Dabei gilt:

Y - Datei wird transferiert
N - Datei wird nicht transferiert, Uebergang zur naechsten Abfrage
A - Datei wird transferiert und alle betreffenden Dateien (ohne Abfrage)
Q - Datei wird nicht transferiert, das Kommando wird abgebrochen

Angaben fuer DATE rel Datum und CDATE rel Datum siehe CAT-Kommando.

Beispiele:
%MOVE DA* P=L P=& L=1/INHALT DATE>850101
Alle mit DA beginnenden Dateien von dem automatisch zugewiesenen Laufwerk 0, deren Datum der letzten Modifikation nach dem 1.1.1985 liegt, werden ohne Abfrage auf das automatisch zugewiesene Laufwerk 1 kopiert. Die Ausgabe der Dateien erfolgt im langen Format in die Datei INHALT auf Laufwerk 1.
%MOVE S=1 D=$NULL F=L P=&
Alle Dateien von Laufwerk 1 werden zum Null-Treiber (dynamisch zugewiesener Arbeitsspeicherbereich) kopiert. Durch F=L wird ein langformatiger Ausdruck auf das Terminal ausgegeben, einschliesslich der Anzahl der bewegten Dateien und der von ihnen belegten Sektoren. Die Abarbeitung eines solchen Kommandos dient auch der Ueberpruefung der verwendeten Diskette.

E/A-Benutzung:
Funktion 0: Directory
Funktion 2: automatische Zuordnung fuer Ausdruck
Funktion 4: Quellendatei
Funktion 5: Zieldatei
Funktion 6: nichtautomatische Zuweisung fuer Ausdruck

## 4.31. PAUSE

PAUSE

Das Kommando erzeugt ununterbrochene READ-STATUS-Abfragen an das Terminal bis ein Zeichen eingegeben wurde. Wurde ein Zeichen eingegeben, wird die Abarbeitung der Kommandokette fortgesetzt, bei Eingabe von ESC abgebrochen.

Beispiel:
Die Kommandodatei DUPLI hat folgenden Inhalt:

```
FORMAT D=1;MOVE,;ECHO INSERT DISKETTES;PAUSE;I;X 4000
%DO DUPLI
```

Das Kommando FORMAT wird geladen und die in Laufwerk 1 befindliche Diskette wird formatiert. Das Kommando MOVE wird geladen, ECHO wird geladen und ausgefuehrt. Das Kommando PAUSE erwartet die Eingabe eines Zeichens (bei ESC wird abgebrochen) und setzt mit der Ausfuehrung von I und MOVE fort.

E/A-Benutzung:
Funktion 1: READ-Aufruf

## 4.32. RELEASE

RELease

Jede geladene Prozedur belegt einen Speicherbereich, der vor der Ladung auf nicht belegt geprueft wurde. Nach einmaliger Abarbeitung des Programms gibt UDOS den Speicherbereich wieder frei. Wurde eine Datei nur geladen und nicht ausgefuehrt, wird die Belegung des Speicherraumes nicht automatisch aufgehoben. Das Kommando RELEASE gibt alle, seit der Durchfuehrung des letzten externen Kommandos belegten Arbeitsspeichersegmente frei, die als Ergebnis des Ladens von Prozeduren belegt wurden.

Beispiel:

```
%CAT,
%STATUS
MEMORY PRETECT VIOLATION
%REL
%STATUS

DRIVE 0   UDOS SYS
1965 SECTORS USED
 595 SECTORS AVAILABLE
```

Da das Kommando CAT nur geladen, aber nicht ausgefuehrt wurde, ist der Speicherplatz noch belegt gesetzt und das Kommando STATUS kann nicht geladen werden. durch die Eingabe von REL wird der Speicher freigegeben und das Kommando STATUS kann dann geladen und ausgefuehrt werden.

E/A-Benutzung:
keine

## 4.33. RENAME

```
RENAME   [alter Dateiname  neuer Dateiname] |
         [$FLOPPY:Drive  ID='neuer Diskettenname']
```

Der alte Dateiname wird auf der Diskette, die die alte Datei enthaelt, in den neuen Dateinamen geaendert. Bei der Aenderung eines Dateinamens beim Betrieb von UDOS im VERBOSE-Modus wird folgende Meldung ausgesandt:

```
     alte datei ---> neue datei
```

Durch die Verwendung von RENAME zusammen mit $FLOPPY:Drive und ID='neuer diskettenname' wird der Name der im angegebenen Laufwerk (Drive:0-3) befindlichen Diskette geaendert. Fuer den Diskettennamen sind maximal 24 Zeichen zulaessig (ohne ';' oder Return).

Beispiele:

```
%RENAME $OS1/PRO1 PRO2
```

Es wird ein ASSIGN- und RENAME-Aufruf fuer den Treiber OS1 erzeugt und der Name PRO1 auf PRO2 geaendert.

```
%RENAME $FLOPPY:3 ID='USER_8'
```

Die Diskette im Laufwerk 3 wird neu benannt als USER_8.

E/A-Benutzung:
Funktion 0: Datei-E/A

## 4.34. RESTORE_TABS

```
RESTORE_TABS  Dateiname
```

Die aktuelle Tabulatorbelegung des Terminals wird durch die Tabulatorspezifikation der angegebenen Datei ersetzt. Diese Datei muss durch das Kommando SAVE_TABS erzeugt worden sein.

Beispiel:

```
%RESTORE_TABS TAB.LISTE
```

Der Momentane Tabulatorstand des Terminals wird durch die Tabulatoren aus der Datei TAB.LISTE ersetzt.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
Funktion 4: Dateizugriff

## 4.35. SAVE_TABS

```
SAVE_TABS  Dateiname
```

Das Kommando speichert den derzeitigen Tabulatorenstand des Terminals in der mit Dateiname angegebenen Datei fuer den spaeteren Aufruf mittels RESTORE_TABS. Ist die angegebene Datei bereits vorhanden, wird sie geloescht und neu geschrieben.

Beispiel:
%SAVE_TABS PR.TAB
Der momentane Tabulatorenstand wird in der Datei PR.TAB auf dem Master-Treiber abgespeichert.

E/A-Benutzung:
Funktion 2: Fehlermeldungen
funktion 4: Dateizugriff

## 4.36. SET

SET [CHRDEL=c] [LINDEL=c] [NULLCT=n] [ECHO ON] | [ECHO OFF] [AUTOLF ON] | [AUTOLF OFF] [TABSIZE=n] [PROPERTIES OF Dateiname TO Props] [TYPE OF Dateiname TO Type] [SUBTYPE OF Dateiname TO Subtype] [LOW_ADDRESS OF Dateiname TO nn] [HIGH_ADDRESS OF Dateiname TO nn] [STACK_SIZE OF Dateiname TO nn] [BYTE_COUNT OF Dateiname TO nn] [ENTRY_POINT OF Dateiname TO nn]

Mit dem Kommando SET koennen die verschiedenen Systemparameter gesetzt werden. Dabei ist jede Kombination in beliebiger Reihenfolge moeglich.

CHRDEL=c (character delete)
Das Symbol zum Loeschen einzelner Zeichen wird auf c gesetzt.

LINDEL=c (line delete)
Das Zeichenloeschsymbol wird auf c gesetzt.

NULLCT=n (null count)
Damit wird die Anzahl der Nullzeichen, die automatisch nach einem Return ausgesandt werden, auf n gesetzt.

TABSIZE=n
Die Tabulatoren werden auf jede n-te Spalte gesetzt, beginnend bei der aeusserst linken als 0 (ohne Angabe wird der Tabulator auf jede 8. Spalte gesetzt).

PPROPERTIES OF Dateiname TO Props
Die Eigenschaften der angegebenen Datei, oder, wenn in runden Klammern eingeschlossen, mehrerer Dateien, werden auf Props gesetzt. Fuer Props koennen einzelne oder mehrere der nachfolgend aufgefuehrten Properties eingesetzt werden:

| | |
|---|---|
| W | (write protected) |
| E | (erase protected) |
| S | (secret) |
| L | (locked) |
| R | (random) |
| F | (force memory allocation) |
| * | (keine Eigenschaften) |

Bei blockierten Dateien (L) koennen die Eigenschaften nicht geaendert werden.

SUBTYPE OF Dateiname TO Subtype
Von der angegebenen Datei wird der Dateisubtype auf den durch Subtype angegebenen Wert gesetzt. Es werden nur die letzten 4 Bits des Wertes verwendet.

TYPE OF Dateiname TO Type
Der Typ der Datei wird auf den angegebenen Wert gesetzt. Fuer Typ ist moeglich:

D (Directory), A (Ascii), B (Binaer), P (Procedure).

LOW_ADDRESS OF Dateiname TO nn
Die untere Grenze des notwendigen Speicherraumes fuer die angegebene Datei wird auf nn gesetzt.

HIGH_ADDRESS OF Dateiname TO nn
die obere Grenze des notwendigen Speicherraumes fuer die angegebene Datei wird auf nn gesetzt.

STACK_SIZE OF Dateiname TO nn
Die Groesse des Anwender-Stacks der angegebenen Datei wird auf nn gesetzt. Bei nn=0 erfolgt keine Stack-Zuweisung, die Datei benutzt dann den System-Stackspeicher.

ECHO [ON] | [OFF]
Der ECHO-Modus wird in CON gesetzt oder geloescht (s. Abschn. 6.4.).

AUTOLP [ON] | [OFF]
Das entsprechende Flag wird in CON gesetzt oder geloescht und damit der automatische Zeilenvorschub ein- oder ausgeschaltet.

ENTRY_POINT OF Dateiname TO nn
Das Feld ENTRY_POINT im Deskriptor der Datei wird auf den Wert nn gesetzt und ist damit gleichzeitig Startadresse der Datei bei Aufruf durch UDOS.

BYTE_COUNT OF Dateiname TO nn
Die Anzahl der Bytes im letzten Satz der Datei wird auf nn gesetzt.

Beispiele:
%SET PROPERTIES OF TEST1 TO WESP
Die Properties der Datei TEST1 werden auf WRITE ÜRPTECT, ERASE PROTECT, SECRET und FORCE gesetzt. Damit kann die Datei nicht neu geschrieben und nicht geloescht werden. Ausserdem wird sie unabhaengig von der aktuellen Speicherbelegung geladen.
%SET PROPERTIES OF TEST2 TO LE
die Properties der Datei werden auf LOCK und ERASE PROTECT gesetzt. Dadurch koennen die Properties der Datei nicht geaendert werden. Da die Datei auch loeschgeschuetzt ist, kann die Datei nur durch Neuformatieren der Diskette beseitigt werden.
%SET ECHO OFF AUTOLF OFF
Die Terminalkontrolle wird auf ECHO OFF und AUTOLF OFF gesetzt.

%SET HIGH_ADDRESS OF TEST1 TO B600
Die obere Grenze des notwendigen Speicherbereiches fuer TEST1 wird auf B600H gesetzt.
%SET LINDEL=* CHRDEL^ NULLCT=4
Das Zeilenloeschsymbol wird auf '*', das Zeichenloeschsymbol wird auf '^' und die Anzahl der Nullen nach einem Wagenruecklauf (Return) wird auf 4 gesetzt (linker Rand).

E/A-Benutzung:
funktion 2: Fehlermeldungen
Funktion 4: Datei-E/A

## 4.37. SETFD

SETFD [F0F1F2F3] [Sd]

Mit diesem Kommando wird die Disketten- und Laufwerkkonfiguration fuer Laufwerk 0 bis 3 entsprechend den Angaben fuer F0....F3 eingetragen. Ohne die optionale Angabe Sd bleibt die Information nur bis zum naechsten Urladen des Systems erhalten. Bei Angabe von Sd (d ist die gewuenschte Laufwerknummer) wird die Information auf die Diskette im angegebenen Laufwerk eingetragen. Damit steht nach erneutem Urladen des Systems die eingestellte Konfiguration zur Verfuegung.

F0....F3
Dafuer kann eingetragen werden:

0 - kein Laufwerk
2 - 5 1/4"-Laufwerk, 40 Spuren mit 16 Sektoren je 256 Bytes, MFM, einseitig
3 - 5 1/4"-Laufwerk, 40 Spuren mit 16 Sektoren je 256 Bytes, MFM, einseitig (auf 80-Spur-Laufwerk)
4 - 5 1/4"-Laufwerk, 80 Spuren mit 16 Sektoren je 256 Bytes, MFM, einseitig
5 - 5 1/4"-Laufwerk, 80 Spuren mit 32 Sektoren je 256 Bytes, MFM, doppelseitig

Bei Angabe von Werten >5 und gleich 1 erscheint die Meldung:

unknown format.

Sd
Fuer d muss eine Laufwerknummer 0-3 eingetragen werden. Auf Laufwerk 0 kann die Konfiguration fuer Laufwerk 0 selbst nicht geaendert werden (Schutz der Systemdiskette).

Beispiele:
%SETFD 5320 S0
Damit ist Laufwerk 0 auf doppelseitiges 80-Spur-Laufwerk mit dem entsprechenden Diskettenformat eingestellt, Laufwerk 1 ist fuer 40-Spur-Disketten im 80-Spur-Laufwerk eingerichtet, Laufwerk 2 ist ein 40-Spur-Laufwerk und Laufwerk 3 ist nicht vorhanden.

Die Information wird auf die im Laufwerk 0 befindliche Diskette auf die Systemspuren geschrieben, fuer Laufwerk 0 selbst wird die Konfiguration nicht geaendert.

```
%SETFD
Drive 0:5 1/4",DD,DS,80 tracks,32 sectors per 256 byte
Drive 1:5 1/4",DD,SS,40 tracks,16 sectors per 256 byte
Drive 2:5 1/4",DD,SS,40 tracks,16 sectors per 256 byte
Drive 3:no drive
```

Entsprechend der eingetragenen Werte wird die Disketten- und Laufwerkkonfiguration an das Terminal ausgegeben.

## 4.38. SETLP

SETLP [lines] [cols] [indent]

Das Kommando SETLP setzt im Druckertreiber entsprechend der angegebenen Werte die Anzahl der Zeilen pro Seite, die Anzahl der Spalten pro Zeile (cols) und den linken Rand (indent). Alle Werte sind dezimal und bis maximal 255 anzugeben. Ohne Angabe von Optionen werden die momentan eingestellten Werte auf das Terminal ausgegeben.

