Das DOS Buch

NEWDOS, GDOS, Colour-DOS
für TRS-80 Model I & III
Genie I, II, III, IIs & IIIs
Colour-Genie

von Hartmut Grosser

RÖCKRATH
MIKROCOMPUTER

Hartmut Grosser:

Das DOS-Buch für TRS-80, Genie und Colour Genie

ISBN 3-925074-10-4

1. Auflage, September 1985

RÖCKRATH
MICROCOMPUTER

Noppiusstraße 19
5100 Aachen

Herstellung: Fotodruck J. Mainz GmbH, Aachen

Hinweis:

Alle in diesem Buch wiedergegebenen Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt und dürfen nicht gewerblich genutzt werden.

Alle Angaben in diesem Buch wurden mit größtmöglicher Sorgfalt erarbeitet und zusammengestellt. Trotzdem lassen sich Fehler nicht völlig ausschließen. Der Verlag und der Autor können deshalb weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen.

Für die Mitteilung eventueller Fehler sind Autor und Verlag jederzeit dankbar.

Inhaltsverzeichnis Seite

Inhaltsverzeichnis (Fortsetzung) Seite

Allgemeines

Alle Zahlenangaben in diesem Buch sind, wenn sie nicht von einem 'D' gefolgt werden, HEXADEZIMAL. Eine Ausnahme bilden Zeitangaben, Frequenzen und Werte, bei denen es sich um eine Anzahl handelt, die stets DEZIMAL angegeben sind.

'*' in einem Disassemblierten Listing kennzeichnet Stellen im Programm, die vom DOS selbst geändert werden können.

'>' in einem Disassemblierten Listing kennzeichnet Stellen im Programm, die unter GDOS oder NEWDOS/80 für TRS-80 Model III gegenüber NEWDOS/80 für Model I abweichen.

'(' in einem Disassemblierten Listing kennzeichnet Stellen im Programm, wo mitten in einen Befehl hineingesprungen wird, oder wo aus einem anderen Grund ein Befehls-Byte mehrere verschiedene Bedeutungen haben kann.

Alle wichtigen Unterprogramme in den Disassemblierten Listings werden durch einen Block in der folgenden Form angeführt, um schnell und übersichtlich die wesentlichen Eigenschaften des jeweiligen Unterprogrammes vorzustellen:

```
*****************************************************
* Name:       max. 6 Zeichen                        *
* Funktion:   kurze Beschreibung                    *
* Input:      Eingabeparameter                      *
* Verändert:  veränderte Register                   *
* Output:     Ausgabeparameter                      *
*****************************************************
```

Die Namen der Unterprogramme wurden so gewählt, daß sie in jedem Assembler als Label verwendet werden können.

Als Eingabe- und Ausgabeparameter zählen nicht nur die Register inkl. Stackpointer, sondern ggf. auch bestimmte RAM-Adressen und Flags, wenn ihr Inhalt zur Steuerung oder als Funktion eines Unterprogrammes wichtig ist.

Bei den veränderten Registern sind auch die Index-Register und der Zweit-Registersatz aufgeführt, sofern sie ihren Wert verändern.

Abkürzungen

#	Nummer, Anzahl
->	bewirkt
CRC	Cyclic Redundancy Check, eine 16-Bit Checksumme
CRT	Cathode Ray Tube (Bildröhre, Bildschirm)
DB	DOS-Befehl
DCB	Device Control Block (Ein/Ausgabe Steuerblock)
DD	Double Density (Doppelte Schreibdichte)
DDGA	Default Directory GRAN Allocation (GDOS: AEIV)
DDSL	Default Directory Starting LUMP (GDOS: SBIV)
DEC	Directory Entry Code, siehe Kapitel 7.2.3
DIR	Directory (Inhaltsverzeichnis)
DIV	Dividieren (/)
DOS	Disk Operating System
DRQ	Data Request, wird vom FDC erzeugt
DS	Double Sided (Doppelseitige Diskette)
DSL	Directory Starting LUMP (LUMP# vom Beginn des DIR)
FCB	File Control Block, siehe Kapitel 7.3
FDC	Floppy Disk Controller, siehe Kapitel 1
FDE	File Directory Entry, siehe Kapitel 7.2.3
FPDE	File Primary Directory Entry, siehe Kapitel 7.2.3.1
FXDE	File Extended Directory Entry, siehe Kapitel 7.2.3.2
GAT	GRAN Allocation Tabelle, siehe Kapitel 7.2.1
GPL	GRANS pro LUMP (GDOS: EIB)
GRAN	(Körnchen), organisatorische Einheit von 5 Sectoren
HIT	Hash Index Tabelle, siehe Kapitel 7.2.2
INTRQ	Interrupt Request, wird vom FDC erzeugt
LRL	Logische Recordlänge (1-256D Bytes)
LSB	Least Significant (niederwertigstes) Bit oder Byte
LUMP	(Klumpen), organisatorische Einheit von 2 - 8 GRANS
ms	Millisekunde
MSB	Most Significant (höchstwertigstes) Bit oder Byte
MULT	Multiplizieren (*)
NMI	Nicht Maskierbarer Interrupt (bewirkt RST 66H)
RAM	Random Access Memory (Schreib/Lesespeicher)
RBA	Relative Byte Adresse (beginnend ab 0)
RCB	ROUTE Control Block, Steuerblock für ROUTE-Befehl
ROM	Read Only Memory (Nur Lesespeicher)
RTC	Real Time Clock, bewirkt 40D Interrupts pro Sekunde
SD	Single Density (Einfache Schreibdichte)
SP	Stackpointer
SPT	Sectors pro Track (GDOS: SEK)
SS	Single Sided (Einseitige Diskette)
TC	Track Count (GDOS: SP)
TD	Type of Drive
TI	Type of Interface
TSR	Track Stepping Rate (GDOS: SWZ)
UP	Unterprogramm
us	Microsekunde

Kapitel 1: Der Floppy Disk Controller (FDC)

Computer: TRS-80 Model I + III
Genie I,II,IIs,III,IIIs
Colour Genie

Wer ins Innere eines Disketten-Betriebssystems (DOS) vorzudringen versucht, braucht grundlegende Kenntnisse von der Hardware und Programmierung des Floppy Disk Controllers (FDC). Im Anhang sind die ausführlichen Datenblätter darüber enthalten, allerdings in Englisch. Das Wesentliche ist daher hier noch einmal zusammengestellt.

Der FDC, ein hochkomplizierter Integrierter Schaltkreis, übernimmt die Kommunikation zwischen dem Computer - genauer gesagt der CPU - und dem Disketten-Laufwerk (Floppy). Je nach Computer finden folgende FDC-Typen Verwendung:

TRS-80 Model I	FD 1771 *)
TRS-80 Model III	FD 1793
Genie I, II	FD 1771 oder FD 1791 *)
Genie III	FD 1771 und FD 1791
Genie IIs, IIIs	FD 2791
Colour Genie	FD 1791

*) Für diese Modelle gibt es Erweiterungs-Platinen mit dem FD 1791.

Der FD 1791 ist eine Erweiterung des FD 1771 und kann zusätzlich auch Disketten in Double-Density (Doppelter Schreibdichte) und Double-Sided (doppelseitig) verarbeiten.

Der FD 1793 leistet das gleiche wie der FD 1791, besitzt jedoch keinen invertierten, sondern einen normalen Datenbus.

Der FD 2791 arbeitet genau wie der FD 1791, hat jedoch einen analogen Data-Separator eingebaut.

Die Programmierung all dieser FDC's ist jedoch bis auf kleine Abweichungen so gut wie identisch, auf die im Text jeweils hingewiesen wird.

1.1 Kommunikation mit den Laufwerken

Die Verbindung zwischen FDC und Floppy erfolgt durch ein 34-poliges Kabel. Im Anhang befindet sich ein Schaltplan der FDC-Einheit im Expansion-Interface des TRS-80 Model I, aus dem die Bedeutung der einzelnen Leitungen hervorgeht.

Je nach verwendetem Computer und Floppy-Laufwerk sind 35, 40 oder 80 Tracks (Spuren) pro Diskettenseite möglich. Die maximale Anzahl der Sectoren pro Track und Seite ergibt sich aus folgender Tabelle:

5.25 Zoll, Single Density:	10 Sectoren
5.25 Zoll, Double Density:	18 Sectoren
8 Zoll, Single Density:	17 Sectoren
8 Zoll, Double Density:	26 Sectoren

1.2 Kommunikation mit der CPU

Die Verbindung zwischen FDC und CPU erfolgt, je nach Computer-Typ, durch

TRS-80 Model I	Adressen 37E0 - 37EF
TRS-80 Model III	Ports F0 - F7
Genie I,II,IIs,III,IIIs	Adressen 37E0 - 37EF
Colour Genie	Adressen FFE0 - FFEF

wobei die Adressen beim Colour Genie gegenüber den Adressen von TRS-80 Model I und Genie I,II,IIs,III,IIIs lediglich um C800 (Hex) verschoben sind, ansonsten aber exakt die gleiche Bedeutung haben.

1.2.1 Adressierung des Floppy Disk Controllers

Die folgende Tabelle gibt die Verwendung der jeweiligen Adressen bzw. Ports an:

Adresse / Port	Verwendung
37E0-37E3 --	Interrupt-Status / Drive-Select Eingabe: Interrupt-Quelle Bit 7: Interrupt durch RTC (Real-Time-Clock) Bit 6: Interrupt durch FDC Ausgabe: Motor an & Drive-Select für Bit 3: Drive 3 Bit 2: Drive 2 Bit 1: Drive 1 Bit 0: Drive 0

Adresse /	Port	Verwendung
37EC	F0	FDC Status/Kommando-Register Eingabe: Status-Register Ausgabe: Kommando-Register
37ED	F1	FDC Track-Register Eingabe: aktuelle Track # Ausgabe: aktuelle Track #
37EE	F2	FDC Sector-Register Eingabe: aktuelle Sector # Ausgabe: aktuelle Sector #
37EF	F3	FDC Daten-Register Eingabe: gelesenes Zeichen Ausgabe: zu schreibendes Zeichen
----	F4-F7	Drive-Select (nur Ausgabe) Bit 7: Double Density (1) oder Single Density (0) Bit 6: wenn gesetzt, wird ein CPU-Wait-Status ausgelöst, der durch Disk-INTRQ, Disk-DRQ, RESET oder spätestens nach 1 ms gelöscht wird Bit 5: Write-Precompensation enable (1) oder disable (0) Bit 4: Disketten-Seitenauswahl Seite 1 (1) oder Seite 0 (0) Bit 3: Motor an & Select Drive 3 Bit 2: Motor an & Select Drive 2 Bit 1: Motor an & Select Drive 1 Bit 0: Motor an & Select Drive 0

Bei Genie I,II,IIs und TRS-80 Model I mit Erweiterung durch FD 1791, beim Colour Genie sowie bei Genie III,IIIs haben einige der oben beschriebenen Adressen noch eine zusätzliche Verwendung (gilt nur für Ausgabe auf diese Adressen, nicht für Eingabe):

37E0-37E3	Bit 4: nur für Genie III,IIIs und Colour Genie: Auswahl der Disketten-Seite Seite 1 (1) oder Seite 0 (0)
	Bit 3: nur für TRS-80 Model I + Genie I,II,IIs: Auswahl der Disketten-Seite Seite 1 (1) oder Seite 0 (0)
37EC	Bit 0: Double (1) oder Single (0) Density gilt nur, wenn Bit 7 bis Bit 3 alle gesetzt sind:
37EE	Bit 6: 8 Zoll (1) oder 5.25 Zoll (0) gilt nur, wenn Bit 7 gesetzt ist:

1.3 Die Register des Floppy Disk Controllers

Die Register des FDC haben folgende Bedeutung:

1.3.1 Das Status-Register

Dieses 8-Bit-Register enthält den Zustand des FDC. Die Bedeutung der einzelnen Bits hängt vom gerade ausgeführten Kommando ab. Wenn ein Kommando ins Kommando-Register geschrieben wird, setzt der FDC das Busy-Bit (Bit 0) und die übrigen Bits werden aktualisiert bzw. gelöscht. Dieses Register kann von der CPU nur gelesen, aber nicht beschrieben werden.

1.3.2 Das Kommando-Register

Dieses 8-Bit-Register enthält jeweils das Kommando, welches gerade ausgeführt wird. Dieses Register kann von der CPU nicht gelesen, sondern nur beschrieben werden. Letzteres sollte nicht geschehen, solange der FDC busy (beschäftigt) ist, außer durch ein FORCE-INTERRUPT-Kommando.
Nachdem ein Kommando abgearbeitet ist, wird ein Interrupt ausgelöst.

1.3.3 Das Track-Register

Dieses 8-Bit-Register enthält die aktuelle Track# des Schreib/Lese-Kopfes. Es wird vom FDC jeweils entsprechend erhöht bzw. erniedrigt, wenn der Schreib/Lese-Kopf nach innen (Richtung Track 79) bzw. nach außen (Richtung Track 0) bewegt wird. Dieses Register kann von der CPU sowohl gelesen als auch beschrieben werden. Letzteres sollte nicht geschehen, solange der FDC busy ist.

1.3.4 Das Sector-Register

Dieses 8-Bit-Register enthält jeweils die Nummer des gewünschten Sectors. Dieses Register kann von der CPU sowohl gelesen als auch beschrieben werden. Letzteres sollte nicht geschehen, solange der FDC busy ist.

1.3.5 Das Daten-Register

Dieses 8-Bit-Register arbeitet als Zwischenspeicher und enthält bei READ-Kommandos die von der Diskette gelesenen Daten bzw. bei WRITE-Kommandos die auf Diskette zu schreibenden Daten. Beim SEEK-Kommando muß in diesem Register die gewünschte Track# abgelegt werden. Dieses Register kann von der CPU sowohl gelesen als auch beschrieben werden.

1.4 Die Programmierung des Floppy Disk Controllers

Im folgenden werden die wichtigsten Kommandos des FDC erklärt:

1.4.1 Das RESTORE-Kommando

Kommando-Format: Bits: 7 6 5 4 3 2 1 0
0 0 0 0 H V R R

H: Schreib/Lese-Kopf laden (ja/nein)
wenn gesetzt, wird bei Beginn dieses Kommandos der Schreib/Lese-Kopf im Floppy-Laufwerk an die Oberfläche der eingelegten Diskette geladen.

Anmerkung: In allen hier beschriebenen Computern ist die Hardware so ausgelegt, daß der Schreib/Lese-Kopf bereits dann geladen wird, wenn durch einen Drive-Select der Motor angeschaltet wird.

V: Track# vergleichen (ja/nein)
wenn gesetzt, wird am Ende dieses Kommandos die Track#, über der der Schreib/Lese-Kopf steht, von der Diskette gelesen und mit dem Inhalt des Track-Registers des FDC verglichen. Wenn diese nicht übereinstimmen, wird Bit 4 im Status-Register (Seek Error) gesetzt.

R: Geschwindigkeit des Schreib/Lese-Kopfes
Diese beiden Bits geben die Geschwindigkeit an, mit der der Schreib/Lese-Kopf von Track zu Track bewegt werden soll. Die Geschwindigkeit hängt von dem verwendeten FDC-Typ und von der Diskettengröße (5.25 Zoll, 8 Zoll) ab:

	FD 1771		FD 1791	
R R	8 Zoll	5.25 Zoll	8 Zoll	5.25 Zoll
0 0	6 ms	12 ms	3 ms	6 ms
0 1	6 ms	12 ms	6 ms	12 ms
1 0	10 ms	20 ms	10 ms	20 ms
1 1	20 ms	40 ms	15 ms	30 ms

Durch das RESTORE-Kommando wird der Schreib/Lese-Kopf solange nach außen (Richtung Track 0) bewegt, bis der FDC durch seinen Eingang TR00 die Position über Track 0 erkennt. Anschließend wird das Track-Register mit Null geladen und ein Interrupt ausgelöst.

Status-Register für RESTORE-Kommando

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Write Protected
ist gesetzt, wenn die eingelegte Diskette schreibgeschützt ist

Bit 5: Head Loaded
ist gesetzt, wenn der Schreib/Lese-Kopf geladen ist

Bit 4: Seek Error
ist gesetzt, wenn Track 0 nicht gefunden wurde

Bit 3: CRC Error
ist gesetzt, wenn ein Checksummen-Fehler auftrat

Bit 2: Track 0
ist gesetzt, wenn der Schreib/Lese-Kopf über Track 0 steht

Bit 1: Index
ist gesetzt, während das Index-Loch der eingelegten Diskette an der Lesevorrichtung vorbeikommt

Bit 0: Busy
ist gesetzt, solange das RESTORE-Kommando ausgeführt wird

1.4.2 Das SEEK-Kommando

Kommando-Format: Bits: 7 6 5 4 3 2 1 0
0 0 0 1 H V R R

H: Schreib/Lese-Kopf laden (ja/nein)
wenn gesetzt, wird bei Beginn dieses Kommandos der Schreib/Lese-Kopf im Floppy-Laufwerk an die Oberfläche der eingelegten Diskette geladen.

Anmerkung: In allen hier beschriebenen Computern ist die Hardware so ausgelegt, daß der Schreib/Lese-Kopf bereits dann geladen wird, wenn durch einen Drive-Select der Motor angeschaltet wird.

V: Track# vergleichen (ja/nein)
wenn gesetzt, wird am Ende dieses Kommandos die Track#, über der der Schreib/Lese-Kopf steht, von der Diskette gelesen und mit dem Inhalt des Track-Registers des FDC verglichen. Wenn diese nicht übereinstimmen, wird Bit 4 im Status-Register (Seek Error) gesetzt.

R: Geschwindigkeit des Schreib/Lese-Kopfes
Diese beiden Bits geben die Geschwindigkeit an, mit der der Schreib/Lese-Kopf von Track zu Track bewegt werden soll. Die Geschwindigkeit hängt von dem verwendeten FDC-Typ und von der Diskettengröße (5.25 Zoll, 8 Zoll) ab:

	FD 1771		FD 1791	
R R	8 Zoll	5.25 Zoll	8 Zoll	5.25 Zoll
0 0	6 ms	12 ms	3 ms	6 ms
0 1	6 ms	12 ms	6 ms	12 ms
1 0	10 ms	20 ms	10 ms	20 ms
1 1	20 ms	40 ms	15 ms	30 ms

Vor dem SEEK-Kommando müssen in das Track-Register die aktuelle Track-Position des Schreib/Lese-Kopfes und in das Daten-Register die gewünschte Track# geschrieben werden. Der FDC bewegt nun den Schreib/Lese-Kopf so weit wie nötig in die erforderliche Richtung, wobei das Track-Register entsprechend erhöht bzw. erniedrigt wird. Anschließend wird ein Interrupt ausgelöst.

Status-Register für SEEK-Kommando

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Write Protected
ist gesetzt, wenn die eingelegte Diskette schreibgeschützt ist

Bit 5: Head Loaded
ist gesetzt, wenn der Schreib/Lese-Kopf geladen ist

Bit 4: Seek Error
ist gesetzt, wenn der gewünschte Track nicht gefunden wurde

Bit 3: CRC Error
ist gesetzt, wenn ein Checksummen-Fehler auftrat

Bit 2: Track 0
ist gesetzt, wenn der Schreib/Lese-Kopf über Track 0 steht

Bit 1: Index
ist gesetzt, während das Index-Loch der eingelegten Diskette an der Lesevorrichtung vorbeikommt

Bit 0: Busy
ist gesetzt, solange das SEEK-Kommando ausgeführt wird

1.4.3 Das STEP-IN-Kommando

Kommando-Format: Bits: 7 6 5 4 3 2 1 0
0 1 0 U H V R R

U: Track-Register aktualisieren (ja/nein)
wenn gesetzt, wird nach jedem STEP-IN-Kommando das Track-Register um 1 erhöht.

H: Schreib/Lese-Kopf laden (ja/nein)
wenn gesetzt, wird bei Beginn dieses Kommandos der Schreib/Lese-Kopf im Floppy-Laufwerk an die Oberfläche der eingelegten Diskette geladen.

Anmerkung: In allen hier beschriebenen Computern ist die Hardware so ausgelegt, daß der Schreib/Lese-Kopf bereits dann geladen wird, wenn durch einen Drive-Select der Motor angeschaltet wird.

V: Track# vergleichen (ja/nein)
wenn gesetzt, wird am Ende dieses Kommandos die Track#, über der der Schreib/Lese-Kopf steht, von der Diskette gelesen und mit dem Inhalt des Track-Registers des FDC verglichen. Wenn diese nicht übereinstimmen, wird Bit 4 im Status-Register (Seek Error) gesetzt.

R: Geschwindigkeit des Schreib/Lese-Kopfes
Diese beiden Bits geben die Geschwindigkeit an, mit der der Schreib/Lese-Kopf von Track zu Track bewegt werden soll. Die Geschwindigkeit hängt von dem verwendeten FDC-Typ und von der Diskettengröße (5.25 Zoll, 8 Zoll) ab:

	FD 1771		FD 1791	
R R	8 Zoll	5.25 Zoll	8 Zoll	5.25 Zoll
0 0	6 ms	12 ms	3 ms	6 ms
0 1	6 ms	12 ms	6 ms	12 ms
1 0	10 ms	20 ms	10 ms	20 ms
1 1	20 ms	40 ms	15 ms	30 ms

Durch das STEP-IN-Kommando wird der Schreib/Lese-Kopf um einen Track nach innen (Richtung Track 79) bewegt. Anschließend wird ein Interrupt ausgelöst.

Status-Register für STEP-IN-Kommando

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Write Protected
ist gesetzt, wenn die eingelegte Diskette schreibgeschützt ist

Bit 5: Head Loaded
ist gesetzt, wenn der Schreib/Lese-Kopf geladen ist

Bit 4: Seek Error
ist gesetzt, wenn der gewünschte Track nicht gefunden wurde

Bit 3: CRC Error
ist gesetzt, wenn ein Checksummen-Fehler auftrat

Bit 2: Track 0
ist gesetzt, wenn der Schreib/Lese-Kopf über Track 0 steht

Bit 1: Index
ist gesetzt, während das Index-Loch der eingelegten Diskette an der Lesevorrichtung vorbeikommt

Bit 0: Busy
ist gesetzt, solange das STEP-IN-Kommando ausgeführt wird

1.4.4 Das READ-SECTOR-Kommando

Kommando-Formate:	Bits:	7	6	5	4	3	2	1	0	
		1	0	0	M	B	E	0	0	für FD 1771
		1	0	0	M	S	E	C	0	für FD 1791

M: Mehrere Sectoren lesen (ja/nein)
wenn gesetzt, wird nicht nur 1 Sector gelesen, sondern ab dem angegebenen Sector können in aufsteigender Reihenfolge alle Sectoren des betreffenden Tracks, bis zu dem mit der höchsten vorhandenen Sector#, von Diskette gelesen werden.

B: Format für Sector-Länge (nur FD 1771)
wenn gesetzt, handelt es sich um das IBM-Format und die Sector-Länge kann 128, 256, 512 oder 1024 Bytes betragen. Ansonsten sind 16, 32, 48, 64, ... 4080, 4096 Bytes pro Sector möglich.
Beim FD 1791 kann nur das IBM-Format verwendet werden.

E: Schreib/Lese-Kopf laden (ja/nein)
wenn gesetzt, muß zu Beginn dieses Kommandos der Schreib/Lese-Kopf im Floppy-Laufwerk zuerst noch an die Oberfläche der eingelegten Diskette geladen und anschließend 10 ms (FD 1771) bzw. 15 ms (FD 1791) gewartet werden (die Zeiten gelten für 8-Zoll-Laufwerke und müssen bei 5.25-Zoll-Laufwerken verdoppelt werden).

Anmerkung: In allen hier beschriebenen Computern ist die Hardware so ausgelegt, daß der Schreib/Lese-Kopf bereits dann geladen wird, wenn durch einen Drive-Select der Motor angeschaltet wird.

S: Auswahl der Disketten-Seite (nur FD 1791)
Es wird die durch S angegebene Seite der Diskette (Seite 0 bzw. Seite 1) ausgewählt.

C: Disketten-Seite prüfen (ja/nein) (nur FD 1791)
wenn gesetzt, wird von der durch Bit S ausgewählten Disketten-Seite die bei dem gewünschten Sector gespeicherte Seiten# gelesen und mit Bit S verglichen. Wenn diese nicht übereinstimmen, wird Bit 4 im Status-Register (Record Not Found) gesetzt.

Vor dem READ-SECTOR-Kommando muß der Schreib/Lese-Kopf über dem richtigen Track positioniert, sowie die gewünschte Track# ins Track-Register und die gewünschte Sector# ins Sector-Register geschrieben werden. Nun wird dieser Sector in dem betreffenden Track gesucht und (falls gefunden) von Diskette gelesen. Dabei erzeugt der FDC nach jedem gelesenen Byte einen DRQ (Data Request, Bit 1 im Status-Register), damit die CPU sich dieses Byte abholen kann.

Falls Bit M gesetzt ist, werden danach auch die restlichen Sectoren dieses Tracks von Diskette gelesen, wobei das Sector-Register jeweils entsprechend erhöht wird.

Zum Schluß dieses Kommandos wird der gefundene Data Adress Mark (siehe später) in Bit 5 und Bit 6 des Status-Registers (beim FD 1791 nur Bit 5) vermerkt. Anschließend wird ein Interrupt ausgelöst.

Der Data Adress Mark ist eine Kennzeichnung, über die jeder Sector verfügt und wird von allen hier besprochenen Disketten-Betriebssystemen dazu benutzt, um Directory-Sectoren von normalen Sectoren zu unterscheiden.

Es gibt 4 verschiedene Data Adress Mark's für den FD 1771 und 2 für den FD 1791:

Data Adress Mark (Hex):		F8	F9	FA	FB
Status-Register nach	Bit 6: *)	1	1	0	0
READ-SECTOR-Kommando	Bit 5:	1	0	1	0
WRITE-SECTOR-Kommando	Bit A1: *)	1	1	0	0
	Bit A0:	1	0	1	0
Möglich auf FD 1771:		X	X	X	X
FD 1791:		X	-	-	X
Verwendung: normale Sectoren:		-	-	-	X
Directory- in Single Density:		S)	-	X	-
-Sectoren: in Double Density:		X	-	-	-

*) Da der FD 1791 nur 2 Data Adress Mark's verarbeiten kann, entfällt Bit A1 im WRITE-SECTOR-Kommando sowie Bit 6 im Status-Register.

S) Data Adress Mark F8 wird nur dann für Directory-Sectoren in Single Density verwendet, wenn SYSTEM-Parameter "BN" in GDOS oder NEWDOS/80 Version 2 auf "Y" gesetzt ist.

Status-Register für READ-SECTOR-Kommando

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Data Adress Mark (nur FD 1771)
MSB vom Data Adress Mark des gelesenen Sectors

Bit 5: Data Adress Mark
LSB vom Data Adress Mark des gelesenen Sectors

Bit 4: Record Not Found
ist gesetzt, wenn der gewünschte Sector nicht gefunden wurde

Bit 3: CRC Error
ist gesetzt, wenn ein Checksummen-Fehler auftrat

Bit 2: Lost Data
ist gesetzt, wenn die CPU nicht schnell genug war, um alle gelesenen Bytes rechtzeitig abzuholen

Bit 1: DRQ (Data Request)
ist gesetzt, wenn die CPU das nächste von Diskette gelesene Byte aus dem Daten-Register des FDC abholen kann

Bit 0: Busy
ist gesetzt, solange das READ-SECTOR-Kommando ausgeführt wird

1.4.5 Das WRITE-SECTOR-Kommando

Kommando-Formate:

Bits:	7	6	5	4	3	2	1	0	
	1	0	1	M	B	E	A	A	für FD 1771
	1	0	1	M	S	E	C	A	für FD 1791

M: Mehrere Sectoren schreiben (ja/nein)
wenn gesetzt, wird nicht nur 1 Sector geschrieben, sondern ab dem angegebenen Sector können in aufsteigender Reihenfolge alle Sectoren des betreffenden Tracks, bis zu dem mit der höchsten vorhandenen Sector#, auf Diskette geschrieben werden.

B: Format für Sector-Länge (nur FD 1771)
wenn gesetzt, handelt es sich um das IBM-Format und die Sector-Länge kann 128, 256, 512 oder 1024 Bytes betragen. Ansonsten sind 16, 32, 48, 64, ... 4080, 4096 Bytes pro Sector möglich.
Beim FD 1791 kann nur das IBM-Format verwendet werden.

E: Schreib/Lese-Kopf laden (ja/nein)
wenn gesetzt, muß zu Beginn dieses Kommandos der Schreib/Lese-Kopf im Floppy-Laufwerk zuerst noch an die Oberfläche der eingelegten Diskette geladen und anschließend 10 ms (FD 1771) bzw. 15 ms (FD 1791) gewartet werden (die Zeiten gelten für 8-Zoll-Laufwerke und müssen bei 5.25-Zoll-Laufwerken verdoppelt werden).

Anmerkung: In allen hier beschriebenen Computern ist die Hardware so ausgelegt, daß der Schreib/Lese-Kopf bereits dann geladen wird, wenn durch einen Drive-Select der Motor angeschaltet wird.

S: Auswahl der Disketten-Seite (nur FD 1791)
Es wird die durch S angegebene Seite der Diskette (Seite 0 bzw. Seite 1) ausgewählt.

C: Disketten-Seite prüfen (ja/nein) (nur FD 1791)
wenn gesetzt, wird von der durch Bit S ausgewählten Disketten-Seite die bei dem gewünschten Sector gespeicherte Seiten# gelesen und mit Bit S verglichen. Wenn diese nicht übereinstimmen, wird Bit 4 im Status-Register (Record Not Found) gesetzt.

A: Auswahl des Data Adress Marks
Der Data Adress Mark ist eine Kennzeichnung, über die jeder Sector verfügt und wird von allen hier besprochenen Disketten-Betriebssystemen dazu benutzt, um Sectoren des Directory von normalen Sectoren zu unterscheiden.

Beim WRITE-SECTOR-Kommando muß jeweils angegeben werden, welcher Data Adress Mark benutzt werden soll. Nach dem READ-SECTOR-Kommando wird der jeweils gefundene Data Adress Mark dann im Status-Register des FDC vermerkt.

Es gibt 4 verschiedene Data Adress Mark's für den FD 1771 und 2 für den FD 1791:

Data Adress Mark (Hex):		F8	F9	FA	FB
WRITE-SECTOR-Kommando	Bit A1: *)	1	1	0	0
	Bit A0:	1	0	1	0
Status-Register nach READ-SECTOR-Kommando	Bit 6: *)	1	1	0	0
	Bit 5:	1	0	1	0
Möglich auf FD 1771:		X	X	X	X
FD 1791:		X	-	-	X
Verwendung: normale Sectoren:		-	-	-	X
Directory- -Sectoren:	in Single Density:	S)	-	X	-
	in Double Density:	X	-	-	-

*) Da der FD 1791 nur 2 Data Adress Mark's verarbeiten kann, entfällt Bit A1 im WRITE-SECTOR-Kommando sowie Bit 6 im Status-Register.

S) Data Adress Mark F8 wird nur dann für Sectoren des Directory in Single Density verwendet, wenn SYSTEM-Parameter "BN" in GDOS oder NEWDOS/80 Version 2 auf "Y" gesetzt ist.

Vor dem WRITE-SECTOR-Kommando muß der Schreib/Lese-Kopf über dem richtigen Track positioniert, sowie die gewünschte Track# ins Track-Register und die gewünschte Sector# ins Sector-Register geschrieben werden. Nun wird geprüft, ob dieser Sector in dem betreffenden Track überhaupt existiert und (falls ja) unter Verwendung des angegebenen Data Adress Marks neu auf Diskette geschrieben. Dabei erzeugt der FDC für jedes zu schreibende Byte einen DRQ (Data Request, Bit 1 im Status-Register), damit die CPU dieses Byte im Daten--Register des FDC bereitstellen kann.

Falls Bit M gesetzt ist, werden danach auch die restlichen Sectoren dieses Tracks auf Diskette geschrieben, wobei das Sector-Register jeweils entsprechend erhöht wird.

Anschließend wird ein Interrupt ausgelöst.

Status-Register für WRITE-SECTOR-Kommando

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Write Protected
ist gesetzt, wenn die eingelegte Diskette schreibgeschützt ist

Bit 5: Write Fault
ist gesetzt, wenn während des WRITE-SECTOR-Kommandos von dem Floppy-Laufwerk ein Fehler in der Schreib-Elektronik gemeldet wurde

Bit 4: Record Not Found
ist gesetzt, wenn der gewünschte Sector nicht gefunden wurde

Bit 3: CRC Error
ist gesetzt, wenn ein Checksummen-Fehler auftrat

Bit 2: Lost Data
ist gesetzt, wenn die CPU nicht schnell genug war, um alle zu schreibenden Bytes rechtzeitig bereitzustellen

Bit 1: DRQ (Data Request)
ist gesetzt, wenn die CPU das nächste auf Diskette zu schreibende Byte in das Daten-Register des FDC schreiben kann

Bit 0: Busy
ist gesetzt, solange das WRITE-SECTOR-Kommando ausgeführt wird

1.4.6 Das WRITE-TRACK-Kommando

Kommando-Format: Bits: 7 6 5 4 3 2 1 0
1 1 1 1 0 E 0 0

E: Schreib/Lese-Kopf laden (ja/nein)
wenn gesetzt, muß zu Beginn dieses Kommandos der Schreib/Lese-Kopf im Floppy-Laufwerk zuerst noch an die Oberfläche der eingelegten Diskette geladen und anschließend 10 ms (FD 1771) bzw. 15 ms (FD 1791) gewartet werden (die Zeiten gelten für 8-Zoll-Laufwerke und müssen bei 5.25-Zoll-Laufwerken verdoppelt werden).

Anmerkung: In allen hier beschriebenen Computern ist die Hardware so ausgelegt, daß der Schreib/Lese-Kopf bereits dann geladen wird, wenn durch einen Drive-Select der Motor angeschaltet wird.

Mit dem WRITE-TRACK-Kommando kann ein Track formatiert werden. Zuvor muß der Schreib/Lese-Kopf über dem gewünschten Track positioniert worden sein. Das Formatieren erledigt der FDC allerdings nicht von selbst, vielmehr muß die CPU alle Bytes, Steuer-Codes und Bit-Muster, aus denen sich der Track zusammensetzen soll, Byte für Byte an den FDC schicken, wofür der FDC jeweils einen DRQ (Data Request) erzeugt.

Der FDC beginnt mit dem Formatieren, wenn das Index-Loch der Diskette die Lesevorrichtung im Laufwerk erreicht und hört genau 1 Umdrehung später auf.

Beim Formatieren setzt sich jeder Track wie folgt zusammen:
1) GAP 1
2) für jeden Sector:
   a) GAP 2
   b) Sector-ID
   c) GAP 3
   d) Benutzer-Feld
3) GAP 4

GAP 1, GAP 2, GAP 3 und GAP 4
sind spezielle Bit-Muster, die es dem FDC erlauben, sich innerhalb eines Tracks zurechtzufinden. Ihr Format ist daher ziemlich genau festgelegt und hängt von folgenden Parametern ab:

- 8 Zoll oder 5.25 Zoll
- Single Density oder Double Density
- FD 1771 oder FD 1791
- Anzahl Bytes pro Sector
- Anzahl Sectoren pro Track

Die Länge von GAP 4 ist innerhalb bestimmter Grenzen variabel und hängt jeweils von der genauen Umdrehungsgeschwindigkeit der Diskette ab, die max. um 2% vom Sollwert abweichen darf.

Der Sector-ID
(ID = Identification) enthält alle notwendigen Angaben, um den nachfolgenden Sector zu spezifizieren und wird mit einem CRC (Cyclic Redundancy Check: eine 16-Bit Checksumme) abgeschlossen.

Ein Byte innerhalb des Sector-ID gibt die Länge des Sectors an und ist im IBM-Format folgendermaßen codiert:
00: 128 Bytes
01: 256 Bytes
02: 512 Bytes
03: 1024 Bytes

Das Benutzer-Feld
enthält den eigentlichen Sector einschließlich CRC. Zum Formatieren wird in Single Density der Wert E5H benutzt, in Double Density immer abwechselnd 6DH und B6H.

In GDOS und NEWDOS/80 werden Disketten folgendermaßen formatiert:

Density:	Single	Single	Double	Double
FDC-Typ:	FD 1771	FD 1771	FD 1791	FD 1791
Diskettengröße:	5.25 Zoll	8 Zoll	5.25 Zoll	8 Zoll
Sectoren/Track:	10D	17D	18D	26D
Dezimal/Hex:	D x H	D x H	D x H	D x H
GAP 1 (entfällt)	./.	./.	./.	./.
GAP 2	11 x FF 6 x 00	10 x FF 6 x 00	16 x 4E 8 x 00 3 x F5	54 x 4E 12 x 00 3 x F5
Sector-ID:				
ID-Adress-Mark	1 x FE	1 x FE	1 x FE	1 x FE
Track# (Dez)	(0-79)	(0-79)	(0-79)	(0-79)
Seiten#	(0-1)	(0-1)	(0-1)	(0-1)
Sector# (Dez)	(0-9)	(0-16)	(0-17)	(0-25)
Sector-Länge	1 x 01	1 x 01	1 x 01	1 x 01
CRC schreiben	1 x F7	1 x F7	1 x F7	1 x F7
GAP 3	11 x FF 6 x 00	11 x FF 6 x 00	22 x 4E 12 x 00 3 x F5	40 x 4E 12 x 00 3 x F5
Benutzer-Feld:				
Data Adress Mark	1 x FB	1 x FB	1 x FB	1 x FB
Sector-Inhalt	256 x E5	256 x E5	128 x 6DB6	128 x 6DB6
CRC schreiben	1 x F7	1 x F7	1 x F7	1 x F7
.				
.				
.				
GAP 4	63- 186 x FF	21- 228 x FF	185- 434 x 4E	68- 483 x 4E

Status-Register für WRITE-TRACK-Kommando

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Write Protected
ist gesetzt, wenn die eingelegte Diskette schreibgeschützt ist

Bit 5: Write Fault
ist gesetzt, wenn während des WRITE-TRACK-Kommandos von dem Floppy-Laufwerk ein Fehler in der Schreib-Elektronik gemeldet wurde

Bit 4: ./.

Bit 3: ./.

Bit 2: Lost Data
ist gesetzt, wenn die CPU nicht schnell genug war, um alle zu schreibenden Bytes rechtzeitig bereitzustellen

Bit 1: DRQ (Data Request)
ist gesetzt, wenn die CPU das nächste auf Diskette zu schreibende Byte in das Daten-Register des FDC schreiben kann

Bit 0: Busy
ist gesetzt, solange das WRITE-TRACK-Kommando ausgeführt wird

1.4.7 Das FORCE-INTERRUPT-Kommando

Kommando-Format: Bits: 7 6 5 4 3 2 1 0
1 1 0 1 0 0 0 0

Das FORCE-INTERRUPT-Kommando ist das einzige Kommando, das jederzeit ins Kommando-Register des FDC geschrieben werden darf - auch dann, wenn gerade ein anderes Kommando ausgeführt wird, welches dadurch sofort abgebrochen wird.

Hier soll nur die in obigem Format angegebene Spezialversion des FORCE-INTERRUPT-Kommandos erklärt werden (da nur sie in den hier beschriebenen Betriebssystemen vorkommt), die lediglich ein evtl. gerade auszuführendes Kommando abbricht und KEINEN Interrupt auslöst.

Status-Register für FORCE-INTERRUPT-Kommando

Wenn das FORCE-INTERRUPT-Kommando gegeben wird, während der FDC gerade ein anderes Kommando ausführt, so wird lediglich das Busy-Bit (Bit 0) zurückgesetzt und die übrigen Bits bleiben UNVERÄNDERT.

Wenn das FORCE-INTERRUPT-Kommando gegeben wird, solange der FDC gerade kein Kommando ausführt, so werden die Bits des Status-Registers gelöscht bzw. gemäß folgender Tabelle aktualisiert:

Bit 7: Not Ready
ist gesetzt, wenn der Floppy-Motor aus ist

Bit 6: Write Protected
ist gesetzt, wenn die eingelegte Diskette schreibgeschützt ist

Bit 5: Head Loaded
ist gesetzt, wenn der Schreib/Lese-Kopf geladen ist

Bit 4: ./.

Bit 3: ./.

Bit 2: Track 0
ist gesetzt, wenn der Schreib/Lese-Kopf über Track 0 steht

Bit 1: Index
ist gesetzt, während das Index-Loch der eingelegten Diskette an der Lesevorrichtung vorbeikommt

Bit 0: ./.

1.4.8 Übersicht über alle Kommandos

In folgender Tabelle sind alle Kommandos des FDC zusammengefaßt:

Kommando	Bits: 7	6	5	4	3	2	1	0	
Restore	0	0	0	0	H	V	R	R	
Seek	0	0	0	1	H	V	R	R	
Step	0	0	1	U	H	V	R	R	
Step In	0	1	0	U	H	V	R	R	
Step Out	0	1	1	U	H	V	R	R	
Read Sector	1	0	0	M	B	E	0	0	(nur FD 1771)
	1	0	0	M	S	E	C	0	(nur FD 1791)
Write Sector	1	0	1	M	B	E	A	A	(nur FD 1771)
	1	0	1	M	S	E	C	A	(nur FD 1791)
Read Adress	1	1	0	0	0	E	0	0	
Read Track	1	1	1	0	0	E	0	0	
Write Track	1	1	1	1	0	E	0	0	
Force Interrupt	1	1	0	1	I	I	I	I	

Die Erläuterungen zu den variablen Bits finden Sie bei den Beschreibungen der betreffenden Kommandos.

1.4.9 Übersicht über das Status-Register

In folgender Tabelle ist die Bedeutung der einzelnen Bits des Status-Registers in Abhängigkeit des ausgeführten Kommandos zusammengefaßt:

Bit	RESTORE SEEK STEP-IN	READ -SECTOR	WRITE -TRACK -SECTOR
7	Not Ready	Not Ready	Not Ready
6	Write Protected	Data Adress Mark	Write Protected
5	Head Loaded	Data Adress Mark	Write Fault
4	Seek Error	Record Not Found	Record Not Found
3	CRC Error	CRC Error	CRC Error
2	Track 0	Lost Data	Lost Data
1	Index	Data Request	Data Request
0	Busy	Busy	Busy

Genaue Erklärungen finden Sie bei den Beschreibungen der betreffenden Kommandos.

Kapitel 2: Initialisierung des DOS

Betriebssystem: NEWDOS/80 Version 2 für TRS-80 Model I

Dieses Kapitel beschreibt die komplette Initialisierung von NEWDOS/80 Version 2 auf TRS-80 Model I. Die Initialisierung anderer Betriebssysteme auf anderen Computern finden Sie im Kapitel 9.

Als erstes wird aus dem BASIC-ROM der sogenannte Urlader aufgerufen, der den BOOT-Sector von Diskette (Drive 0, Track 0, Sector 0) in Single Density ins RAM ab 4200H liest und startet.

Das Programm im BOOT-Sector bewirkt nun, daß SYS0/SYS, welches auf Disketten in Single Density ab Track 0, Sector 5 und auf Disketten in Double Density ab Track 1, Sector 5 stehen muß, ins RAM zwischen 400CH und 51DAH geladen und bei 4D01H gestartet wird.

SYS0 besteht aus mehreren Blocks, die jeweils mit einem Steuer-Code beginnen, der angibt, wieviele Daten folgen und was mit diesen zu geschehen hat.

a) Steuer-Code 01H:

Es folgen 1 Längen-Byte, eine Adresse (LSB, MSB) und soviele Daten, wie das um 2 verminderte Längen-Byte angibt. Die Daten sind im RAM ab der angegebenen Adresse abzulegen.

b) Steuer-Code 02H:

Es folgen 1 Byte ohne Bedeutung und die Startadresse (LSB, MSB), bei der SYS0 zu starten ist. Damit ist das Ende von SYS0 erreicht.

c) Steuer-Code 00H oder 03H bis 1FH:

Es folgen 1 Längen-Byte und soviele Daten, wie das Längen-Byte angibt. Diese Daten sind nur ein Kommentar und brauchen nirgendwo abgespeichert zu werden.

Nach dem Starten beginnt SYS0 mit seiner eigenen Initialisierung, innerhalb derer das Ende des verfügbaren Speichers gesucht, die SYSTEM- und PDRIVE-Parameter von Diskette ins RAM gelesen, ggf. TIME und DATE erfragt und zum Schluß entweder der AUTO-Befehl ausgeführt oder die Befehlseingabe von SYS1 aufgerufen wird.

Diese Initialisierung von SYS0 wird jedoch erst im nächsten Kapitel beschrieben.

Das Programm im BOOT-Sector kann in Abhängigkeit der folgenden PDRIVE-Parameter für Drive 0 leicht variieren:

- Typ des evtl. eingebauten Floppy-Interfaces
- 8 Zoll oder 5.25 Zoll
- Single oder Double Density
- Single oder Double Sided
- Anzahl Sectoren / Track
- Track# ab 0 oder ab 1
- Sector# ab 0 oder ab 1

Nachfolgend wird der BOOT-Sector für die Standard-Version (5.25 Zoll, Single Density, Single Sided, 10 Sectoren pro Track) beschrieben. Die Stellen, wo sich sonst Abweichungen ergeben können, sind durch (*) gekennzeichnet.

2.1 Listing des Urladers im BASIC-ROM

```
0693 31 7D 40     LD      SP,407D      ;Stackpointer auf 407D
0696 3A EC 37     LD      A,(37EC)     ;Status-Register des FDC lesen
0699 3C           INC     A            ;ist dessen Inhalt
069A FE 02        CP      02           ;00H oder FFH ?
069C DA 75 00     JP      C,0075       ;wenn ja: BASIC initialisieren
069F 3E 01        LD      A,01         ;Drive 0
06A1 32 E1 37     LD      (37E1),A     ;anwählen
06A4 21 EC 37     LD      HL,37EC      ;Zeiger auf Kommando/Status-Register
06A7 11 EF 37     LD      DE,37EF      ;Zeiger auf Daten-Register des FDC
06AA 36 03        LD      (HL),03      ;RESTORE-Kommando an FDC senden
06AC 01 00 00     LD      BC,0000      ;960 ms
06AF CD 60 00     CALL    0060         ;warten
06B2 CB 46        BIT     0,(HL)       ;ist der FDC busy ?
06B4 20 FC        JR      NZ,06B2      ;wenn ja: weiter warten
06B6 AF           XOR     A            ;Sect# 0
06B7 32 EE 37     LD      (37EE),A     ;in Sector-Register schreiben
06BA 01 00 42     LD      BC,4200      ;Zeiger auf Buffer für BOOT-Sector
06BD 3E 8C        LD      A,8C         ;READ-SECTOR-Kommando
06BF 77           LD      (HL),A       ;an FDC senden
06C0 CB 4E        BIT     1,(HL)       ;Data Request abfragen
06C2 28 FC        JR      Z,06C0       ;wenn nein: warten
06C4 1A           LD      A,(DE)       ;Datenbyte aus Daten-Register lesen
06C5 02           LD      (BC),A       ;in Buffer schreiben
06C6 0C           INC     C            ;Zeiger +1, Ende des Sectors ?
06C7 20 F7        JR      NZ,06C0      ;wenn nein
06C9 C3 00 42     JP      4200         ;BOOT-Sector starten
```

2.2 Listing des BOOT-Sectors

```
4200 00           ;./.
4201 FE           ;./.
4202*11           ;LUMP#, in welcher der Directory beginnt
4203 F3           DI                  ;Interrupts sperren
4204 21 EC 37     LD      HL,37EC     ;Zeiger auf FDC-Kommando/Status-Register
4207 36*FE        LD      (HL),FE     ;FEH: Single Density oder FFH: DD wählen
4209 36 D0        LD      (HL),D0     ;FORCE-INTERRUPT-Kommando an FDC
420B 23           INC     HL          ;Zeiger auf Track-Register
420C 36 00        LD      (HL),00     ;dort Track 0 eintragen
420E 23           INC     HL          ;Zeiger auf Sector-Register
420F 36*00        LD      (HL),00     ;ggf. 8 Zoll-Laufwerk wählen
                                      ;(dann wird C0H gesendet)
4211 11*05*00     LD      DE,0005     ;D=Track# (TI=J oder TI=K => D=01)
                                      ;E=Sector# (TI=I oder TI=M => E=06)
4214 D9           EXX                 ;gewünschte Track# und Sector# nach DE'
4215 31 E0 41     LD      SP,41E0     ;Stackpointer auf 41E0
4218 21 FF 51     LD      HL,51FF     ;Zeiger auf Ende des Sector-Buffers
                  ----------------------
                  SYS0 einlesen
421B CD 52 42     CALL    4252        ;Steuer-Code lesen
421E FE 20        CP      20          ;größer als 1FH ?
4220 47           LD      B,A         ;B = 1. Byte
4221 30 29        JR      NC,424C     ;wenn ja, Error
4223 57           LD      D,A         ;D = 1. Byte
4224 CD 52 42     CALL    4252        ;nächstes Byte lesen
4227 4F           LD      C,A         ;C = 2. Byte
4228 CD 52 42     CALL    4252        ;nächstes Byte lesen
422B 5F           LD      E,A         ;E = 3. Byte
422C 10 12        DJNZ    4240        ;wenn Steuer-Code <> 01H, dann 4240
                  ----------------------
                  Steuer-Code 01H:
422E CD 52 42     CALL    4252        ;4. Byte nach D lesen,
4231 57           LD      D,A         ;DE zeigt auf zu ladenden Speicher
4232 0D           DEC     C           ;Längen-Byte
4233 0D           DEC     C           ;um 2 vermindern
4234 2C           INC     L           ;Zeiger auf Buffer +1, Buffer zu Ende ?
4235 CC 55 42     CALL    Z,4255      ;wenn ja: nächsten Sector lesen
4238 7E           LD      A,(HL)      ;nächstes Byte aus Buffer holen
4239 12           LD      (DE),A      ;im Speicher ablegen
423A 13           INC     DE          ;Zeiger auf Speicher +1
423B 0D           DEC     C           ;Längen-Byte -1, Block zu Ende ?
423C 20 F6        JR      NZ,4234     ;wenn nein
423E 18 DB        JR      421B        ;wenn ja: nächsten Block bearbeiten
                  ----------------------
4240 10 F9        DJNZ    423B        ;wenn Steuer-Code <> 02H, dann 423B:
                                      ;D hat einen Wert zwischen 00 und 1F,
                                      ;somit werden Kommentare über das
                                      ;BASIC-ROM geladen!
                  ----------------------
                  Steuer-Code 02H:
4242 CD 52 42     CALL    4252        ;4. Byte nach D lesen,
4245 57           LD      D,A         ;DE = Startadresse
4246 1A           LD      A,(DE)      ;ist dort die Kennung
4247 FE A5        CP      A5          ;für NEWDOS/80 bzw. GDOS vorhanden ?
4249 13           INC     DE          ;Startadresse +1
424A D5           PUSH    DE          ;und in den Stack
424B C8           RET     Z           ;wenn ja, starten !
```

```
                    ---------------------
                    Fehlermeldung
424C 21 E5 42       LD      HL,42E5       ;Text CLS + "NO SYS"
424F C3 C3 42       JP      42C3          ;auf Bildschirm ausgeben

     ********************************************************
     * Name:       BYREAD                                   *
     * Funktion:   nächstes Byte von SYS0 holen             *
     * Input:      HL: zeigt auf nächstes Byte im Buffer    *
     *             D': zeigt auf nächste Track#             *
     *             E': zeigt auf nächste Sector#            *
     * Verändert:  HL=HL+1, ggf. DE' erhöhen, HL', B', F    *
     * Output:     A: nächstes Byte von SYS0                *
     ********************************************************

     Wenn beim Lesen eines neuen Sectors ein Disk-Error auftritt, wird eine
     Meldung ausgegeben und BYREAD kehrt nicht zurück (Endlos-Schleife).

4252 2C             INC     L             ;Zeiger auf Buffer +1
4253 7E             LD      A,(HL)        ;nächstes Byte lesen
4254 C0             RET     NZ            ;wenn Buffer nicht zu Ende: RET
4255 D9             EXX                   ;Zweit-Registersatz einschalten
4256 06 0A          LD      B,0A          ;Zähler für max. 10D Versuche
4258 21 E1 37       LD      HL,37E1       ;Drive 0
425B 36 01          LD      (HL),01       ;anwählen
425D D5             PUSH    DE            ;DE = Track# und Sect#
425E C5             PUSH    BC            ;Zähler retten
                    ---------------------
425F 7B             LD      A,E           ;bei doppelseitigen Laufwerken:
4260 D6*0A          SUB     0A            ;wenn "Anzahl der Sectoren / Seite"
4262 38 03          JR      C,4267        ;kleiner als E ist:
4264 5F             LD      E,A           ;Sector# für die Rückseite berechnen
4265 36*09          LD      (HL),09       ;und Seite 1 wählen
                    ---------------------
4267 21 EC 37       LD      HL,37EC       ;Zeiger auf FDC-Kommando/Status-Register
426A CD CE 42       CALL    42CE          ;warten, bis FDC nicht mehr busy ist
426D ED 53 EE 37    LD      (37EE),DE     ;Track# und Sector# an FDC senden
4271 36 1B          LD      (HL),1B       ;SEEK-Kommando an FDC
4273 CD CE 42       CALL    42CE          ;warten, bis FDC nicht mehr busy ist
4276 36 88          LD      (HL),88       ;READ-SECTOR-Kommando an FDC
4278 11 EF 37       LD      DE,37EF       ;Zeiger auf Daten-Register des FDC
427B 01 00 51       LD      BC,5100       ;Zeiger auf Buffer für nächsten Sector
427E CD D7 42       CALL    42D7          ;61 us warten
4281 7E             LD      A,(HL)        ;Status-Register des FDC lesen
4282 E6 83          AND     83            ;Data Request und FDC busy ?
4284 E2 81 42       JP      PO,4281       ;wenn nein, warten
4287 1A             LD      A,(DE)        ;nächstes Byte vom FDC holen
4288 02             LD      (BC),A        ;im Buffer ablegen
4289 03             INC     BC            ;Zeiger auf Buffer +1
428A CB 4E          BIT     1,(HL)        ;Data Request ?
428C C2 87 42       JP      NZ,4287       ;wenn ja
428F CB 4E          BIT     1,(HL)        ;Data Request ?
4291 C2 87 42       JP      NZ,4287       ;wenn ja
4294 CB 4E          BIT     1,(HL)        ;Data Request ?
4296 20 EF          JR      NZ,4287       ;wenn ja
4298 CB 46          BIT     0,(HL)        ;ist FDC noch busy ?
429A 28 08          JR      Z,42A4        ;wenn nein
429C CB 4E          BIT     1,(HL)        ;Data Request ?
429E 20 E7          JR      NZ,4287       ;wenn ja
42A0 CB 7E          BIT     7,(HL)        ;ist Drive noch ready ?
```

```
42A2 28 E6        JR      Z,428A        ;wenn ja
42A4 7E           LD      A,(HL)        ;Status-Register des FDC lesen
42A5 36 D0        LD      (HL),D0       ;FORCE-INTERRUPT-Kommando an FDC
42A7 C1           POP     BC            ;Zähler für Anzahl Versuche zurück
42A8 D1           POP     DE            ;Track# und Sector# zurück
42A9 E6 FC        AND     FC            ;Error-Bits maskieren
42AB 20 0C        JR      NZ,42B9       ;wenn Error
                  ----------------------
42AD 1C           INC     E             ;Sector# +1
42AE 7B           LD      A,E           ;ist die nächste Sector# größer als die
42AF D6*0A        SUB     0A            ;"Anzahl Sectoren pro Track" ?
42B1 20 03        JR      NZ,42B6       ;wenn nein
42B3 14           INC     D             ;Track# +1
42B4 1E*00        LD      E,00          ;E = 1. Sector# im Track (TI=I/M => E=1)
                  ----------------------
42B6 D9           EXX                   ;Zweit-Registersatz zurück
42B7 7E           LD      A,(HL)        ;1. Byte aus dem Buffer holen
42B8 C9           RET                   ;fertig
                  ----------------------
                  Fehlerbehandlung nach Sector lesen
42B9 CD D7 42     CALL    42D7          ;61 us warten
42BC 36 0B        LD      (HL),0B       ;RESTORE-Kommando an FDC
42BE 10 98        DJNZ    4258          ;wenn noch keine 10D Versuche gemacht
42C0 21 DD 42     LD      HL,42DD       ;Text CLS + "ERROR"
                  ----------------------
                  Text (HL) anzeigen und Endlos-Schleife
42C3 7E           LD      A,(HL)        ;nächstes Zeichen holen
42C4 FE 03        CP      03            ;Ende-Markierung ?
42C6 28 FB        JR      Z,42C3        ;wenn ja
42C8 23           INC     HL            ;Zeiger +1
42C9 CD 33 00     CALL    0033          ;Zeichen auf Bildschirm ausgeben
42CC 18 F5        JR      42C3          ;weiter
                  ----------------------
                  Warten, bis FDC nicht mehr busy ist
42CE CD D7 42     CALL    42D7          ;61 us warten
42D1 CB 46        BIT     0,(HL)        ;ist FDC noch busy ?
42D3 20 FC        JR      NZ,42D1       ;wenn ja
42D5 7E           LD      A,(HL)        ;Status-Register lesen
42D6 C9           RET
                  ----------------------
                  Verzögerung 61 us (bei 1.774 MHz)
42D7 3E 06        LD      A,06          ;Zähler setzen
42D9 3D           DEC     A             ;Zähler -1
42DA 20 FD        JR      NZ,42D9       ;wenn <> 0
42DC C9           RET
                  ----------------------
                  Text CLS + "ERROR"
42DD 1C 1F 45 52 52 4F 52 03                        ..ERROR.
                  ----------------------
                  Text CLS + "NO SYS"
42E5 1C 1F 4E 4F 20 53 59 53 03                     ..NO SYS.
                  ----------------------
                  Unbenutzt
42EE FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
                  ----------------------
                  Wichtige PDRIVE-Parameter für Drive 0
                  (werden von SYS0 zum Lesen des PDRIVE-Sectors benötigt)
42FD*00           ;PDRIVE+6 (Erläuterungen siehe Kapitel 4.2)
42FE*05           ;PDRIVE+2                 "
42FF*00           ;PDRIVE+7                 "
```

Kapitel 3: SYS0/SYS

Betriebssystem: NEWDOS/80 Version 2 für TRS-80 Model I

SYS0/SYS ist das Herz des Disketten-Betriebssystems. Im Gegensatz zu den anderen /SYS-Modulen steht es ständig im Speicher, wofür der Bereich 4000H bis 4CFFH reserviert ist. Während der Initialisierung von SYS0 wird zusätzlich der Bereich 4D00H bis 51DAH benutzt.

Funktionen von SYS0: Bereitstellung von Sprungtabellen, RCB's (ROUTE Control Blocks) und FCB's für CHAINING und zum Laden von Programmen und SYS-Files, Verwaltung von SYSTEM- und PDRIVE-Parametern, Interrupts, Uhrzeit und Datum, Erweiterung der Bildschirm- und Tastatur-Routinen, Laden von Programmen und SYS-Files, Verwaltung von Dateien inkl. Lesen und Schreiben von Logischen Records, Positionieren innerhalb von Dateien und der Berechnung der physikalischen Position von Datei-Sectoren, sowie die Bereitstellung von fundamentalen Disk-Routinen wie Lesen und Schreiben von Sectoren und Directory-Sectoren, Auswählen von Drives und Bedienung des Floppy Disk Controllers.

SYS0 wird durch den Lader im BOOT-Sector in folgende Speicherbereiche geladen:

400C - 4014
402D - 4035
403E - 405C
4063 - 407F
4308 - 4317
4368 - 43A8
43B2 - 43DF
4400 - 51DA

Bei den Speicheradressen, in die dabei nichts geladen wird, ist im nachfolgenden Listing daher auch kein Inhalt angegeben (ausgenommen im Bereich 4000H bis 402CH).

Startadresse von SYS0 ist 4D01H.

3.1 Disassembliertes Listing von SYS0/SYS

```
                    ----------------------
                    RESTART-Vektoren für den BASIC-Interpreter
4000 C3 96 1C       JP      1C96        ;RST 08-Vektor
4003 C3 78 1D       JP      1D78        ;RST 10-Vektor
4006 C3 90 1C       JP      1C90        ;RST 18-Vektor (wird vom DOS mitbenutzt)
4009 C3 D9 25       JP      25D9        ;RST 20-Vektor
                    ----------------------
                    RESTART-Vektoren für das DOS
400C C3 C2 4B       JP      4BC2        ;RST 28-Vektor (/SYS-Files laden)
400F C3 09 46       JP      4609        ;RST 30-Vektor (DEBUG)
4012 C3 F2 45       JP      45F2        ;RST 38-Vektor (Interrupts)
                    ----------------------
                    DCB's (Device Control Blocks)
                    Tastatur-DCB
4015 01             ;DCB-Typ
     a) wenn SYSTEM AJ=N:
4016*E3*03          ;Treiberadresse
     b) wenn SYSTEM AJ=Y:
4016*16*45          ;Treiberadresse
4018 00 00 00       ;./.
401B 4B 49          ;KI
                    ----------------------
                    Bildschirm-DCB
401D 07             ;DCB-Typ
     a) wenn SYSTEM AJ=N:
401E*58*04          ;Treiberadresse
     b) wenn SYSTEM AJ=Y:
401E*05*45          ;Treiberadresse
4020 00 3C          ;Cursoradresse
4022 00             ;Cursor-Status (00: Cursor aus, <>00: Zeichen unter Cursor)
4023 44 4F          ;DO
                    ----------------------
                    Drucker-DCB
4025 06             ;DCB-Typ
4026 8D 05          ;Treiberadresse
4028 43             ;Zeilen / Seite
4029 00             ;Zeilenzähler
402A 00             ;./.
402B 50 52          ;PR
```

```
*********************************************************
* Name:       DOSRDY                                    *
* Funktion:   Sprung nach DOS READY (ohne Rückkehr)     *
* Input:      --                                        *
* Verändert:  --                                        *
* Output:     --                                        *
*********************************************************
```

Programme, die ohne Fehler beendet werden, sollten mit einem Sprung nach DOSRDY aufhören. Von dort geht es dann folgendermaßen weiter:

- Wenn DOS-CALL (4419H) aktiv ist, wird ein RETURN zum Aufrufer von DOS-Call ausgeführt.
- Wenn CHAINING aktiv ist, wird der nächste Befehl aus der CHAIN-File gelesen und ausgeführt.
- Ansonsten wird DOS-READY oder MINI-DOS-READY angezeigt und der nächste Befehl von der Tastatur geholt und ausgeführt.

```
402D C3 00 44     JP      4400          ;weiter bei 4400

     ********************************************************
     * Name:       ERROR0                                   *
     * Funktion:   nach einem Fehler: Sprung nach DOS READY*
     * Input:      --                                       *
     * Verändert:  --                                       *
     * Output:     --                                       *
     ********************************************************
```

Programme, die mit einem bereits angezeigten Fehler beendet werden, sollten mit einem Sprung nach ERROR0 aufhören. Von dort geht es dann genau wie bei DOSRDY (402DH) weiter, ein evtl. CHAINING wird jedoch abgebrochen.

```
4030 3E 43        LD      A,43          ;Code für SYS1/SYS
4032 EF           RST     28            ;SYS-File laden und starten
                  ----------------------
                  Hilfssprung für UP's 0013H und 001BH
4033 C3 DB 4A     JP      4ADB          ;weiter bei 4ADB
                  ----------------------
                  Tastatur-Status
4036              ;für Tastaturzeile 3801
4037              ;für Tastaturzeile 3802
4038              ;für Tastaturzeile 3804
4039              ;für Tastaturzeile 3808
403A              ;für Tastaturzeile 3810
403B              ;für Tastaturzeile 3820
403C              ;für Tastaturzeile 3840
                  ----------------------
403D              ;Zwischenspeicher für Port FF
403E 00 00        ;./. (früher: DOS-Kennung)
                  ----------------------
                  Uhrzeit und Datum
4040 00           ;Zähler für 25-ms-Interrupts
4041 00           ;Uhrzeit: Sekunden
4042 00           ;         Minuten
4043 00           ;         Stunden
4044 00           ;Datum:   Jahr
4045 00           ;         Tag     (GDOS: Monat)
4046 00           ;         Monat   (GDOS: Tag)
                  ----------------------
4047 00 52        ;./.
4049 FF FF        ;HIMEM (höchste freie RAM-Adresse für das DOS)
                  ----------------------
                  RAM für DEBUG
404B 00 00        ;Adresse 2. Breakpoint
404D 00           ;Inhalt 2. Breakpoint
404E 00 00        ;Adresse 1. Breakpoint
4050 00           ;Inhalt 1. Breakpoint
                  ----------------------
4051 00 00        ;Zeiger auf Benutzer-Routinen (*name) (s. UP USRINS 4461H)
                  ----------------------
                  Hilfsroutine für Interrupt-Service bei 45F2
4053 F5           PUSH    AF            ;A=(37E0) nach RLCA
4054 07           RLCA                  ;Bit 6 gesetzt ?
4055 DC E8 47     CALL    C,47E8        ;wenn ja: (37EC) lesen
4058 F1           POP     AF            ;AF zurück
4059 C9           RET                   ;weiter bei 45FD
                  ----------------------
```

```
405A 00 00 00     ;./.
                  ----------------------
                  RAM für DEBUG
405D,405E         ;X-Modus: 1. Adresse
405F,4060         ;X-Modus: 2. Adresse
4061,4062         ;X-Modus: 3. Adresse

     ************************************************************
     * Name:       HEXDE                                        *
     * Funktion:   DE hexadezimal nach (HL) ausgeben            *
     * Input:      DE: auszugebende 16-Bit-Zahl                 *
     *             HL: Zeiger auf Bildschirm oder RAM           *
     * Verändert:  AF, HL=HL+4                                  *
     * Output:     --                                           *
     ************************************************************

4063 7A           LD      A,D           ;MSB laden
4064 CD 68 40     CALL    4068          ;und ausgeben
4067 7B           LD      A,E           ;LSB laden

     ************************************************************
     * Name:       HEXA                                         *
     * Funktion:   A hexadezimal nach (HL) ausgeben             *
     * Input:      A: auszugebende 8-Bit-Zahl                   *
     *             HL: Zeiger auf Bildschirm oder RAM           *
     * Verändert:  AF, HL=HL+2                                  *
     * Output:     --                                           *
     ************************************************************

4068 F5           PUSH    AF            ;A retten
4069 0F           RRCA                  ;linkes
406A 0F           RRCA                  ;Digit
406B 0F           RRCA                  ;auswählen
406C 0F           RRCA                  ;
406D CD 71 40     CALL    4071          ;und anzeigen
4070 F1           POP     AF            ;rechtes Digit holen
4071 E6 0F        AND     0F            ;Bits 7..4 löschen
4073 C6 90        ADD     90            ;in
4075 27           DAA                   ;ASCII-
4076 CE 40        ADC     40            ;Code
4078 27           DAA                   ;umformen
4079 77           LD      (HL),A        ;und ausgeben
407A 23           INC     HL            ;Zeiger erhöhen
407B C9           RET
                  ----------------------
                  Tabelle zum Erhöhen der Uhrzeit
407C 3B           ;max. 59 Sekunden
407D 3B           ;max. 59 Minuten
407E 17           ;max. 23 Stunden
407F 00           ;Ende der Tabelle
                  ----------------------
4080-41FF         ;RAM für den BASIC-Interpreter und Stack für DOS
                  ----------------------
4200-42FF         ;Sector-Buffer für DOS
                  ----------------------
                  Position der Schreib/Lese-Köpfe
4300              ;Drive 0: aktuelle Track# des Schreib/Lese-Kopfes
4301              ;Drive 1: aktuelle Track# des Schreib/Lese-Kopfes
4302              ;Drive 2: aktuelle Track# des Schreib/Lese-Kopfes
4303              ;Drive 3: aktuelle Track# des Schreib/Lese-Kopfes
```

```
                    ---------------------
4304-4307           ;./.
                    ---------------------
4308 00             ;aktuelle Drive# (0-3)
4309 00             ;Bit-Muster für aktuelles Drive für 37E1 (Drive-Select)
                    ---------------------
                    PDRIVE-Parameter für das aktuelle Drive:
430A 11      +0     ;wirklicher DSL "Directory Starting Lump"
430B 23      +1     ;Anzahl Lumps pro Diskette
430C 03      +2     ;TI "Type of Interface" und TSR "Track Stepping Rate"
                    ;Bit 7: gesetzt wenn TI=H
                    ;Bit 6: gesetzt wenn TI=K
                    ;Bit 5: gesetzt wenn TI=M
                    ;       TI=A  TI=B  TI=C  TI=D  TI=E
                    ;       ----------------------------
                    ;Bit 4:  0     0     1     0     1
                    ;Bit 3:  0     1     0     1     0
                    ;Bit 2:  1     1     0     0     1
                    ;Bits 1,0: TSR "Track Stepping Rate"
430D 23      +3     ;TC "Track Count"
430E 0A      +4     ;SPT "Sectors pro Track"
430F 02      +5     ;GPL "Grans pro Lump"
4310 00      +6     ;TI "Type of Interface" und TD "Type of Drive"
                    ;(wird nach 37EE geschrieben, siehe Kapitel 1.2.1)
                    ;00: wenn TI=A/C
                    ;80: wenn TI=B/E und TD=A/C/E/G (5.25 Zoll)
                    ;C0: wenn TI=B/E und TD=B/D/F/H (8 Zoll)
                    ;FC: wenn TI=D und TD=A/C (5.25 Zoll und Single Density)
                    ;FD: wenn TI=D und TD=B/D (8 Zoll und Single Density)
                    ;FE: wenn TI=D und TD=E/G (5.25 Zoll und Double Density)
                    ;FF: wenn TI=D und TD=F/H (8 Zoll und Double Density)
4311 00      +7     ;TI "Type of Interface" und TD "Type of Drive"
                    ;Bit 7: 8 Zoll (1) oder 5.25 Zoll (0)
                    ;Bit 6: Diskette doppelseitig (1) oder einseitig (0)
                    ;Bit 5: ./.
                    ;Bit 4: gesetzt wenn TI=I oder TI=M (Sect# ab 1)
                    ;Bit 3: ./.
                    ;Bit 2: gesetzt wenn TI=L
                    ;Bit 1: gesetzt wenn TI=J oder TI=K (Track# ab 1)
                    ;Bit 0: Double (1) oder Single (0) Density
                    ---------------------
                    Hilfssprung für BREAK-Taste
    a) wenn SYSTEM AG=N oder DOS-Befehl BREAK,N:
4312*C3 B0 45       JP      45B0        ;dort: XOR A, RET
    b) wenn SYSTEM AG=Y oder DOS-Befehl BREAK,Y:
4312*C9 B0 45       RET                 ;A unverändert
                    ---------------------
4315 01             ;./. (früher: DEBUG enable / disable)
4316 00             ;./.
                    ---------------------
                    aktuelles /SYS-Modul
4317 00             ;gibt an, welches /SYS-Modul (SYS1 bis SYS29) zuletzt geladen
                    ;wurde und noch verfügbar ist
                    ;00: kein /SYS-Modul verfügbar
                    ;03: SYS1 verfügbar ... 1F: SYS29 verfügbar
                    ---------------------
4318-4367           ;Input-Buffer für DOS
                    ---------------------
4368 A5             ;NEWDOS/80 Kennung
                    ---------------------
```

```
4369 40          ;Bit 7: DOS-Befehl DEBUG,Y (1) oder DEBUG,N (0)
                 ;Bit 6: ist gesetzt, während GETSYS (4BC9H) oder BASIC-Befehl
                 ;       CMD "C" ausgeführt wird, um 'DFG', '123', 'JKL' und
                 ;       CHAINING zu sperren
                 ;Bit 5: gesetzt wenn CHAINING aktiv ist
                 ;Bit 4: SYSTEM AG (BREAK-Taste)
                 ;Bit 3: wird von SYS6 gesetzt, um 'DFG', '123', 'JKL' und
                 ;       CHAINING zu sperren
                 ;Bit 2: SYSTEM AB (RUN-ONLY Modus)
                 ;Bit 1: ./.
                 ;Bit 0: ./.
                 ----------------------
436A 00          ;Bit 7: gesetzt wenn MINI-DOS aktiv ist
                 ;Bit 6: gesetzt während DOS-CALL (4419H)
                 ;Bit 5: ? (SYS1, 4DF5: SET, 4E3E: RES)
                 ;Bit 4: gesetzt wenn CHAINING unter DOS-CALL
                 ;Bit 3: ./.
                 ;Bit 2: wird von SYS4 bei A=46H gesetzt und bewirkt in SYS1
                 ;       "DOS FATAL ERROR!! KEY 'R' FOR RESET"
                 ;Bit 1: ./.
                 ;Bit 0: ./.
                 ----------------------
436B 00          ;anhand dieses Bytes manipuliert SYS6 die Funktion
                 ;von CLOSE (4428H) und EXPAND (444BH)
                 ----------------------
436C 00          ;Bit 7: SYSTEM AA (Passworte)
                 ;Bit 6: SYSTEM AB (RUN-ONLY Modus)
                 ;Bit 5: SYSTEM AG (BREAK-Taste)
                 ;Bit 4: SYSTEM AI (KB-Routine von NEWDOS/80)
                 ;Bit 3: ./.
                 ;Bit 2: ./. (früher: SYSTEM AK)
                 ;Bit 1: SYSTEM AR (Passworte bei COPY)
                 ;Bit 0: SYSTEM AS (Kleinbuchstaben bei BASIC)
                 ----------------------
436D 00          ;Bit 7: SYSTEM AT (CHAINING im Record-Mode oder Einzeltasten)
                 ;Bit 6: SYSTEM BC (CHAINING anhalten und abbrechen)
                 ;Bit 5: SYSTEM BE (DOS-Befehl "R")
                 ;Bit 4: SYSTEM BK (DOS-Befehl "WRDIRP")
                 ;Bit 3: SYSTEM BM (Verify Sectors in FORMAT)
                 ;Bit 2: ./.
                 ;Bit 1: ./.
                 ;Bit 0: ./.
                 ----------------------
436E 00 00       ;./.
                 ----------------------
4370 5A          ;SYSTEM AX (höchster ASCII-Code für Drucker)
                 ----------------------
                 PDRIVE-Parameter für Drive 0:
4371 11     +0   ;wirklicher DSL "Directory Starting Lump" (wird bei
                 ;Bedarf durch den Wert im BOOT-Sector der betreffenden
                 ;Diskette aktualisiert)
4372 23     +1   ;Anzahl Lumps pro Diskette
4373 03     +2   ;TI "Type of Interface" und TSR "Track Stepping Rate"
                 ;Bit 7: gesetzt wenn TI=H
                 ;Bit 6: gesetzt wenn TI=K
                 ;Bit 5: gesetzt wenn TI=M
                 ;      TI=A  TI=B  TI=C  TI=D  TI=E
                 ;      ----------------------------
                 ;Bit 4:  0     0     1     0     1
                 ;Bit 3:  0     1     0     1     0
```

```
                    ;Bit 2:  1     1     0     0     1
                    ;Bits 1,0: TSR "Track Stepping Rate"
4374 23      +3     ;TC "Track Count"
4375 0A      +4     ;SPT "Sectors pro Track"
4376 02      +5     ;GPL "Grans pro Lump"
4377 00      +6     ;TI "Type of Interface" und TD "Type of Drive"
                    ;(wird nach 37EE geschrieben, siehe Kapitel 1.2.1)
                    ;00: wenn TI=A/C
                    ;80: wenn TI=B/E und TD=A/C/E/G (5.25 Zoll)
                    ;C0: wenn TI=B/E und TD=B/D/F/H (8 Zoll)
                    ;FC: wenn TI=D und TD=A/C (5.25 Zoll und Single Density)
                    ;FD: wenn TI=D und TD=B/D (8 Zoll und Single Density)
                    ;FE: wenn TI=D und TD=E/G (5.25 Zoll und Double Density)
                    ;FF: wenn TI=D und TD=F/H (8 Zoll und Double Density)
4378 00      +7     ;TI "Type of Interface" und TD "Type of Drive"
                    ;Bit 7: 8 Zoll (1) oder 5.25 Zoll (0)
                    ;Bit 6: Diskette doppelseitig (1) oder einseitig (0)
                    ;Bit 5: ./.
                    ;Bit 4: gesetzt wenn TI=I oder TI=M (Sect# ab 1)
                    ;Bit 3: ./.
                    ;Bit 2: gesetzt wenn TI=L
                    ;Bit 1: gesetzt wenn TI=J oder TI=K (Track# ab 1)
                    ;Bit 0: Double (1) oder Single (0) Density
4379 11      +8     ;DDSL "Default Directory Starting Lump" (für FORMAT)
437A 02      +9     ;DDGA "Default Directory Grans Allocation" (für FORMAT)
                    ----------------------
                    PDRIVE-Parameter für Drive 1: (Bedeutung siehe bei Drive 0)
437B FF             ;PDRIVE+0
437C 01             ;PDRIVE+1
437D 00             ;PDRIVE+2
437E 01             ;PDRIVE+3
437F 00             ;PDRIVE+4
4380 00             ;PDRIVE+5
4381 00             ;PDRIVE+6
4382 00             ;PDRIVE+7
4383 FF             ;PDRIVE+8
4384 00             ;PDRIVE+9
                    ----------------------
                    PDRIVE-Parameter für Drive 2: (Bedeutung siehe bei Drive 0)
4385 FF             ;PDRIVE+0
4386 01             ;PDRIVE+1
4387 00             ;PDRIVE+2
4388 01             ;PDRIVE+3
4389 00             ;PDRIVE+4
438A 00             ;PDRIVE+5
438B 00             ;PDRIVE+6
438C 00             ;PDRIVE+7
438D FF             ;PDRIVE+8
438E 00             ;PDRIVE+9
                    ----------------------
                    PDRIVE-Parameter für Drive 3: (Bedeutung siehe bei Drive 0)
438F FF             ;PDRIVE+0
4390 01             ;PDRIVE+1
4391 00             ;PDRIVE+2
4392 01             ;PDRIVE+3
4393 00             ;PDRIVE+4
4394 00             ;PDRIVE+5
4395 00             ;PDRIVE+6
4396 00             ;PDRIVE+7
4397 FF             ;PDRIVE+8
```

```
4398 00          ;PDRIVE+9
                 ----------------------
4399 71 43       ;Zeiger auf den PDRIVE-Block der aktuellen Drive
439B 00 00       ;Zwischenspeicher für Stackpointer unter MINI-DOS
439D 00 00       ;Zwischenspeicher für Stackpointer unter DOS-CALL
439F 04          ;SYSTEM AL (Anzahl Drives)
43A0 01          ;SYSTEM AN (Drive# für DIR)
43A1 00          ;SYSTEM AO (Drive# für CREATE von neuen Files)
43A2 01          ;SYSTEM BJ (CPU Speed)
43A3 00          ;./.
43A4 01          ;DEBUG: Modus "S" (00) oder Modus "X" (01)
43A5 00          ;CHAINING: Buffer für nächstes Zeichen aus der CHAIN-File
43A6 00          ;./.
43A7 0D 0D       ;Buffer für DOS-Befehl "R"
43A9,43AA        ;höchste Adresse des vorhandenen RAM (HIMEM)
43AB             ;hat den Inhalt A5H, sobald NEWDOS/80 initialisiert ist
                 ----------------------
                 Buffer für Uhrzeit und Datum
43AC             ;Uhrzeit: Sekunden
43AD             ;         Minuten
43AE             ;         Stunden
43AF             ;Datum:   Jahr
43B0             ;         Tag      (GDOS: Monat)
43B1             ;         Monat    (GDOS: Tag)
                 ----------------------
                 1. RCB (Control Block für DOS-Befehl "ROUTE")
43B2 00 00       ;Adresse des DCB, der umgeleitet werden soll
43B4 00          ;dessen DCB-Typ
43B5 FF FF       ;Zeiger auf weitere RCB's
43B7 FC 4C       ;Adresse des DCB, auf den umgeleitet werden soll
                 ----------------------
                 2. RCB (Control Block für DOS-Befehl "ROUTE")
43B9 00 00       ;Adresse des DCB, der umgeleitet werden soll
43BB 00          ;dessen DCB-Typ
43BC FF FF       ;Zeiger auf weitere RCB's
43BE 00 00       ;Adresse des DCB, auf den umgeleitet werden soll
                 ----------------------
                 3. RCB (Control Block für DOS-Befehl "ROUTE")
43C0 00 00       ;Adresse des DCB, der umgeleitet werden soll
43C2 00          ;dessen DCB-Typ
43C3 FF FF       ;Zeiger auf weitere RCB's
43C5 00 00       ;Adresse des DCB, auf den umgeleitet werden soll
                 ----------------------
                 4. RCB (Control Block für DOS-Befehl "ROUTE")
43C7 00 00       ;Adresse des DCB, der umgeleitet werden soll
43C9 00          ;dessen DCB-Typ
43CA FF FF       ;Zeiger auf weitere RCB's
43CC 00 00       ;Adresse des DCB, auf den umgeleitet werden soll
                 ----------------------
                 Verkürzter FCB zum Laden von /SYS-Files in UP GETSYS (4BC9H)
43CE 80     +0   ;Bit 7: gesetzt wenn FCB geöffnet
                 ;Bits 6-3: ./.
                 ;Bit 2: muß gelöscht sein wegen Abfrage in SYS0 bei 4823H
                 ;Bit 1: gesetzt wenn der FCB nicht zu einer File gehört,
                 ;       sondern zu einer ganzen Diskette
                 ;Bit 0: gesetzt wenn die Sectoren dieser File wie
                 ;       DIR-Sectoren lesegeschützt geschrieben werden sollen
43CF 28     +1   ;Bit 7: gesetzt wenn Logische Recordlänge (LRL) <> 256D
                 ;Bit 6: (1) Der EOF-Wert soll nach dem Schreiben nur dann auf
                 ;           den NEXT-Wert gesetzt werden, wenn dieser größer
```

```
                    ;           als der EOF-Wert ist.
                    ;       (0) Der EOF-Wert soll nach jedem Schreiben auf den
                    ;           NEXT-Wert gesetzt werden.
                    ;Bit 5: gelöscht wenn der aktuelle Sector im Buffer steht
                    ;Bit 4: gesetzt wenn die Daten im Buffer noch auf Diskette
                    ;       geschrieben werden müssen
                    ;Bit 3: gesetzt in NEWDOS/80 Version 2
                    ;Bits 2-0: Zugriffs-Level (0-7) aufgrund eines Passwortes
43D0 00       +2    ;Bit 7: gesetzt wenn ASE=N (siehe DOS-Befehl ATTRIB)
                    ;Bit 6: gesetzt wenn ASC=N (siehe DOS-Befehl ATTRIB)
                    ;Bit 5: gesetzt wenn UDF=Y (siehe DOS-Befehl ATTRIB)
                    ;Bits 4-0: ./.
43D1 00 42    +3    ;Adresse des File-Buffers
43D3 00       +5    ;3. Byte des NEXT-Wertes, gibt die relative Position
                    ;innerhalb des aktuellen Sectors an
43D4 00       +6    ;Drive# der File bzw. der Diskette
43D5 FF       +7    ;DEC (Directory Entry Code) vom FPDE der File
43D6 00       +8    ;3. Byte des EOF-Wertes, gibt die relative Position
                    ;innerhalb des letzten Sectors an
43D7 00       +9    ;Logische Recordlänge (1-256D)
43D8 00 00    +0A   ;2. und 1. Byte des NEXT-Wertes. Die 3 Bytes des NEXT-Wertes
                    ;geben im RBA-Format an, wohin/woher der nächste Record zu
                    ;schreiben/zu lesen ist, wonach der NEXT-Wert jeweils
                    ;entsprechend erhöht wird.
43DA 40 00    +0C   ;2. und 1. Byte des EOF-Wertes (End of File). Die 3 Bytes des
                    ;EOF-Wertes geben im RBA-Format die aktuelle Länge der File
                    ;an.
43DC 00 00    +0E   ;gibt an, wo der 1. Datenblock dieser File auf Diskette steht
                    ;a) 1. Byte = FF: Es steht kein Datenblock mehr auf Diskette.
                    ;b) 1. Byte = FE: Im 2. Byte steht der DEC (Directory Entry
                    ;   Code) für den nächsten FXDE dieser File.
                    ;c) 1. Byte = 00-FD: Gibt die LUMP# an, in der dieser
                    ;   Datenblock auf Diskette beginnt. Bits 7-5 des 2. Bytes
                    ;   geben dann die Anzahl der GRANS an, die zwischen dem
                    ;   Beginn des LUMPS und dem Beginn des Datenblocks liegen
                    ;   und Bits 4-0 des 2. Bytes die um 1 verminderte Anzahl der
                    ;   GRANS, aus denen dieser Datenblock besteht.
43DE FF FF    +10H  ;gibt an, wo der 2. Datenblock dieser File auf Diskette steht
                    ;(Bedeutung der Bytes wie beim 1. Datenblock)
                    ----------------------
43E0-43FF           FCB für CHAINING


     ************************************************************
     * Name:       DOSRDY (Fortsetzung von 402DH)               *
     * Funktion:   Sprung nach DOS READY (ohne Rückkehr)        *
     * Input:      --                                           *
     * Verändert:  --                                           *
     * Output:     --                                           *
     ************************************************************

4400 3E 23          LD      A,23        ;Code für SYS1/SYS
4402 EF             RST     28          ;SYS-File laden und starten
                    ----------------------
4403 00 00          ;Buffer für die übergabe von Parametern unter
                    ;DOS-CALL (4419H) und für die Startadresse von
                    ;Programmen in LOAD (4430H)
```

```
     ********************************************************
     * Name:       DOSCMD                                   *
     * Funktion:   DOS-Befehl (HL) ausführen (ohne Rückkehr)*
     * Input:      HL: zeigt auf einen DOS-Befehl, der mit  *
     *                 ODH abgeschlossen sein muß           *
     * Verändert:  --                                       *
     * Output:     --                                       *
     ********************************************************
```