Beispiele:

```
%SETLP 84 80 2
```

Fuer den Druck sind 84 Zeilen pro Seite, 80 Zeichen pro Zeile und ein Abstand von 2 Zeichen vom linken Rand eingestellt.

```
%SETLP
Device LP: lines=84
           cols=80
         indent=2
```

Die in LP eingestellten Werte werden an das Terminal ausgegeben.

## 4.39. STATUS

STATUS [Drive]

Fuer das spezifizierte Laufwerk (Drive:0..3) wird die Diskettenstatistik ausgegeben. die Statistik gibt die Anzahl der belegten und freien Sektoren auf der Diskette aus. Ohne Angabe von Drive erfolgt die Ausgabe fuer alle aktiven Laufwerke. Das Kommando erkennt zwei fehlerhafte Zustaende:

WARNING: DISK STATISTICS ARE INCONSISTENT

Diese Meldung wird ausgegeben, wenn die Summe der benutzten und der verfuegbaren Sektoren als Gesamtsumme nicht die Gesamtzahl der Sektoren einer Diskette ergeben.

WARNING: ALLOCATION IS INCONSISTENT

Diese Meldung wird ausgegeben, wenn die Anzahl der unbelegten Sektoren aus dem Belegungsplan der Diskette nicht mit

der Zahl der gezaehlten freien Sektoren uebereinstimmt. Dieser Fehler entsteht meistens, wenn von einer Datei benutzte Sektoren nicht als belegt gemeldet werden (z.B. durch Speicherfehler, Diskettenschreibfehler, Loeschen von Dateien mit Zeigerfehlern). Die Diskette darf in diesem Fall nicht weiter beschrieben werden und die Dateien sind auf eine andere Diskette zu uebertragen (was in diesem Fall auch meist ohne Verluste moeglich ist). Anschliessend ist die Diskette neu zu formatieren.
Weiterhin werden von STATUS folgende Fehlermeldungen generiert:

NO UDOS DISK

Bei der eingelegten Diskette handelt es sich nicht um eine UDOS-Diskette.

NO UDOS FORMAT

Der fuer das ausgewaehlte Laufwerk in FDCONF eingetragene Diskettentyp ist fuer UDOS nicht zulaessig

USE SETFD TO TAKE ON FORMAT

Der fuer das ausgewaehlte Laufwerk in FDCONF festgelegte Diskettentyp stimmt nicht mit dem auf der Diskette eingetragenen Typ ueberein.

Beispiel:
```
%STATUS 1

DRIVE 1  USER_SE
1367 SECTORS USED
1193 SECTORS AVAILABLE

WARNING: ALLOCATION IS INCONSISTENT
```
Es wird die Diskettenbelegung von Laufwerk 1 ausgegeben. Da die gezaehlten freien Sektoren nicht mit der Anzahl der unbelegten Sektoren aus dem Belegungsplan uebereinstimmen, wird die entsprechende Warnung ausgegeben.

E/A-Benutzung:
Funktion 2: Ausdruck

## 4.40. VERBOSE (internes Kommando)

Verbose

Es wird der VERBOSE-Modus aufgerufen. So wie vom Computer empfangen, wird jede Kommandokette als ECHO an das Terminal zurueckgegeben. Damit hat der Anwender eine Kontrolle der vom Computer empfangenen Eingaben. Bei manchen UDOS-Kommandos wird dieser Modus vor der Ausgabe von nicht unbedingt notwendigen Meldungen getestet. Wichtig ist es bei der Abarbeitung von Kommandodateien, da ohne VERBOSE nicht sichtbar ist, welches Kommando zur Zeit abgearbeitet wird.

Beispiel:
```
%RENAME 1/TEST1 TEST2
%V
%RENAME 1/TEST1 TEST2
RENAME 1/TEST1 TEST2
TEST1 ---> TEST2
```

E/A-Benutzung:
keine

## 4.41. XEQ (internes Kommando)

Xeq [*] | [nn] [parameterliste]

Bei Angabe von '*' und der optional anzugebenden Parameterliste wird das zuletzt geladene Kommando gestartet (unter Verwendung evtl. angegebener Parameter). Ist nn angegeben, wird der Programmstart auf der angegebenen Adresse vollzogen.

Beispiel:
```
%X * F=L P=&
```
Die zuletzt geladene Prozedur wird gestartet und unter Einbeziehung der angegebenen Parameter abgearbeitet (der Zeilenpufferspeicher zeigt auf Space=20H hinter '*').
```
%X 7400
```
Es wird auf die Adresse 7400 gesprungen.

E/A-Benutzung:
keine

## 4.42. EXPRESSION EVALUATION (internes Kommando)

: Ausdruck

Das Kommando berechnet mathematische Ausdruecke im hexadezimalen Kode von links nach rechts und gibt die Resultate aus. Erlaubte Operatoren sind +, -, *, /. Ein Ueberlauf wird nicht erkannt und Klammern sind unzulaessig.

Beispiel:
```
%: A927-2514
8413
```
Die Subtraktion von 2514H von A927H ergibt 8413H.
```
%: D4A3-3100/200
0051
```
Der Wert 3100H wird von D4A3H subtrahiert und das Ergebnis wird durch 200H geteilt. Das Ergebnis ist 51H.

# 5. Dateiverwaltungssystem NDOS

Im folgenden Abschnitt wird das Dateiverwaltungssystem NDOS des Betriebssystems UDOS beschrieben. Es werden Erlaeuterungen zur Arbeit mit Dateien, zu den NDOS-Aufrufen sowie eine Liste und Erklaerung moeglicherweise entstehender Fehler gegeben.

## 5.1. Dateimerkmale

Mit Hilfe des Dateiverwaltungssystems NDOS ist es moeglich, beliebige Daten in Form von Dateien auf der Diskette abzulegen. Der Zugriff auf die Dateien ist durch Aufruf ihres Namens gegeben. Alle Dateien einer Diskette sind in dem Directory eingetragen. Eine Ausnahme bildet die SCRATCH-Datei. Sie wird bei der Eroeffnung (enthaelt Zuweisung zu einem Dateinamen der Laenge 0) einer Datei fuer eine logische Funktion erzeugt. Die Aufnahme des Dateinamens in das Directory wird verhindert und die gesamte Deskriptorinformation bleibt intern abgespeichert, anstelle auf der Diskette. Alle verwendeten Saetze werden bei Abschluss der Datei automatisch geloescht, und die Datei existiert nicht mehr. Da bei der Arbeit mit solchen Dateien ein Directory-Zugriff (mit der dazugehoerigen Eroeffnungsprozedur) nicht notwendig ist, sind sie schnell im Zugriff und somit ideal zur Speicherung von Zwischendateien.
Das Directory selbst ist eine Datei, die auch unter diesem Namen aufgerufen werden kann, da das Directory selbst als erster Name dieser Datei eingetragen ist. Damit ist der Anfang dieser Datei, im Gegensatz zu allen anderen Dateien, festgelegt und dem System bekannt.

## 5.2. Aufbau einer Datei

### 5.2.1. Dateistruktur

Die Dateistruktur auf der Diskette ist in der Form organisiert, dass die Daten als Folge von Saetzen (Rekords) gespeichert werden. Alle Saetze einer Datei sind gleich lang, wobei die Laenge gleich der Sektorenlaenge einer Diskette (100H=256 Byte) ist. Mit entsprechender Spezifikation koennen auch Daten mit einer Rekordlaenge von 80H, 200H, 400H, 800H und 1000H erstellt werden. Bei einer Rekordlaenge von 80H werden von jedem physischen Sektor (als kleinste Einheit ist auf dem P8000 nur 100H moeglich) nur 80H benutzt. Die Arbeit mit einer Rekordlaenge von 80H ist daher uneffektiv und sollte auf Ausnahmefaelle beschraenkt bleiben.
Zu jeder Datei gehoeren ein oder mehrere Zeigersektoren, in denen die Diskettenadressen aller Sektoren einer Datei abgelegt sind. Die Diskettenadresse des ersten Zeigersektors einer Datei ist im Dateideskriptor eingetragen (s. Abschn. 5.2.2.). Die einzelnen Zeigersektoren einer Datei sind untereinander durch Rueckwaerts- und Vorwaertszeiger verkettet. Diese Zeiger stehen in den letzten Bytes des

Zeigersektors. Der Rueckwaertszeiger des ersten Zeigersektors einer Datei zeigt auf den Deskriptor dieser Datei. Der Vorwaertszeiger des letzten Zeigersektors einer Datei ist FFFFH.

### 5.2.2. Das Directory

Das Directory ist das Inhaltsverzeichnis einer Diskette und wie eine gewoehnliche Datei aufgebaut. Es besitzt den Dateityp D und die Dateieigenschaften WELS (s. abschn. 5.1.). In den Dateisektoren des Directorys (Rekordlaenge=100H) sind je verzeichnete Datei folgende Informationen abgelegt:

- ein Byte, welches in Bit 0-5 die Laenge des Programmnamens (1 bis 32 Zeichen) und in Bit 7 die Kennzeichnung der Dateieigenschaft S (secret) enthaelt
- ein bis 32 Byte, die den eigentlichen Dateinamen enthalten
- zwei Byte, die die Diskettenadresse des Deskriptors dieser Datei angeben

Nach der letzten Eintragung in einem Directory-Sektor steht die Endekennung FFH. Passt eine Dateieintragung nicht mehr in einen Directory-Sektor, erfolgt die Eintragung vollstaendig in den naechsten Datensektor des Directorys. Falls saemtliche in einem, Datensektor des Directorys enthaltene Dateiangaben geloescht wurden, enthaelt das erste Byte den Kode FFH. Sind alle von vornherein vereinbarten Datensektoren des Directorys gefuellt, wird das Directory automatisch um einen Datensektor erweitert.
Beispiel: Im ersten Datensektor des Directorys einer Anwenderdiskette sind folgende Dateien verzeichnet:

DATEI1.S, DATEI1.OBJ, DATEI1.L, DATEI1, DATEI1.MAP

```
0000* 89444952 4543544F 52590016 08444154 *.DIRECTORY...DAT*
0010* 4549312E 5303160A 44415445 49312E4F *EI1.S...DATEI1.O*
0020* 424A0818 08444154 4549312E 410A1806 *BJ...DATEI1.L...*
0030* 44415445 49310718 0A444154 4549312E *DATEI1...DATEI1.*
0040* 4D41500B 18FF0000 00000000 00000000 *map.............*
0050* 00000000 00000000 00000000 00000000 *................*
0060* 00000000 00000000 00000000 00000000 *................*
0070* 00000000 00000000 00000000 00000000 *................*
0080* 00000000 00000000 00000000 00000000 *................*
0090* 00000000 00000000 00000000 00000000 *................*
00A0* 00000000 00000000 00000000 00000000 *................*
00B0* 00000000 00000000 00000000 00000000 *................*
00C0* 00000000 00000000 00000000 00000000 *................*
00D0* 00000000 00000000 00000000 00000000 *................*
00E0* 00000000 00000000 00000000 00000000 *................*
00F0* 00000000 00000000 00000000 00000000 *................*
```

### 5.2.3. Der Deskriptorsektor

Die in der Dateiinformation des Directorys enthaltene Diskettenadresse zeigt auf einen besonderen Sektor der Datei, den Deskriptor. Dieser Sektor gehoert nicht zu den Datenrekords einer Datei und ist nicht in der Anzahl der Rekords einer Datei enthalten. Unabhaengig von der Laenge der Rekords der Datei ist er grundsaetzlich 256 (100H) Byte lang und enthaelt zusaetzliche Informationen ueber die Datei. Von den zur Verfuegung stehenden 256 Bytes definiert UDOS 42 Bytes, die restlichen 214 Bytes sind vom Anwender definierbar. Bei Dateien vom Typ P (procedure) sind nur 126 Bytes vom Anwender definierbar, da zusaetzliche Bytes fuer die Eintragung der Segmentanfangsadressen und der Segmentlaengen benoetigt werden. Im Einzelnen ist der Deskriptor folgendermassen organisiert:

| Byte | Inhalt |
|---|---|
| 00H-05H | unbenutzt |
| 06H/07H | Rueckzeiger zum Directory (Zeiger zu demjenigen Direcotry-Sektor, der den zugehoerigen Datenamen enthaelt) |
| 08H/09H | Zeiger zum ersten Datenrekord der Datei |
| 0AH/0BH | Zeiger zum letzten Datenrekord der Datei |
| 0CH | Dateityp / Subtyp |
| 0DH/0EH | Anzahl der Datenrekords der Datei |
| 0FH/10H | Rekordlaenge(80H,100H,200H,400H,800H,1000H) |
| 11H/12H | Blocklaenge (= Rekordlaenge oder unbenutzt) |
| 13H | Dateieigenschaften (PROPERTIES) |
| 14H/15H | Programmstartadresse (ENTRY POINT) - nur bei P-Dateien |
| 16H/17H | Anzahl der Bytes im letzten Record (BYTE COUNT) |
| 18H-1FH | Erstellungsdatum der Datei (DATE OF CREATION) |
| 20H-27H | Datum der letzten Aenderung der Datei (DATE OF LAST MODIFICATION) |
| 28H/29H | Anfangsadresse des 1. Segmentes - nur bei P-Dateien |
| 2AH/2BH | Laenge des 1. Segmentes - nur bei P-Dateien |
| 7AH/7BH | niedrigste vom Programm benutzte Adresse (LOW ADDRESS) - nur bei P-Dateien |
| 7CH/7DH | hoechste vom Programm benutzte Adresse (HIGH ADDRESS) - nur bei P-Dateien |
| 7EH/7FH | Stackgroesse (STACK SIZE) - nur bei P-Dateien |
| 80H/81H | Zeiger auf den ersten Zeigersektor der Datei |
| 82H-FFH | unbenutzt |

Fuer die Bytes Dateityp und Dateieigenschaften des Deskriptors gilt:

Byte 00H: Dateityp / Subtyp
Bit 0-3: Subtyp (0-15)
Bit 4: Dateityp B
Bit 5: Dateityp A
Bit 6: Dateityp D
Bit 7: Dateityp P

Byte 13H: Dateieigenschaften
Bit 0-1: vom System benutzt
Bit 2: F
Bit 3: R
Bit 4: S
Bit 5: L
Bit 6: E
Bit 7: W

Beispiel: Deskriptor des Directorys

```
0000* 00000000 00000516 05160B16 40050000 *............@...*
0010* 010001F0 00000000 33343031 3234FF00 *........840124..*
0020* 38343031 3234F000 00000000 00000000 *840124..........*
0030* 00000000 00000000 00000000 00000000 *................*
0040* 00000000 00000000 00000000 00000000 *................*
0050* 00000000 00000000 00000000 00000000 *................*
0060* 00000000 00000000 00000000 00000000 *................*
0070* 00000000 00000000 00000000 00000000 *................*
0080* 00000000 00000000 00000000 00000000 *................*
0090* 00000000 00000000 00000000 00000000 *................*
00A0* 00000000 00000000 00000000 00000000 *................*
00B0* 00000000 00000000 00000000 00000000 *................*
00C0* 00000000 00000000 00000000 00000000 *................*
00D0* 00000000 00000000 00000000 00000000 *................*
00E0* 00000000 00000000 00000000 00000000 *................*
00F0* 00000000 00000000 00000000 00000000 *................*
```

Beispiel: Deskriptor einer Prozedurdatei mit 3 Segmenten

```
0000* 00000000 00000516 001F010D 851A0000 *................*
0010* 04000410 C9480004 37383035 3234FF00 *.....E..780524..*
0020* 38343031 3236FF00 0044000C 19530010 *840126...D...S..*
0030* 5872004C 00000000 FFFFFFFF FFFFFFFF *................*
0040* FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *................*
0050* FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *................*
0060* FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF *................*
0070* FFFFFFFF FFFFFFFF FFFF0044 57BE8000 *...........DW...*
0080* 001F0000 00000000 00000000 00000000 *................*
0090* 00000000 00000000 00000000 00000000 *................*
00A0* 00000000 00000000 00000000 00000000 *................*
00B0* 00000000 00000000 00000000 00000000 *................*
00C0* 00000000 00000000 00000000 00000000 *................*
00D0* 00000000 00000000 00000000 00000000 *................*
00E0* 00000000 00000000 00000000 00000000 *................*
00F0* 00000000 00000000 00000000 00000000 *................*
```

Abgesehen von den Bytes 0-0BH ist der Deskriptorinhalt bei folgenden Aktionen verfuegbar:

- Oeffnung der Datei (siehe OPEN)
- Oeffnung der Datei bei Anfrage nach ihren Merkmalen (siehe QUERY ATTRIBUTES)
- Erstellung der Datei
- Aenderung der Merkmale (siehe SET ATTRIBUTES)
- Datei auf den neusten Stand bringen (siehe UPDATE)
- Schliessen der Datei (siehe CLOSE)

### 5.2.4. Der Zeigersektor

Die Diskettenadressen aller Datenrekorde einer Datei sind in speziell dafuer eingerichteten Sektoren, den sogenannten Datensektoren, eingetragen. Reicht ein Sektor fuer die aus jeweils zwei Byte bestehenden Adressen nicht aus, wird ein neuer Zeigersektor eroeffnet (s. Abschn. 5.2.1.). Ein Zeigersektor ist unabhaengig von der Laenge der Rekords einer Datei immer 256 Byte lang. Er gehoert ebenso wie der Deskriptor nicht zu den Datenrekords einer Datei und wird auch nicht zu der Anzahl der Datenrekords dazugezaehlt.
Fuer den Zugriff auf eine Datei muessen drei Diskettenadressen (Zeiger) initialisiert werden:

- Zeiger 1 weist auf den im Momnat behandelten Dateisatz (Zeiger zum aktuellen Datenrekord)
- Zeiger 2 weist auf den vorhergehenden Satz (Zeiger zum vorrangegangenen Datenrekord)
- Zeiger 3 weist auf den naechsten Dateisatz (Zeiger zum nachfolgenden Datenrekord)

Um eine Datei als aktiv zu bezeichnen, muessen diese Zeiger einen gueltigen Inhalt haben. Daher besteht das Oeffnen einer Datei (OPENING A FILE) aus der Lokalisierung der Datei in dem Directory, dem Lesen des zugehoerigen Zeigersektors und der Initialisierung dieser Zeiger. Im Einzelnen ist ein Zeigersektor folgendermassen aufgebaut:

| Byte | Inhalt |
|---|---|
| 00H/01H | - Zeiger zum vorhergehenden Zeigerblock oder, wenn es der erste Zeigerblock ist, zum Deskriptor der Datei |
| 02H/03H | - max. 124 Adressen von Datenrekords der Datei |
| F8H/F9H | |
| FAH/FBH | - ADRCTR - relative Adresse der letzten Eintragung einer Datenrekordadresse im Zeigerblock (zeigt auf das erste Byte der letzten Eintragung) |
| FCH/FDH | - BCKZGR - Diskettenadresse des vorhergehenden Zeigerblockes (Rueckwaertszeiger) |
| FEH/FFH | - FORZGR - Diskettenadresse des nachfolgenden Zeigerblockes (Vorwaertszeiger) |

Beispiel: Zeigerblock einer Datei mit einem Zeigerblock

```
0000* 061F001F 08200020 04200821 00210421 *.... . . .|.|.|*
0010* 08220022 04220823 00230423 08240024 *."."."."#.#.#.$.$*
0020* 04240825 00250425 08260026 04260827 *.$.%.%.%.&.&.&.'*
0030* 00270427 010D0000 00000000 00000000 *.'.'............*
0040* 00000000 00000000 00000000 00000000 *................*
0050* 00000000 00000000 00000000 00000000 *................*
0060* 00000000 00000000 00000000 00000000 *................*
0070* 00000000 00000000 00000000 00000000 *................*
0080* 00000000 00000000 00000000 00000000 *................*
0090* 00000000 00000000 00000000 00000000 *................*
00A0* 00000000 00000000 00000000 00000000 *................*
00B0* 00000000 00000000 00000000 00000000 *................*
00C0* 00000000 00000000 00000000 00000000 *................*
00D0* 00000000 00000000 00000000 00000000 *................*
00E0* 00000000 00000000 00000000 00000000 *................*
00F0* 00000000 00000000 00003400 061FFFFF *................*
```

## 5.3. Dateiwerte und Pufferorganisation

Unter NDOS existieren fuer jede eroeffnete Datei 16 Datei-werte:

| | |
|---|---|
| POIBCK | Zeiger zum vorangegangenen Datenrekord |
| POIAKT | Zeiger zum aktuellen Datenrekord |
| POIVOR | Zeiger zum nachfolgenden Datenrekord |
| DESCAD | Adresse des Dateideskriptors |
| LWADR | Laufwerksnummer |
| PTRDIR | Rueckzeiger in Directory |
| PTRFIR | Zeiger zum ersten Datenrekord |
| PTRLAS | Zeiger zum letzten Datenrekord |
| TYPFIL | Dateityp/Subtyp |
| RECCOU | Anzahl der Datenrekords |
| RECLEN | Rekordlaenge |
| BLOLEN | Blocklaenge |
| PROPER | Dateieigenschaften |
| BLKADR | Adresse des aktuellen Zeigersektors |
| BLKPOS | Zeiger auf die aktuelle Datenrekordadresse im aktuellen Zeigersektor (relative Adresse bezueglich des Anfangs des Zeigersektors) |
| FIRSTBL | Adresse des ersten Zeigersektors der Datei |

Im NDOS-Datenpufferbereich ist ein Speicherbereich von 3*256 Byte reserviert. Das DATFLD (256 Byte) ist fuer gelesene Datensektoren oder fuer den Deskriptorsektor re-serviert, das DIRFLD (256 Byte) ist fuer den gelesenen Directory-Sektor und ein Feld (256 Byte) ist fuer den aktuellen Zeigersektor reserviert. Dieses Feld hat folgen-den Aufbau:

```
ZGRBLK   DEFS  250
ADRCTR   DEFS  2
BCKZGR   DEFS  2
FORZGR   DEFS  2
```

Ausserdem sind zu diesen drei Feldern zwei Byte fuer die

Diskettenadresse des zuletzt eingelesenen Zeigersektors reserviert:

```
BLKSRC   DEFS  2
```

Vor dem erneuten Einlesen eines Zeigersektors wird dessen Adresse mit der in BLKSRC befindlichen verglichen. Besteht Uebereinstimmung, erfolgt kein Einlesen. Dadurch werden unnoetige Diskettenoperationen vermieden.

## 5.4. NDOS-Aufrufe

### 5.4.1. Allgemmeines

Die Realisierung von E/A-Aufrufen durch das Betriebssystem erfolgt mit Hilfe des vom Anwenderprogramm zur Verfuegung gestellten Parametervektors. In diesem Vektor werden die vom Betriebssystem benoetigten Angaben (Parameter) zur Verfuegung gestellt. Der Parametervektor besteht aus 13 aufeinander folgenden Bytes. Das Register IY muss auf das erste Byte zeigen. Der Parametervektor hat folgenden Aufbau:

| | Byte | Inhalt |
|---|---|---|
| IY ---> | 0 | Nummer der logischen Funktion |
| | 1 | Kodenummer der Operation |
| | 2-3 | Datentransferadresse |
| | 4-5 | Laenge des Datenblocks |
| | 6-7 | Ruecksprungadresse |
| | 8-9 | Ruecksprungadresse bei Fehler |
| | 10 | Fertigstellungs-Kode |
| | 11-12 | Startadresse des Hilfsparametervektors |

Eine detaillierte Beschreibung des Parametervektors ist in Abschn. 6.2. nachzulesen.
Die nachfolgend beschriebenen NDOS-Aufrufe sind vom Anwenderprogrammm (alle externen Systemkommandos oder vom Anwender selbst erstellte Programme) ueber den Aufbau des entsprechenden Parametervektors und den nachfolgenden Einsprung in das System ueber CALL 1003H zu erreichen.
Einige allgemeine Bemerkungen zu NDOS-Requests, die sich aus der Arbeit mit Zeigersektoren ergeben:

Leseoperation
Alle Leseoperationen erfolgen immer rekordweise. Dabei zeigt (BLKPOS) auf den aktuellen zu lesenden Datenrekord im Zeigersektor (entsprechend (POIAKT)). Nach dem Lesen zeigt (POIAKT) i.allg. auf den naechsten Datenrekord. Dementsprechend wird auch (BLKPOS) veraendert. Dies geschieht, indem (NLKPOS) um den Wert zwei erhoeht wird, ausser wenn (BLKPOS)=(ADRCTR) , d.h., wenn (BLKPOS) bereits auf die letzte Eintragung im Zeigersektor zeigt. In diesem Fall wird der naechste Zeigersektor eingelesen und (BLKPOS):=0, d.h., (BLKPOS) zeigt auf die erste Eintragung im naechsten Zeigersektor. Ist ein naechster Zeigersektor nicht vorhanden, so ist das Ende der Datei erreicht.

Schreiboperationen
Alle Schreiboperationen erfolgen immer rekordweise. Dabei zeigt (BLKPOS) auf die Stelle im Zeigersektor. wo die naechste Datenrekordadresse eingetragen werden soll. Vor dem Eintragen der Datenrekordadresse in den Zeigersektor wird geprueft, ob danach noch gueltige Datenrekordadressen im Zeigersektor stehen (d.h., wenn (ADRCTR) groesser als (BLKPOS) ist). In diesem Fall werden ab (BLKPOS) alle Datenrekordadressen um zwei Byte nach hinten verschoben und (ADRCTR) wird um zwei erhoeht. Ist der Zeigersektor bereits voll, erfolgt ein Uebertrag in den naechsten Zeigersektor und (ADRCTR) wird in diesem Sektor um zwei erhoeht. Ist kein naechster Zeigersektor vorhanden, wird dieser bereitgestellt.

Loeschoperation
Alle Loeschoperationen erfolgen immer rekordweise. Dabei zeigt (BLKPOS) auf die Datenrekordadresse im Zeigersektor, die geloescht werden soll. Stehen nach der geloeschten Datenrekordadresse noch weitere gueltige Datenrekordadressen (d.h. (ADRCTR) groesser als (BLKPOS)), so werden diese um zwei Byte nach vorn geschoben und (ADRCTR) wird um zwei verkleinert. Ist ein Zeigersektor nach dem Loeschen der Datenrekordadresse leer, so wird er geloescht. Der Vorwaertszeiger des vorhergehenden und der Rueckwaertszeiger des nachfolgenden Zeigersektors (falls vorhanden) werden korrigiert. (BLKPOS) zeigt danach auf die erste Eintragung im nachfolgenden Zeigersektor (falls vorhanden) bzw. auf die letzte Eintragung im vorhandenden Zeigersektor.

### 5.4.2. INITIALIZE

Vektor: Logische Funktion- nicht verwendet
Kode - 00 oder 01
Datenstartadresse- nicht verwendet
Datenblocklaenge - nicht verwendet, es wird 00 zurueckgemeldet
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Alle 16 logischen Funktionen werden als nicht offen markiert. die Tabelle der logischen Funktionen wird geloescht. Der gesamte Anwenderspeicher wird wieder freigegeben. die Diskettenbelegungsplaene werden eingelesen und eine Anzeige (FLAG-WORD) gesetzt, welche ueber die aktiven Laufwerke Auskunft gibt.

Moegliche Fehler:
C4, C5, C6
Der Aufruf wird bei Auftreten eines dieser Fehler nicht fertig durchgefuehrt.
DA
Nicht ausreichender Platz fuer die Belegungsplaene.