Der DOS-Befehl, auf den HL zeigt, wird (unter Umwandlung von Kleinbuchstaben in Großbuchstaben) in den Input-Buffer des DOS (4318-4367) übertragen und ausgeführt. Anschließend kehrt DOSCMD jedoch nicht zum Aufrufer zurück sondern springt nach DOSRDY (402DH).
ACHTUNG! Vor Aufruf von DOSCMD darf bei 4318H keinesfalls ODH stehen, sonst passiert gar nichts!

```
4405 3E 63        LD      A,63          ;Code für SYS1/SYS
4407 EF           RST     28            ;SYS-File laden und starten
                  ----------------------
                  Test auf Errors
4408 C8           RET     Z             ;wenn kein Error: RET
```

```
     ********************************************************
     * Name:       DOSERR                                   *
     * Funktion:   Fehlermeldung (A) ausgeben               *
     * Input:      A: Fehlercode - Bit 7: Rückkehr (J/N)    *
     * Verändert:  F                                        *
     * Output:     --                                       *
     ********************************************************
```

Wenn Bit 7 im A-Register gesetzt ist, wird die entsprechende Fehlermeldung ausgegeben und DOSERR kehrt zum Aufrufer zurück.
Wenn Bit 7 im A-Register nicht gesetzt ist, geschieht folgendes:
- Wenn CHAINING aktiv ist, wird es abgebrochen.
- Wenn DOS-CALL (4419H) aktiv ist, wird der aktuelle DOS-CALL Aufruf beendet, der Fehlercode (A) wird an den Aufrufer von DOS-CALL übergeben und es wird keine Fehlermeldung ausgegeben.
- Ansonsten wird die entsprechende Fehlermeldung ausgegeben und es erfolgt ein Sprung nach DOSRDY (402DH).

```
4409 F5           PUSH    AF            ;Fehlercode retten
440A 3E 26        LD      A,26          ;Code für SYS4/SYS
440C EF           RST     28            ;SYS-File laden und starten
```

```
     ********************************************************
     * Name:       DEBUG                                    *
     * Funktion:   DEBUG aufrufen                           *
     * Input:      --                                       *
     * Verändert:  --                                       *
     * Output:     --                                       *
     ********************************************************
```

```
440D C3 09 46     JP      4609          ;weiter bei 4609
```

```
     ********************************************************
     * Name:       INTINS                                   *
     * Funktion:   Benutzer-Interrupt-Routine einfügen      *
     * Input:      DE: Zeiger auf Kontroll-Block der        *
     *                 Benutzer-Interrupt-Routine           *
     * Verändert:  AF, BC, DE, HL                           *
     * Output:     --                                       *
     ********************************************************

4410 3E 65        LD       A,65          ;Code für SYS3/SYS
4412 EF           RST      28            ;SYS-File laden und starten

     ********************************************************
     * Name:       INTDEL                                   *
     * Funktion:   Benutzer-Interrupt-Routine löschen       *
     * Input:      DE: Zeiger auf Kontroll-Block der        *
     *                 Benutzer-Interrupt-Routine           *
     * Verändert:  AF, BC, DE, HL                           *
     * Output:     --                                       *
     ********************************************************

4413 3E 85        LD       A,85          ;Code für SYS3/SYS
4415 EF           RST      28            ;SYS-File laden und starten

     ********************************************************
     * Name:       MOTONX                                   *
     * Funktion:   Drive-Motoren weiterlaufen lassen        *
     * Input:      --                                       *
     * Verändert:  AF                                       *
     * Output:     --                                       *
     ********************************************************
```

Falls die Motoren der Drives noch an sind, dann wird durch erneuten Drive-Select dafür gesorgt, daß die Motoren weiterlaufen.

```
4416 C3 62 47     JP       4762          ;weiter bei 4762

     ********************************************************
     * Name:       DOSCAL                                   *
     * Funktion:   DOS-Befehl (HL) ausführen und zurück     *
     * Input:      HL: zeigt auf einen DOS-Befehl, der mit  *
     *                 ODH abgeschlossen sein muß           *
     * Verändert:  --                                       *
     * Output:     AF: Fehler-Status (siehe Text)           *
     ********************************************************
```

Der DOS-Befehl, auf den HL zeigt, wird (unter Umwandlung von Kleinbuchstaben in Großbuchstaben) in den Input-Buffer des DOS (4318-4367) übertragen und ausgeführt. Anschließend kehrt DOSCAL zum Aufrufer zurück, wobei im AF-Register folgender Fehler-Status übergeben werden kann:

a) C=1 (Carry): Es ist ein Fehler aufgetreten, der bereits angezeigt wurde.
b) C=0, Z=0: Es ist ein Fehler aufgetreten, im A-Register befindet sich der Fehlercode.
c) C=0, Z=1: Es ist kein Fehler aufgetreten.

Wenn mit DOSCAL z. B. ein Benutzer-Programm aufgerufen werden soll, kann zur Übergabe von Parametern der Buffer bei 4403H,4404H benutzt werden. ACHTUNG! Vor Aufruf von DOSCAL darf bei 4318H keinesfalls 0DH stehen, sonst passiert gar nichts!

```
4419 3E C3        LD      A,C3        ;Code für SYS1/SYS
441B EF           RST     28          ;SYS-File laden und starten
```

```
     ********************************************************
     * Name:       TFSPEC                                   *
     * Funktion:   Filespec (HL) nach FCB (DE) übertragen   *
     * Input:      HL: zeigt auf Filespec                   *
     *             DE: zeigt auf anzulegenden FCB           *
     * Verändert:  BC                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     *             HL: zeigt auf das Ende des Filespecs     *
     ********************************************************
```

Wenn es sich bei dem Text, auf den HL zeigt, um einen möglichen Filespec handelt, wird dieser nach DE übertragen. Es werden max. 32D Bytes übertragen. Wenn weniger als 32D Bytes übertragen wurden, wird ein Byte 03H als Ende-Markierung hinzugefügt. Wenn der Filespec (HL) mit 03H oder 0DH endet, dann zeigt HL anschließend auf dieses Byte, ansonsten auf das 1. Byte nach dem Byte, welches den Filespec beendet.

```
441C 3E 83        LD      A,83        ;Code für SYS1/SYS
441E EF           RST     28          ;SYS-File laden und starten
441F 00           NOP                 ;./.
```

```
     ********************************************************
     * Name:       INIT                                     *
     * Funktion:   File öffen, ggf. neue File anlegen       *
     * Input:      DE: zeigt auf FCB, der Filespec enthält  *
     *             HL: zeigt auf zu benutzenden Buffer      *
     *             B:  gibt Logische Recordlänge an         *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

```
4420 3E 44        LD      A,44        ;Code für SYS2/SYS
4422 EF           RST     28          ;SYS-File laden und starten
4423 00           NOP                 ;./.
```

```
     ********************************************************
     * Name:       OPEN                                     *
     * Funktion:   File öffen, ggf. keine neue File anlegen*
     * Input:      DE: zeigt auf FCB, der Filespec enthält  *
     *             HL: zeigt auf zu benutzenden Buffer      *
     *             B:  gibt Logische Recordlänge an         *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

```
4424 3E 24        LD      A,24        ;Code für SYS2/SYS
4426 EF           RST     28          ;SYS-File laden und starten
                  ----------------------
4427 82           ;Kennung für NEWDOS/80 Version 2
```

```
     ********************************************************
     * Name:       CLOSE                                    *
     * Funktion:   File schließen                           *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

```
4428 3E 25        LD      A,25          ;Code für SYS3/SYS
442A EF           RST     28            ;SYS-File laden und starten
                  ----------------------
442B 01           ;Kennung von NEWDOS/80 Version 2 für TRS-80 Model I

     ********************************************************
     * Name:       KILL                                     *
     * Funktion:   File löschen                             *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

442C 3E 45        LD      A,45          ;Code für SYS3/SYS
442E EF           RST     28            ;SYS-File laden und starten
442F 00           NOP                   ;./.

     ********************************************************
     * Name:       LOAD                                     *
     * Funktion:   Programm laden                           *
     * Input:      DE: zeigt auf FCB, der Filespec enthält  *
     * Verändert:  BC                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     *             HL: Startadresse des Programms           *
     *             4403H,4404H: Startadresse des Programms  *
     ********************************************************
```

Das zu ladende Programm muß im Standard-Format für LOAD-Files auf Diskette stehen. Dieses Format besteht aus mehreren Blocks, die jeweils mit einem Steuer-Code beginnen, der angibt, wieviele Daten folgen und was mit diesen zu geschehen hat:

a) Steuer-Code 01H: Es folgen 1 Längen-Byte, eine Adresse (LSB, MSB) und soviele Daten, wie das um 2 verminderte Längen-Byte angibt. Die Daten sind im RAM ab der angegebenen Adresse abzulegen.
b) Steuer-Code 02H: Es folgen 1 Byte ohne Bedeutung und die Startadresse (LSB, MSB), bei der das Programm zu starten ist. Damit ist das Ende der File erreicht.
c) Steuer-Code 00H oder 03H bis 1FH: Es folgen 1 Längen-Byte und soviele Daten, wie das Längen-Byte angibt. Diese Daten sind nur ein Kommentar und brauchen nirgendwo abgespeichert zu werden.

Zum Laden benutzt das DOS seinen eigenen Sector-Buffer (4200H-42FFH).

```
4430 3E A4        LD      A,A4          ;Code für SYS2/SYS
4432 EF           RST     28            ;SYS-File laden und starten

     ********************************************************
     * Name:       RUN                                      *
     * Funktion:   Programm laden + starten (ohne Rückkehr)*
     * Input:      DE: zeigt auf FCB, der Filespec enthält  *
     * Verändert:  AF, BC                                   *
     * Output:     --:                                      *
     ********************************************************
```

Das Programm muß im Standard-Format für LOAD-Files auf Diskette stehen, siehe bei LOAD (4430H).
Wenn das Programm gestartet wird, sind nur AF und BC verändert. Wenn beim Laden ein Fehler auftritt, erfolgt stattdessen ein Sprung nach DOSERR (4409H).

```
4433 3E C4        LD      A,C4          ;Code für SYS2/SYS
4435 EF           RST     28            ;SYS-File laden und starten

     ********************************************************
     * Name:       READ                                     *
     * Funktion:   nächsten Sector/Record einer File lesen  *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: zeigt auf Record-Buffer (nur wenn    *
     *                 Logische Recordlänge <> 256D ist)    *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4436 C3 FC 49     JP      49FC          ;weiter bei 49FC

     ********************************************************
     * Name:       WRITE                                    *
     * Funktion:   nächsten Sector/Record einer File auf    *
     *             Diskette schreiben (ohne Verify)         *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: zeigt auf Record-Buffer (nur wenn    *
     *                 Logische Recordlänge <> 256D ist)    *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4439 C3 36 4A     JP      4A36          ;weiter bei 4A36

     ********************************************************
     * Name:       VERIFY                                   *
     * Funktion:   nächsten Sector/Record einer File auf    *
     *             Diskette schreiben, anschließend Verify  *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: zeigt auf Record-Buffer (nur wenn    *
     *                 Logische Recordlänge <> 256D ist)    *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

443C C3 32 4A     JP      4A32          ;weiter bei 4A32

     ********************************************************
     * Name:       POS0                                     *
     * Funktion:   NEXT-Feld im FCB auf Beginn der File     *
     *             positionieren                            *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

443F C3 4C 4B     JP      4B4C          ;weiter bei 4BAC
```

```
     ************************************************************
     * Name:       POSBC                                        *
     * Funktion:   NEXT-Feld im FCB auf die in BC ange-         *
     *             gebene Logische Record# positionieren        *
     * Input:      DE: zeigt auf geöffneten FCB                 *
     *             BC: gewünschte Logische Record#              *
     * Verändert:  --                                           *
     * Output:     AF: A=Fehlercode, wenn Z=0                   *
     ************************************************************

4442 C3 73 4B     JP       4B73          ;weiter bei 4B73

     ************************************************************
     * Name:       POSDEC                                       *
     * Funktion:   NEXT-Feld im FCB um 1 Logische Record#       *
     *             decrementieren (-1)                          *
     * Input:      DE: zeigt auf geöffneten FCB                 *
     * Verändert:  --                                           *
     * Output:     AF: A=Fehlercode, wenn Z=0                   *
     ************************************************************

4445 C3 62 4B     JP       4B62          ;weiter bei 4B62

     ************************************************************
     * Name:       POSEOF                                       *
     * Funktion:   NEXT-Feld im FCB auf EOF (End of File)       *
     *             positionieren                                *
     * Input:      DE: zeigt auf geöffneten FCB                 *
     * Verändert:  --                                           *
     * Output:     AF: A=Fehlercode, wenn Z=0                   *
     ************************************************************

4448 C3 54 4B     JP       4B54          ;weiter bei 4B54

     ************************************************************
     * Name:       EXPAND                                       *
     * Funktion:   Wenn die File kürzer ist, als das NEXT-      *
     *             Feld im FCB angibt, wird die File um         *
     *             entsprechend viele GRANS erweitert.          *
     * Input:      DE: zeigt auf geöffneten FCB                 *
     * Verändert:  --                                           *
     * Output:     AF: A=Fehlercode, wenn Z=0                   *
     ************************************************************

444B C3 09 48     JP       4809          ;weiter bei 4809

     ************************************************************
     * Name:       POSRBA                                       *
     * Funktion:   NEXT-Feld im FCB so positionieren, wie       *
     *             in HL und C im RBA-Format angegeben ist      *
     * Input:      DE: zeigt auf geöffneten FCB                 *
     *             HL: 1. und 2. Byte für NEXT-Feld             *
     *             C:  3. Byte für NEXT-Feld                    *
     * Verändert:  --                                           *
     * Output:     AF: A=Fehlercode, wenn Z=0                   *
     ************************************************************

444E C3 47 4B     JP       4B47          ;weiter bei 4B47
```

```
     ********************************************************
     * Name:       PUTEOF                                   *
     * Funktion:   Wenn das EOF-Feld im FCB von dem im FPDE*
     *             abweicht, wird es dort auf Diskette      *
     *             geschrieben                              *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4451 3E C5        LD      A,C5          ;Code für SYS3/SYS
4453 EF           RST     28            ;SYS-File lesen und starten
                  ----------------------
4454 00           NOP                   ;./.
4455 00           NOP                   ;./.
4456 00           NOP                   ;./.
                  ----------------------
                  Fehlerbehandlung
4457 3E 2A        LD      A,2A          ;Fehlercode für "ILLEGAL DOS FUNCTION"
4459 18 AE        JR      4409          ;Fehlermeldung ausgeben (DOSERR)

     ********************************************************
     * Name:       DRVSEL                                   *
     * Funktion:   Drive (A) auswählen und Motor starten    *
     * Input:      A: gewünschte Drive# (0-3)               *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

     Das gewünschte Laufwerk wird als "aktuelles Laufwerk" bei 4308H vermerkt,
     sein Bit-Muster wird nach 4309H sowie seine PDRIVE-Parameter nach
     430AH-4311H übertragen; wenn erforderlich, wird auf Double Density oder
     8 Zoll Laufwerke umgeschaltet und die Motoren der Drives werden gestartet.

445B C3 76 47     JP      4776          ;weiter bei 4776

     ********************************************************
     * Name:       TSTDSK                                   *
     * Funktion:   Drive (A) auswählen, Motor starten und   *
     *             testen, ob Diskette eingelegt            *
     * Input:      A: gewünschte Drive# (0-3)               *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

     Es wird DRVSEL (445BH) aufgerufen und (wenn dabei kein Fehler auftrat)
     anschließend geprüft, ob sich eine drehende Diskette im Laufwerk befindet.

445E C3 EC 47     JP      47EC          ;weiter bei 47EC

     ********************************************************
     * Name:       USRINS                                   *
     * Funktion:   Benutzer-Routine (*name) einfügen        *
     * Input:      HL: zeigt auf Kontroll-Block der         *
     *                 Benutzer-Routine                     *
     * Verändert:  HL, DE, BC                               *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4461 3E 2B        LD      A,2B          ;Code für SYS9/SYS
```

```
4463 EF          RST     28          ;SYS-File laden und starten

     ********************************************************
     * Name:       USRDEL                                   *
     * Funktion:   Benutzer-Routine (*name) löschen         *
     * Input:      HL: zeigt auf Kontroll-Block der         *
     *                 Benutzer-Routine                     *
     * Verändert:  HL, DE, BC                               *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4464 3E 4B       LD      A,4B        ;Code für SYS9/SYS
4466 EF          RST     28          ;SYS-File laden und starten

     ********************************************************
     * Name:       TEXTTV                                   *
     * Funktion:   Text (HL) auf Bildschirm ausgeben        *
     * Input:      HL: zeigt auf Text, Ende = 03H oder 0DH  *
     * Verändert:  AF                                       *
     * Output:     --                                       *
     ********************************************************

4467 C3 A6 4B    JP      4BA6        ;weiter bei 4BA6

     ********************************************************
     * Name:       TEXTLP                                   *
     * Funktion:   Text (HL) auf Drucker ausgeben           *
     * Input:      HL: zeigt auf Text, Ende = 03H oder 0DH  *
     * Verändert:  AF                                       *
     * Output:     --                                       *
     ********************************************************

446A C3 BC 4B    JP      4BBC        ;weiter bei 4BBC

     ********************************************************
     * Name:       TIME                                     *
     * Funktion:   aktuelle Uhrzeit im Buffer (HL) im       *
     *             Format HH:MM:SS ablegen                  *
     * Input:      HL: zeigt auf 8-Byte Buffer              *
     * Verändert:  HL=HL+8, DE, BC, AF                      *
     * Output:     --                                       *
     ********************************************************

446D C3 A7 44    JP      44A7        ;weiter bei 44A7

     ********************************************************
     * Name:       DATE                                     *
     * Funktion:   aktuelles Datum im Buffer (HL) im        *
     *             Format MM/TT/JJ ablegen                  *
     * Input:      HL: zeigt auf 8-Byte Buffer              *
     * Verändert:  HL=HL+8, DE, BC, AF                      *
     * Output:     --                                       *
     ********************************************************

4470 C3 C2 44    JP      44C2        ;weiter bei 44C2
```

```
     ********************************************************
     * Name:       INSEXT                                   *
     * Funktion:   Wenn Filespec (DE) keinen File-Typ       *
     *             enthält, wird File-Typ (HL) eingesetzt   *
     * Input:      DE: zeigt auf Filespec                   *
     *             HL: zeigt auf 3-Byte File-Typ            *
     * Verändert:  HL, AF                                   *
     * Output:     --                                       *
     ********************************************************

4473 3E A3          LD      A,A3          ;Code für SYS1/SYS
4475 EF             RST     28            ;SYS-File laden und starten
                    ----------------------
                    Unbenutzt (?)
4476 00             NOP
4477 00             NOP
4478 00             NOP
4479 00             NOP
447A 00             NOP
447B 00             NOP
447C 18 D9          JR      4457
447E AF             XOR     A
447F C9             RET

                    ----------------------
                    FCB zum Laden und Starten von Benutzer-Programmen
                    (vorbesetzt zum Laden des PDRIVE-Sectors bei 4D3BH)
4480 82       +0    ;Bit 7: gesetzt wenn FCB geöffnet
                    ;Bits 6-3: ./.
                    ;Bit 2: muß gelöscht sein wegen Abfrage in SYS0 bei 4823H
                    ;Bit 1: gesetzt wenn der FCB nicht zu einer File gehört,
                    ;       sondern zu einer ganzen Diskette
                    ;Bit 0: gesetzt wenn die Sectoren dieser File wie
                    ;       DIR-Sectoren lesegeschützt geschrieben werden sollen
4481 20       +1    ;Bit 7: gesetzt wenn Logische Recordlänge (LRL) <> 256D
                    ;Bit 6: (1) Der EOF-Wert soll nach dem Schreiben nur dann auf
                    ;           den NEXT-Wert gesetzt werden, wenn dieser größer
                    ;           als der EOF-Wert ist.
                    ;       (0) Der EOF-Wert soll nach jedem Schreiben auf den
                    ;           NEXT-Wert gesetzt werden.
                    ;Bit 5: gelöscht wenn der aktuelle Sector im Buffer steht
                    ;Bit 4: gesetzt wenn die Daten im Buffer noch auf Diskette
                    ;       geschrieben werden müssen
                    ;Bit 3: gesetzt in NEWDOS/80 Version 2
                    ;Bits 2-0: Zugriffs-Level (0-7) aufgrund eines Passwortes
4482 00       +2    ;Bit 7: gesetzt wenn ASE=N (siehe DOS-Befehl ATTRIB)
                    ;Bit 6: gesetzt wenn ASC=N (siehe DOS-Befehl ATTRIB)
                    ;Bit 5: gesetzt wenn UDF=Y (siehe DOS-Befehl ATTRIB)
                    ;Bits 4-0: ./.
4483 00 42    +3    ;Adresse des File-Buffers
4485 00       +5    ;3. Byte des NEXT-Wertes, gibt die relative Position
                    ;innerhalb des aktuellen Sectors an
4486 00       +6    ;Drive# der File bzw. der Diskette
4487 FF       +7    ;DEC (Directory Entry Code) vom FPDE der File
4488 00       +8    ;3. Byte des EOF-Wertes, gibt die relative Position
                    ;innerhalb des letzten Sectors an
4489 00       +9    ;Logische Recordlänge (1-256D)
448A 02 00    +0A   ;2. und 1. Byte des NEXT-Wertes. Die 3 Bytes des NEXT-Wertes
                    ;geben im RBA-Format an, wohin/woher der nächste Record zu
                    ;schreiben/zu lesen ist, wonach der NEXT-Wert jeweils
```

```
                    ;entsprechend erhöht wird.
448C FF FF    +0C   ;2. und 1. Byte des EOF-Wertes (End of File). Die 3 Bytes des
                    ;EOF-Wertes geben im RBA-Format die aktuelle Länge der File
                    ;an.
448E FF FF    +0E   ;gibt an, wo der 1. Datenblock dieser File auf Diskette steht
                    ;a) 1. Byte = FF: Es steht kein Datenblock mehr auf Diskette.
                    ;b) 1. Byte = FE: Im 2. Byte steht der DEC (Directory Entry
                    ;   Code) für den nächsten FXDE dieser File.
                    ;c) 1. Byte = 00-FD: Gibt die LUMP# an, in der dieser
                    ;   Datenblock auf Diskette beginnt. Bits 7-5 des 2. Bytes
                    ;   geben dann die Anzahl der GRANS an, die zwischen dem
                    ;   Beginn des LUMPS und dem Beginn des Datenblocks liegen
                    ;   und Bits 4-0 des 2. Bytes die um 1 verminderte Anzahl der
                    ;   GRANS, aus denen dieser Datenblock besteht.
4490 FF FF    +10H  ;gibt an, wo der 2. Datenblock dieser File auf Diskette steht
                    ;(Bedeutung der Bytes wie beim 1. Datenblock)
4492 FF FF    +12H  ;gibt an, wo der 3. Datenblock dieser File auf Diskette steht
                    ;(Bedeutung der Bytes wie beim 1. Datenblock)
4494 FF FF    +14H  ;gibt an, wo der 4. Datenblock dieser File auf Diskette steht
                    ;(Bedeutung der Bytes wie beim 1. Datenblock)
4496 FF FF    +16H  ;wenn <> FFFF: in den folgenden 8 Bytes sind Angaben über
                    ;die 4 Datenblöcke eines FXDE, der zu dieser File gehört,
                    ;enthalten und hier steht die Anzahl der GRANS, die
                    ;zwischen dem Beginn der File und dem Beginn des
                    ;1. Erweiterungs-Datenblocks dieses FXDE's liegen.
4498 FF FF    +18H  ;gibt an, wo der 1. Erweiterungs-Datenblock eines FXDE's auf
                    ;Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)
449A FF FF    +1A   ;gibt an, wo der 2. Erweiterungs-Datenblock eines FXDE's auf
                    ;Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)
449C FF FF    +1C   ;gibt an, wo der 3. Erweiterungs-Datenblock eines FXDE's auf
                    ;Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)
449E FF FF    +1E   ;gibt an, wo der 4. Erweiterungs-Datenblock eines FXDE's auf
                    ;Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)

                    ----------------------
                    Interrupt-Routine für CLOCK-Befehl
44A0 00 00          ;Zeiger auf Interrupt-Kette
44A2 28             ;40D Aufrufe pro Sekunde
44A3 01             ;Interrupt-Zähler
44A4 21 35 3C       LD      HL,3C35       ;Zeiger auf Bildschirm-RAM

     ********************************************************
     * Name:       TIME (Fortsetzung von 446DH)             *
     * Funktion:   aktuelle Uhrzeit im Buffer (HL) im       *
     *             Format HH:MM:SS ablegen                  *
     * Input:      HL: zeigt auf 8-Byte Buffer              *
     * Verändert:  HL=HL+8, DE, BC, AF                      *
     * Output:     --                                       *
     ********************************************************

44A7 11 43 40       LD      DE,4043       ;Zeiger auf HH,MM,SS
44AA 06 3A          LD      B,3A          ;Trennzeichen ":"
44AC 0E 03          LD      C,03          ;Zähler: 3 Werte
44AE 36 2F          LD      (HL),2F       ;Umwandlung in Dezimalzahl
44B0 1A             LD      A,(DE)
44B1 34             INC     (HL)          ;Zehnerstelle
44B2 D6 0A          SUB     0A
44B4 30 FB          JR      NC,44B1
44B6 23             INC     HL            ;Zeiger auf Buffer +1
44B7 C6 3A          ADD     3A
```

```
44B9 77          LD      (HL),A       ;Einerstelle
44BA 23          INC     HL           ;Zeiger auf Buffer +1
44BB 1B          DEC     DE           ;nächster Wert
44BC 0D          DEC     C
44BD C8          RET     Z
44BE 70          LD      (HL),B       ;Trennzeichen
44BF 23          INC     HL
44C0 18 EC       JR      44AE         ;weiter

     ********************************************************
     * Name:       DATE (Fortsetzung von 4470H)             *
     * Funktion:   aktuelles Datum im Buffer (HL) im        *
     *             Format MM/TT/JJ ablegen                  *
     * Input:      HL: zeigt auf 8-Byte Buffer              *
     * Verändert:  HL=HL+8, DE, BC, AF                      *
     * Output:     --                                       *
     ********************************************************

44C2 11 46 40    LD      DE,4046      ;Zeiger auf MM,TT,JJ
44C5 06 2F       LD      B,2F         ;Trennzeichen "/"
44C7 18 E3       JR      44AC         ;weiter wie TIME
                 ----------------------
                 Interrupt-Routine zum weitersetzen der Uhr
44C9 EB 44       ;Zeiger auf Interrupt-Kette
44CB 28          ;40D Aufrufe pro Sekunde
44CC 01          ;Interrupt-Zähler
44CD 21 41 40    LD      HL,4041      ;Zeiger auf Uhrzeit und Datum
44D0 E5          PUSH    HL
44D1 11 AC 43    LD      DE,43AC      ;nach 43AC-43B1 übertragen
44D4 01 06 00    LD      BC,0006
44D7 ED B0       LDIR
44D9 11 7C 40    LD      DE,407C      ;Tabelle zum erhöhen der Uhrzeit
44DC E1          POP     HL
44DD 06 03       LD      B,03         ;Zähler: 3 Werte
44DF 34          INC     (HL)
44E0 1A          LD      A,(DE)
44E1 96          SUB     (HL)
44E2 D0          RET     NC           ;wenn kein Übertrag
44E3 71          LD      (HL),C
44E4 13          INC     DE           ;nächster Wert
44E5 23          INC     HL
44E6 10 F7       DJNZ    44DF
44E8 23          INC     HL           ;Tag erhöhen
44E9 34          INC     (HL)
44EA C9          RET                  ;
                 ----------------------
                 Interrupt-Routine für blinkenden Cursor
44EB 00 00       ;Zeiger auf Interrupt-Kette
44ED 14          ;20D Aufrufe pro Sekunde
44EE 01          ;Interrupt-Zähler
44EF DD E5       PUSH    IX           ;steht IX
44F1 E1          POP     HL           ;auf
44F2 11 1D 40    LD      DE,401D      ;DCB für
44F5 DF          RST     18           ;Bildschirm ?
44F6 C8          RET     Z            ;wenn ja
44F7 3A 22 40    LD      A,(4022)     ;Cursor off ?
44FA B7          OR      A
44FB C8          RET     Z            ;wenn ja
44FC 2A 20 40    LD      HL,(4020)    ;Cursor-Position holen
44FF BE          CP      (HL)
```

```
4500 36*8F        LD      (HL),8F       ;SYSTEM BI (Cursor Zeichen)
     a) wenn SYSTEM BH=N oder DOS-Befehl BLINK,N:
4502*C9           RET
     b) wenn SYSTEM BH=Y oder DOS-Befehl BLINK,Y:
4502*C8           RET     Z
4503 77           LD      (HL),A        ;Zeichen unter Cursor anzeigen
4504 C9           RET
                  ----------------------
                  NEWDOS/80's Erweiterung der Bildschirm-Routine
     a) wenn SYSTEM BF=N oder DOS-Befehl LCDVR,N:
4505*18 0C        JR      4513
     b) wenn SYSTEM BF=Y oder DOS-Befehl LCDVR,Y:
4505*38 0C        JR      C,4513
4507 CD 53 48     CALL    4853          ;HL=Cursor-Position
450A 79           LD      A,C           ;Zeichen prüfen
450B D6 20        SUB     20
450D FE 60        CP      60
450F 79           LD      A,C
4510 DA 7D 04     JP      C,047D        ;wenn 20H .. 7FH
4513 C3 58 04     JP      0458          ;wenn 00H .. 1FH oder 80H .. FFH
                  ----------------------
                  NEWDOS/80's Erweiterung der Tastatur-Routine
4516 3A 69 43     LD      A,(4369)      ;ist
4519 EE 20        XOR     20            ;CHAINING
451B E6 68        AND     68            ;aktiv ?
451D 3E CB        LD      A,CB          ;Code für SYS9/SYS
451F CC DD 49     CALL    Z,49DD        ;wenn ja: SYS-File laden und starten
4522 C8           RET     Z
4523 21 BE 45     LD      HL,45BE       ;Abfrage von 'DFG', '123' und 'JKL'
4526 36 C9        LD      (HL),C9       ;sperren
4528 E5           PUSH    HL
4529 21*36*40     LD      HL,4036       ;Zeiger auf Tastatur-Status
452C 7E           LD      A,(HL)        ;der letzten Taste
452D B7           OR      A
452E 3E*00        LD      A,00          ;SYSTEM AV (Repeat-Faktor)
4530 20 03        JR      NZ,4535
4532 32 80 45     LD      (4580),A
4535 AF           XOR     A
4536 F6*00        OR      00            ;Repeat-Zähler, wird bei jedem
                                        ;RTC-Interrupt um 1 vermindert
     a) wenn SYSTEM AU=N (Repeat Funktion aus):
4538*18 09        JR      4543
     b) wenn SYSTEM AU=Y (Repeat Funktion ein):
4538*20 09        JR      NZ,4543
453A B6           OR      (HL)          ;Repeat-Taste noch gedrückt ?
453B 20 04        JR      NZ,4541
453D 3D           DEC     A
453E 32 73 45     LD      (4573),A
4541 36 00        LD      (HL),00       ;Status der Repeat-Taste löschen
4543 2E 36        LD      L,36          ;Zeiger auf Tastatur-Status
4545 01 01 38     LD      BC,3801       ;Zeiger auf Tastatur-Matrix
4548 16 FF        LD      D,FF
454A 0A           LD      A,(BC)        ;Tastatur-Matrix lesen
454B 5F           LD      E,A
454C AE           XOR     (HL)          ;mit Status vergleichen
454D 73           LD      (HL),E
454E A3           AND     E
454F 20 0D        JR      NZ,455E       ;wenn eine neue Taste gedrückt ist
4551 7A           LD      A,D
4552 C6 08        ADD     08
```

```
4554 57          LD      D,A
4555 2C          INC     L            ;nächste Matrix-Zeile
4556 CB 01       RLC     C
4558 F2 4A 45    JP      P,454A
455B AF          XOR     A            ;Ende der Matrix
455C 18 0F       JR      456D
455E 5F          LD      E,A          ;eine neue Taste ist gedrückt
455F 14          INC     D
4560 0F          RRCA
4561 30 FC       JR      NC,455F
4563 C5          PUSH    BC           ;Register retten
4564 D5          PUSH    DE
4565 22 2A 45    LD      (452A),HL    ;Matrix-Zeile für Repeat retten
4568 CD 0B 04    CALL    040B         ;ASCII-Code berechnen, BREAK -> RST 28H
456B E1          POP     HL
456C C1          POP     BC           ;Register zurück
456D B7          OR      A
456E 57          LD      D,A          ;D = ASCII-Code
456F 28 18       JR      Z,4589       ;wenn keine neue Taste gedrückt
4571 7C          LD      A,H
4572 FE*FF       CP      FF
4574 32 73 45    LD      (4573),A
4577 3E*00       LD      A,00         ;SYSTEM AV (Repeat-Faktor)
4579 20 06       JR      NZ,4581
457B 0A          LD      A,(BC)
457C A5          AND     L
     a) wenn SYSTEM AC=Y (Tasten-Entprellung von NEWDOS/80):
457D*28 EE       JR      Z,456D
     b) wenn SYSTEM AC=N (keine Tasten-Entprellung):
457D 3E EE       LD      A,EE
457F 3E*FF       LD      A,FF
4581 32 37 45    LD      (4537),A     ;Repeat-Zähler setzen
4584 3E 02       LD      A,02
4586 32 80 45    LD      (4580),A
4589 CD BF 45    CALL    45BF         ;'DFG', '123' und 'JKL' abfragen
458C E1          POP     HL           ;HL=45BE
458D 36 00       LD      (HL),00      ;Abfrage 'DFG', '123' und 'JKL' erlauben
     a) wenn SYSTEM AQ=N (CLEAR-Taste verboten):
458F FE*1F       CP      1F           ;mit CLEAR-Taste vergleichen
     b) wenn SYSTEM AQ=Y (CLEAR-Taste erlaubt):
458F FE*00       CP      00
4591 28 1D       JR      Z,45B0
     a) wenn SYSTEM BF=N oder DOS-Befehl LCDVR,N:
4593*C9          RET
     b) wenn SYSTEM BF=Y oder DOS-Befehl LCDVR,Y:
4593*4F          LD      C,A
4594 E6 DF       AND     DF           ;Buchstabe ?
4596 D6 41       SUB     41
4598 FE 1A       CP      1A
459A 79          LD      A,C
459B 38 15       JR      C,45B2       ;wenn Buchstabe
459D FE 20       CP      20           ;Leertaste ?
459F C0          RET     NZ           ;wenn nein
45A0 21 7F 38    LD      HL,387F      ;gesamte Tastatur-Matrix
45A3 7E          LD      A,(HL)       ;abfragen
45A4 23          INC     HL           ;Zeiger auf SHIFT-Tasten
45A5 A6          AND     (HL)         ;SHIFT gedrückt ?
45A6 0F          RRCA                 ;SHIFT + Leertaste ?
45A7 79          LD      A,C
45A8 D0          RET     NC           ;wenn nein
```

```
45A9 21 B4 45     LD      HL,45B4       ;SHIFT + Leertaste:
45AC 7E           LD      A,(HL)        ;Toggle Flag für Umschaltung
45AD EE C9        XOR     C9            ;a-z -> A-Z oder a-z -> a-z
45AF 77           LD      (HL),A
45B0 AF           XOR     A
45B1 C9           RET
45B2 EE 20        XOR     20            ;Umschaltung für a-z:
     a) wenn SYSTEM BG=Y oder DOS-Befehl LC,Y:
45B4*C9           RET
     b) wenn SYSTEM BG=N oder DOS-Befehl LC,N:
45B4*00           NOP

     ************************************************************
     * Name:       UPCASE                                       *
     * Funktion:   wandelt Kleinbuchstaben in                   *
     *             Großbuchstaben um                            *
     * Input:      A: ASCII-Code eines Zeichens                 *
     * Verändert:  F                                            *
     * Output:     A: ASCII-Code für Großbuchstaben             *
     ************************************************************

45B5 FE 61        CP      61            ;< "a" ?
45B7 D8           RET     C             ;wenn ja
45B8 FE 7B        CP      7B            ;> "z" ?
45BA D0           RET     NC            ;wenn ja
45BB D6 20        SUB     20            ;a-z -> A-Z
45BD C9           RET
                  ----------------------
                  Abfrage von 'DFG', '123' und 'JKL'
     a) wenn Tastatur-Routine nicht aktiv ist:
45BE*00           NOP                   ;Abfrage erlaubt
     b) wenn Tastatur-Routine gerade aktiv ist:
45BE*C9           RET                   ;Abfrage nicht erlaubt
45BF 21 69 43     LD      HL,4369       ;Status des DOS prüfen:
45C2 7E           LD      A,(HL)        ;CHAINING aktiv oder RST 28H aktiv
45C3 E6 6C        AND     6C            ;oder RUN-ONLY-Modus ?
45C5 20 26        JR      NZ,45ED       ;wenn ja: Abfrage nicht erlaubt
                  ----------------------
45C7 3A 01 38     LD      A,(3801)      ;'DFG' gedrückt ?
45CA FE D0        CP      D0
     a) wenn SYSTEM AF=N ('DFG' verboten):
45CC*18 05        JR      45D3
     b) wenn SYSTEM AF=Y ('DFG' erlaubt):
45CC*20 05        JR      NZ,45D3       ;wenn nein
45CE 3E E3        LD      A,E3          ;Code für SYS1/SYS
45D0 0E 04        LD      C,04
45D2 EF           RST     28            ;SYS-File laden und starten
                  ----------------------
45D3 3A 10 38     LD      A,(3810)      ;'123' gedrückt ?
45D6 FE 0E        CP      0E
     a) wenn SYSTEM AE=N ('123' verboten):
45D8*18 0E        JR      45E8
     b) wenn SYSTEM AE=Y ('123' erlaubt):
45D8*20 0E        JR      NZ,45E8       ;wenn nein
45DA 3A BE 45     LD      A,(45BE)      ;erfolgt die Abfrage von
45DD D6 C9        SUB     C9            ;der Tastatur-Routine her ?
45DF CA 0D 44     JP      Z,440D        ;wenn ja: DEBUG über 440D starten
45E2 F1           POP     AF            ;bei Abfrage durch die
45E3 C1           POP     BC            ;Interrupt-Routine:
45E4 D1           POP     DE            ;Register zurück
```

```
45E5 E1           POP     HL            ;und
45E6 18 22        JR      460A          ;DEBUG über 460A
                  ----------------------
45E8 3A 02 38     LD      A,(3802)      ;'JKL' gedrückt ?
45EB FE 1C        CP      1C
45ED 7A           LD      A,D           ;Tastatur-Zeichen zurück
     a) wenn SYSTEM AD=N ('JKL' verboten):
45EE*C9           RET
     b) wenn SYSTEM AD=Y ('JKL' erlaubt):
45EE*C0           RET     NZ            ;wenn nein
45EF 3E A5        LD      A,A5          ;Code für SYS3/SYS
45F1 EF           RST     28            ;SYS-File laden und starten
                  ----------------------
                  Interrupt-Routine über RST 38H
45F2 F5           PUSH    AF            ;AF retten
45F3 E5           PUSH    HL            ;HL retten
45F4 D5           PUSH    DE            ;DE retten
45F5 C5           PUSH    BC            ;BC retten
45F6 3A E0 37 ┐   LD      A,(37E0)      ;Interrupt-Status lesen
45F9 07       │   RLCA                  ;Interrupt durch FDC ?
45FA CD 53 40 ┘   CALL    4053          ;wenn ja: (37EC) lesen
45FD DC 10 46     CALL    C,4610        ;wenn RTC-Interrupt: alle definierten
                                        ;Interrupt-Routinen bearbeiten
4600 CD BE 45     CALL    45BE          ;'DFG', '123' und 'JKL' abfragen
4603 C1           POP     BC            ;BC zurück
4604 D1           POP     DE            ;DE zurück
4605 E1           POP     HL            ;HL zurück
4606 F1           POP     AF            ;AF zurück
4607 FB           EI                    ;Interrupts wieder erlauben
4608 C9           RET                   ;zurück zum Hauptprogramm

     ************************************************************
     * Name:       DEBUG (Fortsetzung von 440DH)                *
     * Funktion:   DEBUG aufrufen                               *
     * Input:      --                                           *
     * Verändert:  --                                           *
     * Output:     --                                           *
     ************************************************************

4609 F5           PUSH    AF            ;AF retten
460A 3E 27        LD      A,27          ;Code für SYS5/SYS
460C EF           RST     28            ;SYS-File laden und starten
                  ----------------------
                  Interrupt-Routine aufrufen
460D 70           LD      (HL),B        ;Interrupt-Zähler setzen
460E 23           INC     HL            ;HL = Startadresse der Routine
460F E9           JP      (HL)          ;Interrupt-Routine aufrufen
                  ----------------------
                  Bearbeitung von RTC-Interrupts (Real-Time-Clock)
4610 21 40 40     LD      HL,4040       ;Zeiger auf Interrupt-Zähler
4613 34           INC     (HL)          ;Zähler +1
4614 21 37 45     LD      HL,4537       ;Zeiger auf Tastatur Repeat-Zähler
4617 7E           LD      A,(HL)        ;Zähler = 0 ?
4618 B7           OR      A
4619 28 01        JR      Z,461C        ;wenn ja
461B 35           DEC     (HL)          ;wenn nein: Zähler -1
                  ----------------------
                  Bearbeitung der RTC-Interrupt-Kette
461C 21 C9 44     LD      HL,44C9       ;Zeiger auf 1. Kontroll-Block
461F 7C           LD      A,H           ;Zeiger = 0000 ?
```

```
4620 B5          OR      L
4621 C8          RET     Z              ;wenn ja, Ende der Kette
4622 5E          LD      E,(HL)         ;Zeiger auf Kontroll-Block der
4623 23          INC     HL             ;nächsten Interrupt-Routine
4624 56          LD      D,(HL)         ;aus der Kette holen
4625 D5          PUSH    DE             ;Zeiger retten
4626 23          INC     HL             ;Zeiger auf Kontroll-Block +1
4627 46          LD      B,(HL)         ;Aufruf alle wieviel Interrupts
4628 23          INC     HL             ;Zeiger +1
4629 35          DEC     (HL)           ;Interrupt-Zähler -1
462A CC 0D 46    CALL    Z,460D         ;wenn 0: Interrupt-Routine aufrufen
462D E1          POP     HL             ;Zeiger auf nächsten Kontroll-Block
462E 18 EF       JR      461F           ;nächste Interrupt-Routine bearbeiten
```

```
*********************************************************
* Name:       READS                                     *
* Funktion:   liest einen Sector von Diskette           *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             HL: Zeiger auf zu benutzenden Buffer      *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

READS liest einen physikalischen Sector von Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
4630 3E 88       LD      A,88           ;READ-SECTOR-Kommando für FDC
4632 18 0E       JR      4642           ;weiter bei 4642
```

```
*********************************************************
* Name:       TESTS                                     *
* Funktion:   testet einen Sector auf Fehler (Verify)   *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

TESTS testet einen physikalischen Sector auf Diskette, ob er ohne Fehler lesbar ist. Dazu wird versucht, diesen Sector von Diskette zu lesen, wobei als Buffer einfach das BASIC-ROM im Bereich 0100H-02FEH benutzt wird. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
4634 E5          PUSH    HL             ;HL retten
4635 26 01       LD      H,01           ;Buffer = 01XX
4637 CD 30 46    CALL    4630           ;Sector lesen
463A E1          POP     HL             ;HL zurück
463B C9          RET
```

```
     ********************************************************
     * Name:       WRITDS                                   *
     * Funktion:   schreibt einen Sector des Directory auf  *
     *             Diskette                                 *
     * Input:      DE: gewünschte Sector# (Disk-relativ)    *
     *             HL: Zeiger auf zu benutzenden Buffer     *
     *             (4308H): gewünschte Drive# (0-3)         *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

WRITDS schreibt einen physikalischen Sector mit dem Data Adress Mark für Directory-Sectoren auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

a) wenn SYSTEM BN=N (Data Adress Mark für TRS-80 Model I Disketten):

```
463C 3E*A9        LD      A,A9          ;WRITE-SECTOR-Kommando für Directory mit
                                        ;Data Adress Mark FAH für FD 1771 (SD)
                                        ;bzw. F8H für FD 1791 (DD)
```

b) wenn SYSTEM BN=Y (Data Adress Mark für TRS-80 Model III Disketten):

```
463C 3E*AB        LD      A,AB          ;WRITE-SECTOR-Kommando für Directory mit
                                        ;Data Adress Mark F8H für FD 1771 (SD)
                                        ;und FD 1791 (DD: aus "AB" wird "A9" !)
463E 18 02        JR      4642          ;weiter bei 4642
```

```
     ********************************************************
     * Name:       WRITES                                   *
     * Funktion:   schreibt einen Sector auf Diskette       *
     * Input:      DE: gewünschte Sector# (Disk-relativ)    *
     *             HL: Zeiger auf zu benutzenden Buffer     *
     *             (4308H): gewünschte Drive# (0-3)         *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

WRITES schreibt einen physikalischen Sector auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
4640 3E A8        LD      A,A8          ;WRITE-SECTOR-Kommando für FDC mit
                                        ;"normalem" Data Adress Marker FBH.
                    ----------------------
4642 32 C4 46     LD      (46C4),A      ;FDC-Kommando nach 46C4
4645 E6 20        AND     20            ;READ oder WRITE ?
4647 C5           PUSH    BC            ;BC retten
4648 01 1A 02     LD      BC,021A       ;"LD  A,(DE)" + "LD  (BC),A"
464B 28 05        JR      Z,4652        ;wenn READ
464D 3E 08        LD      A,08          ;Offset für Fehlercode bei WRITE = 08H
464F 01 12 0A     LD      BC,0A12       ;"LD  (DE),A" + "LD  A,(BC)"
4652 32 31 47     LD      (4731),A      ;Offset für Fehlercode (READ=0/WRITE=8)
4655 ED 43 FC 46  LD      (46FC),BC     ;Richtung des Daten-Transfers nach 46FC
4659 06*0A        LD      B,0A          ;SYSTEM AM: Anzahl Versuche bei Errors
465B CD 73 47     CALL    4773          ;Select Drive (4308H) und Motor starten
465E 20 18        JR      NZ,4678       ;wenn Error
                    ----------------------
4660 C5           PUSH    BC            ;B (Zähler für Anzahl Versuche) retten
4661 D5           PUSH    DE            ;DE (Disk-relative Sector#) retten
```

```
4662 E5          PUSH    HL          ;HL (Zeiger auf Buffer) retten
4663 3A 0E 43    LD      A,(430E)    ;A = PDRIVE+4: SPT (Sectors pro Track)
4666 EB          EX      DE,HL       ;=> HL = gewünschte Sector#
4667 CD B4 4C    CALL    4CB4        ;1) C=A (Sectors pro Track)
                                     ;2) DIV: HL=INT(HL/A), A=Rest
466A 55          LD      D,L         ;D = gesuchte Track#
466B 5F          LD      E,A         ;E = gesuchte Sector#
466C 21 0D 43    LD      HL,430D     ;HL zeigt auf PDRIVE+3: TC (Track Count)
466F 7A          LD      A,D         ;ist die berechnete Track# größer als
4670 BE          CP      (HL)        ;die Anzahl der vorhandenen Tracks ?
4671 38 09       JR      C,467C      ;wenn nein
                 ---------------------
4673 3E 14       LD      A,14        ;Fehlercode für "TRACK # TOO HIGH"
4675 E1          POP     HL          ;HL (Zeiger auf Buffer) zurück
4676 D1          POP     DE          ;DE (Disk-relative Sector#) zurück
4677 C1          POP     BC          ;B (Zähler für Anzahl Versuche) zurück
4678 B7          OR      A           ;Z-Flag löschen, A=Fehlercode
4679 C3 37 47    JP      4737        ;weiter bei 4737
                 ---------------------
467C 3A 11 43    LD      A,(4311)    ;PDRIVE+7: TI und TD
467F 47          LD      B,A         ;nach B
4680 CB 40       BIT     0,B         ;Double Density (DD) ?
4682 28 05       JR      Z,4689      ;wenn nein
4684 21 C4 46    LD      HL,46C4     ;Zeiger auf Kommando für FDC
4687 CB 8E       RES     1,(HL)      ;aus ABH wird A9H (Rest unverändert)
                 ---------------------
4689 21 09 43    LD      HL,4309     ;Zeiger auf Bit-Maske für aktuelle Drive
468C CB 48       BIT     1,B         ;TI=J oder TI=K (Track# ab 1) ?
468E 28 01       JR      Z,4691      ;wenn nein
4690 14          INC     D           ;wenn ja: Track# +1
                 ---------------------
4691 D5          PUSH    DE          ;D (gewünschte Track#) retten
4692 CB 50       BIT     2,B         ;TI=L (doppelte Step-Impulse) ?
4694 28 02       JR      Z,4698      ;wenn nein
4696 CB 02       RLC     D           ;Pseudo-Track# für SEEK verdoppeln
                 ---------------------
4698 CB 70       BIT     6,B         ;Diskette doppelseitig (DS) ?
469A 28 09       JR      Z,46A5      ;wenn nein
469C CB 09       RRC     C           ;=> C = Sectors pro Track pro Seite
469E 7B          LD      A,E         ;ist die gewünschte Sector#
469F 91          SUB     C           ;auf der Rückseite der Diskette ?
46A0 38 03       JR      C,46A5      ;wenn nein
46A2 5F          LD      E,A         ;E = gesuchte Sector# auf der Rückseite
46A3 CB DE       SET     3,(HL)      ;Bit für Rückseite (Seite 1) setzen
46A5 CD 67 47    CALL    4767        ;gewünschte Disk-Seite anwählen
                 ---------------------
46A8 CB 60       BIT     4,B         ;TI=I oder TI=M (Sector# ab 1) ?
46AA 28 01       JR      Z,46AD      ;wenn nein
46AC 1C          INC     E           ;wenn ja: Sector# +1
                 ---------------------
46AD ED 53 EE 37 LD      (37EE),DE   ;zu suchende Track# und Sector# an FDC
46B1 0E 18       LD      C,18        ;SEEK-Kommando an FDC senden
46B3 CD 47 47    CALL    4747        ;und warten, bis FDC nicht mehr busy ist
46B6 F1          POP     AF          ;A = gewünschte Track#
46B7 C1          POP     BC          ;BC = Zeiger auf Buffer
46B8 C5          PUSH    BC          ;BC (Zeiger auf Buffer) retten
46B9 32 ED 37    LD      (37ED),A    ;gewünschte Track# ins Track-Register
46BC D5          PUSH    DE          ;D (Pseudo-Track# für SEEK) retten
46BD 11 EF 37    LD      DE,37EF     ;Zeiger auf Daten-Register des FDC
46C0 21 EC 37    LD      HL,37EC     ;Zeiger auf Kommando/Status-Register
```

```
46C3 36*00        LD      (HL),00       ;Kommando (READ/WRITE) an FDC senden
46C5 CD E3 47     CALL    47E3          ;55 us warten und Status-Register lesen
46C8 F3           DI                    ;Interrupts sperren
46C9 CB 46        BIT     0,(HL)        ;ist FDC busy ?
46CB 28 4A        JR      Z,4717        ;wenn nein, Kommando abbrechen
46CD 3A C4 46     LD      A,(46C4)      ;ist das auszuführende Kommando
46D0 CB 6F        BIT     5,A           ;das READ-Kommando ?
46D2 28 22        JR      Z,46F6        ;wenn ja
                  ----------------------
                  WRITE-SECTOR:
46D4 3E 83        LD      A,83          ;Status-Register des FDC auf
46D6 A6           AND     (HL)          ;Data Request und busy prüfen
46D7 E2 D4 46     JP      PO,46D4       ;wenn nein, warten
46DA 0A           LD      A,(BC)        ;1. Zeichen aus dem Buffer
46DB 12           LD      (DE),A        ;an Daten-Register des FDC schicken
46DC 03           INC     BC            ;Zeiger auf Buffer +1
46DD 0A           LD      A,(BC)        ;2. Zeichen aus dem Buffer
46DE 32 E8 46     LD      (46E8),A      ;nach 46E8
46E1 03           INC     BC            ;Zeiger auf Buffer +1
46E2 3E 01        LD      A,01          ;Status-Register des FDC auf
46E4 BE           CP      (HL)          ;busy und kein Data Request prüfen
46E5 28 FD        JR      Z,46E4        ;wenn ja, warten
46E7 3E*00        LD      A,00          ;2. Zeichen aus dem Buffer
46E9 12           LD      (DE),A        ;an Daten-Register des FDC schicken
46EA 0A           LD      A,(BC)        ;3. Zeichen aus dem Buffer holen
46EB 03           INC     BC            ;Zeiger auf Buffer +1
46EC CB 4E        BIT     1,(HL)        ;Data Request ?
46EE 20 0C        JR      NZ,46FC       ;wenn ja
46F0 CB 4E        BIT     1,(HL)        ;Data Request ?
46F2 20 08        JR      NZ,46FC       ;wenn ja
46F4 18 09        JR      46FF          ;weiter bei 46FF
                  ----------------------
                  READ-SECTOR:
46F6 3E 83        LD      A,83          ;Status-Register des FDC auf
46F8 A6           AND     (HL)          ;Data Request und busy prüfen
46F9 E2 F6 46     JP      PO,46F6       ;wenn nein, warten
                  ----------------------
                  READ + WRITE:
46FC*1A           LD      A,(DE)        ;bei READ: nächstes Zeichen vom FDC
46FD*02           LD      (BC),A        ;holen und in Buffer schreiben
                                        ;bei WRITE: Zeichen (A) an FDC schicken
                                        ;+ nächstes Zeichen aus dem Buffer holen
46FE 03           INC     BC            ;Zeiger auf Buffer +1
46FF CB 4E        BIT     1,(HL)        ;Data Request ?
4701 20 F9        JR      NZ,46FC       ;wenn ja
4703 CB 4E        BIT     1,(HL)        ;Data Request ?
4705 20 F5        JR      NZ,46FC       ;wenn ja
4707 CB 4E        BIT     1,(HL)        ;Data Request ?
4709 20 F1        JR      NZ,46FC       ;wenn ja
470B CB 46        BIT     0,(HL)        ;ist FDC noch busy ?
470D 28 08        JR      Z,4717        ;wenn nein
470F CB 4E        BIT     1,(HL)        ;Data Request ?
4711 20 E9        JR      NZ,46FC       ;wenn ja
4713 CB 7E        BIT     7,(HL)        ;ist Drive noch ready ?
4715 28 E8        JR      Z,46FF        ;wenn ja
                  ----------------------
                  Ende des READ/WRITE-Kommandos
4717 7E           LD      A,(HL)        ;Status-Register des FDC lesen
4718 36 D0        LD      (HL),D0       ;FORCE-INTERRUPT-Kommando an FDC
471A 23           INC     HL            ;Zeiger auf Track-Register des FDC
```

```
471B C1          POP     BC          ;B = Pseudo-Track# für SEEK
471C 70          LD      (HL),B      ;ins Track-Register schreiben
471D E1          POP     HL          ;HL (Zeiger auf Buffer) zurück
471E D1          POP     DE          ;DE (Disk-relative Sector#) zurück
471F C1          POP     BC          ;BC (Zähler für Anzahl Versuche) zurück
4720 FB          EI                  ;Interrupts erlauben
4721 E6 FC       AND     FC          ;Error-Bits maskieren
4723 28 12       JR      Z,4737      ;wenn kein Error
                 ----------------------
                 Fehlerbehandlung
4725 4F          LD      C,A         ;Error-Bits retten
4726 E6 9C       AND     9C          ;wenn "Write Protected", "Write Fault"
4728 28 06       JR      Z,4730      ;oder "Data Adress Mark": abbrechen !
472A 4F          LD      C,A         ;maskierte Error-Bits
472B 87          ADD     A           ;wenn "Drive Not Ready":
472C 28 02       JR      Z,4730      ;abbrechen (kein weiterer Versuch)
472E 10 0A       DJNZ    473A        ;schon "SYSTEM AM" Versuche gemacht ?
                 ----------------------
                 Fehlercode berechnen
4730 3E*00       LD      A,00        ;Offset für Fehlercode (READ=0, WRITE=8)
4732 3C          INC     A           ;Fehlercode +1
4733 CB 09       RRC     C           ;Error-Bit (C) nach rechts schieben
4735 30 FB       JR      NC,4732     ;solange, bis Error-Bit im Carry ist
4737 FB          EI                  ;Interrupts erlauben
4738 C1          POP     BC          ;BC (ursprünglicher Wert) zurück
4739 C9          RET                 ;READS/TESTS/WRITDS/WRITES fertig
                 ----------------------
                 Fehlerbehandlung: neuen Versuch starten
473A CB 61       BIT     4,C         ;"Record Not Found" Error ?
473C C4 42 47    CALL    NZ,4742     ;wenn ja: alle 2 Versuche RESTORE an FDC
473F C3 60 46    JP      4660        ;nächster Versuch
                 ----------------------
4742 CB 40       BIT     0,B         ;gerade Versuchs# ?
4744 C8          RET     Z           ;wenn ja
```

```
*********************************************************
* Name:       RESTOR                                    *
* Funktion:   RESTORE-Kommando an FDC senden und        *
*             warten, bis der FDC nicht mehr busy ist   *
* Input:      --                                        *
* Verändert:  AF, C                                     *
* Output:     --                                        *
*********************************************************
```

Im RESTORE-Kommando wird die Track Stepping Rate (TSR) verwendet, die in den PDRIVE-Parametern für das betreffende Laufwerk angegeben ist. Während darauf gewartet wird, daß der FDC nicht mehr busy ist, wird durch wiederholten Drive-Select dafür gesorgt, daß die Drive-Motoren weiterlaufen.

```
4745 0E 08       LD      C,08        ;RESTORE-Kommando nach C
```

```
*********************************************************
* Name:       FDCCMD                                    *
* Funktion:   Kommando (C) mit TSR an FDC senden und    *
*             warten, bis der FDC nicht mehr busy ist   *
* Input:      C: Kommando für den FDC                   *
* Verändert:  AF                                        *
* Output:     --                                        *
*********************************************************
```

Im Kommando für den FDC wird die Track Stepping Rate (TSR) eingesetzt, die in den PDRIVE-Parametern für das betreffende Laufwerk angegeben ist. Während darauf gewartet wird, daß der FDC nicht mehr busy ist, wird durch wiederholten Drive-Select dafür gesorgt, daß die Motoren der Drives weiterlaufen.

```
4747 3A 0C 43     LD      A,(430C)      ;PDRIVE+2: TI und TSR
474A E6 03        AND     03            ;TSR (Track Stepping Rate) maskieren
474C B1           OR      C             ;mit Kommando für FDC verknüpfen
474D 32 EC 37     LD      (37EC),A      ;ins Kommando-Register des FDC schreiben
```

```
*********************************************************
* Name:       WNBUSY                                    *
* Funktion:   warten, bis der FDC nicht mehr busy ist   *
* Input:      --                                        *
* Verändert:  AF                                        *
* Output:     --                                        *
*********************************************************
```

Während darauf gewartet wird, daß der FDC nicht mehr busy ist, wird durch wiederholten Drive-Select dafür gesorgt, daß die Motoren der Drives weiterlaufen.

```
4750 CD E3 47     CALL    47E3          ;55 us warten und Status des FDC lesen
4753 CB 47        BIT     0,A           ;ist der FDC noch busy ?
4755 C8           RET     Z             ;wenn nein
4756 07           RLCA                  ;ist Drive noch ready ?
4757 38 05        JR      C,475E        ;wenn nein: FORCE-INTERRUPT-Kommando
4759 CD 67 47     CALL    4767          ;Drive-Select: Motor weiterlaufen lassen
475C 18 F2        JR      4750          ;nächste Runde
```

```
*********************************************************
* Name:       FBREAK                                    *
* Funktion:   FORCE-INTERRUPT-Kommando an FDC senden    *
*             und warten, bis FDC nicht mehr busy ist   *
* Input:      --                                        *
* Verändert:  AF                                        *
* Output:     --                                        *
*********************************************************
```

```
475E 3E D0        LD      A,D0          ;FORCE-INTERRUPT-Kommando
4760 18 EB        JR      474D          ;an FDC senden
```

```
*********************************************************
* Name:       MOTONX (Fortsetzung von 4416H)            *
* Funktion:   Drive-Motoren weiterlaufen lassen         *
* Input:      --                                        *
* Verändert:  AF                                        *
* Output:     --                                        *
*********************************************************
```

Falls die Motoren der Drives noch an sind, dann wird durch erneuten Drive-Select dafür gesorgt, daß die Motoren weiterlaufen.

```
4762 3A EC 37     LD      A,(37EC)      ;Status-Register des FDC lesen
4765 07           RLCA                  ;laufen die Drive-Motoren noch ?
4766 D8           RET     C             ;wenn nein
```

```
      ********************************************************
      * Name:       MOTON                                    *
      * Funktion:   Drive-Motoren starten                    *
      * Input:      --                                       *
      * Verändert:  A                                        *
      * Output:     --                                       *
      ********************************************************

4767 3A 09 43     LD      A,(4309)      ;Bit-Muster für aktuelles Laufwerk
476A 32 E1 37     LD      (37E1),A      ;Drive-Select und Motor an
476D C9           RET
                  ----------------------
                  DRVSEL (Drive-Select) für Betrieb mit FCB
476E DD 7E 06     LD      A,(IX+06)     ;gewünschte Drive# bei FCB+6 holen
4771 18 03        JR      4776          ;weiter bei 4776
                  ----------------------
                  DRVSEL (Drive-Select) für aktuelles Laufwerk
4773 3A 08 43     LD      A,(4308)      ;Drive# für aktuelles Laufwerk

      ********************************************************
      * Name:       DRVSEL (Fortsetzung von 445BH)           *
      * Funktion:   Drive (A) auswählen und Motor starten    *
      * Input:      A: gewünschte Drive# (0-3)               *
      * Verändert:  --                                       *
      * Output:     AF: A=Fehlercode, wenn Z=0               *
      ********************************************************

      Das gewünschte Laufwerk wird als "aktuelles Laufwerk" bei 4308H vermerkt,
      sein Bit-Muster wird nach 4309H sowie seine PDRIVE-Parameter nach
      430AH-4311H übertragen; wenn erforderlich, wird auf Double Density oder
      8 Zoll Laufwerke umgeschaltet und die Motoren der Drives werden gestartet.

4776 E5           PUSH    HL            ;HL retten
4777 D5           PUSH    DE            ;DE retten
4778 C5           PUSH    BC            ;BC retten
4779 FE*01        CP      01            ;Drive# mit SYSTEM AL vergleichen
477B 4F           LD      C,A           ;C = gewünschte Drive#
477C 3E 20        LD      A,20          ;Fehlercode "ILLEGAL OR MISSING DRIVE #"
477E 30 5E        JR      NC,47DE       ;wenn Drive# größer als SYSTEM AL ist
4780 CD 5E 47     CALL    475E          ;FORCE-INTERRUPT-Kommando an FDC,
                                        ;warten bis FDC nicht mehr busy ist
4783 21 08 43     LD      HL,4308       ;Zeiger auf "aktuelle Drive#"
4786 7E           LD      A,(HL)        ;bisherige Drive# nach A
4787 71           LD      (HL),C        ;neue Drive# nach 4308
4788 B9           CP      C             ;alte Drive# = neue Drive# ?
4789 F5           PUSH    AF            ;Ergebnis in F retten
478A 6F           LD      L,A           ;HL = "aktuelle Track#" für altes Drive
478B 3A ED 37     LD      A,(37ED)      ;Position des Schreib/Lese-Kopfes im
478E 77           LD      (HL),A        ;alten Laufwerk nach 4300H-4303H retten
478F 69           LD      L,C           ;HL = "aktuelle Track#" für neues Drive
4790 E5           PUSH    HL            ;HL retten
                  ----------------------
                  Bit-Muster für gewünschte Drive# berechnen
4791 41           LD      B,C           ;Zähler = neue Drive#
4792 3E 80        LD      A,80          ;Bit 7 in A setzen
4794 04           INC     B             ;Zähler +1
4795 07           RLCA                  ;gesetztes Bit nach links routieren
4796 10 FD        DJNZ    4795          ;entsprechend oft wiederholen
4798 32 09 43     LD      (4309),A      ;errechnetes Bit-Muster nach 4309
                  ----------------------
```

```
                    PDRIVE-Block der gewünschten Drive# holen
479B 79             LD      A,C             ;HL=
479C 07             RLCA                    ;4371H
479D 07             RLCA                    ;+
479E 81             ADD     C               ;10D
479F 87             ADD     A               ;*
47A0 C6 71          ADD     71              ;neue
47A2 6F             LD      L,A             ;Drive#
47A3 22 99 43       LD      (4399),HL       ;Adresse des PDRIVE-Blocks nach 4399
47A6 0E 08          LD      C,08            ;die ersten 8 Bytes des PDRIVE-Blocks
47A8 11 0A 43       LD      DE,430A         ;nach 430A-4311
47AB ED B0          LDIR                    ;übertragen
                    ----------------------
47AD 21 EC 37       LD      HL,37EC         ;Status-Register des FDC lesen
47B0 5E             LD      E,(HL)          ;Ergebnis (Motor an J/N) nach E retten
                    ----------------------
                    Auswahl Single Density (SD) / Double Density (DD)
47B1 F3             DI                      ;Interrupts sperren
47B2 3A 11 43       LD      A,(4311)        ;PDRIVE+7: Bit 0 ist gesetzt wenn DD
47B5 F6 FE          OR      FE              ;für DD wird auf FD 1791 umgeschaltet
47B7 77             LD      (HL),A          ;(siehe Kapitel 1.2.1: 37EC)
47B8 36 D0          LD      (HL),D0         ;FORCE-INTERRUPT-Kommando an FDC senden
47BA FB             EI                      ;Interrupts wieder erlauben
                    ----------------------
47BB 23             INC     HL              ;Zeiger auf Track-Register des FDC
47BC C1             POP     BC              ;BC zeigt auf "aktuelle Track" des
47BD 0A             LD      A,(BC)          ;gewünschten Laufwerks, diese Track#
47BE 77             LD      (HL),A          ;ins Track-Register des FDC eintragen
                    ----------------------
                    Auswahl 5.25 Zoll / 8 Zoll Laufwerk
47BF 23             INC     HL              ;Zeiger auf 37EE
47C0 3A 10 43       LD      A,(4310)        ;PDRIVE+6 nach 37EE schreiben
47C3 77             LD      (HL),A          ;(siehe Kapitel 1.2.1: 37EE)
                    ----------------------
47C4 CD 5E 47       CALL    475E            ;FORCE-INTERRUPT-Kommando
47C7 CD 59 47       CALL    4759            ;Drive-Motor starten
47CA 06 80          LD      B,80            ;Verzögerung ca. 480 ms
47CC CB 7B          BIT     7,E             ;war der Drive-Motor schon an ?
47CE C4 ED 4C       CALL    NZ,4CED         ;wenn nein: 480 ms warten
                    ----------------------
47D1 F1             POP     AF              ;ist alte Drive# = neue Drive# ?
47D2 28 09          JR      Z,47DD          ;wenn ja
47D4 3A 0C 43       LD      A,(430C)        ;PDRIVE+2 holen
47D7 07             RLCA                    ;ist TI=H ?
47D8 06 0C          LD      B,0C            ;Verzögerung ca. 45 ms
47DA DC ED 4C       CALL    C,4CED          ;wenn ja: 45 ms warten
                    ----------------------
47DD AF             XOR     A               ;kein Fehler
47DE C1             POP     BC              ;BC zurück
47DF D1             POP     DE              ;DE zurück
47E0 E1             POP     HL              ;HL zurück
47E1 B7             OR      A               ;Z-Flag löschen, wenn A<>00
47E2 C9             RET
```

```
     ********************************************************
     * Name:       DELAY1                                   *
     * Funktion:   ca. 55 us warten (bei 1.774 MHz)         *
     *             und Status-Register des FDC lesen        *
     * Input:      --                                       *
     * Verändert:  F                                        *
     * Output:     A: Status des FDC                        *
     ********************************************************

     ********** FEHLER! Vor dem Laden des PDRIVE-Sectors   **********
     ********** in der Initialisierung von SYS0 steht hier **********
     ********** immer noch der Wert für SYSTEM BJ=1 !       **********
47E3 3E*06        LD      A,06          ;6 * SYSTEM BJ (CPU Speed)
47E5 3D           DEC     A             ;Zähler -1
47E6 20 FD        JR      NZ,47E5       ;wenn <> 0
47E8 3A EC 37     LD      A,(37EC)      ;Status-Register des FDC lesen
47EB C9           RET

     ********************************************************
     * Name:       TSTDSK (Fortsetzung von 445EH)           *
     * Funktion:   Drive (A) auswählen, Motor starten und   *
     *             testen, ob Diskette eingelegt            *
     * Input:      A: gewünschte Drive# (0-3)               *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

     Es wird DRVSEL (445BH) aufgerufen und (wenn dabei kein Fehler auftrat)
     anschließend geprüft, ob sich eine drehende Diskette im Laufwerk befindet.

47EC CD 76 47     CALL    4776          ;DRVSEL aufrufen
47EF C0           RET     NZ            ;wenn Error
47F0 E5           PUSH    HL            ;HL retten
47F1 D5           PUSH    DE            ;DE retten
47F2 C5           PUSH    BC            ;BC retten
47F3 11*00*00     LD      DE,0000       ;Zeit-Zähler: 2400H * SYSTEM BJ (Speed)
47F6 21 EC 37     LD      HL,37EC       ;Zeiger auf Status-Register des FDC
47F9 46           LD      B,(HL)        ;Status nach B
47FA 7E           LD      A,(HL)        ;Status nach A
47FB A8           XOR     B             ;beide vergleichen
47FC E6 02        AND     02            ;hat sich das Index-Bit verändert ?
47FE 20 DD        JR      NZ,47DD       ;wenn ja: ohne Fehler zurück
4800 1B           DEC     DE            ;ist die Zeit
4801 7A           LD      A,D           ;für 1.32 Umdrehungen
4802 B3           OR      E             ;abgelaufen ?
4803 20 F5        JR      NZ,47FA       ;wenn nein
4805 3E 08        LD      A,08          ;Fehlercode "DEVICE NOT AVAILABLE"
4807 18 D5        JR      47DE          ;mit Fehler zurück

     ********************************************************
     * Name:       EXPAND (Fortsetzung von 444BH)           *
     * Funktion:   Wenn die File kürzer ist, als das NEXT-  *
     *             Feld im FCB angibt, wird die File um     *
     *             entsprechend viele GRANS erweitert.      *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