### 5.4.3. ASSIGN

Vektor: Logische Funktion
- Kode - 02 oder 03
- Datenstartadresse - nicht verwendet
- Datenblocklaenge - nicht verwendet
- Rueckkehradresse
- Fehlerrueckkehradresse
- Fertigstellungskode
- Folgevektor - Zeiger zu weiterem Vektor mit folgendem Inhalt:
  - Byte1 von NDOS nicht verwendet
  - Byte2 Kodezeichen des verwendeten Laufwerkes (0...3)
  - Byte3 Bytezahl im Dateinamen, max. 32 falls 0, dann SCRATCH-Datei
  - Byte4...Dateiname

Aktion: Der angegebene Prozedurname wird mit einer logischen Funktion fuer nachfolgende E/A-Operationen verknuepft. Vorangehende Zuweisungen werden geloescht, der Prozedurname wird in einen Pufferspeicher fuer den spaeteren Gebrauch abgelegt (bei Aufruf der logischen Funktion).
Falls die Prozedur in der Zwischenzeit einmal eroeffnet und wieder geschlossen wird, geht die Verbindung mit der logischen Funktion verloren, der Aufruf ASSIGN muss wiederholt werden.

Moegliche Fehler:

CC
Versuch der Verknuepfung mit einer aktiven logischen Funktion.

CE
Ungueltige Laufwerknummer.

CD
Dateinamenpuffer voll.

83
Dateiname war zu lang, wurde abgeschnitten.

### 5.4.4. OPEN

Vektor: Logische Funktion
- Kode - 04 oder 05
- Datenstartadresse - Zeiger zu Bereich mit Inhalt der Merkmale fuer zu erstellende Datei, bzw. mit Merkmalen fuer bereits bestehende Datei. Falls 0, werden die ueblichen Merkmale der zu erstellenden Datei beigegeben.
- Datenblocklaenge - Anzahl der Bytes fuer gespeicherte Merkmale. Falls 0 oder zu klein fuer die ueblichen Merkmale, nicht verwendet.
- Rueckkehradresse
- Fehlerrueckkehradresse

Fertigstellungskode
Folgevektor - Zeiger zu einem Bereich mit folgendem Inhalt:
Byte1 Information ueber durchzufuehrende Aktion
Byte2 Kennzeichen des Laufwerks, auf dem die Aktion durchgefuehrt wurde
Byte3 Laenge des Dateinamens
Der OPEN-Aufruf taetigt seine eigenen Zuweisungen (ASSIGN), wodurch nicht zwei Aufrufre von NDOS noetig sind. Die benoetigten Informationen muessen denen des ASSIGN-Aufrufs entsprechen. Falls kein ASSIGN stattfinden soll, muss dieses Byte 0FFH gesetzt sein.
Byte4..Dateiname, falls vorhanden

Zur Beachtung:

Bei der Erstellung einer neuen Datei wird der erste Zeigersektor dieser Datei initialisiert und auf Diskette geschrieben.

Varianten des OPEN-Aufrufs:

Die verschiedenen Arten der Aktivierung einer Datei werden durch das erste Byte des Folgevektors spezifiziert. Eine Datei kann fuer folgende Zugriffe eroeffnet werden:

a) Wahlfreier Zugriff (random access) spezifiziert durch Setzen von Bit 3 dieses Bytes, es ergibt sich: der READ DIRECT-Zugriff wird angenommen (ansonsten invalid Request- Fehler) und bei jeder satzorientierten Aktion wird die Adresse des ersten angesprochenen Satzes an das aufrufende Programm zurueckgeliefert (im Folgevektor des Parametervektors).

b) Sequentieller Zugriff
Es existieren fuenf einander gegenseitig ausschliessende Verfahren fuer die Handhabung der Faelle, ob die Datei gefunden wurde oder nicht. Sie werden in den unteren drei Bits des ersten Bytes des Folgevektors angegeben:

Open for INPUT Kode 0
Sofern die Datei existiert, wird sie aktiviert, wobei der Zeiger auf den Beginn des ersten Satzes weist, bei Nichtexistenz erfolgt die Fehlermeldung FILE NOT FOUND (Kode C7)

Open for OUTPUT Kode 1
Sofern die Datei existiert, wird sie aktiviert und ihre saemtlichen Saetze werden geloescht; bei Nichtexistenz wird sie erstellt.

Open NEW FILE Kode 2
Oeffnen fuer nicht zerstoerende Ausgabe. Sofern die Datei existiert, folgt die Fehlermeldung DUPLICATE FILE (Kode D0) und die Datei wird nicht aktiviert; bei Nichtexistenz wird sie erstellt.

Open for APPEND Kode 3
Sofern die Datei existiert, wird sie aktiviert und der Zeiger weist auf das Ende des letzten Satzes; bei Nichtexistenz wird sie erstellt.

Open for UPDATE Kode 4
Sofern die Datei existiert, wird sie aktiviert und der Zeiger zeigt zum Beginn des ersten Satzes; bei Nichtexistenz wird sie erstellt.

Aktion: Sofern die Zuweisung eines Dateinamens gewuenscht wurde (Byte 3 des Folgevektors ist nicht 0FFE), wird eine ASSIGN-Subroutine aufgerufen, als ob ein ASSIGN-Aufruf erfolgt waere. In diesem Fall wird das Directory des aufgerufenen Laufwerkes nach dem Dateinamen durchsucht (beim Kode * erfolgt eine Ueberpruefung der READY-Information aller vorhandenen Laufwerke), bis die Datei gefunden wurde oder saemtliche moeglichen Laufwerke durchsucht sind.
Die ID (Identifikation) jener Diskette, welche entweder die Datei enthaelt oder auf der die Datei erstellt wird, kommt in einen Pufferspeicher und wird mit der ID der korrespondierenden Liste im Speicher verglichen. Wenn sie nicht uebereinstimmen und keine andere Datei im Moment auf demselben Diskettenlaufwerk offen ist, wird eine neue Liste (und ID) angelegt und in den Speicher geschrieben. Falls eine Datei offen ist, erfolgt eine Fehlermeldung WRONG DISK die Datei wird nicht aktiviert.
Beim Auffinden der Datei wird ihr Deskriptor gelesen, seine wichtigen Teile in die Tabelle der aktiven Dateien uebernommen und sofern gewuenscht, in den Datentransferbereich transferiert. Anschliessend wird die Datei als OFFEN markiert.
Soll eine Datei erst erstellt werden, erzeugt man einen Deskriptorsatz in einem Pufferspeicher, transferiert ihn zur Tabelle der aktiven Dateien und falls gewuenscht, zum Datenspeicherbereich des Anwenders. Hat die Datei einen Namen, wird sie auch zur Diskette geschrieben und erhaelt einen Platz im Directory.

Moegliche Fehler:
Grundsaetzlich sind alle DISK ERRORS moeglich.

NOT READY Kode C2
Das angesprochene Diskettenlaufwerk war beim Aufruf nicht bereit.

PROTECTION Kode C3
Die Diskette ist schreibgeschuetzt, bzw. die aufge-

rufene Datei schreib- oder loeschgeschuetzt gegen OPEN for OUTPUT (mit anschliessendem Loeschen ihrer Saetze). Im letzteren Fall wird die Datei eroeffnet, aber nicht geloescht.

UNIT ALREADY OPEN Kode CC
Die logische Einheit ist bereits aktiv. Sie muss entweder geschlossen oder neu initialisiert werden, um fuer OPEN bereit zu sein. In diesem Fall wird keine Aktion durchgefuehrt.

WRONG DISK Kode D1
ID der Diskette im Laufwerk ist nicht identisch mit ID im Speicher. Das heisst, dass die Disketten nach der letzten INITIALIZE-Aktion ausgetauscht wurden, bzw. dass die Listen im Speicher von einem Programm ueberschrieben wurden. Die Datei wird nicht eroeffnet.

FILE NOT FOUND Kode C7
Der OPEN-Aufruf war fuer eine Eingabe gedacht und die aufgerufene Datei existiert nicht.

POINTER ERROR Kode CA
Die Sektoranzeiger fuer die Verbindung der verschiedenen Teile sind falsch oder zerstoert oder die Zeiger des Deskriptors sind inkorrekt.

DUPLICATE FILE Kode D0
Aufruf zum Oeffnen einer neuen Datei, waehrend die Datei bereits existiert. Die Datei wird nicht aktiviert.

INVALID ATTRIBUTE Kode D2
Eine spezifizierte Eigenschaft war ungueltig oder die Satzlaenge ist inkorrekt. Die Datei wird mit den automatisch zugewiesenen Eigenschaften anstelle der inkorrekten aktiviert.

DISK FULL Kode D3
Es ist kein Platz fuer den Deskriptor. Es ist kein weiterer Directory-Sektor verfuegbar. Dieser Fehler kann nur bei der Dateierstellung auftreten.

FILE ALREADY OPEN Kode D6
Die zu oeffnende Datei ist bereits bei einer anderen logischen Funktion aktiviert.

PROPERTIES PROTECTION Kode D8
Versuch, die Eigenschaften bei einer blockierten Datei zu aendern.

INVALID OPEN REQUEST Kode D9
Ungueltiger Aufruf (weder INPUT, OUTPUT, NEWFILE, APPEND, UPDATE)

INSUFFFICIENT MEMORY Kode DA
Der Speicherplatz fuer die Belegungsplaene der Disketten reicht nicht aus.

Warnungen:

DIRECTORY FORMAT ERROR Kode 81
Das Format eines oder mehrerer Directory-Saetze war falsch. Der Satz kann gelesen werden, aber die Daten koennen falsch sein.

SCRATCH FILE Kode 82
Eine SCRATCH-Datei wurde erstellt.

ATTRIBUTES TOO LONG Kode 84
Mehr als 116 Bytes Merkmale (nur 116 Bytes uebertragen).

### 5.4.5. CLOSE

Vektor: Logische Funktion
Kode - 06 oder 07
Datenstartadresse- Zeiger zu einem Speicherbereich mit Eigenschaften, die die der Datei ersetzen. Format wie bei OPEN. Wenn eine Aenderung der Eigenschaften unerwuenscht ist, =0.
Datenblocklaenge - Anzahl der zum Deskriptor zu transferierenden Bytes (max. 116), ansonsten =0.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Sofern die Datei eine SCRATCH-Datei war, wird sie geloescht. Bei einer Aenderung der Belegung oder der Eigenschaften der zu schliessenden Datei, liest das System den Deskriptor, bringt ihn auf den letzten Stand und aendert den Belegungsplan. Die Datei wird anschliessend als geschlossen markiert.

Moegliche Fehler

FILE NOT OPEN KODE CB
Die logische Funktion war nicht aktiv.

WRONG DISK KODE D1
Die Diskettenidentifikation ist nicht mit der im Speicher identisch. Das kann nach Austausch der Diskette ohne INITIALIZE oder Ueberschreiben der Belegungsplaene durch ein Programm auftreten. die Datei wird nicht abgeschlossen. Zur Behebung des Fehlers muss entweder die korrekte Diskette wieder eingesetzt oder INITIALIZE ausgefuehrt werden.

INVALID ATTRIBUTE Kode D2
Entsteht, falls der Dateityp nicht zu den Eigenschaften passt.

Alle DISK-ERRORS sind moeglich.

POINTER ERROR Kode CA
Zeigt einen Zeigerfehler an, der waehrend des Loeschens einer SCRATCH-Datei auftritt, bzw. dass die Zeiger zur Verkettung der Zeigerbloecke unrichtig sind.

Warnung:

ATTRIBUTES TOO LONG Kode 84
Es wurden mehr als 116 Bytes Eigenschaften angegeben, jedoch nur 116 Bytes transferiert (maximal).

### 5.4.6 REWIND

Vektor: Logische Funktion
Kode - 08 oder 09
Datenstartadresse- nicht verwendet
Datenblocklaenge - nicht verwendet
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Der Dateizeiger wird zum Deskriptor positioniert, wobei als naechster Satz der erste Datensatz markiert wird. Diese Position wird beim Eroeffnen der Datei (OPEN FILE) oder beim Erstellen (CREATING) eingenommen, nicht aber beim Erweitern (APPEND). Bei leerer Datei ist der Zeiger auf den naechsten Satz (NEXT RECORD POINTER) gleich 0.

Moegliche Fehler:

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert. Es wird keine Aktion durchgefuehrt.

### 5.4.7. READ BINARY

Vektor: Logische Funktion
Kode - 0A oder 0B
Datenstartadresse- Anfangsadresse, ab der die Daten gespeichert werden.
Datenblocklaenge - Anzahl der zu lesenden Bytes. Falls diese Zahl kein Vielfaches der Satzlaenge ist, wird sie aufgerundet. Enthaelt bei Rueckkehr die Zahl der tatsaechlich transferierten Bytes.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Ist die Datei fuer wahlfreien Zugriff offen, enthaelt dieses Feld die Adresse eines Bereichs von drei Bytes, in welchen die Diskettenadresse des ersten gelesenen Satzes geschrieben wird.

Aktion: Die Daten werden vom naechstfolgenden Satz der Datei gelesen und ab der Datenstartadresse abgespeichert. Der Zeiger zeigt auf den letzten gelesenen Satz.
Wurde die Datei fuer wahlfreien Zugriff eroeffnet, enthaelt nach der Rueckkehr der Folgevektor die Adresse des ersten gelesenen Satzes. Das dritte Byte dieser Adresse ist stets 0.

Moegliche Fehler:
Alle Diskettenfehler ausser PROTECTION moeglich.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert. Es wird keine Aktion durchgefuehrt.

END OF FILE Kode C9
Der letzte Satz der Datei wurde gelesen, ohne dass die Zahl der gewuenschten Bytes erreicht wurde. Die rueckgemeldete Datenblocklaenge liefert eine Information ueber die tatsaechliche Bytezahl.

### 5.4.8. WRITE BINARY

Vektor: Logische Funktion
Kode - 0E oder 0F
Datenstartadresse- Adresse, ab der die zu schreibenden Daten zu finden sind.
Datenblocklaenge - Die Zahl der zu schreibenden Bytes wird auf ein ganzzahliges Vielfaches der Satzlaenge aufgerundet. Bei Rueckkehr steht hier die Zahl der tatsaechlich uebertragenen Bytes.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Bei wahlfreiem Zugriff zeigt die hier enthaltene Adresse auf einen Drei-Byte-Bereich, der die Adresse des ersten beschriebenen Satzes enthaelt.
Ansonsten nicht verwendet.