```
4809 CD 80 49     CALL    4980        ;PUSH Register, IX=FCB, IY=4380, A=00
480C CD B7 49     CALL    49B7        ;Zugriffs-Level WRITE oder mehr ?
480F AF           XOR     A           ;Flag: File darf auch erweitert werden
```

```
*****************************************************
* Name:       FILPOS                                *
* Funktion:   Berechnet die physikalische Position des*
*             Sectors, auf den das NEXT-Feld zeigt  *
* Input:      IX: zeigt auf geöffneten FCB          *
*             IY: muß auf 4380H zeigen              *
*             A:  00: File darf erweitert werden    *
*                 B6: File darf nicht erweitert werden*
*             vorher: Call PUSHR (4980H)!           *
* Verändert:  BC                                    *
* Output:     AF: A=Fehlercode, wenn Z=0            *
*             DE: gesuchte Sector# (Disk-relativ)   *
*             HL: zeigt auf Buffer der File         *
*****************************************************
```

FILPOS berechnet, wo derjenige Sector einer File, auf den das NEXT-Feld zeigt, letztendlich auf Diskette steht. Falls die File kürzer ist, als das NEXT-Feld angibt, wird sie dabei - sofern das erlaubt ist - um entsprechend viele GRANS erweitert.
Vor dem Aufruf von FILPOS muß UP PUSHR (4980H) aufgerufen worden sein, damit im Falle eines Errors der Notaussprung über 49CDH funktioniert!

Arbeitsweise von FILPOS: Es werden zunächst die 4 Datenblocks überprüft, deren Angaben im FCB+0E bis FCB+15H stehen, ob sie den gesuchten Sector enthalten. Wenn nein, werden nun die 4 Erweiterungs-Datenblocks überprüft, deren Angaben zur Zeit im FCB+16H bis FCB+1F enthalten sind. Falls auch sie den gesuchten Sector nicht enthalten, werden aus dem Directory nacheinander alle FDE's dieser File gelesen und überprüft, bis die Informationen über den gesuchten Sector gefunden sind oder das Ende der File erreicht wurde. In letztem Fall wird - sofern die File erweitert werden darf - nun SYS2/SYS damit beauftragt, freie GRANS zu suchen und für diese File zu belegen.

```
4810 32 BB 48     LD      (48BB),A    ;Flag: darf File auch erweitert werden ?
4813 CD 6E 47     CALL    476E        ;Drive (FCB+6) auswählen + Motor starten
4816 20 5A        JR      NZ,4872     ;wenn Error
4818 CD 68 49     CALL    4968        ;HLC = NEXT-Wert
481B EB           EX      DE,HL       ;=> DE = NEXT-Wert (Bytes 1 und 2)
481C DD 7E 00     LD      A,(IX+00)   ;gehört der FCB nicht zu
481F 0F           RRCA                ;einer File, sondern zu
4820 0F           RRCA                ;einer ganzen Diskette ?
4821 38 30        JR      C,4853      ;wenn ja
4823 0F           RRCA                ;Bit 2,FCB+0 gesetzt ?
4824 3E 2D        LD      A,2D        ;Fehlercode "BAD FCB DATA"
4826 38 4A        JR      C,4872      ;wenn ja, Error
4828 EB           EX      DE,HL       ;=> HL = NEXT-Wert (Bytes 1 und 2)
4829 CD B2 4C     CALL    4CB2        ;DIV: HL=INT(HL/5), A=Rest
482C 0E 0E        LD      C,0E
482E EB           EX      DE,HL       ;=> DE = relative GRAN# innerhalb der
                                      ;File, die den gesuchten Sector enthält
482F DD E5        PUSH    IX          ;HL auf FCB+0 setzen
4831 E1           POP     HL
4832 09           ADD     HL,BC       ;HL auf FCB+0E setzen
4833 F5           PUSH    AF          ;A = relative Sector# innerhalb des
                                      ;GRANS, das den gesuchten Sector enthält
4834 E5           PUSH    HL          ;FCB+0E retten
```

```
4835 D5          PUSH    DE          ;gesuchte GRAN# retten
4836 3E 08       LD      A,08        ;Zähler: 8 Datenblocks pro FCB
4838 08          EX      AF,AF'
4839 7E          LD      A,(HL)      ;EOF erreicht ?
483A 3C          INC     A
483B 23          INC     HL
483C 28 54       JR      Z,4892      ;wenn ja
483E 7E          LD      A,(HL)      ;enthält der jetzt bearbeitete
483F CD E9 48    CALL    48E9        ;Datenblock den gesuchten Sector ?
4842 30 31       JR      NC,4875     ;wenn nein
                 ----------------------
                 Datenblock wurde gefunden, der den gesuchten Sector enthält
4844 09          ADD     HL,BC
4845 EB          EX      DE,HL       ;=> DE = gesuchte GRAN#
4846 AE          XOR     (HL)        ;Anzahl der GRANS berechnen, die
4847 07          RLCA                ;zwischen dem Beginn des LUMPS, auf
4848 07          RLCA                ;das HL zeigt, und dem Beginn des
4849 07          RLCA                ;Datenblocks liegen, in dem der
484A 83          ADD     E           ;gesuchte Sector steht; und zur
484B 5F          LD      E,A         ;gesuchten GRAN# hinzuaddieren
484C F1          POP     AF          ;./. (gesuchte GRAN#)
484D F1          POP     AF          ;./. (FCB+0E)
484E F1          POP     AF          ;A = relative Sector# im gesuchten GRAN
484F 2B          DEC     HL
4850 CD 7C 4C    CALL    4C7C        ;DE = gesuchte Sector# (Disk-relativ)
4853 DD 6E 03    LD      L,(IX+03)   ;Bufferadresse der File nach HL
4856 DD 66 04    LD      H,(IX+04)
4859 AF          XOR     A           ;kein Error
485A C9          RET                 ;ggf. nach 49A8: POP Register und RET
                 ----------------------
                 Nächsten FXDE der File aus dem Directory lesen und überprüfen
485B C1          POP     BC          ;./. (FDE+16H)
485C C1          POP     BC          ;BC = Anzahl GRANS, die vor dem Beginn
                                     ;der 4 Datenblocks im letzten FDE liegen
485D E1          POP     HL          ;HL = gesuchte GRAN#
485E ED 52       SBC     HL,DE       ;Anzahl der GRANS berechnen, die vor
4860 09          ADD     HL,BC       ;Beginn der 4 Datenblocks im nächsten
4861 44          LD      B,H         ;zu untersuchenden FDE liegen
4862 4D          LD      C,L         ;und in BC merken
4863 F5          PUSH    AF          ;DEC des nächsten FXDE's
4864 CD 2F 49    CALL    492F        ;DIR-Sector mit dem nächsten FXDE nach
                                     ;4200H laden und HL auf FXDE+0 setzen
4867 20 09       JR      NZ,4872     ;wenn Error
4869 3E*FF       LD      A,FF        ;DEC des zuvor untersuchten FDE's
486B 23          INC     HL          ;Überprüfung
486C BE          CP      (HL)        ;der Backward-Chain
486D 2B          DEC     HL          ;(Rück-Verkettung)
486E 28 2F       JR      Z,489F      ;wenn kein Error
4870 3E 2C       LD      A,2C        ;Fehlercode "BAD DIRECTORY DATA"
4872 C3 CD 49    JP      49CD        ;Error-Exit über Notaussprung
                 ----------------------
                 Nächsten (Erweiterungs-)Datenblock im FCB überprüfen
4875 EB          EX      DE,HL       ;HL auf das 1. Byte der Angaben
4876 23          INC     HL          ;über den nächsten Datenblock setzen
4877 DD CB 01 5E BIT     3,(IX+01)   ;Format des FCB: NEWDOS/80 Version 2 ?
487B 37          SCF
487C 28 A6       JR      Z,4824      ;wenn nein, Error
487E 08          EX      AF,AF'      ;Zähler: Anzahl Datenblocks pro FCB
487F FE 05       CP      05          ;kommen nun die Erweiterungs-Datenbl. ?
4881 20 0C       JR      NZ,488F     ;wenn nein
```

```
4883 4E          LD      C,(HL)        ;BC = Anzahl der GRANS, die vor
4884 23          INC     HL            ;dem Beginn des 1. Erweiterungs-
4885 46          LD      B,(HL)        ;Datenblocks liegen
4886 23          INC     HL
4887 D1          POP     DE            ;gesuchte GRAN#
4888 D5          PUSH    DE
4889 EB          EX      DE,HL         ;liegt die gesuchte GRAN#
488A ED 42       SBC     HL,BC         ;vor dem Beginn des
488C EB          EX      DE,HL         ;1. Erweiterungs-Datenblocks ?
488D 38 03       JR      C,4892        ;wenn ja
488F 3D          DEC     A             ;nächsten (Erweit.-)Datenblock prüfen
4890 20 A6       JR      NZ,4838       ;wenn noch nicht alle 8 Blocks überprüft
                 ----------------------
                 EOF oder Ende der Erweiterungs-Datenblocks erreicht
4892 01 00 00    LD      BC,0000       ;Anzahl GRANS, die vor dem Beginn der
                                       ;nächsten 4 Datenblocks liegen
4895 D1          POP     DE            ;DE = gesuchte GRAN#
4896 CD 4B 49    CALL    494B          ;DIR-Sector mit dem FPDE (FCB+7) nach
                                       ;4200H laden und HL auf FPDE+0 setzen
4899 20 CC       JR      NZ,4867       ;wenn Error
489B DD 7E 07    LD      A,(IX+07)     ;DEC des FPDE holen
489E F5          PUSH    AF            ;A retten (DEC des FPDE's)
489F F1          POP     AF            ;A = DEC des gerade geladenen FDE's
48A0 32 6A 48    LD      (486A),A      ;für Prüfung der Backward-Chain merken
48A3 CB 66       BIT     4,(HL)        ;ist der gerade geladene FDE aktiv ?
48A5 28 C9       JR      Z,4870        ;wenn nein, Error
48A7 D5          PUSH    DE            ;gesuchte GRAN#
48A8 C5          PUSH    BC            ;Anzahl GRANS, die vor Beginn der
                                       ;nächsten 4 Datenblocks liegen
48A9 7D          LD      A,L           ;HL auf FDE+16H setzen
48AA C6 16       ADD     16
48AC 6F          LD      L,A
48AD E5          PUSH    HL            ;FDE+16H
48AE 7E          LD      A,(HL)        ;nächsten Datenblock im FDE prüfen
48AF FE FE       CP      FE            ;Zeiger auf nächsten FXDE ?
48B1 23          INC     HL
48B2 7E          LD      A,(HL)
48B3 23          INC     HL
48B4 38 10       JR      C,48C6        ;wenn normaler Datenblock
48B6 28 A3       JR      Z,485B        ;wenn nächster FXDE gelesen werden muß
                 ----------------------
                 EOF erreicht. File erweitern ?
48B8 3E 3E       LD      A,3E          ;Fehlercode "CAN'T EXTEND FILE VIA READ"
    a) wenn File erweitert werden darf:
48BA 18*00       JR      48BC          ;Error ignorieren
    b) wenn File nicht erweitert werden darf:
48BA 18*B6       JR      4872          ;Error weiterleiten
48BC F1          POP     AF            ;./. (FDE+16H)
48BD 3E 64       LD      A,64          ;Code für SYS2/SYS
48BF CD DD 49    CALL    49DD          ;SYS-File laden und starten
48C2 C1          POP     BC            ;Anzahl GRANS, die vor dem Beginn der
                                       ;nächsten 4 Datenblocks liegen
48C3 D1          POP     DE            ;gesuchte GRAN#
48C4 18 DD       JR      48A3          ;neues Spiel, neues Glück
                 ----------------------
                 Nächsten Datenblock im FDE überprüfen
48C6 CB 65       BIT     4,L           ;HL zeigt auf nächsten Datenblock im FDE
48C8 28 A6       JR      Z,4870        ;"BAD DIRECTORY DATA" ?
48CA CD E9 48    CALL    48E9          ;enthält der jetzt überprüfte
48CD EB          EX      DE,HL         ;Datenblock den gesuchten Sector ?
```

```
48CE 30 DE        JR      NC,48AE       ;wenn nein: nächsten Datenblock prüfen
                  ----------------------
                  Datenblock wurde gefunden, der den gesuchten Sector enthält
48D0 C1           POP     BC            ;BC = FDE+16H
48D1 D1           POP     DE            ;DE = Anzahl GRANS, die vor dem Beginn
                                        ;der gefundenen 4 Datenblocks liegen
48D2 F1           POP     AF            ;./. gesuchte GRAN#
48D3 E1           POP     HL            ;HL = FCB+0E
48D4 F1           POP     AF            ;A = relative Sector# im gesuchten GRAN
48D5 C5           PUSH    BC            ;BC = FDE+16H
48D6 01 08 00     LD      BC,0008       ;8 Bytes für 4 Datenblocks
48D9 7A           LD      A,D           ;zeigte BC auf einen FPDE oder FXDE ?
48DA B3           OR      E
48DB 28 05        JR      Z,48E2        ;wenn FPDE
48DD 09           ADD     HL,BC         ;HL auf FCB+16H setzen
48DE 73           LD      (HL),E        ;dort die Anzahl der GRANS
48DF 23           INC     HL            ;eintragen, die vor dem
48E0 72           LD      (HL),D        ;Beginn der gefundenen
48E1 23           INC     HL            ;4 Datenblocks liegen
48E2 EB           EX      DE,HL         ;=> DE = FCB+0E bzw. FCB+18H
48E3 E1           POP     HL            ;HL = FDE + 16H
48E4 ED B0        LDIR                  ;Angaben über die gefundenen 4 Daten-
                                        ;blocks vom FDE in den FCB übertragen
48E6 C3 13 48     JP      4813          ;nun müsste es aber klappen
                  ----------------------
                  Enthält der zu prüfende Datenblock den gesuchten Sector ?
48E9 E6 1F        AND     1F            ;BC = Anzahl der
48EB 4F           LD      C,A           ;GRANS, aus denen
48EC 06 00        LD      B,00          ;der zu prüfende
48EE 03           INC     BC            ;Datenblock besteht
48EF EB           EX      DE,HL         ;=> HL = gesuchte GRAN#
48F0 ED 42        SBC     HL,BC         ;Länge des Datenblocks abziehen
48F2 C9           RET                   ;C=1 (Carry): der gesuchte Sector liegt
                                        ;     in dem geprüften Datenblock
                                        ;C=0 (Carry): der gesuchte Sector liegt
                                        ;     hinter dem geprüften Datenblock

                  ----------------------
                  Position des Directory aus dem BOOT-Sector entnehmen
48F3 3A 02 42     LD      A,(4202)      ;DSL (Directory Starting Lump) holen
48F6 2A 99 43     LD      HL,(4399)     ;Zeiger auf aktuellen PDRIVE-Block
48F9 77           LD      (HL),A        ;wirklichen DSL bei PDRIVE+0 eintragen
48FA 3A 30 49     LD      A,(4930)      ;gewünschte DIR-Sector#
                  ----------------------
                  DIR-Sector von Diskette lesen
48FD CD 74 4C     CALL    4C74          ;Position des DIR-Sectors berechnen:
                                        ;DE = gesuchte Sector# (Disk-relativ)
                                        ;HL = 4200H (Buffer für DIR-Sectoren)
4900 CD 30 46     CALL    4630          ;Sector von Diskette lesen
4903 20 02        JR      NZ,4907       ;wenn (Data Adress Mark ?) Error
4905 F6 31        OR      31            ;Fehlercode "WRONG DISKETTE RECORD TYPE"
4907 FE 06        CP      06            ;Z=1, wenn Data Adress Mark des DIR
4909 C9           RET
```

```
*********************************************************
* Name:       DIRR                                      *
* Funktion:   liest einen Sector des Directory          *
* Input:      A: gewünschte Sector# (DIR-relativ)       *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*             HL: 4200H (Buffer für DIR-Sectoren)       *
*             (4930H): gelesene Sector# (DIR-relativ)   *
*********************************************************
```

DIRR berechnet, wo ein gewünschter Sector des Directory auf Diskette steht und liest ihn nach 4200H. Falls der gelesene Sector nicht mit dem Data Adress Mark für DIR-Sectoren versehen ist, wird die wirkliche Position des Directory aus dem BOOT-Sector der betreffenden Diskette gelesen, bei PDRIVE+0 dieses Laufwerks eingetragen und anschließend damit ein neuer Versuch gestartet.
Register A gibt an, um den wievielten Sector innerhalb des Directory (beginnend ab 0) es sich dabei handelt.
Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
490A D5          PUSH    DE          ;DE retten
490B C5          PUSH    BC          ;BC retten
490C CD FD 48    CALL    48FD        ;Position des DIR-Sectors berechnen und
                                     ;DIR-Sector nach 4200H lesen
490F 28 0B       JR      Z,491C      ;wenn kein Error
4911 11 00 00    LD      DE,0000     ;BOOT-Sector der gleichen
4914 CD 30 46    CALL    4630        ;Diskette nach 4200H lesen
4917 CC F3 48    CALL    Z,48F3      ;wenn kein Error: wirkliche Position des
                                     ;DIR entnehmen und neuen Versuch machen
491A 3E 11       LD      A,11        ;ggf. Fehlercode "DIRECTORY READ ERROR"
491C C1          POP     BC          ;BC zurück
491D D1          POP     DE          ;DE zurück
491E C9          RET
```

```
*********************************************************
* Name:       DIRW                                      *
* Funktion:   schreibt einen Sector des Directory       *
* Input:      (4930H): gewünschte Sector# (DIR-relativ)
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*             HL: 4200H (Buffer für DIR-Sectoren)       *
*********************************************************
```

DIRW berechnet, wo ein gewünschter Sector des Directory auf Diskette steht und schreibt den Inhalt des DIR-Buffers (4200H) mit dem Data Adress Mark für DIR-Sectoren dorthin.
(4930H) gibt an, um den wievielten Sector innerhalb des Directory (beginnend ab 0) es sich dabei handelt.
Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
491F 3A 30 49    LD      A,(4930)    ;gewünschte Sector# (DIR-relativ)
4922 D5          PUSH    DE          ;DE retten
4923 C5          PUSH    BC          ;BC retten
4924 CD 74 4C    CALL    4C74        ;Position des DIR-Sectors berechnen:
                                     ;DE = gesuchte Sector# (Disk-relativ)
                                     ;HL = 4200H (Buffer für DIR-Sectoren)
```

```
4927 B4          OR      H           ;DIR-Sector mit Verify
4928 CD B8 4A    CALL    4AB8        ;auf Diskette schreiben
492B 3E 12       LD      A,12        ;ggf. Fehlercode "DIRECTORY WRITE ERROR"
492D 18 ED       JR      491C        ;POP BC, POP DE und RET
                 ----------------------
                 FDE einer File aus dem Directory holen, wenn der benötigte
                 DIR-Sector möglicherweise schon/noch im Buffer steht
492F 2E*FF       LD      L,FF        ;Sector# des DIR-Sectors im Buffer
4931 18 05       JR      4938        ;weiter wie GETFDE
                 ----------------------
                 FPDE einer File aus dem Directory holen
4933 DD 7E 07    LD      A,(IX+07)   ;DEC des FPDE bei FCB+7 lesen
```

```
*********************************************************
* Name:       GETFDE                                    *
* Funktion:   holt einen FDE aus dem Directory          *
* Input:      A: DEC des gewünschten FDE's              *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*             HL: zeigt im Buffer (42XXH) auf FDE+0     *
*             (4930H): gelesene DIR-Sector#             *
*********************************************************
```

GETFDE berechnet, in welchem Sector des Directory ein gewünschter FDE (File Directory Entry) auf Diskette steht und liest diesen Sector nach 4200H. Falls der gelesene Sector nicht mit dem Data Adress Mark für DIR-Sectoren versehen ist, wird die wirkliche Position des Directory aus dem BOOT-Sector der betreffenden Diskette gelesen, bei PDRIVE+0 dieses Laufwerks eingetragen und anschließend damit ein neuer Versuch gestartet. Register A gibt den DEC (Directory Entry Code) des gewünschten FDE's an. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
4936 2E FF       LD      L,FF        ;Flag: der benötigte DIR-Sector
                                     ;steht noch nicht im Buffer
4938 F5          PUSH    AF          ;A retten (DEC des gesuchten FDE's)
4939 E6 1F       AND     1F          ;Sector# des benötigten
493B 3C          INC     A           ;DIR-Sectors berechnen
493C 3C          INC     A           ;(DIR-relativ)
493D BD          CP      L           ;steht der benötigte Sector
                                     ;bereits im Buffer ?
493E C4 0A 49    CALL    NZ,490A     ;wenn nein: Position dieses Sectors
                                     ;berechnen und Sector nach 4200H lesen
4941 E1          POP     HL          ;H = DEC des gesuchten FDE's
4942 C0          RET     NZ          ;wenn Error
4943 3E E0       LD      A,E0        ;Adresse
4945 A4          AND     H           ;vom FDE+0
4946 6F          LD      L,A         ;im Buffer
4947 26 42       LD      H,42        ;berechnen
4949 BF          CP      A           ;Z=1 (kein Error)
494A C9          RET
                 ----------------------
                 FPDE einer File aus dem DIR holen und FCB auf Fehler prüfen
494B CD 6E 47    CALL    476E        ;Drive (FCB+6) auswählen + Motor starten
494E CC 33 49    CALL    Z,4933      ;wenn kein Error: FPDE der File nach
                                     ;4200H lesen und HL auf FPDE+0 setzen
4951 C0          RET     NZ          ;wenn Error
4952 E5          PUSH    HL          ;HL retten (FPDE+0)
4953 C6 16       ADD     16          ;HL auf FPDE+16H
```

```
4955 6F          LD      L,A
4956 DD 7E 0E    LD      A,(IX+0E)     ;stimmen die Angaben über den 1. Daten-
4959 BE          CP      (HL)          ;block in FCB+0E und FPDE+16H überein ?
495A E1          POP     HL            ;HL zurück (FPDE+0)
495B 28 03       JR      Z,4960        ;wenn ja
495D 3C          INC     A             ;EOF ?
495E 20 05       JR      NZ,4965       ;wenn nein: Error
4960 7E          LD      A,(HL)        ;(FPDE+0) testen
4961 E6 90       AND     90            ;handelt es sich um
4963 FE 10       CP      10            ;einen aktiven FPDE ?
4965 3E 2D       LD      A,2D          ;ggf. Fehlercode "BAD FCB DATA"
4967 C9          RET
```

```
*****************************************************
* Name:       NXTEOF                                *
* Funktion:   NEXT-Wert aus dem FCB nach HLC laden  *
*             und NEXT-Wert mit EOF-Wert vergleichen*
* Input:      IX: zeigt auf einen geöffneten FCB    *
* Verändert:  A                                     *
* Output:     HL: 1. und 2. Byte des NEXT-Wertes    *
*             C:  3. Byte des NEXT-Wertes           *
*             F:  C=1 und Z=0, wenn NEXT < EOF      *
*                 C=0 und Z=1, wenn NEXT = EOF      *
*                 C=0 und Z=0, wenn NEXT > EOF      *
*****************************************************
```

```
4968 DD 4E 05    LD      C,(IX+05)     ;C = 3. Byte des NEXT-Wertes
496B DD 6E 0A    LD      L,(IX+0A)     ;L = 2. Byte des NEXT-Wertes
496E DD 66 0B    LD      H,(IX+0B)     ;H = 1. Byte des NEXT-Wertes
4971 7C          LD      A,H           ;1. Byte des NEXT-Wertes mit
4972 DD BE 0D    CP      (IX+0D)       ;1. Byte des EOF-Wertes vergleichen
4975 C0          RET     NZ            ;wenn <>
4976 7D          LD      A,L           ;2. Byte des NEXT-Wertes mit
4977 DD BE 0C    CP      (IX+0C)       ;2. Byte des EOF-Wertes vergleichen
497A C0          RET     NZ            ;wenn <>
497B 79          LD      A,C           ;3. Byte des NEXT-Wertes mit
497C DD BE 08    CP      (IX+08)       ;3. Byte des EOF-Wertes vergleichen
497F C9          RET
```

```
*****************************************************
* Name:       PUSHR                                 *
* Funktion:   wenn FCB geöffnet ist: PUSH Register, *
*             IX=FCB+0, IY=4380H, A=00, SP=SP-16D   *
* Input:      DE: zeigt auf einen geöffneten FCB    *
* Verändert:  SP: wird um 8 Ebenen erniedrigt       *
* Output:     IX: zeigt auf FCB+0                   *
*             IY: zeigt auf 4380H                   *
*             AF: 0044H (wenn kein Error)           *
*             (49CEH): geretteter Stackpointer      *
*****************************************************
```

PUSHR ist kein UP im gewöhnlichen Sinne, da es erhebliche Manipulationen am SP (Stackpointer) vornimmt und auch nicht immer zu seinem Aufrufer zurückkehrt.

Wenn Register DE beim Aufruf von PUSHR nicht auf einen geöffneten FCB zeigt, wird der Fehlercode "FILE NOT OPEN" nach AF geladen, die Rücksprung-Adresse zum Aufrufer von PUSHR aus dem Stack entfernt und in die nächsthöhere Ebene zurückgesprungen, ohne ein weiteres Register zu verändern.