Aktion: Es werden neue Saetze erzeugt und mit Daten aus dem Datenspeicher gefuellt. Die neuen Saetze werden hinter dem aktuellen eingeschoben. Der Zeiger weist auf den letzten eingeschriebenen Satz, der Zeiger fuer den naechsten Satz (NEXT RECORD POINTER) zeigt auf denselben Satz wie vor der Operation.

Moegliche Fehler:
Alle Diskettenfehler ausser CRC (Kode C6) koennen auftreten.

PROTECTION Kode C3
Meldung, falls die Datei schreibgeschuetzt war.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion war nicht aktiviert. Keine Aktion wird durchgefuehrt.

DISK FULL Kode D3
Auf der Diskette ist kein Platz fuer weitere Daten, wobei die Diskette bereits mit einigen neuen Saetzen beschrieben worden sein kann. die Datenblocklaenge enthaelt die Anzahl der tatsaechlich aufgezeichneten Bytes bis zum Zeitpunkt der vollstaendigen Auffuellung.

### 5.4.9. WIRTE CURRENT

Vektor: Logische Funktion
Kode - 12 oder 13
Datenstartadresse- Anfangsadresse des Datenspeicherbereichs
Datenblocklaenge - Falls = 0, werden keine Daten transferiert, ansonsten ein Satz. Bei der Rueckkehr steht hier die Zahl der tatsaechlich uebertragenen Bytes.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Bei Aufruf fuer wahlfreien Zugriff enthaelt dieses Feld die Adresse eines Drei-Byte-Bereichs mit der Diskettenadresse des Satzes; ansonsten nicht verwendet.

Aktion: Es wird ein Satz vom Speicher zur Datei uebertragen, welcher die Daten des momentanen markierten Satzes ueberschreibt. Dabei wird kein neuer Satz erzeugt und der Satz-Zeiger (RECORD POINTER) bleibt unveraendert.

Moegliche Fehler:
Alle Diskettenfehler sind moeglich, ausser CRC (KODE C6).

PROTECTION (Kode C3) wird bei schreibgeschuetzten Dateien gemeldet.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert.

### 5.4.10. DELETE

Vektor: Logische Funktion
Kode - 16 oder 17
Datenstartadresse- nicht verwendet
Datenblocklaenge - Anzahl der von der Datei geloeschten Bytes, wird auf einen

vollen Satz aufgerundet. Enthaelt bei der Rueckkehr die Zahl der geloeschten Bytes.

Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Beginnend mit dem aktuellen Satz werden solange Saetze von der Datei geloescht und ihr Platz freigegeben (deallocated), bis die gewuenschte Anzahl von Bytes geloescht wurde. Der Satzzeiger (CURRENT RECORD POINTER) bleibt auf dem letzten, den geloeschten Saetzen vorausgehenden Satz, stehen. Steht der Zeiger gerade auf dem Deskriptor, wird er zum ersten Datensatz positioniert, bevor die Aktion ablaeuft. Anschliessend zeigt er wieder auf den Deskriptor.

Moegliche Fehler:

Alle Diskettenfehler einschliesslich PROTECTION, falls die Datei schreib- oder loeschgeschuetzt war, sind moeglich.

END OF FILE Kode C9
Der letzte Satz der Datei wurde geloescht, ohne die gewuenschte Zahl der zu loeschenden Bytes zu erreichen. die gemeldete Zahl von Bytes entspricht den tatsaechlich geloeschten Bytes, inklusive des letzten Satzes. Der Satzzeiger (CURRENT RECORD POINTER) zeigt auf den, dem ersten geloeschten Satz vorangehenden Satz; der Zeiger fuer den naechsten Satz (NEXT RECORD POINTER) ist 0.

POINTER ERROR Kode CA
Waherend des Loeschens wurde ein Zeigerfehler entdeckt. Alle Saetze bis zum Auftreten des Fehlers werden geloescht.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiv.

### 5.4.11. DELETE REMAINING RECORDS

Vektor: Logische Funktion
Kode - 18 oder 19
Datenstartadresse- nicht verwendet
Datenblocklaenge - nicht verwendet, rueckgemeldet wird die Anzahl geloeschter Bytes
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Alle Saetze einer Datei vom aktuellen bis zum letzten werden geloescht. Der Zeiger zeigt auf den letzten Satz vor dem geloeschten. Der Zeiger fuer den naechsten Satz ist 0. Zeigt der Zeiger vor der

Operation auf den Deskriptor, wird er vor dem Loeschen auf den ersten Datensatz gesetzt.

Moegliche Fehler:
Alle Diskettenfehler einschliesslich PROTECTION.

POINTER ERROR Kode CA
Waehrend des Loeschens wurde ein Zeigerfehler entdeckt. Alle Saetze bis zum Auftreten des Fehlers werden geloescht.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert.

## 5.4.12. ERASE

Vektor: Logische Funktion
Kode - 1A oder 1B
Datenstartadresse- nicht verwendet
Datenblocklaenge - nicht verwendet, die Zahl der geloeschten Bytes wird zurueckgemeldet.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Alle Saetze einer Datei einschliesslich Deskriptor werden freigegeben (deallocated). Der Einsprung in das Directory wird geloescht. Es muss kein OPEN-Aufruf erfolgen, jedoch ASSIGN ist noetig.

Moegliche Fehler:
Alle Diskettenfehler sind moeglich, einschliesslich PROTECTION,

NOT READY Kode C2
Entsteht, falls das angegebene Laufwerk nicht aktiv ist.

FILE NOT FOUND Kode C7
Die genannte Datei kann bei den spezifizierten Laufwerken nicht aufgefunden werden.

POINTER ERROR Kode CA
Beim Aufruf bzw. Loeschen der Saetze wurde ein Zeigerfehler entdeckt. Alle nachfolgenden Saetze bleiben als belegt markiert (allocated).

INVALID DRIVE Kode CE
Das angegebene Laufwerk war ungleich 0-3; * ist keine gueltige Spezifikation.

WRONG DISK Kode D1
Die Disketten-ID entspricht nicht der Angabe im Speicher. Das kann beim Wechseln einer Diskette

ohne nachfolgendem INITIALIZE oder Ueberschreiben der Belegungsplaene geschehen.

FILE NOT FOUND IN DIRECTORY Kode D4
Zeigt an, dass kein Directory-Einsprung im vom Deskriptor angegebenen Directory-Satz existiert. Die Saetze werden zwar freigegeben, jedoch der Directory-Einsprung bleibt irgendwo erhalten. Bei Auftreten dieses Fehlers sollte die Diskette (nach Kopie der funktionsfaehigen Dateien) neu formatiert werden, um ungewollte Fehler durch eventuellen Zugriff zu dieser ungueltigen Datei zu vermeiden.

FILE ALREADY OPEN (ON ANOHER UNIT) Kode D6
Entsteht beim Versuch des Loeschens einer Datei, die bei einer anderen logischen Funktion aktiviert ist. Es wird keine Aktion durchgefuehrt.

### 5.4.13. READ AND DELETE

Vektor: Logische Funktion
Kode - 1C oder 1D
Datenstartadresse - ab dieser Adresse werden die gelesenen Daten abgespeichert
Datenblocklaenge - Anzahl der zu lesenden Bytes; wird auf eine volle Satzlaenge aufgerundet, enthaelt bei Rueckkehr die Anzahl der tatsaechlich gelesenen Bytes.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Enthaelt bei wahlfreiem Zugriff (random I/O) in einem Drei-Byte-Feld die Diskettenadresse des ersten gelesenen Satzes.

Aktion: Dieser Aufruf ist nuetzlich, wenn Daten gelesen, veraendert und neu gespeichert werden. Vom Satz beginnend, nach dem gerade markierten, werden die Daten aus der Datei in den Speicher transportiert und gleichzeitig geloescht (DEALLOCATION der Saetze). Der markierte Satz bleibt unveraendert, ebenso der Zeiger; dadurch besteht die Moeglichkeit, von der gleichen Stelle an neue Daten wieder einzuschreiben. Der Satz nach dem letzten gelesenen und geloeschten Satz bleibt als NEXT RECORD markiert.

Moegliche Fehler:
Alle Diskettenfehler sind moeglich, einschliesslich PROTECTION.

END OF FILE Kode C8
Der letzte Satz wurde gelesen, ohne die gewuenschte Byteanzahl zu erreichen. Die tatsaechliche Zahl wird beim Ruecksprung im Datenblocklaengefeld geliefert, der NEXT-RECORD-Zeiger enthaelt 0.

POINTER ERROR Kode CA
Der Datentransfer wird beim Auftreten des Zeigerfehlers beendet. Das Datenblocklaengefeld enthaelt die Zahl der bis dahin uebertragenen Bytes.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert.

### 5.4.14. READ CURRENT

Vektor: Logische Funktion
Kode - 1E oder 1F
Datenstartadresse- Daten werden ab dieser Adresse abgespeichert
Datenblocklaenge - Anzahl der zu uebertragenden Daten. Falls 0, erfolgt keine Aktion, ansonsten wird ein Satz uebertragen. die tatsaechlich uebertragene Anzahl ist nach Ruecksprung hier enthalten.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - enthaelt nur bei wahlfreiem Zugriff (random I/O) in einem Drei-Byte-Feld die Adresse des momentanen Satzes. Ansonsten nicht verwendet.

Aktion: Die Daten des gerade markierten Satzes (CURRENT RECORD) werden in den Speicher transferiert. Der Zeiger bleibt unveraendert.

Moegliche Fehler:
Alle Diskettenfehler ausser PROTECTION

POINTER ERROR Kode CA
Entsteht bei falschem Inhalt des PREVIOUS-RECORD-Zeiger. Der Datentransfer erfolgt trotzdem.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert. Es erfolgt kein Transfer.

BEGINNING OF FILE Kode D5
Der Zeiger weist auf den nicht lesbaren Deskriptor. Es erfolgt kein Datentransfer und eine Null-Adresse wird bei wahlfreiem Zugriff (random I/O) zurueckgegeben.

### 5.4.15. READ PREVIOUS

Vektor: Logische Funktion
Kode - 20 oder 21
Datenstartadresse- Beginn des Datenspeicherbereichs
Datenblocklaenge - Anzahl der zu lesenden Bytes;

falls =0, erfolgt keine aktion, ansonsten wird ein Satz uebertragen. Nach dem Ruecksprung ist hier die Zahl der tatsaechlich transferierten Bytes zu finden.

Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Nur bei wahlfreiem Zugriff (random I/O) sollte dieser Vektor einen Drei-Byte-Bereich angeben, der nach Ruecksprung die Diskettenadresse des vorhergehenden Satzes (PREVIOUS RECORD) enthaelt, ansonsten nicht verwendet.

Aktion: Unter der Voraussetzung, dass die Datenblocklaenge ungleich 0 ist, wird der vorhergehende Satz des momentan markierten Satzes gelesen. Grundsaetzlich wird der Satzzeiger (RECORD POINTER) um einen Satz rueckwaerts positioniert.

Moegliche Fehler:
Alle Diskettenfehler ausser PROTECTION.

POINTER ERROR Kode CA
Entsteht, falls der Vorwaertszeiger des PREVIOUS RECORD nicht auf den aktuellen Vorwaertszeiger zeigt. Die Daten werden trotzdem transferiert.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert, keine Aktion.

BEGINNING OF FILE Kode D5
Es wird versucht, den Deskriptor zu lesen. Dabei wird die Diskettenadresse 0 zurueckgegeben.

### 5.4.16. READ DIRECT

Vektor: Logische Funktion
Kode - 22 oder 23
Datenstartadresse- Anfangsadresse des Datenspeichers
Datenblocklaenge - Anzahl der zu transferierenden Bytes, wird auf einen vollen Satz gerundet. Enthaelt bei Ruecksprung die Zahl der tatsaechlich transferierten Bytes.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Enthaelt ein Drei-Byte-Feld mit der diskettenadresse des ersten zu lesenden Satzes. Das dritte Byte ist 0. Das aufrufende Programm muss sicherstellen, dass

die Adresse zur aufgerufenen Datei gehoert.

Aktion: Der aufgerufene Satz wird wie der naechste Satz gelesen. Falls seine Daten nicht die gesamte Datenblocklaenge liefern, werden auch die nachfolgenden Saetze gelesen, jedoch alle ohne Zeigerkontrolle. Der Aufruf ist nur bei Dateien zulaessig, die fuer wahlfreien Zugriff (opening for random I/P) eroeffnet wurden.

Moegliche Fehler

INVALID REQUEST Kode C1
Entsteht, wenn die Datei nicht fuer wahlfreien Zugriff eroeffnet ist.
Zusaetzlich alle Fehler des Aufrufs READ BINARY.

## 5.4.17. SKIP FORWARD

Vektor: Logische Funktion
Kode - 24 oder 25
Datenstartadresse- nicht verwendet
Datenblocklaenge - Anzahl der Saetze, die uebersprungen werden. bei Rueckkehr ist hier die Anzahl der tatsaechlich uebersprungenen Saetze enthaelten.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Ist bei wahlfreiem Zugriff (random I/O) ein Drei-Byte-Feld, in dem nach Rueckkehr die Diskettenadresse des ersten uebersprungenen Satzes zu finden ist.

Aktion: Der Zeiger auf den momentanen Satz (CURRENT RECORD POINTER) wird in der Datei um den angegebenen Wert vorwaerts bewegt. Es erfolgt kein Datentransfer.

Moegliche Fehler:
Alle Diskettenfehler, ausser PROTECTION.

END OF FILE Kode C3
Erreichen des letzten Satzes waherend der Zeigerbewegung. Der Zeiger markiert diesen; der Zeiger fuer den naechsten Satz (NEXT RECORD POINTER) ist 0.

POINTER ERROR Kode CA
Dieser Fehler kann waehrend des Laufes ueber die einzelnen Saetze entstehen. In diesem Fall bleibt der Zeiger CURRENT RECORD auf dem letzten fehlerfreien Satz stehen.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert.

### 5.4.18. SKIP BACKWARD

Vektor: Logische Funktion
Kode - 26 oder 27
Datenstartadresse- nicht verwendet
Datenblocklaenge - Anzahl der zu ueberspringenden Saetze. Bei Rueckkehr die tatsaechlich uebersprungene Zahl.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Enthaelt nur bei wahlfreiem Zugriff (random I/O) nach der Rueckkehr die Diskettenadresse des vorhergehenden Satzes.