Wenn Register DE beim Aufruf von PUSHR jedoch auf einen geöffneten FCB zeigt, werden die Register IY, HL, DE, BC, AF', IX, der Zwischenspeicher für den SP (49CEH) sowie 49A8H als nächste Rücksprung-Adresse in den Stack geschrieben, Register IX wird auf FCB+0, IY auf 4380H (als Zeiger auf den RAM-Bereich 4300H bis 43FFH) und AF auf 0044H gesetzt, der jetzige Wert des SP wird bei 49CEH vermerkt und nun wird zum Aufrufer von PUSHR zurückgekehrt. Das nächste RET, daß nun von dort ausgeführt wird, verzweigt aufgrund der manipulierten Rücksprung-Adresse nach 49A8H, wo alle Register (mit Ausnahme von AF) aus dem Stack geholt werden und so wieder ihren ursprünglichen Wert erhalten, den sie beim Aufruf von PUSHR hatten.

```
4980 1A           LD      A,(DE)        ;Zeiger auf FCB+0
4981 07           RLCA                  ;FCB geöffnet ?
4982 3E 26        LD      A,26          ;ggf. Fehlercode "FILE NOT OPEN"
4984 30 01        JR      NC,4987       ;wenn nein
4986 AF           XOR     A             ;wenn ja: kein Error
4987 08           EX      AF,AF'        ;Fehlerstatus nach AF'
4988 E3           EX      (SP),HL       ;HL = Rücksprung-Adresse, (SP) = HL
4989 22 A6 49     LD      (49A6),HL     ;Rücksprung-Adresse retten
498C E1           POP     HL            ;HL zurück
498D FD E5        PUSH    IY            ;IY retten
498F E5           PUSH    HL            ;HL retten
4990 D5           PUSH    DE            ;DE retten
4991 C5           PUSH    BC            ;BC retten
4992 F5           PUSH    AF            ;AF' retten
4993 D5           PUSH    DE            ;IX retten und
4994 DD E3        EX      (SP),IX       ;IX auf FCB+0 setzen
4996 E5           PUSH    HL            ;Zwischenspeicher
4997 2A CE 49     LD      HL,(49CE)     ;des SP in den
499A E3           EX      (SP),HL       ;Stack schreiben
499B ED 73 CE 49  LD      (49CE),SP     ;jetzigen SP merken
499F FD 21 80 43  LD      IY,4380       ;IY auf 4380H setzen
49A3 08           EX      AF,AF'        ;Fehlerstatus zurück nach AF
49A4 B7           OR      A             ;wenn kein Error:
49A5 CC*00*00     CALL    Z,0000        ;Rücksprung-Adresse 49A8H in den Stack
                                        ;und zurück zum Aufrufer von PUSHR
                  ----------------------
                  POP Register
49A8 E1           POP     HL            ;Zwischenspeicher des
49A9 22 CE 49     LD      (49CE),HL     ;SP zurück nach 49CEH
49AC 08           EX      AF,AF'        ;neuen Fehlerstatus in AF' retten
49AD DD E1        POP     IX            ;IX zurück
49AF F1           POP     AF            ;AF' zurück
49B0 C1           POP     BC            ;BC zurück
49B1 D1           POP     DE            ;DE zurück
49B2 E1           POP     HL            ;HL zurück
49B3 FD E1        POP     IY            ;IY zurück
49B5 08           EX      AF,AF'        ;Fehlerstatus zurück nach AF
49B6 C9           RET                   ;nicht zum Aufrufer von PUSHR, sondern
                                        ;in die nächsthöhere Ebene
                  ----------------------
                  Zugriffs-Level aufgrund eines Passwortes prüfen
49B7 06 05        LD      B,05          ;Zugriffs-Level WRITE oder mehr ?
49B9 DD 7E 01     LD      A,(IX+01)     ;erlaubten Zugriffs-Level
49BC E6 07        AND     07            ;aus dem FCB holen und mit dem
49BE B8           CP      B             ;erwünschten Zugriffs-Level vergleichen
49BF D8           RET     C             ;wenn erlaubt
49C0 3E 25        LD      A,25          ;"ILLEGAL ACCESS TO PROTECTED FILE"
```

```
49C2 18 09        JR      49CD          ;Error-Exit über Notaussprung
                  ----------------------
                  Wurde EOF erreicht oder überschritten ?
49C4 CD 68 49     CALL    4968          ;NEXT-Feld < EOF-Feld ?
49C7 D8           RET     C             ;wenn ja
49C8 3E 1C        LD      A,1C          ;Fehlercode "END OF FILE ENCOUNTERED"
49CA 28 01        JR      Z,49CD        ;wenn NEXT-Feld = EOF-Feld
49CC 3C           INC     A             ;Fehlercode "PAST END OF FILE"
                  ----------------------
                  Error-Exit über Notaussprung: zurück mit Fehlerstatus in AF
                  zur nächsthöheren Ebene als zum Aufrufer von PUSHR (4980H).
                  (Nur zulässig, wenn vorher PUSHR aufgerufen wurde)
49CD 31*00*00     LD      SP,0000       ;zuletzt gemerkten SP aus
                                        ;dem Zwischenspeicher holen
49D0 B7           OR      A             ;Z=0, wenn A=Fehlercode
49D1 18 D5        JR      49A8          ;POP Register und RET
                  ----------------------
                  Routine zum Aufruf von SYS-Files (wird z.B. von SYS1 benutzt)
49D3 C2 09 44     JP      NZ,4409       ;wenn Error
49D6 AF           XOR     A             ;kein Error
49D7 20 F4        JR      NZ,49CD       ;Error-Exit über Notaussprung
49D9 E3           EX      (SP),HL       ;Codes für SYS-File aus dem Stack holen
49DA 7C           LD      A,H           ;A = Code für gewünschte SYS-File
49DB 4D           LD      C,L           ;C = Hilfscode für SYS-File
49DC E1           POP     HL            ;HL zurück
49DD EF           RST     28            ;SYS-File laden und starten
                  ----------------------
                  Hilfsroutine für UP's READ (4436H) und WRITE (4439H)
49DE DD CB 01 7E  BIT     7,(IX+01)     ;LRL (Logische Recordlänge) = 256D ?
49E2 C8           RET     Z             ;wenn ja
49E3 D1           POP     DE            ;Rücksprung-Adresse löschen
49E4 DD 46 09     LD      B,(IX+09)     ;B = Logische Recordlänge
49E7 F5           PUSH    AF            ;F retten (C=1: READ / C=0: WRITE)
49E8 E5           PUSH    HL            ;HL retten (Zeiger auf Record-Buffer)
49E9 C5           PUSH    BC            ;B retten (Zähler für LRL)
49EA 4E           LD      C,(HL)        ;falls WRITE: nächstes Zeichen aus
                                        ;dem Record-Buffer holen
49EB CD ED 4A     CALL    4AED          ;nächsten Zeichen lesen oder schreiben
49EE 20 DD        JR      NZ,49CD       ;wenn Error: Exit über Notaussprung
49F0 5F           LD      E,A           ;falls READ: gelesenes Zeichen retten
49F1 C1           POP     BC            ;B zurück (Zähler für LRL)
49F2 E1           POP     HL            ;HL zurück (Zeiger auf Record-Buffer)
49F3 F1           POP     AF            ;F zurück (C=1: READ / C=0: WRITE)
49F4 30 01        JR      NC,49F7       ;wenn WRITE
49F6 73           LD      (HL),E        ;wenn READ: gelesenes Zeichen in
                                        ;den Record-Buffer übertragen
49F7 23           INC     HL            ;Zeiger auf Record-Buffer +1
49F8 10 ED        DJNZ    49E7          ;Zähler für LRL -1, nächstes Byte ?
49FA AF           XOR     A             ;kein Error
49FB C9           RET                   ;nach 49A8H: POP Register und RET
```

```
     ********************************************************
     * Name:       READ (Fortsetzung von 4436H)             *
     * Funktion:   nächsten Sector/Record einer File lesen  *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: zeigt auf Record-Buffer (nur wenn    *
     *                 Logische Recordlänge <> 256D ist)    *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

49FC CD 80 49     CALL    4980        ;PUSH Register, IX=FCB, IY=4380, A=00
49FF 37           SCF                 ;Flag für READ statt WRITE
4A00 CD DE 49     CALL    49DE        ;wenn LRL < 256D (Logische Recordlänge):
                                      ;dort abarbeiten, kein RET hierher
4A03 CD 19 4A     CALL    4A19        ;nächsten Sector von Diskette lesen
4A06 28 03        JR      Z,4A0B      ;wenn kein Error
4A08 FE 06        CP      06          ;Directory-Sector ?
4A0A C0           RET     NZ          ;wenn nein
4A0B DD 34 0A     INC     (IX+0A)     ;1. und 2. Byte
4A0E 20 03        JR      NZ,4A13     ;des NEXT-Wertes
4A10 DD 34 0B     INC     (IX+0B)     ;um 1 erhöhen
4A13 DD CB 01 EE  SET     5,(IX+01)   ;vermerken, daß der aktuelle Sector
                                      ;noch nicht im Buffer steht
4A17 B7           OR      A           ;Z=0, wenn A=Fehlercode
4A18 C9           RET                 ;ggf. nach 49A8: POP Register und RET
                  ----------------------
                  Nächsten Sector einer File von Diskette lesen
4A19 06 06        LD      B,06        ;Zugriffs-Level READ:
4A1B CD B9 49     CALL    49B9        ;prüfen, ob Zugriffs-Level READ erlaubt
4A1E CD C4 49     CALL    49C4        ;wenn NEXT = EOF: "END OF FILE ENCOUNT."
                                      ;wenn NEXT > EOF: "PAST END OF FILE"
4A21 3E B6        LD      A,B6        ;Flag: File beim Lesen nicht erweitern
4A23 CD 10 48     CALL    4810        ;DE = gewünschte Sector# (Disk-relativ),
                                      ;HL = Adresse des Buffers
4A26 CD 30 46     CALL    4630        ;Sector von Diskette lesen
     ********** FEHLER! Auch wenn beim Lesen ein Fehler auftrat wird **********
     ********** vermerkt, daß der aktuelle Sector im Buffer steht!  **********
4A29 DD CB 01 AE  RES     5,(IX+01)   ;vermerken, daß der aktuelle Sector im
4A2D DD CB 01 A6  RES     4,(IX+01)   ;Buffer steht und keine Daten enthält,
4A31 C9           RET                 ;die noch auf Diskette zu schreiben sind

     ********************************************************
     * Name:       VERIFY (Fortsetzung von 443CH)           *
     * Funktion:   nächsten Sector/Record einer File auf    *
     *             Diskette schreiben, anschließend Verify  *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: zeigt auf Record-Buffer (nur wenn    *
     *                 Logische Recordlänge <> 256D ist)    *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4A32 3E F6        LD      A,F6        ;"OR  nn": Kennung für UP VERIFY
4A34 18 02        JR      4A38        ;weiter wie UP WRITE
```

```
     ********************************************************
     * Name:       WRITE (Fortsetzung von 4439H)            *
     * Funktion:   nächsten Sector/Record einer File auf    *
     *             Diskette schreiben (ohne Verify)         *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: zeigt auf Record-Buffer (nur wenn    *
     *                 Logische Recordlänge <> 256D ist)    *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4A36 3E E6        LD      A,E6          ;"AND  nn": Kennung für UP WRITE
4A38 32 8B 4A     LD      (4A8B),A      ;Kennung für VERIFY oder WRITE
4A3B CD 80 49     CALL    4980          ;PUSH Register, IX=FCB, IY=4380, A=00
4A3E CD DE 49     CALL    49DE          ;wenn LRL < 256D (Logische Recordlänge):
                                        ;dort abarbeiten, kein RET hierher
4A41 CD 67 4A     CALL    4A67          ;Buffer-Inhalt auf Diskette schreiben
4A44 C0           RET     NZ            ;wenn Error
4A45 DD 7E 05     LD      A,(IX+05)     ;3. Byte des NEXT-Wertes testen
4A48 B7           OR      A             ;wenn 00:
4A49 CC 0B 4A     CALL    Z,4A0B        ;1. und 2. Byte des NEXT-Wertes +1,
                                        ;und vermerken, daß der aktuelle Sector
                                        ;noch nicht im Buffer steht
4A4C CD 68 49     CALL    4968          ;HLC = NEXT-Wert
4A4F 30 06        JR      NC,4A57       ;wenn NEXT-Wert >= EOF-Wert
4A51 DD CB 01 76  BIT     6,(IX+01)     ;soll der EOF-Wert nach jedem Schreiben
                                        ;auf den NEXT-Wert gesetzt werden ?
4A55 20 09        JR      NZ,4A60       ;wenn nein
4A57 DD 71 08     LD      (IX+08),C     ;EOF-Wert := NEXT-Wert
4A5A DD 75 0C     LD      (IX+0C),L
4A5D DD 74 0D     LD      (IX+0D),H
4A60 AF           XOR     A             ;kein Error
4A61 C9           RET                   ;ggf. nach 49A8: POP Register und RET
                  ----------------------
                  Nächsten Sector einer File auf Diskette schreiben
4A62 DD CB 01 66  BIT     4,(IX+01)     ;muß der Inhalt des Buffers noch auf
                                        ;Diskette geschrieben werden ?
4A66 C8           RET     Z             ;wenn nein
4A67 CD 0C 48     CALL    480C          ;wenn die File kürzer ist, als das
                                        ;NEXT-Feld angibt, wird die File um
                                        ;entsprechend viele GRANS erweitert,
                                        ;DE = gewünschte Sector# (Disk-relativ),
                                        ;HL = Adresse des File-Buffers
4A6A DD CB 02 6E  BIT     5,(IX+02)     ;ist das Update-Flag schon gesetzt ?
4A6E 20 18        JR      NZ,4A88       ;wenn ja
4A70 DD CB 00 4E  BIT     1,(IX+00)     ;gehört der FCB nicht zu einer File,
                                        ;sondern zu einer ganzen Diskette ?
4A74 20 12        JR      NZ,4A88       ;wenn ja
                  ----------------------
                  Update-Flag im Directory setzen
4A76 E5           PUSH    HL            ;HL retten (Zeiger auf Buffer der File)
4A77 CD 4B 49     CALL    494B          ;DIR-Sector mit FPDE lesen, HL = FPDE+0
4A7A 20 03        JR      NZ,4A7F       ;wenn Error
4A7C 23           INC     HL            ;HL auf FPDE+1
4A7D CB EE        SET     5,(HL)        ;Update-Flag im FPDE setzen
4A7F CC 1F 49     CALL    Z,491F        ;wenn kein Error: DIR-Sector schreiben
4A82 E1           POP     HL            ;HL zurück (Zeiger auf Buffer der File)
4A83 C0           RET     NZ            ;wenn Error
4A84 DD CB 02 EE  SET     5,(IX+02)     ;Update-Flag im FCB setzen
                  ----------------------
```

```
                Verify gewünscht ? - Normaler Sector oder DIR-Sector ?
    a) wenn DOS-Befehl VERIFY,N:
4A88*DD*7E*01     LD      A,(IX+01)     ;wird Verify gewünscht ?
    b) wenn DOS-Befehl VERIFY,Y:
4A88*00*3E*FF     LD      A,FF          ;Verify wird gewünscht
    a) wenn Aufruf über UP WRITE (4439H):
4A8B*E6 90        AND     90            ;wird Verify gewünscht ?
    b) wenn Aufruf über UP VERIFY (443CH):
4A8B*F6 90        OR      90            ;Verify wird gewünscht
4A8D DD CB 00 46  BIT     0,(IX+00)     ;normaler Sector oder DIR-Sector ?
4A91 CD B8 4A     CALL    4AB8          ;Sector auf Diskette schreiben
4A94 C0           RET     NZ            ;wenn Error
4A95 AF           XOR     A             ;kein Error
4A96 18 95        JR      4A2D          ;vermerken, daß der Buffer keine Daten
                                        ;mehr enthält, die noch auf Diskette zu
                                        ;schreiben sind und RET

    ********************************************************
    * Name:       WRITEB (Fortsetzung von 001BH)           *
    * Funktion:   nächstes Byte in eine File schreiben     *
    * Input:      DE: zeigt auf geöffneten FCB             *
    *             A:  zu schreibendes Byte                 *
    * Verändert:  --                                       *
    * Output:     AF: A=Fehlercode, wenn Z=0               *
    ********************************************************

    Obige Angaben gelten nur für einen Aufruf über 001BH

4A98 28 12        JR      Z,4AAC        ;wenn aktueller Sector im Buffer steht
                  ----------------------
                  Aktueller Sector steht noch nicht im Buffer
4A9A C5           PUSH    BC            ;C retten (zu schreibendes Zeichen)
4A9B CD 0C 48     CALL    480C          ;wenn die File kürzer ist, als das
                                        ;NEXT-Feld angibt, wird die File um
                                        ;entsprechend viele GRANS erweitert
4A9E CD 68 49     CALL    4968          ;NEXT-Wert < EOF-Wert ?
4AA1 20 01        JR      NZ,4AA4       ;wenn ja
4AA3 B7           OR      A             ;3. Byte des NEXT-Wertes > 0 ?
4AA4 C4 21 4A     CALL    NZ,4A21       ;wenn ja: aktuellen Sector von
                                        ;Diskette in den Buffer lesen
4AA7 C1           POP     BC            ;C zurück (zu schreibendes Zeichen)
4AA8 C0           RET     NZ            ;wenn Error
4AA9 CD 29 4A     CALL    4A29          ;vermerken, daß der aktuelle Sector im
                                        ;Buffer steht und keine Daten enthält,
                                        ;die noch auf Diskette zu schreiben sind
                  ----------------------
4AAC CD 0B 4B     CALL    4B0B          ;HL zeigt auf nächstes Zeichen im Buffer
4AAF 71           LD      (HL),C        ;zu schreibendes Zeichen in den Buffer
4AB0 DD CB 01 E6  SET     4,(IX+01)     ;vermerken, daß der Buffer Daten enthält
                                        ;die noch auf Diskette zu schreiben sind
4AB4 28 8B        JR      Z,4A41        ;wenn Buffer jetzt voll ist
4AB6 18 94        JR      4A4C          ;wenn Buffer noch nicht voll ist
```

```
*********************************************************
* Name:       WRITXV                                    *
* Funktion:   schreibt einen normalen Sector oder       *
*             einen Sector des Directory auf Diskette,  *
*             anschließend Verify (optional)            *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             HL: Zeiger auf zu benutzenden Buffer      *
*             A:  Verify nur durchführen, wenn A <> 00  *
*             F:  normaler (Z=1) oder DIR-Sector (Z=0)  *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  BC                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

WRITXV schreibt einen physikalischen Sector mit dem Data Adress Mark für normale Sectoren oder Directory-Sectoren auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).
Wenn Register A <> 00 ist, wird der Sector nach dem Schreiben getestet, ob er auch ohne Fehler lesbar ist. Wenn nein, wird das Schreiben und anschl. Testen so oft wiederholt, wie SYSTEM AW angibt.

```
4AB8 4F          LD      C,A          ;Flag für Verify (00 = nein)
4AB9 06*03       LD      B,03         ;SYSTEM AW (Anzahl Versuche für Verify)
4ABB 20 0D       JR      NZ,4ACA      ;wenn DIR-Sector
```

```
*********************************************************
* Name:       WRITEV                                    *
* Funktion:   schreibt einen normalen Sector auf        *
*             Diskette, anschließend Verify (optional)  *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             HL: Zeiger auf zu benutzenden Buffer      *
*             C:  Verify nur durchführen, wenn C <> 00  *
*             B:  max. Anzahl Verify-Versuche           *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  B                                         *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

WRITEV schreibt einen physikalischen Sector mit dem Data Adress Mark für normale Sectoren auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).
Wenn Register C <> 00 ist, wird der Sector nach dem Schreiben getestet, ob er auch ohne Fehler lesbar ist. Wenn nein, wird das Schreiben und anschl. Testen so oft wiederholt, wie Register B angibt.

```
4ABD CD 40 46    CALL    4640         ;Sector schreiben
4AC0 C0          RET     NZ           ;wenn Error
4AC1 79          LD      A,C          ;Verify gewünscht ?
4AC2 B7          OR      A
4AC3 C4 34 46    CALL    NZ,4634      ;wenn ja: Sector testen
4AC6 C8          RET     Z            ;wenn kein Error
4AC7 10 F4       DJNZ    4ABD         ;nächster Versuch
4AC9 C9          RET
```

```
*********************************************************
* Name:       WRITDV                                    *
* Funktion:   schreibt einen Sector des Directory auf   *
*             Diskette, anschließend Verify (optional)  *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             HL: Zeiger auf zu benutzenden Buffer      *
*             C:  Verify nur durchführen, wenn C <> 00  *
*             B:  max. Anzahl Verify-Versuche           *
*             (4308H): gewünschte Drive# (0-3)          *
* Verändert:  B                                         *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

WRITDV schreibt einen physikalischen Sector mit dem Data Adress Mark für Directory-Sectoren auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 4308H die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).
Wenn Register C <> 00 ist, wird der Sector nach dem Schreiben getestet, ob er auch ohne Fehler lesbar ist. Wenn nein, wird das Schreiben und anschl. Testen so oft wiederholt, wie Register B angibt.

```
4ACA CD 3C 46     CALL    463C        ;Sector schreiben
4ACD C0           RET     NZ          ;wenn Error
4ACE 79           LD      A,C         ;Verify gewünscht ?
4ACF B7           OR      A
4AD0 C8           RET     Z           ;wenn nein
4AD1 CD 34 46     CALL    4634        ;Sector testen
4AD4 CD 03 49     CALL    4903        ;Sector mit Data Adress Mark des DIR ?
4AD7 C8           RET     Z           ;wenn kein Error
4AD8 10 F0        DJNZ    4ACA        ;nächster Versuch
4ADA C9           RET
                  ----------------------
                  Fortsetzung der UP's 0013H und 001BH
4ADB 21*00*00     LD      HL,0000     ;Zeiger auf 1. RCB (Route-Control-Block)
4ADE 1A           LD      A,(DE)      ;DCB-Typ holen
4ADF FE C0        CP      C0          ;liegt eine Umleitung durch ROUTE vor ?
-> 4AE1 28 5D     JR      Z,4B40      ;wenn ja
                  ----------------------
4AE3 CD 80 49     CALL    4980        ;PUSH Register, IX=FCB, IY=4380, A=00
4AE6 DD CB 01 FE  SET     7,(IX+01)   ;vermerken, daß LRL <> 256D ist
4AEA 78           LD      A,B         ;DCB-Typ prüfen:
4AEB FE 02        CP      02          ;Input (0013H) oder Output (001BH) ?
4AED DD CB 01 6E  BIT     5,(IX+01)   ;steht der aktuelle Sector im Buffer ?
4AF1 30 A5        JR      NC,4A98     ;wenn Output
```

```
*********************************************************
* Name:       READB (Fortsetzung von 0013H)             *
* Funktion:   nächstes Byte aus einer File lesen        *
* Input:      DE: zeigt auf geöffneten FCB              *
* Verändert:  --                                        *
* Output:     AF: wenn Z=1: A = nächstes Byte           *
*                 wenn Z=0: A = Fehlercode              *
*********************************************************
```

Obige Angaben gelten nur für einen Aufruf über 0013H

```
4AF3 C4 19 4A     CALL    NZ,4A19     ;aktuellen Sector von Diskette lesen,
                                      ;wenn er nicht schon im Buffer steht
4AF6 C0           RET     NZ          ;wenn Error
```

```
4AF7 CD C4 49     CALL    49C4          ;wenn NEXT = EOF: "END OF FILE ENCOUNT."
                                        ;wenn NEXT > EOF: "PAST END OF FILE"
4AFA CD 0B 4B     CALL    4B0B          ;HL zeigt auf nächstes Zeichen im Buffer
4AFD 20 09        JR      NZ,4B08       ;wenn noch nicht am Ende des Buffers
4AFF E5           PUSH    HL            ;HL retten (Zeiger auf nächstes Zeichen)
4B00 CD 62 4A     CALL    4A62          ;wenn notwendig, alten Buffer-Inhalt
                                        ;auf Diskette schreiben
4B03 E1           POP     HL            ;HL zurück (Zeiger auf nächstes Zeichen)
4B04 C0           RET     NZ            ;wenn Error
4B05 CD 0B 4A     CALL    4A0B          ;1. und 2. Byte des NEXT-Wertes +1,
                                        ;und vermerken, daß der aktuelle Sector
                                        ;noch nicht im Buffer steht
4B08 7E           LD      A,(HL)        ;nächstes Zeichen nach A lesen
4B09 BF           CP      A             ;Z=1 (kein Error)
4B0A C9           RET                   ;ggf. nach 49A8: POP Register und RET
                  ----------------------
                  HL auf nächstes Byte im Buffer positionieren
                  und das 3. Byte des NEXT-Feldes um 1 erhöhen
4B0B CD 53 48     CALL    4853          ;HL zeigt auf File-Buffer
4B0E DD 5E 05     LD      E,(IX+05)     ;E = 3. Byte des NEXT-Wertes
4B11 57           LD      D,A           ;D=00
4B12 19           ADD     HL,DE         ;HL auf nächstes Byte im Buffer setzen
4B13 DD 34 05     INC     (IX+05)       ;3. Byte des NEXT-Feldes +1
4B16 C9           RET
                  ----------------------
                  Umleitung eines DCB's durch ROUTE
4B17 7E           LD      A,(HL)        ;HL = RCB+0 (Route-Control-Block)
4B18 BB           CP      E             ;ist in diesem RCB
4B19 23           INC     HL            ;eine Umleitung
4B1A 20 02        JR      NZ,4B1E       ;für den DCB
4B1C 7E           LD      A,(HL)        ;eingetragen,
4B1D BA           CP      D             ;auf den DE   '
4B1E 23           INC     HL            ;gerade zeigt ?
4B1F 20 1A        JR      NZ,4B3B       ;wenn nein
                  ----------------------
4B21 E5           PUSH    HL            ;wenn ja:
4B22 D5           PUSH    DE            ;Register retten
4B23 C5           PUSH    BC
4B24 7E           LD      A,(HL)        ;Original DCB-Typ holen
4B25 A0           AND     B
4B26 23           INC     HL
4B27 23           INC     HL
4B28 23           INC     HL
4B29 5E           LD      E,(HL)        ;Adresse des DCB nach
4B2A 23           INC     HL            ;DE holen, auf den
4B2B 56           LD      D,(HL)        ;umgeleitet werden soll
4B2C D5           PUSH    DE
4B2D DD E1        POP     IX            ;DCB-Adresse nach IX
4B2F CD D4 03     CALL    03D4          ;Umleitung aufrufen
4B32 C1           POP     BC            ;Register zurück
4B33 D1           POP     DE
4B34 E1           POP     HL
                  ----------------------
                  Sind für diesen DCB mehrere Umleitungen angelegt ?
4B35 CB 40        BIT     0,B           ;wurde ein Output-DCB umgeleitet ?
4B37 28 02        JR      Z,4B3B        ;wenn ja
4B39 B7           OR      A             ;Input-DCB: ist ein Zeichen vorhanden ?
4B3A C0           RET     NZ            ;wenn ja
4B3B 23           INC     HL            ;Zeiger
4B3C 7E           LD      A,(HL)        ;auf
```

```
4B3D 23          INC     HL          ;weitere
4B3E 66          LD      H,(HL)      ;RCB's
4B3F 6F          LD      L,A         ;nach HL
4B40 7C          LD      A,H         ;sind weitere RCB's
4B41 B5          OR      L           ;zu bearbeiten ?
4B42 20 D3       JR      NZ,4B17     ;wenn ja
4B44 C3 F8 4C    JP      4CF8        ;wenn nein

     ********************************************************
     * Name:       POSRBA (Fortsetzung von 444EH)           *
     * Funktion:   NEXT-Feld im FCB so positionieren, wie   *
     *             in HL und C im RBA-Format angegeben ist  *
     * Input:      DE: zeigt auf geöffneten FCB             *
     *             HL: 1. und 2. Byte für NEXT-Feld         *
     *             C:  3. Byte für NEXT-Feld                *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4B47 CD 80 49    CALL    4980        ;PUSH Register, IX=FCB, IY=4380, A=00
4B4A 18 34       JR      4B80        ;NEXT-Feld auf HLC setzen

     ********************************************************
     * Name:       POSO (Fortsetzung von 443FH)             *
     * Funktion:   NEXT-Feld im FCB auf Beginn der File     *
     *             positionieren                            *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4B4C CD 80 49    CALL    4980        ;PUSH Register, IX=FCB, IY=4380, A=00
4B4F 67          LD      H,A         ;H=00
4B50 6F          LD      L,A         ;L=00
4B51 4F          LD      C,A         ;C=00
4B52 18 3C       JR      4B90        ;NEXT-Feld auf HLC setzen

     ********************************************************
     * Name:       POSEOF (Fortsetzung von 4448H)           *
     * Funktion:   NEXT-Feld im FCB auf EOF (End of File)   *
     *             positionieren                            *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************

4B54 CD 80 49    CALL    4980        ;PUSH Register, IX=FCB, IY=4380, A=00
4B57 DD 4E 08    LD      C,(IX+08)   ;C = EOF-Wert 3. Byte
4B5A DD 6E 0C    LD      L,(IX+0C)   ;L = EOF-Wert 2. Byte
4B5D DD 66 0D    LD      H,(IX+0D)   ;H = EOF-Wert 1. Byte
4B60 18 22       JR      4B84        ;NEXT-Feld auf HLC setzen

     ********************************************************
     * Name:       POSDEC (Fortsetzung von 4445H)           *
     * Funktion:   NEXT-Feld im FCB um 1 Logische Record#   *
     *             decrementieren (-1)                      *
     * Input:      DE: zeigt auf geöffneten FCB             *
     * Verändert:  --                                       *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

```
4B62 CD 80 49     CALL    4980          ;PUSH Register, IX=FCB, IY=4380, A=00
4B65 CD 68 49     CALL    4968          ;HLC = NEXT-Wert
4B68 AF           XOR     A
4B69 DD 96 09     SUB     (IX+09)       ;Logische Recordlänge (LRL)
4B6C 81           ADD     C             ;vom 3. Byte des
4B6D 4F           LD      C,A           ;NEXT-Wertes abziehen
4B6E 38 10        JR      C,4B80        ;wenn kein Übertrag
4B70 2B           DEC     HL            ;wenn übertrag
4B71 18 0D        JR      4B80          ;NEXT-Feld auf HLC setzen

      ********************************************************
      * Name:       POSBC (Fortsetzung von 4442H)            *
      * Funktion:   NEXT-Feld im FCB auf die in BC ange-     *
      *             gebene Logische Record# positionieren    *
      * Input:      DE: zeigt auf geöffneten FCB             *
      *             BC: gewünschte Logische Record#          *
      * Verändert:  --                                       *
      * Output:     AF: A=Fehlercode, wenn Z=0               *
      ********************************************************

4B73 CD 80 49     CALL    4980          ;PUSH Register, IX=FCB, IY=4380, A=00
4B76 60           LD      H,B           ;gewünschte logische
4B77 69           LD      L,C           ;Record# nach HL
4B78 DD 7E 09     LD      A,(IX+09)     ;Logische Recordlänge (LRL)
4B7B B7           OR      A             ;= 0 ?
4B7C 4F           LD      C,A           ;nach C
4B7D C4 9D 4C     CALL    NZ,4C9D       ;wenn <>0: MULT: HLC = HL*A
                  ----------------------
                  NEXT-Feld auf HLC setzen
4B80 DD CB 01 F6  SET     6,(IX+01)     ;der EOF-Wert soll nach dem Schreiben
                                        ;nur dann auf den NEXT-Wert gesetzt
                                        ;werden, wenn dieser größer als der
                                        ;EOF-Wert ist - wozu ???
4B84 7C           LD      A,H           ;prüfen, ob der
4B85 DD BE 0B     CP      (IX+0B)       ;Sector des gewünschten
4B88 20 06        JR      NZ,4B90       ;NEXT-Wertes mit dem
4B8A 7D           LD      A,L           ;aktuellen Sector
4B8B DD BE 0A     CP      (IX+0A)       ;übereinstimmt
4B8E 28 11        JR      Z,4BA1        ;wenn ja
                  ----------------------
                  Gewünschter NEXT-Wert liegt in einem neuen Sector
4B90 E5           PUSH    HL            ;HLC retten
4B91 C5           PUSH    BC            ;(gewünschter NEXT-Wert)
4B92 CD 62 4A     CALL    4A62          ;wenn notwendig, alten Buffer-Inhalt
                                        ;auf Diskette schreiben
4B95 C1           POP     BC            ;HLC zurück
4B96 E1           POP     HL            ;(gewünschter NEXT-Wert)
4B97 C0           RET     NZ            ;wenn Error
4B98 CD 13 4A     CALL    4A13          ;vermerken, daß der aktuelle Sector
                                        ;noch nicht im Buffer steht
4B9B DD 75 0A     LD      (IX+0A),L     ;2. Byte des NEXT-Feldes = L
4B9E DD 74 0B     LD      (IX+0B),H     ;1. Byte des NEXT-Feldes = H
4BA1 DD 71 05     LD      (IX+05),C     ;3. Byte des NEXT-Feldes = C
4BA4 AF           XOR     A             ;kein Error
4BA5 C9           RET                   ;nach 49A8: POP Register und RET
```

```
     ********************************************************
     * Name:       TEXTTV (Fortsetzung von 4467H)           *
     * Funktion:   Text (HL) auf Bildschirm ausgeben        *
     * Input:      HL: zeigt auf Text, Ende = 03H oder 0DH  *
     * Verändert:  AF                                       *
     * Output:     --                                       *
     ********************************************************

4BA6 D5           PUSH    DE          ;DE retten
4BA7 11 1D 40     LD      DE,401D     ;Zeiger auf Bildschirm-DCB
4BAA E5           PUSH    HL          ;HL retten
4BAB 7E           LD      A,(HL)      ;nächstes Zeichen holen
4BAC FE 03        CP      03          ;Textende ?
4BAE 28 09        JR      Z,4BB9      ;wenn ja
4BB0 CD 1B 00     CALL    001B        ;Zeichen ausgeben
4BB3 7E           LD      A,(HL)      ;ausgegebenes Zeichen
4BB4 FE 0D        CP      0D          ;Textende ?
4BB6 23           INC     HL          ;Zeiger +1
4BB7 20 F2        JR      NZ,4BAB     ;wenn nein, weiter
4BB9 E1           POP     HL          ;HL zurück
4BBA D1           POP     DE          ;DE zurück
4BBB C9           RET

     ********************************************************
     * Name:       TEXTLP (Fortsetzung von 446AH)           *
     * Funktion:   Text (HL) auf Drucker ausgeben           *
     * Input:      HL: zeigt auf Text, Ende = 03H oder 0DH  *
     * Verändert:  AF                                       *
     * Output:     --                                       *
     ********************************************************

4BBC D5           PUSH    DE          ;DE retten
4BBD 11 25 40     LD      DE,4025     ;Zeiger auf Drucker-DCB
4BC0 18 E8        JR      4BAA        ;weiter bei 4BAA
                  ----------------------
                  RST 28H: BREAK-Taste und /SYS-Files laden
4BC2 33           INC     SP          ;RETURN-Addresse im
4BC3 33           INC     SP          ;Stack vergessen
4BC4 FE 20        CP      20          ;ist A < 20H ?
4BC6 DA 12 43     JP      C,4312      ;wenn ja: BREAK-Taste

     ********************************************************
     * Name:       GETSYS                                   *
     * Funktion:   SYS-File (A) laden und starten           *
     * Input:      A: Code für SYS-File (siehe Text)        *
     * Verändert:  HL, DE, BC                               *
     * Output:     AF: A=Fehlercode, wenn Z=0               *
     ********************************************************
```

Die SYS-File, die durch Register A angegeben ist, wird geladen (falls sie nicht schon im Speicher steht) und gestartet, wobei alle Register unverändert übergeben werden.
Das Format des 8-Bit-Registers A ist xxxbbsss, wobei

- bbsss+2 die gewünschte SYS-File (SYS1 - SYS29) bestimmt, indem
- sss angibt, im wievielten Sector des Directory der FPDE dieser SYS-File enthalten ist
- bb angibt, der wievielte FPDE in diesem Directory-Sector der gesuchte FPDE ist und
- xxx eine von evtl. mehreren Funktionen auswählt, die diese SYS-File enthält. Bei mehr als 7 möglichen Funktionen wird zusätzlich

Register C zur Auswahl bestimmter Funktionen benutzt.

Falls die ausgewählte Funktion mit einem RETURN endet, kehrt GETSYS zu seinem Aufrufer zurück (jedoch nicht zum Aufrufer eines RST 28H, sondern 1 Ebene höher) und übergibt alle Register so, wie sie durch die ausgewählte Funktion gesetzt wurden. Es gibt aber auch Funktionen, die nicht mit einem RETURN, sondern mit einem Sprung nach DOSRDY (402DH), ERROR0 (4030H) oder DOSERR (4409H) enden.

```
4BC9 E5           PUSH    HL          ;HL retten
4BCA D5           PUSH    DE          ;DE retten
4BCB C5           PUSH    BC          ;BC retten
4BCC F5           PUSH    AF          ;AF retten
4BCD 21 69 43     LD      HL,4369     ;Bit 6,(4369) setzen (dann sind 'JKL',
4BD0 CB F6        SET     6,(HL)      ;'DFG', '123' und CHAINING verboten)
4BD2 21 BE 45     LD      HL,45BE     ;Abfrage von 'JKL', 'DFG' und '123'
4BD5 36 00        LD      (HL),00     ;erlauben - wozu ?
4BD7 FB           EI                  ;Interrupts erlauben
4BD8 E6 1F        AND     1F          ;Bits 4-0 (benötigte SYS-File) maskieren
4BDA 21 17 43     LD      HL,4317     ;Zeiger auf zuletzt geladene SYS-File
4BDD BE           CP      (HL)        ;ist die benötigte SYS-File im Speicher?
4BDE 28 39        JR      Z,4C19      ;wenn ja
                  ----------------------
                  SYS-File muß geladen werden
4BE0 77           LD      (HL),A      ;neue SYS-File vermerken
4BE1 E6 07        AND     07          ;benötigten DIR-Sector maskieren
4BE3 4F           LD      C,A         ;und in C merken
4BE4 AF           XOR     A           ;NEXT-Wert bei FCB+0A
4BE5 32 D8 43     LD      (43D8),A    ;auf 0 setzen
4BE8 CD 76 47     CALL    4776        ;Drive 0 auswählen und Motor starten
4BEB 7E           LD      A,(HL)      ;DEC (Directory Entry Code)
4BEC 91           SUB     C           ;der gewünschten
4BED 07           RLCA                ;SYS-File berechnen
4BEE 07           RLCA                ;Format: 0bb00sss
4BEF 81           ADD     C           ;(siehe oben)
4BF0 CD 36 49     CALL    4936        ;benötigten DIR-Sector nach 4200H
                                      ;einlesen und HL auf FPDE+0 setzen
4BF3 20 19        JR      NZ,4C0E     ;wenn Error
4BF5 CB 76        BIT     6,(HL)      ;gehört dieser FPDE zu einer SYS-File ?
4BF7 28 15        JR      Z,4C0E      ;wenn nein, Error
     ********** FEHLER! Es wird überhaupt nicht geprüft, **********
     ********** ob die SYS-File evtl. gelöscht ist !     **********
4BF9 C6 16        ADD     16          ;HL auf FPDE+16H
4BFB 6F           LD      L,A         ;setzen
4BFC 5E           LD      E,(HL)      ;die Angaben darüber, wo der
4BFD 23           INC     HL          ;1. Datenblock dieser SYS-File
4BFE 56           LD      D,(HL)      ;auf Diskette steht, von FPDE+16H
4BFF ED 53 DC 43  LD      (43DC),DE   ;nach FCB+0E übertragen
4C03 21 CE 43     LD      HL,43CE     ;HL = FCB zum Laden von SYS-Files
4C06 CD 28 4C     CALL    4C28        ;SYS-File laden
4C09 22 1E 4C     LD      (4C1E),HL   ;Startadresse der SYS-File nach 4C1E
4C0C 28 0B        JR      Z,4C19      ;wenn kein Error
                  ----------------------
                  Fehlerbehandlung
4C0E 3A 17 43     LD      A,(4317)    ;sollte SYS4/SYS
4C11 FE 06        CP      06          ;geladen werden ?
4C13 3E 2E        LD      A,2E        ;Fehlercode "SYSTEM PROGRAM NOT FOUND"
4C15 C2 09 44     JP      NZ,4409     ;wenn nein
4C18 76           HALT                ;NMI auslösen (nicht mask. Interrupt)
                  ----------------------
```

```
                    SYS-File starten
4C19 F1             POP     AF          ;AF zurück
4C1A C1             POP     BC          ;BC zurück
4C1B D1             POP     DE          ;DE zurück
4C1C E1             POP     HL          ;HL zurück
4C1D CD*00*00       CALL    0000        ;SYS-File starten
                    ----------------------
                    Wenn die SYS-File zurückkehrt
4C20 E5             PUSH    HL          ;HL retten
4C21 21 69 43       LD      HL,4369     ;Bit 6,(4369) löschen (Verbot für 'JKL',
4C24 CB B6          RES     6,(HL)      ;'DFG', '123' und CHAINING aufheben)
4C26 E1             POP     HL          ;HL zurück
4C27 C9             RET

      ************************************************************
      * Name:       LOADP                                        *
      * Funktion:   Programm laden                               *
      * Input:      HL: zeigt auf geöffneten FCB                 *
      * Verändert:  DE, BC                                       *
      * Output:     AF: A=Fehlercode, wenn Z=0                   *
      *             HL: Startadresse des Programms               *
      ************************************************************
```

Das zu ladende Programm muß im Standard-Format für LOAD-Files auf Diskette stehen. Dieses Format besteht aus mehreren Blocks, die jeweils mit einem Steuer-Code beginnen, der angibt, wieviele Daten folgen und was mit diesen zu geschehen hat:

a) Steuer-Code 01H: Es folgen 1 Längen-Byte, eine Adresse (LSB, MSB) und soviele Daten, wie das um 2 verminderte Längen-Byte angibt. Die Daten sind im RAM ab der angegebenen Adresse abzulegen.

b) Steuer-Code 02H: Es folgen 1 Byte ohne Bedeutung und die Startadresse (LSB, MSB), bei der das Programm zu starten ist. Damit ist das Ende der File erreicht.

c) Steuer-Code 00H oder 03H bis 1FH: Es folgen 1 Längen-Byte und soviele Daten, wie das Längen-Byte angibt. Diese Daten sind nur ein Kommentar und brauchen nirgendwo abgespeichert zu werden.