Aktion: Der Zeiger auf den aktuellen Satz (CURRENT RECORD POINTER) wird um die gewuenschte Zahl rueckwaerts bewegt. Es erfolgt kein Datentransfer.

Moegliche Fehler:
Alle Diskettenfehler, ausser PROTECTION.

POINTER ERROR Kode CA
Der Fehler kann waehrend des Laufes erkannt werden. Der Zeiger bleibt auf dem letzten fehlerfreien Satz stehen.

BEGINNING OF FILE Kode D5
Der Anfang der Datei wurde vor Ablauf der Satzzahl erreicht. er Zeiger bleibt am Deskriptor stehen.

### 5.4.19. SKIP TO END

Vektor: Logische Funktion
Kode - 28 oder 29
Datenstartadresse- nicht verwendet
Datenblocklaenge - nicht verwendet, 0 wird zurueckgegeben.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Zeigt auf ein Drei-Byte-Feld, das nach Rueckkehr die Diskettenadresse des letzten Satzes enthaelt, ansonsten nicht verwendet.

Aktion: Positioniert den CURRENT RECORD-Zeiger auf den letzten Satz.

Moegliche Fehler:
Alle Diskettenfehler, ausser PROTECTION.

POINTER ERROR Kode CA
Zeigerfehler beim letzten Satz (NEXT RECORD POINTER=0); damit Anzeige, dass der letzte lesbare Satz nicht der letzte Satz der Datei war.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion war nicht aktiviert.

5.4.20. RENAME

Vektor: Logische Funktion
Kode - 2A oder 2B
Datenstartadresse- nicht verwendet
Datenblocklaenge - nicht verwendet
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - Zeiger zu Bereich mit folgendem Inhalt:
Byte 1 Namenslaenge
Byte 2..neuer Name

Aktion: Die aufgerufene Datei erhaelt den neuen Namen aus dem Folgevektor. die Datei muss eroeffnet sein (OPEN) oder eine gueltige Zuweisung (ASSIGN) muss existieren, damit sie eroeffnet werden kann. Falls noch nicht eroeffnet, wird sie aktiviert, der Directory-Einsprung entfernt und ein neuer erzeugt (unter dem neuen Namen). anschliessend wird die Datei (falls nicht bereits eroeffnet) wieder geschlossen.
Eine offene SCRATCH-Datei kann zu einer normalen Datei (ein Deskriptor wird erzeugt), aber eine normale Datei nicht in eine SCRATCH-Datei umgewandelt werden.

Moegliche Fehler:
Alle Diskettenfehler sind moeglich.

FILE NOT FOUND Kode C7
Die aufgerufene Funktion war nicht eroeffnet und die zugeordnete Datei ebenfalls, es erfolgt keine aktion.

POINTER ERROR Kode CA
Der Rueckwaertszeiger im Deskriptor zeigt nicht zum Directory, keine weitere Aktion.

DUPLICATE FILE Kode D0
Es existiert bereits eine Datei gleichen Namens, keine Aktion.

DISK FULL Kode D3
Ein neuer Directory Sektor war noetig, um den Namen aufzunehmen. Die Diskette enthielt nicht mehr ausreichend Platz, keine Aktion.

FILE NOT IN PROPPER DIRECTORY RECORD Kode D4
Es existiert kein Directory-Einsprung; keine Aktion.

FILE ALREADY OPEN IN ANOTHER UNIT Kode D6
Die aufgerufene Datei ist bereits auf einer anderen logischen Funktion eroeffnet; keine Aktion.

INVALID RENAME Kode D7
Versuch der Erzeugung einer SCRATCH-Datei aus einer Normalen Datei, bzw. Dateiname mit mehr als 32 Zeichen; keine Aktion.

## 5.4.21. UPDATE

Vektor: Logische Funktion
Kode - 2C oder 2D
Datenstartadresse- Adresse der Eigenschaften (Format siehe OPEN-Aufruf)
Datenblocklaenge - Zahl der Bytes der Eigenschaften-Information.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner

Aktion: Bei Aenderung der Datei oder ihrer Eigenschaften erfolgt ein Aufruf des Deskriptors, Lesen, Aenderung und Rueckschreiben, ebenso ein Rueckschreiben des Belegungsplans (ALLOCATION MAP). die Aenderung mittels UPDATE erfolgt wie bei CLOSE. Die Datei bleibt aktiv.

Moegliche Fehler:
Alle Diskettenfehler.

POINTER ERROR Kode CA
entsteht, falls der Deskriptorzeiger nicht auf das Directory weist.

FILE NOT FOUND Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert.

WRONG DISK Kode D1
Die Diskettenidentifikation (ID) des Laufwerkes fuer diese Datei ist nicht mit der im Speicher abgelegten ID identisch. Zeigt zumeist, dass ein Austausch der Disketten oder ein Ueberschreiben des Belegungsplan stattfand. Keine Aktion.

INVALID ATTRIBUTES Kode D2
Hier werden die Eigenschaften Typ und Satzlaenge (RECORD LENGTH) geprueft. Die Fehlerhaften Eigenschaften bleiben unveraendert.

PROPERTY PROTECTION Kode D8
Blockierte Eigenschaften (LOCKED ATTRIBUTES), d.h., die Eigenschaften der Datei sind nicht aenderbar.

### 5.4.22. SET ATTRIBUTES

Vektor: Logische Funktion
Kode - 2E oder 2F
Datenstartadresse- Adresse der zu uebertragenden Eigenschaften (siehe Format des OPEN-Aufrufs)
Datenblocklaenge - Anzahl der Bytes der Eigenschaften
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner
Aktion: Der Deskriptor wird gelesen und mit den neuen Eigenschaften versehen, zurueckgeschrieben.

Moegliche Fehler:
Alle Diskettenfehler.

POINTER ERROR Kode CA
Der Deskriptorzeiger weist nicht zum Directory.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion ist nicht aktiviert.

INVALID ATTRIBUTES Kode D2
Satzlaenge oder Typ sind ungueltig.

PROPERTY PROTECTION CODE Kode D8
Blockierte Dateieigenschaften.

### 5.4.23. QUERY ATTRIBUTES

Vektor: Logische Funktion
Kode - 30 oder 31
Datenstartadresse- Startadresse zum Einschreiben der Eigenschaften in den Speicher.
Datenblocklaenge - Hier wird die Anzahl der Bytes der Eigenschaften zurueckgegeben.
Rueckkehradresse
Fehlerrueckkehradresse
Fertigstellungskode
Folgevektor - keiner
Aktion: Lesen des Deskriptors zur Information ueber eine eroeffnete Datei. Format siehe OPEN-Aufruf.

Moegliche Fehler:
Alle Diskettenfehler, ausser PROTECTION.

POINTER ERROR Kode CA
Der Deskriptorzeiger weist nicht zum Directory.

FILE NOT OPEN Kode CB
Die aufgerufene logische Funktion war nicht aktiviert.

ATTRIBUTE LIST TRUNCATED Kode 84
Warnung! Es wurden mehr als 116 Bytes verlangt, aber nur 116 koennen transferiert werden.

# 6. Realisierung von Ein/Ausgabe-Zugriffen

## 6.1. Ein/Ausgabe-Tabellen

Die Ein/Ausgabe-Struktur des UDOS-Betriebssystems erfuellt zwei Anforderungen:

- Die Entwicklung und Implementierung von Interface-Treiberprozeduren fuer System- und Anwenderprogramme ist ohne Schwierigkeiten moeglich.
- Der Zugriff zu allen Ein/Ausgabe-Treibern ist vereinfacht und standardisiert. Alle Ein/Ausgabe-Zugriffe erfolgen ueber UDOS mit dem Aufruf einer logischen Funktion. UDOS bestimmt den gueltigen Ablauf fuer diese logische Einheit und ruft die entsprechende Treiberprozedur auf.

Fuer die Vereinheitlichung des Zugriffs sind von UDOS Ein-Ausgabe-Tabellen eingerichtet:

- Tabelle der aktiven Treiberprozeduren (ADT)
  ACTIVE DEVICE TABLE
  Die Tabelle enthaelt Einsprungspunkt und Prozedurnamen fuer jedes momentan aktive Treiberprogramm. Mittels UDOS-Kommando ACTIVATE (s. Abschn. 4.2.) werden die Treiber in die Tabelle eingefuegt oder aber mittels DEACTIVATE (s. Abschn. 4.12.) entfernt. Die Namen aktiver Treiber koennen zur "Qualifizierung" von Dateinamen verwendet werden. Ist keine "Qualifizierung" des gerade verwendeten Dateinamen vorgenommen, so wird der Dateiname automatisch mit dem Master-Treiberprogramm qualifiziert (s. Abschn. 4.29.). Die Tabelle kann mittels UDOS-Kommando LADT (s. Abschn. 4.28.) aufgerufen werden.
- Tabelle fuer logischen Prozeduren (LFT)
  LOGICAL FILE TABLE
  Die Tabelle der logischen Prozeduren enthaelt die Verbindung zwischen logischer Einheit und aktiver Treiberprozedur, speziell den Eintrittspunkt des Treibers, zu dem die logische Funktion verbunden wurde. Bevor Ein/Ausgabe-Aufrufe erfolgen koennen, ist eine Definition der zugehoerigen logischen Funktion noetig. diese Definition erfolgt mittels ASSIGN (s. Abschn. 6.4.) oder dem UDOS-Kommando DEFINE (s. Abschn. 4.15.).
  Mit der Initialisierung des Systems wird automatisch die Definition der logischen Funktionen 1,2 und 3 als Terminaleingabe, Terminalausgabe und als schneller Drucker vorgenommen. Diese Funktionen sind zwar neu definierbar, UDOS erwartet aber aequivalente Treiber fuer 1,2 und 3.

## 6.2. Ein/Ausgabe-Operationen mittels Systemaufruf

Eine vom Anwender benoetigte Ein/Ausgabe-Operation kann auch als Systemaufruf mittels CALL auf die Einsprungadresse SYSTEM 1003H ausgefuehrt werden. Dazu muss das Indexregister IY die Startadresse eines Aufrufvektors beinhalten. dieser Vektor, auch UDOS-Parametervektor genannt, besteht aus 13 aufeinanderfolgenden Bytes.
Der Ein/Ausgabe-Aufruf kann auf zwei Arten erfolgen (siehe auch Byte 2 im Parametervektor):

- RETURN ON COMPLETION
  Hierbei fuehrt der Treiber die gesamte Ein/Ausgabe-Operation durch und kehrt anschliessend zum aufrufenden Programm zurueck.

- IMMEDIATE RETURN
  Der Treiber fuehrt die Initialisierung der ein/Ausgabe-Operation durch, kehrt sofort zum aufrufenden Programm zurueck, arbeitet dort weiter und setzt den ein/Ausgabe-Aufruf unter Interrupt fort.

Ein Fertigstellungskode (COMPLETION CODE) gibt Auskunft ueber fehlerhaften oder erfolgreichen Abschluss des Ein/Ausgabe-Aufrufs. Er wird in ein Byte des Parametervektors geschrieben und kann nach Rueckkehr ins aufrufende Programm abgefragt und entsprechend ausgewertet werden.
Im Einzelnen hat der Parametervektor folgenden Aufbau:

| | Byte | Inhalt |
|---|---|---|
| IY ---> | 1 | Nummer der logischen Funktion<br>Es ist eine Zahl zwischen 9 und 20, wobei 1,2 und 3 von UDOS als CONIN, CONOUT und SYSLST vordefiniert sind. |
| | 2 | Kodenummer der Operation<br>Sie identifiziert die gewuenschte Ein/Ausgabe-Operation. soll eine Rueckkehr nach Fertigstellung der Operation erfolgen, ist Bit0=0. Wird Bit 0 mit 1 belegt, erfolgt Interrupt-Abarbeitung. |
| | 3,4 | Startadresse des Datenblocks<br>Anfangsadresse des zu transferierenden Datenblocks. Wenn keine Datenuebertragung erforderlich, ist die Adresse mit 0000 zu belegen. |
| | 5,6 | Laenge des Datenblocks<br>Es ist die Anzahl der zu transferierenden Bytes einzutragen. Nach Ablauf der Prozedur wird die Laenge auf die Anzahl der tatsaechlich transferierten Bytes gesetzt. |
| | 7,8 | Ruecksprungadresse<br>Falls in der Kodenummer der Operation Bit=1 ist, wird so schnell wie moeglich aus der Treiberprozedur in das aufrufende Programm zurueckgekehrt. Die Wei- |

terarbeit im Treiber erfolgt dann durch Interrupts, die durch den Treiber selbst initialisiert wurden. Bei Beendigung des Ein/Ausgabe-Aufrufs erfolgt der Ruecksprung auf die in Byte 7,8 angegebene Adresse, auf der eine Interrupt-Service-Routine beginnen sollte. Ist Bit0=0 (d.h. Ruecksprung erst nach Beendigung) wird die Ruecksprungadresse ignoriert.

9,10 Ruecksprungadresse bei Fehler
hier kann die Einsprungadresse einer Fehler-Service-Routine eingetragen werden, falls bei der Abarbeitung der angeforderten Operation ein Fehler auftrat. Der Abschluss sollte RET sein, um eine Rueckkehr zur aufrufenden Routine zu gewaehrleisten. Ist als Rueckspungadresse 0000 eingetragen, so wird sie ignoriert.

11 Fertigstellungskode
Der Fertigstellungskode wird immer von der aufgerufenen Treiberprozedur geschrieben. Er liefert nach Rueckkehr zur aufrufenden Routine die Information ueber die Fertigstellung des Aufrufs, wobei alle Treiber mit den gleichen Kodes auskommen.
Es gilt: Bit7=1 markiert den vollstaendigen Ablauf
Bit6=1 markiert fehlerhafte Uebertragung
Die restlichen Bits dienen der Kennzeichnung der Fehlerart.

12,13 Hilfsparameter (optional)
Diese Eintragung liefert die Adresse eines Hilfsvektors, der zusaetzliche Aufgaben enthalten kann. Reichen zwei Byte Zusatzinformation aus, koennen sie in Byte 12,13 eingetragen werden und der Hilfsvektor entfaellt (wie z.B. in NDOS-Aufrufen).

## 6.3. Ein/Ausgabe-Aufruf ASSIGN

Wird ein Systemaufruf mit dem Kode fuer ASSIGN durchgefeuhrt, so prueft UDOS auf:

- Definition einer logischen Funktion
- Manipulation des Hilfsparametervektors

Das erste Byte (Flagbyte) des Hilfsparametervektors gibt den Ablauf fuer den Ein/Ausgabe-Aufruf ASSIGN an. folgender Zusammenhang zwischen dem Inhalt des Flagbytes und der Wirkung des ASSIGN-Aufrufs gilt:

| Flagbyte | Wirkung |
|---|---|
| 00H | UDOS formatiert den Hilfsparametervektor. Dieser enthaelt dann den Namen der Floppy-Station, die Laenge des Dateinamens und den Dateinamen. Die Datentransferadresse des aufrufenden Parametervektors ist die Startadresse der Dateinamenzeichenkette im Datenfeld. |
| 02H | UDOS formatiert den Hilfsparametervektor, wobei das Dateinamenfeld die Dateinamenzeichenkette enthaelt. |
| 80H | Parameter- und Hilfsparametervektor werden als im korrekten Format befindlich betrachtet, d.h. alle Felder enthalten gueltige Informationen. UDOS uebergibt in diesem Fall den Aufruf direkt dem Treiberprogramm. Eine vorhergehende Zuweisung des Treibers zur |
| 81H | logischen Funktion ist notwendig. Parameter- und Hilfsparametervektor werden als im korrekten Format befindlich betrachten. UDOS uebergibt den Aufruf dem Master-Treiber. |

Beispiele:

a) Inhalt des Flagbytes: 02H
als Parameter benoetigt das Anwenderprogramm einen qualifizierten oder unqualifizierten Dateinamen. Diese Zeichenkette soll folgende Informationen liefern, die dann von UDOS in den Hilfsparametervektor geschrieben werden:

- Treibername (wenn vorhanden)
- Floppy-Station (wenn vorhanden)
- Dateiname und Dateilaenge

b) Inhalt des Flagbytes: 00H
Die Datentransferadresse im Parametervektor zeigt auf den Beginn der Dateinamenzeichenkette (muss mit einem gueltigen Begrenzungszeichen abgeschlossen sein). Daraufhin wird von UDOS der Dateiname in das Dateinamenfeld des Hilfsparametervektors eingetragen und die Felder fuer den Namen des Laufwerks und die Dateinamenlaenge auf den richtigen Wert gesetzt. Ist kein Laufwerk angegeben, wird * eingesetzt, wenn kein Dateinamen angegeben ist, ist die Laenge 0.
Die im Parametervektor angegebene logische Funktion wird dem Treiberprogramm (in der Dateinamenzeichenkette angegeben) zugewiesen. Ist kein Treiber angegeben, erfolgt die Zuweisung zum Master-Treiber.

## 6.4. UDOS Standard-Treiber

Von UDOS werden nach der System-Initialisierung fuenf Treiber (devices) verwendet, die die UDOS Standard-Treiber darstellen. Es handelt sich um folgende Treiber:

- NDOS
- NULL
- CON
- PCON
- FLOPPY

Zu NDOS:
NDOS ist das Dateiverwaltungssystem von UDOS. Als einziger Treiber im Standardsystem kann es zwischen den verschiedenen logischen Funktionen unterscheiden und mit Namen versehene Dateien bearbeiten (s. Abschn. 5).

Zu NULL:
NULL ist ein Pseudo-Treiber. Er kann durch alle Aufruf-Kodes aktiviert werden. Die durchgefuehrte Aktion ist in den meisten Faellen Null, es wird also nichts durchgefuehrt. Es wird trotzdem ein Durchfuehrungskode geliefert, der Informationen ueber die Datei gibt.

| Ein/Ausgabe-Aufruf | Aktion |
|---|---|
| READ LINE | Durchfuehrungskode 09H (Ende der Datei) |
| READ BINARY | Datenlaenge=0 |
| alle uebrigen | Durchfuehrungskode 80H (Aktion abgeschlossen) |

Mit diesem Treiber koennen beispielsweise die Dateien einer Diskette geprueft werden, da beim Kopieren der Dateien zu Null eine vollstaendige Leseoperation durchgefuehrt wird.

Zu CON:
CON ist der automatisch zugewiesene UDOS-Terminaltreiber. Zur Kommunikation mit CON wird der Standard-E/A-Vektor (Parametervektor) verwendet. Mit CON ist die Definition der Zeichen- und Zeilenloeschsymbole und die Tabulation innerhalb der maximalen Zeilenlaenge von 134 Zeichen moeglich. Folgende Funktionen werden von CON ausgefuehrt:

Loeschen von Zeichen
Die Eingabe des Zeichenloeschsymbols BACKSPACE (08H) bewirkt die logische Loeschung des letzten in den Eingabepufferspeicher eingegebenen Zeichens waehrend READ-Aktionen. Zum Loeschen des Zeichens auf dem Bildschirm und zur Positionierung des Kursors wird zum Terminal die Folge BACKSPACE, SPACE, BACKSPACE ausgesandt.

Zeilenvorschub
Die Eingabe des Zeichens LINE FEED (=aH) setzt den Kursor auf den Anfang der naechsten Zeile. In den Eingabepufferspeicher wird das Zeichen SPACE (20H) eingetragen und damit wird der Kursor zur naechsten

Zeile gebracht, ohne die Eingabe zu beenden. Nur nach Eingabe von Return (Wagenruecklauf=0DH) wird die eingabe beendet, bei Eingabe anderer Zeichen bildet die Eingabe logisch gesehen eine Zeile.

Loeschen von Zeilen
Bei Eingabe von DELETE (7FH) werden vom Schirm und aus dem Eingabepufferspeicher alle Zeichen bis einschliesslich des vorherigen Return geloescht. Eine Ausnahme bilden eingegebene LINE FEED-Zeichen. Wurden diese eingegeben, wird nicht die gesamte Eingabe vom Schirm geloescht.

Loeschen der Eingabe
Mit CONTROL-X wird der Eingabe-Puffer geloescht und liefert als Echo Backslash ('\') gefolgt von Return. Nur beim Aufruf von READ BINARY gibt es einen Unterschied zwischen Loeschen der Eingabe und Loeschen von Zeilen.

Eingabe von Spezialzeichen
Ausser Return kann jedes Spezialzeichen mit Hilfe von '\' eingegeben werden. So erfolgt z.B. die Eingabe von DELETE als '\'DELETE oder aber die Eingabe von '\' als '\\'.

Der Terminaltreiber arbeitet nicht im Interrupt-Betrieb und unterscheidet nicht zwischen logischen Funktionen. Die folgenden Betriebsparameter koennen mit Hilfe des SET-Kommandos (s. Abschn. 4.36.) veraendert werden:

AUTOLF ON (implizit eingeschaltet)
Nach jedem Return wird ein LINE FEED-Zeichen ausgesandt.

NULLCT (implizit NULLCT=1)
Es werden nach jedem Return Nullzeichen (00H) ausgesandt, damit Kursor oder Druckkopf Zeit haben, sich an den Anfang der Zeile zu positionieren.

ECHO ON (implizit gilt ECHO OFF)
Nach dem Lesen wird jedes Zeichen als Echo zum Terminal zurueckgegeben.

TABULATOR
Das Zeichen fuer Tabulator (CONTROL-I) wird nur fuer die Ausgabe mit der entsprechenden Zahl von Zwischenraeumen versehen. Dadurch koennen ASCII-Dateien mit vielen Zwischenraeumen effektiver gespeichert werden. Das Setzen des Tabulators erfolgt durch Hinfuehren des Kursors in die gewuenschte Spalte und Eingabe von CONTROL-T (14H) gefolgt von T. Zum loeschen wird der Kursor in die gewuenschte Spalte gesetzt und anschliessend CONTROL-T gefolgt von SPACE eingegeben. Die implizit vereinbarten Tabulatoren befinden sich in jeder 8.Spalte, beginnend mit der aeusserst linken als 0. Das implizit bereinbarte Setzen der Tabulatoren kann durch das SET-Kommando oder auch mittels DEBUG-Kommando im OS veraendert werden.

Durch CON werden folgende Ein/Ausgabe-Aufrufe durchgefuehrt:

Kodierung Bedeutung

00H INITIALIZE
Die momentanen Daten werden in die zugewiesene ATTRIBUTE-Tabelle eingetragen (siehe OPEN).

02H ASSIGN
Es wird keine Aktion durchgefuehrt, Antwort ist OPERATION COMPLETE.

04H OPEN
Ist die Datenlaenge=0, dann wird keine Aktion durchgefuehrt. Sonst koennen maximal 20 Bytes von der ATTRIBUTE-Tabelle abgerufen werden:

| | |
|---|---|
| Type | 20H |
| Satzanzahl | 0 |
| Satzlaenge | 80H |
| Blocklaenge | 80H |
| Eigenschaften | 0 |
| Startadresse | 0 |
| Ladeadresse | 0 |
| Entstehungsdatum | momentanes Datum |

06H CLOSE
Es wird keine Aktion durchgefuehrt. Antwort ist OPERATION COMPLETE

0AH READ BINARY
Es werden Daten vom Terminal gesandt, wobei die Eingabe von CONTROL-D eine Markierung fuer END OF FILE (FFH) im Puffer erzeugt und die Eingabe beendet. Fuer die Datenlaenge wird die tatsaechliche Zeichenzahl eingetragen und die Paritaetsbits der Zeichen sind zurueckgesetzt.

0EH WRITE BINARY
Es werden Daten zum Terminal gesandt, bis die Markierung END OF FILE auftritt. In der Datenlaenge ist die tatsaechliche Zeichenzahl eingetragen.

10H WRITE LINE
Es werden solange Daten zum Terminal gesandt, bis ein Return auftritt. Return wird noch uebertragen und anschliessend der Schreibvorgang beendet.

44H ACTIVATE
Es wird keine Aktion durchgefuehrt, Antwort ist OPERATION COMPLETE.

42H WRITE STATUS
Von der angegebenen Datentransferadresse werden Daten zum CON-Statusbereich transferiert (siehe READ STATUS).

40H READ STATUS
Vom CON-Statusbereich werden Daten zu dem Speicherbereich transferiert, dessen Beginn

durch die Datentransferadresse festgelegt ist. Der Terminalstatus ist folgendermassen definiert:

| Byte | Beschreibung |
|---|---|
| 0 | FLAGBYTE<br>Bit0 Interne FLAG<br>Bit1 AUTO LINEFEED<br>0N=1 (implizit)<br>OFF=0<br>Bit2 ECHO<br>ON=1<br>OFF=0 (implizit)<br>Bit3 Eingabe Puffer (TIB)<br>VOLL=1 (implizit)<br>LEER=0<br>Bit 4 ECHO RETURN<br>OFF=1<br>ON=0 (implizit)<br>Bit5 ESC (ESCAPE) gueltig<br>ungueltig=1<br>gueltig=0 (implizit)<br>Bit6 Interne Flag<br>Bit7 VOLL/HALB-DUPLEX<br>HALB=1<br>VOLL=0 (implizit) |
| 1 | reserviert |
| 2 | TIB<br>enthaelt das letzte, vom seriellen Kanal eingegebene, noch nicht mit READ gelesene Zeihen |
| 3 | Kursoradresse |
| 4 | reserviert |
| 5...138 | Tabulatortabelle: 134 Positionen, um gesetzte Tabulatoren zu markieren (Wert ungleich 0) |

Zu PCON:
Der im U880-Softwaremonitor befindliche Treiber ist der einfachste Ein/Ausgabe-Treiber zum und vom Terminal. Einzelheiten sind in der Beschreibung des U880-Softwaremonitors im P8000 Dokumentationsband Einfuehrung in die Software des Geraetesystems P8000 nachzulesen.

Zu FLOPPY:
Der im U880-Softwaremonitor befindliche Floppy-Disk-Treiber wird von NDOS als einfachster Zugriff zu den Diskettenlaufwerken benutzt. Einzelheiten sind ebenfalls in der Beschreibung des U880-Softwaremonitors im P8000 Dokumentationsband Einfuehrung in die Software des Geraetesystems P8000 nachzulesen

# 7. Hinweise zur Erstellung von Anwenderprogrammen

## 7.1. Speicherzuordnung

Das UDOS-Betriebssystem belegt folgende Speicherbereiche:

| Programmteil | Speicherbereich |
|---|---|
| U880-Softwaremonitor | 0 - BFF |
| RAM-Bereich des U880-Softwaremonitors | 000 - FFF |
| UDOS-Operaionssysem (OS) | 1000 - 20FF |
| UDOS-Dateiverwaltungs-system (NDOS) | 2580 - 3FFF |
| RAM-Bereich des NDOS | FB00 - FFFF |
| Konsolentreiber | 2100 - 2500 |
| Anwenderbereich | 4000 - FAFF |

Die UDOS-Kommandos sind wie alle anderen UDOS-Prozeduren als Unterprogramme geschrieben, d.h., beim Start des Programms wird die Rueckkehradresse zum System im Stack abgelegt. Die UDOS-Kommandoprozeduren laufen grundsaetzlich im untersten dem Anwender zur Verfuegung stehenden Speicherbereich. Allgemein ist fuer die Programmdurchfuehrung mittels UDOS notwendig:

- das Programm muss ladbar sein, d.h., der benoetigte Speicherbereich ist frei
- ein Stackbereich, d.h. der fuer den Stack notwendige Speicher, muss vorhanden sein

Ist die entsprechende Prozedur geladen, belegt sie den zugehoerigen Speicherbereich. Dieser wird mit Hilfe des Belegungsplans vor Doppelbelegung geschuetzt. Der Belegungsplan kann mit DISPLAY ausgelesen und mit ALLOCATE bzw. DEALLOATE geaendert werden.
Nach der Identifizierung des Dateinamens einer Prozedur durch den Kommandozeichenketten-Interpreter wird die Prozedur geladen. Dem Zeiger auf den Eingabepufferspeicher (INPTR) wird die Adresse des Begrenzungszeichens nach dem Programmnamen zugeordnet. Die dem Begrenzungszeichen evtl. folgende Parameterzeichenkette kann bis einschliesslich des naechsten Begrenzers (Wagenruecklauf oder ';') geaendert werden. Diese Adresse wird vor der durchfuehrung der Prozedur in den Anwenderstack, gefolgt von der Systemrueckkehradresse, gerettet. Insgesamt werden von UDOS vor dem Start eines UDOS-Programms folgende Aktivitaeten vorgenommen:

- Rettung des System-Stackpointers
- Zuordnung eines Anwenderstackbereiches
- Retten der Adresse der Parameterzeichenkette
- Retten der Rueckkehradresse zu UDOS

Die fuer ein Anwenderprogramm zur Verfuegung gestellte Stackgroesse kann mittels LINK oder IMAGE ueber deren Option ST=nn bestimmt werden. Automatische Zuordnung sind

immer 80H, was bei der Arbeit mit UDOS-Programmen zu beachten ist. Aus Anwenderprogrammen zu uebermittelnde Fehlermeldungen koennen an UDOS durch die Variable ERCODE uebergeben werden. Das Setzen von Bit 6 weist auf den Fehler hin, und entsprechend UDOS-Fehlerliste wird der Text zum Fehler ausgegeben oder nicht.