Zum Laden benutzt das DOS seinen eigenen Sector-Buffer (4200H-42FFH).

```
4C28 22 6A 4C       LD      (4C6A),HL   ;FCB retten
4C2B 11 FF 42       LD      DE,42FF     ;Zeiger auf Ende des Sector-Buffers
                    ----------------------
                    Steuer-Code lesen
4C2E CD 65 4C       CALL    4C65        ;nächstes Byte lesen
4C31 4F             LD      C,A         ;C = 1. Byte
4C32 3E 1F          LD      A,1F        ;ist das 1. Byte
4C34 B9             CP      C           ;größer als 1FH ?
4C35 3E 22          LD      A,22        ;Fehlercode "LOAD FILE FORMAT ERROR"
4C37 D8             RET     C           ;wenn ja
4C38 CD 65 4C       CALL    4C65        ;nächstes Byte lesen
4C3B 47             LD      B,A         ;B = 2. Byte
4C3C CD 65 4C       CALL    4C65        ;nächstes Byte lesen
4C3F 6F             LD      L,A         ;L = 3. Byte
4C40 61             LD      H,C         ;H = 1. Byte
4C41 0D             DEC     C           ;Steuer-Code 01H ?
4C42 28 09          JR      Z,4C4D      ;wenn ja
4C44 0D             DEC     C           ;Steuer-Code 02H ?
                    ----------------------
4C45 20 1A          JR      NZ,4C61     ;wenn Steuer-Code <> 02H, dann 4C61:
                                        ;H hat einen Wert zwischen 00 und 1F,
                                        ;somit werden Kommentare über das
```

```
                                        ;BASIC-ROM geladen!
                        ----------------------
                        Steuer-Code 02H:
4C47 CD 65 4C           CALL    4C65    ;4. Byte nach H lesen
4C4A 67                 LD      H,A     ;HL = Startadresse
4C4B AF                 XOR     A       ;kein Error
4C4C C9                 RET
                        ----------------------
                        Steuer-Code 01H:
4C4D CD 65 4C           CALL    4C65    ;4. Byte nach H lesen
4C50 67                 LD      H,A     ;HL zeigt auf zu ladenden Speicher
4C51 05                 DEC     B       ;Längen-Byte
4C52 05                 DEC     B       ;um 2 vermindern
4C53 CD 65 4C           CALL    4C65    ;nächstes Byte lesen
4C56 77                 LD      (HL),A  ;im Speicher ablegen
4C57 BE                 CP      (HL)    ;ist dort RAM vorhanden ?
4C58 23                 INC     HL      ;Zeiger auf Speicher +1
4C59 28 06              JR      Z,4C61  ;wenn ja
4C5B 3E 24              LD      A,24    ;Fehler "TRIED TO LOAD READ ONLY MEMORY"
4C5D 0C                 INC     C       ;Steuer-Code 01H oder
4C5E 0D                 DEC     C       ;Kommentar ?
4C5F 28 62              JR      Z,4CC3  ;wenn Steuer-Code 01H: Error
4C61 10 F0              DJNZ    4C53    ;Block zu Ende ?
4C63 18 C9              JR      4C2E    ;wenn ja: nächsten Block bearbeiten
                        ----------------------
                        Nächstes Byte der File holen
4C65 1C                 INC     E       ;Zeiger auf Buffer +1
4C66 1A                 LD      A,(DE)  ;nächstes Byte aus dem Buffer holen
4C67 C0                 RET     NZ      ;wenn Buffer noch nicht zu Ende
4C68 D5                 PUSH    DE      ;DE retten
4C69 11*00*00           LD      DE,0000 ;DE mit FCB laden
4C6C CD 36 44           CALL    4436    ;nächsten Sector lesen
4C6F D1                 POP     DE      ;DE zurück
4C70 20 29              JR      NZ,4C9B ;wenn Error: POP BC + RET
4C72 1A                 LD      A,(DE)  ;1. Byte aus dem Buffer holen
4C73 C9                 RET
```

```
*********************************************************
* Name:       DIRPOS                                    *
* Funktion:   Position eines DIR-Sectors berechnen      *
* Input:      A: gibt an, um den wievielten Sector      *
*                innerhalb des Directory (beginnend     *
*                ab 0) es sich handelt                  *
* Verändert:  AF, BC                                    *
* Output:     DE: gesuchte Sector# (Disk-relativ)       *
*             HL: Buffer für DIR-Sectoren (4200H)       *
*********************************************************
```

DIRPOS berechnet, der wievielte Sector innerhalb einer Diskette (beginnend ab 0) ein bestimmter Directory-Sector ist. Zuvor muß durch DRVSEL (445BH) oder TSTDSK (445EH) das gewünschte Laufwerk ausgewählt worden sein. DIRPOS ist scheinbar etwas "merkwürdig" programmiert, weils es auch zum Berechnen der disk-relativen Sector# eines bestimmten Sectors aus einer File benutzt wird.

```
4C74 32 30 49           LD      (4930),A    ;gewünschte DIR-Sector# bei 4930 merken
4C77 1E 00              LD      E,00        ;E = 00
4C79 2A 99 43           LD      HL,(4399)   ;Zeiger auf PDRIVE-Block
4C7C 4F                 LD      C,A         ;C = gewünschte DIR-Sector#
4C7D 6E                 LD      L,(HL)      ;L = LUMP# vom Beginn des Directory
```

```
4C7E 3A 0F 43     LD      A,(430F)      ;A = GPL (Grans pro Lump)
4C81 CD 92 4C     CALL    4C92          ;MULT: HL = L * A, A = Überlauf
4C84 47           LD      B,A           ;B = Überlauf
4C85 57           LD      D,A           ;D = Überlauf
4C86 19           ADD     HL,DE         ;HL = GRAN# vom Beginn des Directory
4C87 3E 05        LD      A,05          ;Anzahl Sectoren pro Gran = 5
4C89 CD 94 4C     CALL    4C94          ;MULT: HL = L * A, A = Überlauf
4C8C 09           ADD     HL,BC         ;HL = Sector# des gesuchten Sectors
4C8D EB           EX      DE,HL         ;DE = Sector# des gesuchten Sectors
4C8E 21 00 42     LD      HL,4200       ;HL = Buffer für DIR-Sectoren
4C91 C9           RET

     ************************************************************
     * Name:       MULTL                                        *
     * Funktion:   Multipliziere L * A                          *
     * Input:      L: 1. Faktor                                 *
     *             A: 2. Faktor                                 *
     * Verändert:  F                                            *
     * Output:     A=00, HL=Ergebnis                            *
     ************************************************************

4C92 26 00        LD      H,00          ;H=00

     ************************************************************
     * Name:       MULTHL                                       *
     * Funktion:   Multipliziere HL * A                         *
     * Input:      HL: 1. Faktor (falsche Ergebnisse ab         *
     *             A:  2. Faktor              HL > 8080H) !     *
     * Verändert:  F                                            *
     * Output:     AHL (Ergebnis = 65536D * A + HL)             *
     ************************************************************

4C94 C5           PUSH    BC            ;BC retten
4C95 CD 9D 4C     CALL    4C9D          ;MULT: HLC = HL * A
4C98 7C           LD      A,H           ;A = Ergebnis * 65536D
4C99 65           LD      H,L           ;H = Ergebnis * 256D
4C9A 69           LD      L,C           ;L = Ergebnis * 1
4C9B C1           POP     BC            ;BC zurück
4C9C C9           RET

     ************************************************************
     * Name:       MULTC                                        *
     * Funktion:   Multipliziere HL * A                         *
     * Input:      HL: 1. Faktor (falsche Ergebnisse ab         *
     *             A:  2. Faktor              HL > 8080H) !     *
     * Verändert:  F                                            *
     * Output:     HLC (Ergebnis = 256D * HL + C)               *
     ************************************************************

4C9D D5           PUSH    DE            ;DE retten
4C9E EB           EX      DE,HL         ;1. Faktor nach DE
4C9F 0E 80        LD      C,80          ;Zähler für 8 Runden
4CA1 21 00 00     LD      HL,0000       ;Ergebnis mit 0 vorbesetzen
4CA4 0F           RRCA                  ;jeweils Bit 0,A testen
4CA5 30 01        JR      NC,4CA8       ;wenn nicht gesetzt
4CA7 19           ADD     HL,DE         ;Ergebnis + 1. Faktor
4CA8 CB 3C        SRL     H             ;HLC
4CAA CB 1D        RR      L             ;durch 2
4CAC CB 19        RR      C             ;teilen
4CAE 30 F4        JR      NC,4CA4       ;wenn noch keine 8 Runden
```

```
4CB0 D1          POP     DE          ;DE zurück
4CB1 C9          RET

     ********************************************************
     * Name:       DIV05                                    *
     * Funktion:   Dividiere HL / 05 (# Sectoren pro Gran)  *
     * Input:      HL: Dividend                             *
     * Verändert:  C=Divisor, B=00                          *
     * Output:     HL: Quotient = INT(HL/5)                 *
     *             AF: A = Rest, wenn Z=0                   *
     ********************************************************

4CB2 3E 05       LD      A,05        ;Anzahl Sectoren pro Gran = 5

     ********************************************************
     * Name:       DIVA                                     *
     * Funktion:   Dividiere HL / A                         *
     * Input:      HL: Dividend                             *
     *             A:  Divisor                              *
     * Verändert:  C=Divisor, B=00                          *
     * Output:     HL: Quotient = INT(HL/A)                 *
     *             AF: A = Rest, wenn Z=0                   *
     ********************************************************

4CB4 4F          LD      C,A         ;C = Divisor
4CB5 06 10       LD      B,10        ;Zähler für 16D Runden
4CB7 AF          XOR     A           ;Rest mit 0 vorbesetzen
4CB8 29          ADD     HL,HL       ;jeweils Bit 15D,HL testen
4CB9 17          RLA                 ;und nach Bit 0,A übertragen
4CBA 38 03       JR      C,4CBF      ;wenn Bit 7,A gesetzt
4CBC B9          CP      C           ;mit Divisor vergleichen
4CBD 38 02       JR      C,4CC1      ;wenn A < Divisor
4CBF 91          SUB     C           ;Divisor subtrahieren
4CC0 2C          INC     L           ;Ergebnis +1
4CC1 10 F5       DJNZ    4CB8        ;wenn noch keine 16D Runden
4CC3 B7          OR      A           ;Z=1 wenn Rest=00
4CC4 C9          RET

     ********************************************************
     * Name:       CPBCHL                                   *
     * Funktion:   Vergleiche Texte (BC) und (HL)           *
     * Input:      BC: Text1 (Ende = 00H)                   *
     *             HL: Text2 (ohne Ende-Markierung)         *
     * Verändert:  BC                                       *
     * Output:     a) wenn Text1 = Text2:                   *
     *                A=00, Z=1, HL=Ende von Text2          *
     *             b) wenn Text1 <> Text2:                  *
     *                A=34, Z=0, HL ist unverändert         *
     ********************************************************

4CC5 E5          PUSH    HL          ;HL (Zeiger auf Text2) retten
4CC6 0A          LD      A,(BC)      ;nächstes Zeichen von Text1 holen
4CC7 B7          OR      A           ;Ende-Markierung ?
4CC8 03          INC     BC          ;Zeiger auf Text1 +1
4CC9 20 03       JR      NZ,4CCE     ;wenn nein
4CCB E3          EX      (SP),HL     ;wenn ja (Text1 = Text2):
4CCC E1          POP     HL          ;2 * INC SP
4CCD C9          RET                 ;fertig
4CCE BE          CP      (HL)        ;Text1 mit Text2 vergleichen
4CCF 23          INC     HL          ;Zeiger auf Text2 +1
```

```
4CD0 28 F4        JR      Z,4CC6        ;wenn gleich: weiter
4CD2 E1           POP     HL            ;HL zurück
4CD3 18 14        JR      4CE9          ;weiter bei 4CE9: A=34, Z=0, RET

     ********************************************************
     * Name:       NEXTC1                                   *
     * Funktion:   Nächstes Zeichen von (HL) holen und      *
     *             in Abhängigkeit vom Inhalt Flags setzen  *
     * Input:      HL: Zeiger auf Text (z. B. Input-Buffer) *
     * Verändert:  --                                       *
     * Output:     HL, AF (siehe Text)                      *
     ********************************************************
```

In Abhängigkeit von dem Zeichen, auf das HL gerade zeigt, werden HL, A, sowie die Flags Z (Zero) und C (Carry) folgendermaßen gesetzt:

Zeichen	Z	C	A	HL zeigt auf
0DH (ENTER)	Z	NC	0DH	unverändert
2CH (Komma)	NZ	NC	34H	1. Zeichen nach dem Komma
20H (Blank)	NZ	NC	34H	1. Zeichen, das ggf. mehreren Blanks folgt
Rest	NZ	C	34H	unverändert

```
4CD5 7E           LD      A,(HL)        ;Zeichen holen
4CD6 FE 0D        CP      0D            ;= 0DH ?
4CD8 C8           RET     Z             ;wenn ja

     ********************************************************
     * Name:       NEXTC2                                   *
     * Funktion:   Nächstes Zeichen von (HL) holen und      *
     *             in Abhängigkeit vom Inhalt Flags setzen  *
     * Input:      HL: Zeiger auf Text (z. B. Input-Buffer) *
     * Verändert:  --                                       *
     * Output:     HL, AF (siehe Text)                      *
     ********************************************************
```

In Abhängigkeit von dem Zeichen, auf das HL gerade zeigt, werden HL, A, sowie die Flags Z (Zero) und C (Carry) folgendermaßen gesetzt:

Zeichen	Z	C	A	HL zeigt auf
2CH (Komma)	NZ	NC	34H	1. Zeichen nach dem Komma
20H (Blank)	NZ	NC	34H	1. Zeichen, das ggf. mehreren Blanks folgt
Rest	NZ	C	34H	unverändert

```
4CD9 7E           LD      A,(HL)        ;Zeichen holen
4CDA FE 2C        CP      2C            ;= Komma (,) ?
4CDC 23           INC     HL            ;Zeiger +1
4CDD 28 0A        JR      Z,4CE9        ;wenn ja
4CDF FE 20        CP      20            ;= Blank ?
4CE1 2B           DEC     HL            ;Zeiger -1
4CE2 37           SCF                   ;C-Flag setzen
4CE3 20 05        JR      NZ,4CEA       ;wenn nein
4CE5 23           INC     HL            ;Zeiger +1
4CE6 BE           CP      (HL)          ;Blank ?
4CE7 28 FC        JR      Z,4CE5        ;wenn ja
4CE9 B7           OR      A             ;Flags Z=0 und C=0
4CEA 3E 34        LD      A,34          ;Fehlercode "ILLEGAL KEYWORD OR
4CEC C9           RET                   ;            SEPARATOR OR TERMINATOR"
```

```
      ********************************************************
      * Name:       DELAY2                                   *
      * Funktion:   ca. B * 3.75 ms warten                   *
      * Input:      B: Zeitfaktor                            *
      * Verändert:  DE, BC, AF                               *
      * Output:     --                                       *
      ********************************************************

4CED 50           LD      D,B         ;B (Zeitfaktor) retten
4CEE 1E*01        LD      E,01        ;SYSTEM BJ (CPU Speed)
4CF0 42           LD      B,D         ;B (Zeitfaktor) zurück
4CF1 CD 60 00     CALL    0060        ;Verzögerung B * 3.75 ms (1.774 MHz)
4CF4 1D           DEC     E           ;schon SYSTEM BJ mal ?
4CF5 20 F9        JR      NZ,4CF0     ;wenn nein
4CF7 C9           RET
                  ----------------------
                  Hilfsroutine für DOS-Befehl ROUTE
4CF8 CB 40        BIT     0,B         ;DCB für Input oder Output ?
4CFA C0           RET     NZ          ;wenn Input
4CFB 79           LD      A,C         ;wenn Output: Zeichen zurück
4CFC C9           RET
                  ----------------------

(4CFC C9)         DCB für ROUTE 'NL'
4CFD A4 4B        ;DCB-Typ
4CFF 00           ;Treiberadresse
                  ;./.


                  ----------------------
                  Initialisierung von SYS0/SYS
4D00 A5           ;Stackpointer steht auf 41E0
4D01 ED 56                            ;Kennung von NEWDOS/80 und GDOS
                  IM      1           ;Interrupts über RST 38H
                  ----------------------
                  Ende des RAM ab FFFFH abwärts suchen
4D03 21 FF FF     LD      HL,FFFF     ;Zeiger auf FFFFH
4D06 7E           LD      A,(HL)      ;Speicherinhalt lesen
4D07 2F           CPL                 ;alle Bits invertieren
4D08 77           LD      (HL),A      ;und zurückschreiben
4D09 BE           CP      (HL)        ;RAM vorhanden ?
4D0A 2B           DEC     HL          ;Zeiger -1
4D0B 20 F9        JR      NZ,4D06     ;wenn nein
4D0D 2F           CPL                 ;alten
4D0E 23           INC     HL          ;Speicherinhalt
4D0F 77           LD      (HL),A      ;zurückschreiben
4D10 22 A9 43     LD      (43A9),HL   ;höchste RAM-Adresse nach 43A9
4D13 22 49 40     LD      (4049),HL   ;und nach 4049 (HIMEM)
                  ----------------------
                  Uhrzeit + Datum nach 4041-4046 zurück
4D16 21 AB 43     LD      HL,43AB     ;war NEWDOS/80 oder GDOS
4D19 3E A5        LD      A,A5        ;schon vor dem
4D1B BE           CP      (HL)        ;RESET aktiv ?
4D1C 20 09        JR      NZ,4D27     ;wenn nein
4D1E 23           INC     HL          ;wenn ja:
4D1F 11 41 40     LD      DE,4041     ;Uhrzeit + Datum
4D22 01 06 00     LD      BC,0006     ;aus dem Buffer 43AC-43B1
4D25 ED B0        LDIR                ;nach 4041-4046 übertragen
                  ----------------------
                  PDRIVE-Sector von Drive 0 laden
4D27 FD 21 80 43  LD      IY,4380     ;Zeiger auf 4300 - 43FF
```

```
4D2B ED 4B FE 42  LD      BC,(42FE)     ;PDRIVE-Parameter aus dem BOOT-Sector
4D2F FD 70 F8     LD      (IY+F8),B     ;4378 = PDRIVE+7 für Drive 0
4D32 FD 71 F3     LD      (IY+F3),C     ;4373 = PDRIVE+2 für Drive 0
4D35 3A FD 42     LD      A,(42FD)      ;PDRIVE-Parameter aus dem BOOT-Sector
4D38 FD 77 F7     LD      (IY+F7),A     ;4377 = PDRIVE+6 für Drive 0
     ********** FEHLER! Das Lesen des PDRIVE-SECTORS erfolgt **********
     ********** mit den Zeitkonstanten für SYSTEM BJ=1 !     **********
4D3B 11 80 44     LD      DE,4480       ;DE auf FCB (File Control Block)
4D3E CD 36 44     CALL    4436          ;READ PDRIVE-Sector nach 4200
4D41 C2 D3 4D     JP      NZ,4DD3       ;wenn Error aufgetreten
4D44 3A EF 42     LD      A,(42EF)      ;Kennzeichnung für den
4D47 FE A5        CP      A5            ;PDRIVE-Sector vorhanden ?
4D49 C2 D1 4D     JP      NZ,4DD1       ;wenn nein, Error
                  ----------------------
                  Auswertung der SYSTEM-Parameter aus dem PDRIVE-Sector
4D4C 21 F8 42     LD      HL,42F8       ;SYSTEM-Parameter von
4D4F 11 5B 50     LD      DE,505B       ;42F8-42FF nach
4D52 01 08 00     LD      BC,0008       ;505B-5062
4D55 ED B0        LDIR                  ;zwischenspeichern
4D57 2A F0 42     LD      HL,(42F0)     ;(436C) = (PDRIVE-Sector Byte F0)
4D5A 22 6C 43     LD      (436C),HL     ;(436D) = (PDRIVE-Sector Byte F1)
4D5D 2A F2 42     LD      HL,(42F2)     ;(436E) = (PDRIVE-Sector Byte F2)
4D60 22 6E 43     LD      (436E),HL     ;(436F) = (PDRIVE-Sector Byte F3)
                  ----------------------
4D63 3A A0 42     LD      A,(42A0)      ;SYSTEM AL (Anzahl Drives)
4D66 32 9F 43     LD      (439F),A      ;nach 439F
4D69 32 7A 47     LD      (477A),A      ;und 477A
4D6C 3D           DEC     A             ;wenn AL < 1
4D6D FE 04        CP      04            ;oder > 4 ist
4D6F 30 60        JR      NC,4DD1       ;dann Error
                  ----------------------
4D71 3A A1 42     LD      A,(42A1)      ;SYSTEM AW (Wiederhol. b. Verify-Errors)
4D74 32 BA 4A     LD      (4ABA),A      ;nach 4ABA
4D77 3A A2 42     LD      A,(42A2)      ;SYSTEM AN (Drive# für DIR)
4D7A 32 A0 43     LD      (43A0),A      ;nach 43A0
4D7D 3A A3 42     LD      A,(42A3)      ;SYSTEM AO (Drive# f. CREATE neue Files)
4D80 32 A1 43     LD      (43A1),A      ;nach 43A1
4D83 3A A6 42     LD      A,(42A6)      ;SYSTEM AM (Wiederhol. bei Disk-Errors)
4D86 32 5A 46     LD      (465A),A      ;nach 465A
                  ----------------------
4D89 3A A9 42     LD      A,(42A9)      ;SYSTEM BJ (CPU Speed)
4D8C 32 A2 43     LD      (43A2),A      ;nach 43A2
4D8F 32 EF 4C     LD      (4CEF),A      ;und 4CEF
4D92 2E 06        LD      L,06          ;1. Multiplikations-Faktor für BJ
4D94 11 00 24     LD      DE,2400       ;2. Multiplikations-Faktor für BJ
4D97 F5           PUSH    AF            ;A = SYSTEM BJ
4D98 CD 92 4C     CALL    4C92          ;MULT: HL = L * A
4D9B B4           OR      H             ;HL > 255D ?
4D9C 20 33        JR      NZ,4DD1       ;wenn ja, Error
4D9E 7D           LD      A,L           ;Ergebnis: 6 * BJ
4D9F 32 E4 47     LD      (47E4),A      ;nach 47E4
4DA2 F1           POP     AF            ;A = SYSTEM BJ
4DA3 EB           EX      DE,HL         ;HL = 2. Multiplikations-Faktor 2400H
4DA4 CD 94 4C     CALL    4C94          ;MULT: HL = HL * A, A = Überlauf
4DA7 B7           OR      A             ;Überlauf ?
4DA8 20 27        JR      NZ,4DD1       ;wenn ja, Error
4DAA 22 F4 47     LD      (47F4),HL     ;Ergebnis: 2400H * BJ nach 47F4,47F5
                  ----------------------
4DAD 3A A8 42     LD      A,(42A8)      ;SYSTEM AX (höchster Drucker-ASCII-Code)
4DB0 32 70 43     LD      (4370),A      ;nach 4370
```

```
                        ----------------------
4DB3 2A 49 40     LD    HL,(4049)     ;HL = (HIMEM)
4DB6 ED 5B D0 42  LD    DE,(42D0)     ;DE = SYSTEM AP (HIMEM)
4DBA 7A           LD    A,D           ;ist SYSTEM AP
4DBB B3           OR    E             ; = 0000H ?
4DBC 28 06        JR    Z,4DC4        ;wenn ja
4DBE B7           OR    A             ;ansonsten:
4DBF ED 52        SBC   HL,DE         ;ist SYSTEM AP > HIMEM ?
4DC1 38 01        JR    C,4DC4        ;wenn ja
4DC3 EB           EX    DE,HL         ;HL = SYSTEM AP
4DC4 22 49 40     LD    (4049),HL     ;nach HIMEM speichern
                        ----------------------
                  Auswertung der PDRIVE-Parameter aus dem PDRIVE-Sector
4DC7 AF           XOR   A             ;Zähler = 0
4DC8 11 71 43     LD    DE,4371       ;Zeiger auf 1. PDRIVE-Block im RAM
4DCB DD 21 00 42  LD    IX,4200       ;Zeiger auf 1. PDRIVE-Block im Buffer
4DCF 18 06        JR    4DD7          ;weiter bei 4DD7
                        ----------------------
                  Fehlerbehandlung
4DD1 3E 27        LD    A,27          ;Fehlercode "ILLEGAL INITIALISATION DATA
                                      ;ON SYSTEM DISKETTE"
4DD3 F5           PUSH  AF            ;Fehlercode retten
4DD4 3E 46        LD    A,46          ;Code für SYS4/SYS
4DD6 EF           RST   28            ;SYS-File laden und starten
                        ----------------------
                  Fortsetzung: Auswertung der PDRIVE-Parameter
4DD7 01 0A 00     LD    BC,000A       ;Anzahl Bytes pro PDRIVE-Block = 10D
4DDA F5           PUSH  AF            ;sind schon soviele PDRIVE-Blocks über-
4DDB FD BE 1F     CP    (IY+1F)       ;tragen, wie (439F) SYSTEM AL angibt ?
4DDE 30 05        JR    NC,4DE5       ;wenn ja
4DE0 DD E5        PUSH  IX            ;ansonsten nächsten PDRIVE-Block
4DE2 E1           POP   HL            ;aus dem Disk-Buffer
4DE3 ED B0        LDIR                ;nach 4371/437B/4385/438F übertragen
4DE5 DD 7E 02     LD    A,(IX+02)     ;sind die PDRIVE-Parameter für
4DE8 E6 1C        AND   1C            ;dieses Laufwerk fehlerhaft ?
4DEA 28 E5        JR    Z,4DD1        ;wenn ja, Error
4DEC 0E 10        LD    C,10          ;Zeiger im Disk-Buffer auf den
4DEE DD 09        ADD   IX,BC         ;nächsten PDRIVE-Block setzen
4DF0 F1           POP   AF            ;Zähler f. schon erledigte PDRIVE-Blocks
4DF1 3C           INC   A             ;Zähler +1
4DF2 FE 0A        CP    0A            ;schon 10D PDRIVE-Blocks bearbeitet ?
4DF4 38 E1        JR    C,4DD7        ;wenn nein
                        ----------------------
                  Weitere Bearbeitung der SYSTEM-Parameter
4DF6 3A FA 42     LD    A,(42FA)      ;SYSTEM BN=Y ?
4DF9 CB 7F        BIT   7,A           ;(Data Adress Mark von Model III)
4DFB 28 05        JR    Z,4E02        ;wenn nein
4DFD 3E AB        LD    A,AB          ;neuer WRITE-SECTOR Befehl für FDC
4DFF 32 3D 46     LD    (463D),A      ;nach 463D
                        ----------------------
4E02 ED 4B F8 42  LD    BC,(42F8)     ;diverse SYSTEM-Parameter nach BC
4E06 ED 5B 6C 43  LD    DE,(436C)     ;diverse SYSTEM-Parameter nach DE
                        ----------------------
4E0A CB 73        BIT   6,E           ;SYSTEM AB=Y ? (RUN-ONLY Modus)
4E0C 28 08        JR    Z,4E16        ;wenn nein
4E0E FD CB E9 D6  SET   2,(IY+E9)     ;SET 2,(4369)
4E12 FD CB ED B6  RES   6,(IY+ED)     ;RES 6,(436D) => SYSTEM BC=N
                        ----------------------
4E16 CB 41        BIT   0,C           ;SYSTEM AU=Y ? (Tastatur mit Repeat)
4E18 28 0E        JR    Z,4E28        ;wenn nein
```

```
4E1A 3E 20      LD      A,20        ;"JR  NZ,"
4E1C 32 38 45   LD      (4538),A    ;nach 4538
4E1F 3A A7 42   LD      A,(42A7)    ;SYSTEM AV (Repeat-Faktor)
4E22 32 78 45   LD      (4578),A    ;nach 4578
4E25 32 2F 45   LD      (452F),A    ;und 452F
                ---------------------
4E28 3A A5 42   LD      A,(42A5)    ;SYSTEM BI=00H ?
4E2B B7         OR      A           ;(Cursor Zeichen)
4E2C 28 03      JR      Z,4E31      ;wenn ja
4E2E 32 01 45   LD      (4501),A    ;Cursor Zeichen nach 4501
                ---------------------
4E31 CB 49      BIT     1,C         ;SYSTEM AJ=Y ? (Tastatur- und
                                    ;Bildschirm-Routine von NEWDOS/80)
4E33 28 13      JR      Z,4E48      ;wenn nein
4E35 3A 40 38   LD      A,(3840)    ;Tastatur abfragen
4E38 CB 5F      BIT     3,A         ;Taste "Hochpfeil" gedrückt ?
4E3A 20 0C      JR      NZ,4E48     ;wenn ja
4E3C 21 16 45   LD      HL,4516     ;neue Tastatur-Routine
4E3F 22 16 40   LD      (4016),HL   ;bei 4016 eintragen
4E42 21 05 45   LD      HL,4505     ;neue Bildschirm-Routine
4E45 22 1E 40   LD      (401E),HL   ;bei 401E eintragen
                ---------------------
4E48 21 00 3C   LD      HL,3C00     ;Zeiger auf Bildschirm-RAM
4E4B 3E 61      LD      A,61        ;testen ob das
4E4D 77         LD      (HL),A      ;Bildschirm-RAM
4E4E BE         CP      (HL)        ;auf Kleinbuchstaben
4E4F 36 20      LD      (HL),20     ;umgerüstet wurde
4E51 20 17      JR      NZ,4E6A     ;wenn nein
4E53 CB 50      BIT     2,B         ;SYSTEM BF=Y ?
                                    ;(Bildschirm mit Kleinbuchstaben)
4E55 28 0A      JR      Z,4E61      ;wenn nein
4E57 3E 4F      LD      A,4F        ;"LD  C,A"
4E59 32 93 45   LD      (4593),A    ;nach 4593
4E5C 3E 38      LD      A,38        ;"JR  C,"
4E5E 32 05 45   LD      (4505),A    ;nach 4505
4E61 CB 48      BIT     1,B         ;SYSTEM BG=Y ?
                                    ;(Tastatur mit Kleinbuchstaben)
4E63 20 05      JR      NZ,4E6A     ;wenn ja
4E65 3E 00      LD      A,00        ;"NOP"
4E67 32 B4 45   LD      (45B4),A    ;nach 45B4
                ---------------------
4E6A CB 40      BIT     0,B         ;SYSTEM BH=Y ? (blinkender Cursor)
4E6C 28 05      JR      Z,4E73      ;wenn nein
4E6E 3E C8      LD      A,C8        ;"RET  Z"
4E70 32 02 45   LD      (4502),A    ;nach 4502
                ---------------------
4E73 CB 79      BIT     7,C         ;SYSTEM AC=Y ?
                                    ;(Tastatur-Entprellung von NEWDOS/80)
4E75 20 05      JR      NZ,4E7C     ;wenn ja
4E77 3E 3E      LD      A,3E        ;"LD  A,"
4E79 32 7D 45   LD      (457D),A    ;nach 457D
                ---------------------
4E7C CB 71      BIT     6,C         ;SYSTEM AD=Y ? ('JKL' erlaubt)
4E7E 28 05      JR      Z,4E85      ;wenn nein
4E80 3E C0      LD      A,C0        ;"RET  NZ"
4E82 32 EE 45   LD      (45EE),A    ;nach 45EE
                ---------------------
4E85 CB 61      BIT     4,C         ;SYSTEM AF=Y ? ('DFG' erlaubt)
4E87 28 05      JR      Z,4E8E      ;wenn nein
4E89 3E 20      LD      A,20        ;"JR  NZ,"
```

```
4E8B 32 CC 45     LD      (45CC),A      ;nach 45CC
                  ----------------------
4E8E CB 51        BIT     2,C           ;SYSTEM AQ=Y ? (CLEAR-Taste erlaubt)
4E90 28 04        JR      Z,4E96        ;wenn nein
4E92 AF           XOR     A             ;"00"
4E93 32 90 45     LD      (4590),A      ;nach 4590
                  ----------------------
4E96 CB 69        BIT     5,C           ;SYSTEM AE=Y ? ('123' erlaubt)
4E98 28 05        JR      Z,4E9F        ;wenn nein
4E9A 3E 20        LD      A,20          ;"JR  NZ,"
4E9C 32 D8 45     LD      (45D8),A      ;nach 45D8
                  ----------------------
4E9F CB 6B        BIT     5,E           ;SYSTEM AG=Y ? (BREAK-Taste erlaubt)
4EA1 28 04        JR      Z,4EA7        ;wenn nein
4EA3 FD CB E9 E6  SET     4,(IY+E9)     ;SET 4,(4369)
                  ----------------------
4EA7 21 A8 4F     LD      HL,4FA8       ;Zeiger auf Tabelle "CLS"
4EAA CD 67 44     CALL    4467          ;Bildschirm löschen
                  ----------------------
4EAD 3A F9 42     LD      A,(42F9)      ;SYSTEM BA=Y ?
4EB0 CB 6F        BIT     5,A           ;(ROUTE DO,NL)
4EB2 28 18        JR      Z,4ECC        ;wenn nein
4EB4 F3           DI                    ;Interrupts sperren
4EB5 21 B2 43     LD      HL,43B2       ;Zeiger auf 1. RCB (ROUTE Control Block)
4EB8 22 DC 4A     LD      (4ADC),HL     ;nach 4ADC eintragen
4EBB 11 1D 40     LD      DE,401D       ;DCB der Bildschirm-Routine
4EBE 73           LD      (HL),E        ;in RCB+0
4EBF 23           INC     HL            ;und RCB+1
4EC0 72           LD      (HL),D        ;eintragen
4EC1 23           INC     HL            ;Zeiger auf RCB+2
4EC2 1A           LD      A,(DE)        ;DCB-Typ der Bildschirm-Routine
4EC3 77           LD      (HL),A        ;bei RCB+2 speichern
4EC4 23           INC     HL            ;Zeiger auf RCB+3
4EC5 3E C0        LD      A,C0          ;neuen DCB-Typ
4EC7 12           LD      (DE),A        ;nach 401D (DCB+0) eintragen
4EC8 AF           XOR     A             ;im RCB+3 und RCB+4
4EC9 77           LD      (HL),A        ;den Zeiger auf
4ECA 23           INC     HL            ;weitere RCB's
4ECB 77           LD      (HL),A        ;löschen
                  ----------------------
4ECC FB           EI                    ;Interrupts erlauben
4ECD 21 AB 4F     LD      HL,4FAB       ;Text "NEWDOS-80 ... Vers. 2.052 ..."
4ED0 CD 67 44     CALL    4467          ;auf Bildschirm ausgeben
                  ----------------------
                  Datum und Uhrzeit abfragen und anzeigen
4ED3 3A AB 43     LD      A,(43AB)      ;prüfen, ob NEWDOS/80 oder GDOS
4ED6 FE A5        CP      A5            ;schon vor dem RESET aktiv war,
4ED8 3A F9 42     LD      A,(42F9)      ;SYSTEM AY und SYSTEM AZ
4EDB 20 02        JR      NZ,4EDF       ;(Datum und Uhrzeit eingeben)
4EDD CB BF        RES     7,A           ;prüfen
4EDF E6 C0        AND     C0            ;und
4EE1 C4 3D 4F     CALL    NZ,4F3D       ;ggf. INPUT Datum und Uhrzeit
4EE4 21 89 50     LD      HL,5089       ;Zeiger auf Buffer für Datum + Uhrzeit
4EE7 E5           PUSH    HL            ;Datum im MM/DD/YY-Format
4EE8 CD C2 44     CALL    44C2          ;nach 5089-5090 übertragen
4EEB 21 93 50     LD      HL,5093       ;Uhrzeit im HH:MM:SS-Format
4EEE CD A7 44     CALL    44A7          ;nach 5093-509A übertragen
4EF1 E1           POP     HL            ;Datum und Uhrzeit
4EF2 CD 67 44     CALL    4467          ;auf Bildschirm ausgeben
                  ----------------------
```

```
                        GAT-Sector von Drive 0 lesen wegen evtl. AUTO-Befehl
4EF5 AF                 XOR     A               ;Nummer der DIR-Sectors (GAT=0)
4EF6 32 30 49           LD      (4930),A        ;nach 4930
4EF9 21 05 4F           LD      HL,4F05         ;Fortsetzungsadresse 4F05
4EFC E5                 PUSH    HL              ;in den Stack schreiben
4EFD D5                 PUSH    DE              ;DE und BC
4EFE C5                 PUSH    BC              ;ebenfalls in den Stack
4EFF 21 00 42           LD      HL,4200         ;Buffer = 4200
4F02 C3 11 49           JP      4911            ;GAT-Sector nach 4200 lesen
4F05 C2 D3 4D           JP      NZ,4DD3         ;wenn Error
4F08 11 18 43           LD      DE,4318         ;Zeiger auf Input-Buffer des DOS
4F0B D5                 PUSH    DE              ;Zeiger retten
4F0C C5                 PUSH    BC              ;wozu BC retten ?
4F0D 21 E0 42           LD      HL,42E0         ;Zeiger auf AUTO-Befehl im GAT-Sector
4F10 01 20 00           LD      BC,0020         ;max. Länge des AUTO-Befehls
4F13 ED B0              LDIR                    ;AUTO-Befehl in Input-Buffer übertragen
4F15 21 AB 43           LD      HL,43AB         ;vermerken, daß NEWDOS/80
4F18 36 A5              LD      (HL),A5         ;bzw. GDOS initialisiert ist
4F1A C1                 POP     BC              ;BC zurück
4F1B E1                 POP     HL              ;Zeiger auf AUTO-Befehl im Input-Buffer
4F1C AF                 XOR     A               ;überflüssig
4F1D FD CB EC 76        BIT     6,(IY+EC)       ;SYSTEM AB=Y ? (RUN-ONLY Modus)
4F21 20 14              JR      NZ,4F37         ;wenn ja
4F23 3A 5C 50           LD      A,(505C)        ;SYSTEM BD=Y ?
4F26 CB 5F              BIT     3,A             ;(AUTO-Befehl durch ENTER stoppen)
4F28 28 0D              JR      Z,4F37          ;wenn nein
4F2A 3A 40 38           LD      A,(3840)        ;Tastatur abfragen
4F2D 0F                 RRCA                    ;ENTER gedrückt ?
4F2E DA 00 44           JP      C,4400          ;wenn ja: Sprung nach DOS READY
4F31 7E                 LD      A,(HL)          ;Zeiger auf AUTO-Befehl
4F32 FE 0D              CP      0D              ;ist ein AUTO-Befehl vorhanden ?
4F34 CA 00 44           JP      Z,4400          ;wenn nein: Sprung nach DOS READY
4F37 CD 67 44           CALL    4467            ;AUTO-Befehl auf Bildschirm anzeigen
4F3A C3 05 44           JP      4405            ;AUTO-Befehl ausführen
                        ----------------------
                        Eingabe des Datums
4F3D 21 63 50           LD      HL,5063         ;Text "Date?  (MM/DD/YY) "
4F40 CD 64 4F           CALL    4F64            ;anzeigen und Eingabe holen
4F43 01 9C 50           LD      BC,509C         ;Tabelle für zulässige Datums-Eingabe
4F46 11 46 40           LD      DE,4046         ;Zeiger auf RAM-Buffer für Datum
4F49 3E 2F              LD      A,2F            ;Trennzeichen "/"
4F4B CD 6F 4F           CALL    4F6F            ;Datum in Buffer eintragen
4F4E 20 ED              JR      NZ,4F3D         ;wenn Eingabefehler
                        ----------------------
                        Eingabe der Uhrzeit
4F50 21 76 50           LD      HL,5076         ;Text "Time?  (HH:MM:SS) "
4F53 CD 64 4F           CALL    4F64            ;anzeigen und Eingabe holen
4F56 01 A2 50           LD      BC,50A2         ;Tabelle für zulässige Uhrzeit-Eingabe
4F59 11 43 40           LD      DE,4043         ;Zeiger auf RAM-Buffer für Uhrzeit
4F5C 3E 3A              LD      A,3A            ;Trennzeichen ":"
4F5E CD 6F 4F           CALL    4F6F            ;Uhrzeit in Buffer eintragen
4F61 20 ED              JR      NZ,4F50         ;wenn Eingabefehler
4F63 C9                 RET
                        ----------------------
                        Datum oder Uhrzeit eingeben
4F64 CD 67 44           CALL    4467            ;Text anzeigen
4F67 21 18 43           LD      HL,4318         ;Zeiger auf Input-Buffer
4F6A 06 09              LD      B,09            ;max. 9 Zeichen holen
4F6C C3 40 00           JP      0040            ;Zeilen-Eingabe des BASIC-Interpreters
                        ----------------------
```

```
                    Datum oder Uhrzeit auf Eingabefehler prüfen
                    und in ihren Buffer übertragen
4F6F 32 A0 4F       LD      (4FA0),A      ;Trennzeichen retten
4F72 F3             DI
4F73 C5             PUSH    BC
4F74 06 03          LD      B,03          ;Zähler: 3 Werte
4F76 7E             LD      A,(HL)        ;nächstes Digit
4F77 D6 30          SUB     30
4F79 FE 0A          CP      0A
4F7B 23             INC     HL
4F7C 30 26          JR      NC,4FA4       ;wenn keine Zahl
4F7E 4F             LD      C,A
4F7F 07             RLCA
4F80 07             RLCA
4F81 81             ADD     C             ;C = 10D * 1. Digit
4F82 87             ADD     A
4F83 4F             LD      C,A
4F84 7E             LD      A,(HL)        ;nächstes Digit
4F85 D6 30          SUB     30
4F87 FE 0A          CP      0A
4F89 23             INC     HL
4F8A 30 18          JR      NC,4FA4       ;wenn keine Zahl
4F8C 81             ADD     C
4F8D 12             LD      (DE),A        ;10 * 1. Digit + 2. Digit in Buffer
4F8E 1B             DEC     DE
4F8F E3             EX      (SP),HL       ;Tabelle der zulässigen Eingaben
4F90 96             SUB     (HL)
4F91 23             INC     HL
4F92 BE             CP      (HL)          ;Eingabe prüfen
4F93 23             INC     HL
4F94 30 0E          JR      NC,4FA4       ;wenn Fehler
4F96 E3             EX      (SP),HL
4F97 10 05          DJNZ    4F9E          ;Trennzeichen prüfen + nächster Wert
4F99 C1             POP     BC
4F9A FB             EI
4F9B C3 D5 4C       JP      4CD5          ;Fehler, wenn kein 0DH folgt
                    ----------------------
4F9E 7E             LD      A,(HL)
4F9F FE*00          CP      00            ;Trennzeichen prüfen
4FA1 23             INC     HL
4FA2 28 D2          JR      Z,4F76        ;wenn OK
                    ----------------------
                    Eingabefehler
4FA4 FB             EI
4FA5 F1             POP     AF
4FA6 B7             OR      A             ;Z-Flag löschen
4FA7 C9             RET
                    ----------------------
                    Tabelle zum Löschen des Bildschirms
4FA8 1C             ;Cursor Home
4FA9 1F             ;Rest des Bildschirms löschen
4FAA 03             ;Ende der Tabelle
                    ----------------------
                    Text "NEWDOS-80 ... Apparat Inc. ... Vers. 2.052 ... Model I"
4FAB BF 83 AF BF 83 BF 83 B3 B3 B3 BF 83 BF BF 83 BF  ................
4FBB 83 B3 B3 8B BF 87 B3 B3 8B BF 87 B3 B3 8B BF BF  ................
4FCB BF BF BF BF 87 B3 B3 8B BF 87 B3 93 8B BF 20 41  .............. A
4FDB 70 70 61 72 61 74 20 49 6E 63 2E 0A               pparat Inc..
                    ----------------------
4FE7 BF 80 B4 8B 80 BF 80 B3 B3 BB BF 80 87 8B 80 BF  ................
```

```
4FF7 80 BF BF 80 BF 80 BF BF 80 BF B4 B3 B3 8B BF B7  ................
5007 B3 B3 BB BF 84 B3 B3 88 BF 80 87 B8 80 BF 20 56  .............. V
5017 65 72 73 2E 20 32 2E 30 35 32 0A                 ers. 2.052.
                     ----------------------
5022 BF B0 BF BD B0 BF B0 B3 B3 B3 BF B0 BE BD B0 BF  ................
5032 B0 B3 B3 B8 BF B4 B3 B3 B8 BF B4 B3 B3 B8 BF BF  ................
5042 BF BF BF BF B4 B3 B3 B8 BF B4 B2 B3 B8 BF 20 C2  .............. .
5052 4D 6F 64 65 6C 20 49 0A 0D                       Model I..
                     ----------------------
                     Buffer für SYSTEM-Parameter aus dem PDRIVE-Sector
505B 00              ;PDRIVE-Sector Byte F8
505C 00              ;PDRIVE-Sector Byte F9
505D 00              ;PDRIVE-Sector Byte FA
505E 00              ;PDRIVE-Sector Byte FB
505F 00              ;PDRIVE-Sector Byte FC
5060 00              ;PDRIVE-Sector Byte FD
5061 00              ;PDRIVE-Sector Byte FE
5062 00              ;PDRIVE-Sector Byte FF
                     ----------------------
                     Texte für Datum und Uhrzeit
5063 44 61 74 65 3F 20 20 28 4D 4D 2F 44 44 2F 59 59  Date?  (MM/DD/YY
5073 29 20 03                                         ) .
                     ----------------------
5076 54 69 6D 65 3F 20 20 28 48 48 3A 4D 4D 3A 53 53  Time?  (HH:MM:SS
5086 29 20 03                                         ) .
                     ----------------------
5089 4D 4D 2F 44 44 2F 59 59 20 20 48 48 3A 4D 4D 3A  MM/DD/YY  HH:MM:
5099 53 53 0D                                         SS.
                     ----------------------
                     Tabelle für die zulässige Eingabe des Datums
509C 01 0C           ;Monat (1-12)
509E 01 1F           ;Tag   (1-31)
50A0 00 64           ;Jahr  (0-99)
                     ----------------------
                     Tabelle für die zulässige Eingabe der Uhrzeit
50A2 00 18           ;Stunde  (0-23)
50A4 00 3C           ;Minute  (0-59)
50A6 00 3C           ;Sekunde (0-59)
                     ----------------------
                     Frei für Erweiterungen und ZAPS
50A8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50B8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50D8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5108 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5118 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5128 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5138 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5148 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5158 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5168 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5178 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5188 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5198 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
51A8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
51B8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
51C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
51D8 00 00 00
```

3.2 Unterprogramme und Ansprungadressen

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I,
GDOS für Genie I,II,IIs,III,IIIs

Die Unterprogramme und Ansprungadressen anderer Betriebssysteme finden Sie in Kapitel 9.