## 7.2. Systemaufrufe

Systemaufrufe aus UDOS-Prozeduren sind mittels Unterprogrammaufruf (CALL) zum Systemeintrittspunkt SYSTEM (Adresse 1003H) wie ein Ein/Ausgabe-Aufruf zu realisieren. Wie bereits in Abschn. 6.2. beschrieben, muss im Indexregister IY die Startadresse eines Vektors mit folgendem Format uebergeben werden:

| Byte | Inhalt |
|---|---|
| 1 | Null: bedeutet einen Systemaufruf anstelle eines Ein/Ausgabe-Aufrufs |
| 2 | nicht verwendet |
| 2-4 | Adresse zur Zeichenkette des Befehls: Adresse zum ersten Zeichen der Kommandoeingabe, Zeichenkette kann beliebig lang sein, Abschluss muss mit Return erfolgen, Format entspricht dem Eingabeformat ueber das Terminal |
| 5-8 | nicht verwendet |
| 9-10 | Adresse der Fehlerroutine: Routine wird von UDOS beim Auftreten von Fehlern aufgerufen, wenn Null, wird sie ignoriert |
| 11 | Fertigstellungskode: wird von UDOS erzeugt oder als Fehlerkode an UDOS gemeldet, wenn Bit6=1, so ist ein Fehler entstanden |

Zur Beachtung:
Werden Systemaufrufe durch externe Prozeduren ausgefuehrt, die ihrerseits wieder durch externe Prozeduren aufgerufen werden, ist der momentane Status des Speicherbelegungsplanes zu retten, damit die Kontrolle ueber den Platzbedarf und die Belegung nicht verloren geht. Dieser Plan wird auf 44H Bytes im Anwenderstack gespeichert. Bei Anwenderprogrammen mit dieser Moeglichkeit, ist eine entsprechende Stackgroesse vorzusehen.

## 7.3. Interrupt-Status

Fuer die Initialisierung und zur Arbeit mit dem Betriebssystem wird der Interrupt-Mode 2 eingestellt und das I-Register auf die Basisadresse des Interrupt-Vektors, fuer UDOS ist es 0FH, gesetzt. Der Interrupt wird erlaubt. Soll in Treibern oder Anwenderprogrammen mit Interrupt gearbeitet werden, so ist das ohne Einschraenkungen moeglich, wenn die im I-Register stehende Basisadresse des Interrupt-Vektors nicht veraendert wird. Als Tabelle fuer die Interrupt-Adressen ist freier Speicher im RAM-Bereich des U880-

Softwaremonitors (ab Adresse F40H, 96 Bytes) zu verwenden. Ist in Anwenderprogrammen eine Aenderung des Interrupt-Modus oder der Basisadresse im I-Register unumgaenglich, so muss nach dem letzten Diskettenzugriff 2s mit der Aenderung gewartet werden. Weiterhin muessen die gueltigen Bedingungen unter allen Umstaenden wieder restauriert werden, bevor ein Systemaufruf mit Terminalaktivitaet oder aber mit Diskettenaktivitaet erfolgt. wird die rechtzeitige Wiederherstellung des alten Zustandes versaeumt, kommt es unweigerlich zu einem Absturz des Systems.

## 7.4. Ein/Ausgabe-Definition der logischen Funktionen

Logische Funktionen koennen zwischen 0 und 20 definiert werden. Alle logischen Funktionen, mit Ausnahme von 0, kann der Anwender neu definieren. Die Funktion 0 wird in jedem Fall von UDOS benoetigt und ist nicht zugreifbar. Die Funktionen 1,2 und 3 sind von UDOS als Terminaleingabe, Terminalausgabe und Druckerausgabe vordefiniert. Die Umdefinition dieser Funktionen kann ein abnormales Verhalten des Systems beim Ruecksprung zu UDOS hervorrufen. Die Funktionen 4-20 sind als Master-Treiber initialisiert, koennen aber umdefiniert oder fuer 1 bis 2 aequivalente Treiber vorgesehen werden.

Anhang A

UDOS-Kommandos und Syntax

| | |
|---|---|
| ACTIVATE | $Treibername [Adresse] |
| Allocate | untere Adresse obere Adresse Blockgroesse |
| Brief | |
| CAT | [String] [T=Type] [P=Properties] [D=Drive] [F=Format] [L=Listingdisposition] [DATE rel Datum] [CDATE rel Datum] |
| Close | Unit \| * |
| COMPARE | datei1 datei2 |
| COPY | datei1 datei2 ([RL=Rekordlaenge] [T=Type] [A \| U \| O]) |
| COPY.DISK | |
| COPYSD | dateiname |
| DATE | [JJMMTT] |
| DEACTIVATE | $Treibername |
| DEAllocate | Blockadresse Blocklaenge |
| Debug | |
| DEFINE | log. Funktion Dateiname \| log. Funktion Treibername \| log. Funktion * \| * [A] \| [O] \| [U] |
| DELETE | [String] [T=Type] [P=Properties] [D=Drive] [Q=Frage] [DATE rel Datum] [CDATE rel Datum] |
| DISPLAY | |
| DO | Kommandodaei [Parameterliste] |
| DUMP | Dateiname [m] [n] |
| ECHO | Zeichenkette |
| ERROR | Fehlerkode |
| ERRORS | |
| EXTRACT | Dateiname |
| Force | Kommandokette |
| FORMAT | [S] [D=Drive] [ID=Diskettenname] [Q=Frage] |

| | |
|---|---|
| IMAGE | Dateiname erster Speicherplatz letzter Speicherplatt [E=Eintrittspunkt] [RL=Satzlaenge] [ST=Stackgroesse] |
| Initialize | [$Treibername] [Parameterliste] |
| LADT | |
| MASTER | [$Treibername] |
| MOVE | [String] [T=Type] [P=Properties] [F=Format] [D=Zieltreiber] [S=Quellentreiber] [L=Ausgabetreiber [Q=Frage] [DATE rel Datum] [CDATE rel Datum] |
| PAUSE | |
| RELease | |
| RENAME | [alter Dateiname neuer Dateiname] \| [$FLOPPY:Drive ID='neuer Diskettenname'] |
| RESTORE_TABS | Dateiname |
| SAVE_TABS | Dateiname |
| SET | [CHRDEL=c] [LINDEL=c] [NULLCT=n] [ECHO ON] \| [ECHO OFF] [AUTOLF ON] \| [AUTOLF OFF] [TABSIZE=n] [PROPERTIES OF Dateiname TO Props] [TYPE OF Dateiname TO Type] [SUBTYPE OF Dateiname TO Subtype] [LOW_ADDRESS OF Dateiname TO nn] [HIGH_ADDRESS OF Dateiname TO nn] [STACK_SIZE OF Dateiname TO nn] [BYTE_COUNT OF Dateiname TO nn] [ENTRY_POINT OF Dateiname TO nn] |
| SETFD | [F0F1F2F3][Sd] |
| SETLP | [lines] [cols] [indent] |
| STATUS | [Drive] |
| Verbose | |
| Xeq | [*] \| [nn] [parameterliste] |
| : | Ausdruck |

Anhang B

UDOS-Systemadressen

| Symbol | Adresse | |
|---|---|---|
| BRKFLG | 0FC4 | |
| BRKRTN | 0FC5 | |
| CHRBEL | 0FC3 | |
| CONIBF | 0D8A | |
| CONIVC | 0ECE | |
| CONOBF | 0D04 | |
| CONOVC | 0EC3 | |
| DATE | 0FA2 | |
| DECODE | 0FB4 | |
| INITIALIZATION COMMAND AREA | 14D9 | |
| EXTRET | 0FB5 | |
| FDCONF | 0EEB | |
| INPTR | 0FBB | |
| INTERRUPT VECTOR | 0F40 | (fuer Anwenderprogramme) |
| LINDEL | 0FC2 | |
| MEMMGR | 1009 | |
| NULLCT | 0FBF | |
| OUTPTR | 0FBD | |
| DEBUG | 0BFA | |
| PROMPT | 0FC1 | |
| PCON | 0BEE | |
| SYSFLG | 100E | |
| SYSTEM | 1003 | |
| SYSTEM REENTRY POINT | 1000 | |

Die Adressen fuer den Sprungverteiler im U880-Softwaremonitor siehe Beschreibung des U880-Softwaremonitors im P8000 Dokumentationsband Einfuehrung in die Software des Geraetesystems P8000.

Anhang C

Nutzung von MEMMGR

Dieser Anhang beschreibt die Registerinhalte vor und nach einem MEMMGR-Aufruf. In Anhang B ist die Adresse von MEMMGR aufgefuehrt.

ALLOCATE

Vor dem MEMMGR CALL:
<A>=0 (Kode fuer allocate)
<HL>=untere Adresse
<DE>=obere Adresse
<BC>=Anzahl der benoetigten Bytes

Nach dem MEMMGR CALL:
<A>=80 (Kode fuer vollstaendige Durchfuehrung)
<HL>=untere Adresse des Bereiches
<DE>=obere Adresse des Bereiches
<BC>=Anzahl der Bytes des Bereiches
<A>=4A (Kode fuer nicht belegbaren Speicher)
<HL>=Beginn des unteren Bereiches innerhalb der Grenzen
<BC>=Anzahl der Bytes
(Wenn <BC>=0, dann ist <HL> undefiniert)

DEALLOCATE

Vor dem MEMMGR CALL:
<A>=1 (Kode fuer deallocate)
<HL>=Anfangsadresse des Bereiches
<BC>=Anzahl der Bytes

Nach dem MEMMGR CALL:
<A>=80 (Kode fuer vollstaendige Durchfuehrung)
<A>=43 (<HL> war nicht an der Grenze eines Bereiches)
<A>=44 (nicht alle Bloecke des Bereiches waren belegt)

Anhang D

UDOS-Fehlerliste

| Fehlerkode (hex.) | Beschreibung |
|---|---|
| C1 | Ungueltige Operation |
| C2 | Angesprochenes Geraet nicht bereit |
| C3 | Schreibgeschuetzt |
| C4 | Sektorfehler |
| C5 | Spurfehler |
| C6 | Datentransferfehler |
| C7 | Datei nicht auffindbar |
| C8 | --- |
| C9 | End of File - Fehler |
| CA | Zeigerkontrollenfehler |
| CB | Datei ist nicht eroeffnet |
| CC | Logische Funktion ist bereits aktiv |
| CD | Zuweisungspufferspeicher ist voll |
| CE | Ungueltige Diskettenstation |
| CF | Tabelle der logischen Funktionen ist voll |
| D0 | Datei-Duplizierung |
| D1 | Disketten-Identifizierungsfehler |
| D2 | Ungueltige Merkmale |
| D3 | Diskette ist voll |
| D4 | Datei ist nicht in entsprechendem Directory enthalten |
| D5 | Dateianfang ist falsch |
| D6 | Datei ist bereits geoeffnet |
| D7 | Ungueltige Umbenennung |
| D8 | Datei ist gesperrt (Versuch der Merkmalaenderung) |
| D9 | Ungueltiger Aufruf zur Eroeffnung |
| DA | Zuwenig Speicherplatz fuer den Belegungsplan |

Warnungen (Bit 6 nicht gesetzt)

| | |
|---|---|
| 81 | Formatfehler im Directory |
| 82 | SCRATCH-Datei eroeffnet |
| 83 | Dateiname ist zu lang (wird abgeschnitten) |
| 84 | Liste der Merkmale ist zu lang (wird abgeschnitten) |

---
Hinweise des Lesers zu diesem Dokumentationsband
---

Wir sind staendig bemueht, unsere Unterlagen auf einem qualitativ hochwertigen Stand zu halten. Sollten Sie deshalb Hinweise zur Verbesserung dieses Dokumentationsbandes bzw. zur Beseitigung von Fehlern haben, so bitten wir Sie, diesen Fragebogen auszufuellen und an uns zurueckzusenden.

Titel des Dokumentationsbandes: UDOS-Systemhandbuch

Ihr Name / Tel.-Nr.:

Name und Anschrift des Betriebes:

Genuegt diese Dokumentation Ihren Anspruechen? ja / nein
Falls nein, warum nicht?

Was wuerde diese Dokumentation verbessern?

Sonstige Hinweise:

Fehler innerhalb dieser Dokumentation:

Unsere Anschrift: Kombinat VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"
Abteilung Basissoftware
Hoffmannstrasse 15-26
BERLIN
1193

Kombinat VEB
**ELEKTRO-APPARATE-WERKE**
BERLIN-TREPTOW >FRIEDRICH EBERT<
Hoffmannstraße 15-26, Berlin, DDR-1193
011 2263 eaw 011 2264 eaw

Die Angaben über technische Daten entsprechen dem bei Redaktionsschluß vorliegenden Stand. Änderungen im Sinne der technischen Weiterentwicklung behalten wir uns vor.

Ausgabe August 1986