Nach Adressen sortiert:

0013	READB	liest ein Byte aus einer File (siehe 4AF3H)
001B	WRITEB	schreibt ein Byte in eine File (siehe 4A98H)
402D	DOSRDY	Sprung nach DOS READY (ohne Rückkehr)
4030	ERROR0	nach einem Fehler: Sprung nach DOS READY
4063	HEXDE	DE hexadezimal nach (HL) ausgeben
4068	HEXA	A hexadezimal nach (HL) ausgeben
4405	DOSCMD	DOS-Befehl ausführen (ohne Rückkehr)
4409	DOSERR	Fehlermeldung des DOS ausgeben
440D	DEBUG	DEBUG aufrufen
4410	INTINS	Benutzer-Interrupt-Routine einfügen
4413	INTDEL	Benutzer-Interrupt-Routine löschen
4416	MOTONX	Drive-Motoren weiterlaufen lassen
4419	DOSCAL	DOS-Befehl ausführen und zurück
441C	TFSPEC	Filespec (HL) nach FCB (DE) übertragen
4420	INIT	File öffnen, ggf. neue File anlegen
4424	OPEN	File öffnen, ggf. keine neue File anlegen
4428	CLOSE	File schließen
442C	KILL	File löschen
4430	LOAD	Programm laden
4433	RUN	Programm laden und starten (keine Rückkehr)
4436	READ	nächsten Sector/Record aus einer File lesen
4439	WRITE	nächsten Sector/Record in eine File schreiben
443C	VERIFY	Sector/Record in eine File schreiben + testen
443F	POS0	FCB auf Beginn einer File positionieren
4442	POSBC	FCB auf Logische Record# (BC) positionieren
4445	POSDEC	FCB um 1 Record zurückpositionieren
4448	POSEOF	FCB auf Ende der File (EOF) positionieren
444B	EXPAND	File erweitern, wenn File zu kurz ist
444E	POSRBA	FCB auf RBA-Format (HLC) positionieren
4451	PUTEOF	EOF im FPDE auf den neuesten Stand bringen
445B	DRVSEL	Drive auswählen und Motor starten
445E	TSTDSK	Drive auswählen, prüfen ob Diskette eingelegt
4461	USRINS	Benutzer-Routine (*name) einfügen
4464	USRDEL	Benutzer-Routine (*name) löschen
4467	TEXTTV	Text auf Bildschirm ausgeben
446A	TEXTLP	Text auf Drucker ausgeben
446D	TIME	Uhrzeit im Format HH:MM:SS in Buffer ablegen
4470	DATE	Datum im Format MM/TT/JJ in Buffer ablegen
4473	INSEXT	File-Typ (/EXT) in Filespec einfügen
45B5	UPCASE	Kleinbuchstaben in Großbuchstaben umwandeln
4630	READS	Sector von Diskette lesen
4634	TESTS	Sector auf Lesbarkeit testen
463C	WRITDS	Directory-Sector auf Diskette schreiben
4640	WRITES	normalen Sector auf Diskette schreiben

4745	RESTOR	RESTORE-Kommando an FDC senden
4747	FDCCMD	Kommando mit richtiger TSR an FDC senden
4750	WNBUSY	warten, bis der FDC nicht mehr busy ist
475E	FBREAK	FORCE-INTERRUPT-Kommando an FDC senden
4767	MOTON	Drive-Motoren starten
47E3	DELAY1	ca. 55 us warten und Status des FDC lesen
4810	FILPOS	Disk-Position eines File-Sectors berechnen
490A	DIRR	liest einen Sector des Directory
491F	DIRW	schreibt einen Sector des Directory
4936	GETFDE	holt einen FDE aus dem Directory
4968	NXTEOF	NEXT-Wert aus FCB holen + mit EOF vergleichen
4980	PUSHR	Push Register, IX-FCB+0, IY-4380H, A-00
4AB8	WRITXV	Sector oder DIR-Sector schreiben, ggf. Verify
4ABD	WRITEV	normalen Sector schreiben, ggf. Verify
4ACA	WRITDV	Directory-Sector schreiben, ggf. Verify
4BC9	GETSYS	SYS-File laden und starten
4C28	LOADP	Programm laden
4C74	DIRPOS	Disk-Position eines DIR-Sectors berechnen
4C92	MULTL	Multipliziere AHL - L*A
4C94	MULTHL	Multipliziere AHL - HL*A
4C9D	MULTC	Multipliziere HLC - HL*A
4CB2	DIV05	Dividiere HL-INT(HL/5), A-Rest
4CB4	DIVA	Dividiere HL-INT(HL/A), A-Rest
4CC5	CPBCHL	Vergleiche Text (BC) mit Text (HL)
4CD5	NEXTC1	Nächstes Zeichen holen und Flags setzen
4CD9	NEXTC2	Nächstes Zeichen holen und Flags setzen
4CED	DELAY2	ca. B * 3.75 ms warten

Alphabetisch sortiert:

CLOSE	4428	File schließen
CPBCHL	4CC5	Vergleiche Text (BC) mit Text (HL)
DATE	4470	Datum im Format MM/TT/JJ in Buffer ablegen
DEBUG	440D	DEBUG aufrufen
DELAY1	47E3	ca. 55 us warten und Status des FDC lesen
DELAY2	4CED	ca. B * 3.75 ms warten
DIRPOS	4C74	Disk-Position eines DIR-Sectors berechnen
DIRR	490A	liest einen Sector des Directory
DIRW	491F	schreibt einen Sector des Directory
DIV05	4CB2	Dividiere HL-INT(HL/5), A-Rest
DIVA	4CB4	Dividiere HL-INT(HL/A), A-Rest
DOSCAL	4419	DOS-Befehl ausführen und zurück
DOSCMD	4405	DOS-Befehl ausführen (ohne Rückkehr)
DOSERR	4409	Fehlermeldung des DOS ausgeben
DOSRDY	402D	Sprung nach DOS READY (ohne Rückkehr)
DRVSEL	445B	Drive auswählen und Motor starten
ERROR0	4030	nach einem Fehler: Sprung nach DOS READY
EXPAND	444B	File erweitern, wenn File zu kurz ist
FBREAK	475E	FORCE-INTERRUPT-Kommando an FDC senden
FDCCMD	4747	Kommando mit richtiger TSR an FDC senden
FILPOS	4810	Disk-Position eines File-Sectors berechnen
GETFDE	4936	holt einen FDE aus dem Directory
GETSYS	4BC9	SYS-File laden und starten
HEXA	4068	A hexadezimal nach (HL) ausgeben
HEXDE	4063	DE hexadezimal nach (HL) ausgeben
INIT	4420	File öffnen, ggf. neue File anlegen

INSEXT	4473	File-Typ (/EXT) in Filespec einfügen
INTDEL	4413	Benutzer-Interrupt-Routine löschen
INTINS	4410	Benutzer-Interrupt-Routine einfügen
KILL	442C	File löschen
LOAD	4430	Programm laden
LOADP	4C28	Programm laden
MOTON	4767	Drive-Motoren starten
MOTONX	4416	Drive-Motoren weiterlaufen lassen
MULTC	4C9D	Multipliziere HLC = HL*A
MULTHL	4C94	Multipliziere AHL = HL*A
MULTL	4C92	Multipliziere AHL = L*A
NEXTC1	4CD5	Nächstes Zeichen holen und Flags setzen
NEXTC2	4CD9	Nächstes Zeichen holen und Flags setzen
NXTEOF	4968	NEXT-Wert aus FCB holen + mit EOF vergleichen
OPEN	4424	File öffnen, ggf. keine neue File anlegen
POS0	443F	FCB auf Beginn einer File positionieren
POSBC	4442	FCB auf Logische Record# (BC) positionieren
POSDEC	4445	FCB um 1 Record zurückpositionieren
POSEOF	4448	FCB auf Ende der File (EOF) positionieren
POSRBA	444E	FCB auf RBA-Format (HLC) positionieren
PUSHR	4980	Push Register, IX-FCB+0, IY-4380H, A-00
PUTEOF	4451	EOF im FPDE auf den neuesten Stand bringen
READ	4436	nächsten Sector/Record aus einer File lesen
READB	0013	liest ein Byte aus einer File (siehe 4AF3H)
READS	4630	Sector von Diskette lesen
RESTOR	4745	RESTORE-Kommando an FDC senden
RUN	4433	Programm laden und starten (keine Rückkehr)
TESTS	4634	Sector auf Lesbarkeit testen
TEXTLP	446A	Text auf Drucker ausgeben
TEXTTV	4467	Text auf Bildschirm ausgeben
TFSPEC	441C	Filespec (HL) nach FCB (DE) übertragen
TIME	446D	Uhrzeit im Format HH:MM:SS in Buffer ablegen
TSTDSK	445E	Drive auswählen, prüfen ob Diskette eingelegt
UPCASE	45B5	Kleinbuchstaben in Großbuchstaben umwandeln
USRDEL	4464	Benutzer-Routine (*name) löschen
USRINS	4461	Benutzer-Routine (*name) einfügen
VERIFY	443C	Sector/Record in eine File schreiben + testen
WNBUSY	4750	warten, bis der FDC nicht mehr busy ist
WRITDS	463C	Directory-Sector auf Diskette schreiben
WRITDV	4ACA	Directory-Sector schreiben, ggf. Verify
WRITE	4439	nächsten Sector/Record in eine File schreiben
WRITEB	001B	schreibt ein Byte in eine File (siehe 4A98H)
WRITES	4640	normalen Sector auf Diskette schreiben
WRITEV	4ABD	normalen Sector schreiben, ggf. Verify
WRITXV	4AB8	Sector oder DIR-Sector schreiben, ggf. Verify

Kapitel 4: SYSTEM- und PDRIVE-Parameter

Der 3. Sector auf jeder Diskette enthält die SYSTEM- und PDRIVE-Parameter des Betriebssystems und heißt daher auch PDRIVE-Sector. Er wird in NEWDOS/80 und GDOS dadurch gekennzeichnet, daß er im Sector-relativen Byte EFH den Wert A5H enthält.

Bei GDOS-Disketten ist außerdem in den relativen Bytes E0H bis EEH die Meldung 'Trommeschläger' gespeichert. Unter GDOS für Genie III und IIIs befinden sich zusätzlich in den Bytes C0H bis CFH die Werte für die Initialisierung des CRT-Controllers.

4.1 SYSTEM-Parameter

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I
GDOS für Genie I,II,IIs,III,IIIs

Die folgende Tabelle zeigt, wie die SYSTEM-Parameter AA bis BN im PDRIVE-Sector und im RAM gespeichert werden.

Für SYSTEM-Parameter, bei denen nur "N" oder "Y" möglich ist, wird nur 1 Bit zur Speicherung benutzt: "N" <=> "0" und "Y" <=> "1".

Einige SYSTEM-Parameter führen während der Initialisierung von SYS0 zusätzlich noch an bestimmten Stellen im RAM Programmänderungen durch.

	PDRIVE-Sector		RAM		SYS0
	Byte	Bit	Adresse	Bit	Programmänderung
AA:	F0	7	436C	7	
AB:	F0	6	436C	6	
			4369	2	
AC:	F8	7	505B	7	"N" => 457D = 3E
AD:	F8	6	505B	6	"Y" => 45EE = C0
AE:	F8	5	505B	5	"Y" => 45D8 = 20
AF:	F8	4	505B	4	"Y" => 45CC = 20
AG:	F0	5	436C	5	
			4369	4	
AH:	F8	3	505B	3	(nur NEWDOS/80 Version 1)
AI:	F0	4	436C	4	
AJ:	F8	1	505B	1	"Y" => 4016 = 1645 401E = 0545

	PDRIVE-Sector Byte	Bit	RAM Adresse	Bit	SYS0 Programmänderung
AK:	F0	2	436C	2	(nur NEWDOS/80 Version 1)
AL:	A0		439F		
			477A		
AM:	A6		465A		
AN:	A2		43A0		
AO:	A3		43A1		
AP:	D0		4049		
	D1		404A		
AQ:	F8	2	505B	2	"Y" => 4590 = 00
AR:	F0	1	436C	1	
AS:	F0	0	436C	0	
AT:	F1	7	436D	7	
AU:	F8	0	505B	0	"Y" => 4538 = 20
AV:	A7		452F		
			4578		
AW:	A1		4ABA		
AX:	A8		4370		
AY:	F9	7	505C	7	
AZ:	F9	6	505C	6	
BA:	F9	5	505C	5	"Y" => 401D = C0 43B2 = 1D40 43B4 = 07 43B5 = 0000 4ADC = B243
BB:	F9	4	504C	4	(nur TRS-80 Model III)
BC:	F1	6	436D	6	
BD:	F9	3	505C	3	
BE:	F1	5	436D	5	
BF:	F9	2	505C	2	"Y" => 4505 = 38 4593 = 4F
BG:	F9	1	505C	1	"N" => 45B4 = 00
BH:	F9	0	505C	0	"Y" => 4502 = C8
BI:	A5		4501		
BJ:	A9		43A2		
			4CEF		
			47E4		(6 * BJ / GDOS: 8 * BJ)
			47F4,F5		(2400H * BJ)
BK:	F1	4	436D	4	
BL:	--	-	----	-	
BM:	F1	3	436D	3	(entfällt unter GDOS)
BN:	FA	7	505D	7	"Y" => 463D = AB

4.2 PDRIVE-Parameter

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I+III
GDOS für Genie I,II,IIs,III,IIIs

In den ersten 160D Bytes des PDRIVE-Sectors sind die PDRIVE-Parameter der Drives 0 bis 9 gespeichert, wobei jeweils 16D Bytes pro Drive benutzt werden. Diese 16D Bytes haben folgende Bedeutung (in Klammern sind jeweils die deutschen Bezeichnungen von GDOS angegeben):

Byte 0 — DDSL "Default Directory Starting Lump" (SBIV)

Byte 1 — Anzahl Lumps (Blocks) pro Diskette

Byte 2

Bit 7: gesetzt wenn TI-H
Bit 6: gesetzt wenn TI-K
Bit 5: gesetzt wenn TI-M

	TI-A	TI-B	TI-C	TI-D	TI-E	ERROR *)
Bit 4:	0	0	1	0	1	0
Bit 3:	0	1	0	1	0	0
Bit 2:	1	1	0	0	1	0

*) fehlerhafte PDRIVE-Parameter für dieses Laufwerk

Bits 1,0: TSR "Track Stepping Rate" (SWZ)

Byte 3 — TC "Track Count" (SP)

Byte 4 — SPT "Sectors pro Track" (SEK)

Byte 5 — GPL "Grans pro Lump" (EIB)

Byte 6 — Auswahl 5.25 Zoll / 8 Zoll und SD / DD
(wird nach 37EE geschrieben, siehe Kapitel 1.2.1)

00: TI=A/C
80: TI=B/E und TD=A/C/E/G (5.25 Zoll)
C0: TI=B/E und TD=B/D/F/H (8 Zoll)

nur für TRS-80 Model III:
FC: TI=D und TD=A/C (5.25 Zoll und Single Density)
FD: TI=D und TD=B/D (8 Zoll und Single Density)
FE: TI=D und TD=E/G (5.25 Zoll und Double Density)
FF: TI=D und TD=F/H (8 Zoll und Double Density)

Byte 7

Bit 7: 8 Zoll (1) oder 5.25 Zoll (0)
Bit 6: Diskette doppelseitig (1) oder einseitig (0)
Bit 5: ./.
Bit 4: gesetzt wenn TI=I oder TI=M (Sect# ab 1)
Bit 3: ./.
Bit 2: gesetzt wenn TI=L
Bit 1: gesetzt wenn TI=J oder TI=K (Track# ab 1)
Bit 0: Double (1) oder Single (0) Density

Byte	Bedeutung
8	DDSL "Default Directory Starting Lump" (SBIV)
9	DDGA "Default Directory Grans Allocation" (AEIV)
A	./.
B	./.
C	TSR "Track Stepping Rate" (SWZ)
D	TI "Type of Interface" Bit 7: gesetzt wenn TI-H Bit 6: ./. Bit 5: ./. Bit 4: gesetzt wenn TI-E Bit 3: gesetzt wenn TI-D Bit 2: gesetzt wenn TI-C Bit 1: gesetzt wenn TI-B Bit 0: gesetzt wenn TI-A
E	TI "Type of Interface" Bit 7: ./. Bit 6: ./. Bit 5: ./. Bit 4: gesetzt wenn TI-M Bit 3: gesetzt wenn TI-L Bit 2: gesetzt wenn TI-K Bit 1: gesetzt wenn TI-J Bit 0: gesetzt wenn TI-I
F	TD "Type of Drive" 00: TD-A 01: TD-B 02: TD-C 03: TD-D 04: TD-E 05: TD-F 06: TD-G 07: TD-H

Die ersten 10D Bytes der PDRIVE-Parameter für die Laufwerke 0 bis 3 werden folgendermaßen im RAM gespeichert:

	TRS-80 Model I Genie I,II,III,IIIs	TRS-80 Model III
Drive 0:	4371 - 437A	4291 - 429A
Drive 1:	437B - 4384	429B - 42A4
Drive 2:	4385 - 438E	42A5 - 42AE
Drive 3:	438F - 4398	42AF - 42B8

Zusätzlich werden die ersten 8 Bytes des jeweils zuletzt ausgewählten Laufwerks im RAM bei 430A - 4311 gespeichert (TRS-80 Model III: 4280 - 4287).

Kapitel 5: Die Befehlstabelle in SYS1/SYS

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I
NEWDOS/80 Version 2 für TRS-80 Model III
GDOS für Genie I,II,IIs
GDOS für Genie III
GDOS für Genie IIIs

Die Befehlstabelle in SYS1/SYS nimmt eine weitere zentrale Stelle in NEWDOS/80 und GDOS ein. Sie enthält sämtliche Befehle, und von welcher SYS-File jeder Befehl ausgeführt wird. Zusätzlich gibt es für jeden Befehl eine 8-Bit-Maske, die spezielle Eigenschaften einiger Befehle anzeigt.

Die hier ausgedruckten Tabellen haben folgenden Aufbau:

- Adresse in der Tabelle in SYS1/SYS, wo der Befehl steht
- ein (*), wenn der Befehl unter MINI-DOS verboten ist
- Name des Befehls
- die SYS-File, die den Befehl ausführt
- die Einsprung-Bedingungen für diese SYS-File in Register A und C
- eine 8-Bit-Maske, die folgende Bedeutung hat:
  Bit 7: wenn gesetzt, muß für diesen Befehl eine File geöffnet werden, siehe Bit 6
  Bit 6: wenn gesetzt, darf dafür ggf. auch eine neue File angelegt werden
  Bit 5: wenn gesetzt, benötigt dieser Befehl 2 Filespecs
  Bit 4: wenn gesetzt, dürfen dem Befehl (ausgenommen Filespecs gemäß Bit 7+5) keine Parameter folgen
  Bit 3: wenn gesetzt, wird der Zeiger auf weitere Parameter in den Stack geschrieben
  Bit 2: ./.
  Bit 1: wenn gesetzt: wenn die File gemäß Bit 7 keinen Typ hat, bekommt sie den Typ /JCL (GDOS: /JOB)
  Bit 0: wenn gesetzt: wenn die File gemäß Bit 7 keinen Typ hat, bekommt sie den Typ /CMD

5.1 Befehlstabelle NEWDOS/80 Version 2 für TRS-80 Model I

4F58	*APPEND	SYS6/SYS	A-68	C-00	Maske: 0000 0000
4F61	ATTRIB	SYS7/SYS	A-E9	C-05	Maske: 1000 1000
4F6A	AUTO	SYS7/SYS	A-E9	C-04	Maske: 0000 0000
4F71	BASIC2	SYS9/SYS	A-EB	C-06	Maske: 0000 0000
4F7A	BLINK	SYS3/SYS	A-E5	C-01	Maske: 0000 0000
4F82	BOOT	SYS9/SYS	A-EB	C-0A	Maske: 0001 0000
4F89	BREAK	SYS3/SYS	A-E5	C-05	Maske: 0000 0000
4F91	*CHAIN	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
4F99	*CHNON	SYS9/SYS	A-EB	C-05	Maske: 0000 0000
4FA1	CLEAR	SYS14/SYS	A-F0	C-04	Maske: 0000 0000
4FA9	CLOCK	SYS3/SYS	A-E5	C-02	Maske: 0000 0000
4FB1	CLS	SYS1/SYS	A-E3	C-09	Maske: 0001 0000
4FB7	*COPY	SYS6/SYS	A-48	C-00	Maske: 0000 0000
4FBE	CREATE	SYS14/SYS	A-F0	C-02	Maske: 0100 0000
4FC7	DATE	SYS7/SYS	A-E9	C-0B	Maske: 0000 0000
4FCE	DEBUG	SYS3/SYS	A-E5	C-03	Maske: 0000 0000
4FD6	DIR	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FDC	*DO	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
4FE1	DUMP	SYS7/SYS	A-E9	C-07	Maske: 1100 1000
4FE8	ERROR	SYS14/SYS	A-F0	C-07	Maske: 0000 0000
4FF0	*FORMAT	SYS6/SYS	A-28	C-00	Maske: 0000 0000
4FF9	FREE	SYS8/SYS	A-4A	C-00	Maske: 0000 0000
5000	HIMEM	SYS7/SYS	A-E9	C-02	Maske: 0000 0000
5008	JKL	SYS3/SYS	A-A5	C-00	Maske: 0001 0000
500E	KILL	SYS3/SYS	A-45	C-00	Maske: 1001 0000
5015	LC	SYS3/SYS	A-E5	C-09	Maske: 0000 0000
501A	LCDVR	SYS3/SYS	A-E5	C-08	Maske: 0000 0000
5022	LIB	SYS1/SYS	A-E3	C-02	Maske: 0000 0000
5028	LIST	SYS14/SYS	A-F0	C-05	Maske: 1000 1000
502F	LOAD	SYS2/SYS	A-A4	C-00	Maske: 0101 0000
5036	MDBORT	SYS1/SYS	A-E3	C-05	Maske: 0000 0000
503F	MDCOPY	SYS9/SYS	A-EB	C-02	Maske: 1011 0000
5048	MDRET	SYS1/SYS	A-E3	C-06	Maske: 0000 0000
5050	PAUSE	SYS9/SYS	A-EB	C-08	Maske: 0000 0000
5058	PDRIVE	SYS7/SYS	A-E9	C-03	Maske: 0000 0000
5061	PRINT	SYS14/SYS	A-F0	C-06	Maske: 1000 1000
5069	PROT	SYS7/SYS	A-E9	C-06	Maske: 0000 0000
5070	PURGE	SYS7/SYS	A-E9	C-09	Maske: 0000 0000
5078	R	SYS1/SYS	A-23	C-00	Maske: 0000 0000
507C	RENAME	SYS2/SYS	A-E4	C-01	Maske: 1011 0000
5085	ROUTE	SYS14/SYS	A-F0	C-01	Maske: 0000 0000
508D	STMT	SYS9/SYS	A-EB	C-09	Maske: 0000 0000
5094	SYSTEM	SYS7/SYS	A-E9	C-01	Maske: 0000 0000
509D	TIME	SYS7/SYS	A-E9	C-0A	Maske: 0000 0000
50A4	VERIFY	SYS3/SYS	A-E5	C-04	Maske: 0000 0000
50AD	WRDIRP	SYS17/SYS	A-53	C-00	Maske: 0000 0000

5.2 Befehlstabelle NEWDOS/80 Version 2 für TRS-80 Model III

4F6F	*APPEND	SYS6/SYS	A-68	C-00	Maske:	0000 0000
4F78	ATTRIB	SYS7/SYS	A-E9	C-05	Maske:	1000 1000
4F81	AUTO	SYS7/SYS	A-E9	C-04	Maske:	0000 0000
4F88	BLINK	SYS3/SYS	A-E5	C-01	Maske:	0000 0000
4F90	BOOT	SYS9/SYS	A-EB	C-0A	Maske:	0001 0000
4F97	BREAK	SYS3/SYS	A-E5	C-05	Maske:	0000 0000
4F9F	*CHAIN	SYS9/SYS	A-EB	C-03	Maske:	1000 1010
4FA7	*CHNON	SYS9/SYS	A-EB	C-05	Maske:	0000 0000
4FAF	CLEAR	SYS14/SYS	A-F0	C-04	Maske:	0000 0000
4FB7	CLOCK	SYS3/SYS	A-E5	C-02	Maske:	0000 0000
4FBF	CLS	SYS1/SYS	A-E3	C-09	Maske:	0001 0000
4FC5	*COPY	SYS6/SYS	A-48	C-00	Maske:	0000 0000
4FCC	CREATE	SYS14/SYS	A-F0	C-02	Maske:	0100 0000
4FD5	DATE	SYS7/SYS	A-E9	C-0B	Maske:	0000 0000
4FDC	DEBUG	SYS3/SYS	A-E5	C-03	Maske:	0000 0000
4FE4	DIR	SYS8/SYS	A-2A	C-00	Maske:	0000 0000
4FEA	*DO	SYS9/SYS	A-EB	C-03	Maske:	1000 1010
4FEF	DUMP	SYS7/SYS	A-E9	C-07	Maske:	1100 1000
4FF6	ERROR	SYS14/SYS	A-F0	C-07	Maske:	0000 0000
4FFE	*FORMAT	SYS6/SYS	A-28	C-00	Maske:	0000 0000
5007	FORMS	SYS15/SYS	A-F1	C-01	Maske:	0000 0000
500F	FREE	SYS8/SYS	A-4A	C-00	Maske:	0000 0000
5016	HIMEM	SYS7/SYS	A-E9	C-02	Maske:	0000 0000
501E	JKL	SYS3/SYS	A-A5	C-00	Maske:	0001 0000
5024	KILL	SYS3/SYS	A-45	C-00	Maske:	1001 0000
502B	LC	SYS3/SYS	A-E5	C-09	Maske:	0000 0000
5030	LIB	SYS1/SYS	A-E3	C-02	Maske:	0000 0000
5036	LIST	SYS14/SYS	A-F0	C-05	Maske:	1000 1000
503D	LOAD	SYS2/SYS	A-A4	C-00	Maske:	0101 0000
5044	MDBORT	SYS1/SYS	A-E3	C-05	Maske:	0000 0000
504D	MDCOPY	SYS9/SYS	A-EB	C-02	Maske:	1011 0000
5056	MDRET	SYS1/SYS	A-E3	C-06	Maske:	0000 0000
505E	PAUSE	SYS9/SYS	A-EB	C-08	Maske:	0000 0000
5066	PDRIVE	SYS7/SYS	A-E9	C-03	Maske:	0000 0000
506F	PRINT	SYS14/SYS	A-F0	C-06	Maske:	1000 1000
5077	PROT	SYS7/SYS	A-E9	C-06	Maske:	0000 0000
507E	PURGE	SYS7/SYS	A-E9	C-09	Maske:	0000 0000
5086	R	SYS1/SYS	A-23	C-00	Maske:	0000 0000
508A	RENAME	SYS2/SYS	A-E4	C-01	Maske:	1011 0000
5093	ROUTE	SYS14/SYS	A-F0	C-01	Maske:	0000 0000
509B	SETCOM	SYS15/SYS	A-F1	C-02	Maske:	0000 0000
50A4	STMT	SYS9/SYS	A-EB	C-09	Maske:	0000 0000
50AB	SYSTEM	SYS7/SYS	A-E9	C-01	Maske:	0000 0000
50B4	TIME	SYS7/SYS	A-E9	C-0A	Maske:	0000 0000
50BB	VERIFY	SYS3/SYS	A-E5	C-04	Maske:	0000 0000
50C4	WRDIRP	SYS17/SYS	A-53	C-00	Maske:	0000 0000

5.3 Befehlstabelle GDOS für Genie I,II,IIs

4F58	0	SYS14/SYS	A-F0	C-04	Maske: 0000 0000
4F5C	§	SYS14/SYS	A-F0	C-01	Maske: 0000 0000
4F60	AIK	SYS17/SYS	A-53	C-00	Maske: 0000 0000
4F66	*APPEND	SYS6/SYS	A-68	C-00	Maske: 0000 0000
4F6F	ATTRIB	SYS7/SYS	A-E9	C-05	Maske: 1000 1000
4F78	AUTO	SYS7/SYS	A-E9	C-04	Maske: 0000 0000
4F7F	B2	SYS9/SYS	A-EB	C-06	Maske: 0000 0000
4F84	BL	SYS3/SYS	A-E5	C-01	Maske: 0000 0000
4F89	BOOT	SYS9/SYS	A-EB	C-0A	Maske: 0001 0000
4F90	BREAK	SYS3/SYS	A-E5	C-05	Maske: 0000 0000
4F98	CLS	SYS1/SYS	A-E3	C-09	Maske: 0001 0000
4F9E	*CONT	SYS9/SYS	A-EB	C-05	Maske: 0000 0000
4FA5	*COPY	SYS6/SYS	A-48	C-00	Maske: 0000 0000
4FAC	CREATE	SYS14/SYS	A-F0	C-02	Maske: 0100 0000
4FB5	DATUM	SYS7/SYS	A-E9	C-0B	Maske: 0000 0000
4FBD	DIR	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FC3	DISK	SYS29/SYS	A-FF	C-03	Maske: 0000 0000
4FCA	*DO	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
4FCF	DR	SYS28/SYS	A-FE	C-02	Maske: 0000 0000
4FD4	DUMP	SYS7/SYS	A-E9	C-07	Maske: 1100 1000
4FDB	E	SYS14/SYS	A-F0	C-07	Maske: 0000 0000
4FDF	FORM	SYS28/SYS	A-FE	C-08	Maske: 0000 0000
4FE6	FREE	SYS8/SYS	A-4A	C-00	Maske: 0000 0000
4FED	HIMEM	SYS7/SYS	A-E9	C-02	Maske: 0000 0000
4FF5	I	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FF9	INFO	SYS29/SYS	A-FF	C-01	Maske: 0000 0000
5000	JKL	SYS3/SYS	A-A5	C-00	Maske: 0001 0000
5006	KILL	SYS3/SYS	A-45	C-00	Maske: 1001 0000
500D	LC	SYS3/SYS	A-E5	C-08	Maske: 0000 0000
5012	LF	SYS28/SYS	A-FE	C-01	Maske: 0000 0000
5017	LIB	SYS1/SYS	A-E3	C-02	Maske: 0000 0000
501D	LIST	SYS14/SYS	A-F0	C-05	Maske: 1000 1000
5024	LOAD	SYS2/SYS	A-A4	C-00	Maske: 0101 0000
502B	LWT	SYS23/SYS	A-F9	C-01	Maske: 0000 0000
5031	N	SYS2/SYS	A-E4	C-01	Maske: 1011 0000
5035	*NDF	SYS6/SYS	A-28	C-00	Maske: 0000 0000
503B	PAUSE	SYS9/SYS	A-EB	C-08	Maske: 0000 0000
5043	PD	SYS7/SYS	A-E9	C-03	Maske: 0000 0000
5048	PORT	SYS29/SYS	A-FF	C-02	Maske: 0000 0000
504F	PRINT	SYS14/SYS	A-F0	C-06	Maske: 1000 1000
5057	PROT	SYS7/SYS	A-E9	C-06	Maske: 0000 0000
505E	PURGE	SYS7/SYS	A-E9	C-09	Maske: 0000 0000
5066	R	SYS1/SYS	A-23	C-00	Maske: 0000 0000
506A	S	SYS7/SYS	A-E9	C-01	Maske: 0000 0000
506E	STMT	SYS9/SYS	A-EB	C-09	Maske: 0000 0000
5075	UHR	SYS3/SYS	A-E5	C-02	Maske: 0000 0000
507B	V+	SYS3/SYS	A-E5	C-04	Maske: 0000 0000
5080	V24	SYS29/SYS	A-FF	C-07	Maske: 0000 0000
5086	Z	SYS29/SYS	A-FF	C-06	Maske: 0000 0000
508A	ZEIT	SYS7/SYS	A-E9	C-0A	Maske: 0000 0000
5091	&	SYS3/SYS	A-E5	C-03	Maske: 0000 0000
5095	!	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
5099	;	SYS1/SYS	A-E3	C-06	Maske: 0000 0000

509D	/	SYS1/SYS	A-E3	C-05	Maske: 0000 0000
50A1	?	SYS1/SYS	A-E3	C-02	Maske: 0000 0000
50A5	*>	SYS6/SYS	A-48	C-00	Maske: 0000 0000
50A9	M>	SYS9/SYS	A-EB	C-02	Maske: 1011 0000
50AE	*DDE	SYS15/SYS	A-F1	C-00	Maske: 0000 0000

5.4 Befehlstabelle GDOS für Genie III

4F58	0	SYS14/SYS	A-F0	C-04	Maske: 0000 0000
4F5C	§	SYS14/SYS	A-F0	C-01	Maske: 0000 0000
4F60	AIK	SYS17/SYS	A-53	C-00	Maske: 0000 0000
4F66	*APPEND	SYS6/SYS	A-68	C-00	Maske: 0000 0000
4F6F	ATTRIB	SYS7/SYS	A-E9	C-05	Maske: 1000 1000
4F78	AUTO	SYS7/SYS	A-E9	C-04	Maske: 0000 0000
4F7F	B2	SYS9/SYS	A-EB	C-06	Maske: 0000 0000
4F84	BL	SYS3/SYS	A-E5	C-01	Maske: 0000 0000
4F89	BOOT	SYS9/SYS	A-EB	C-0A	Maske: 0001 0000
4F90	BREAK	SYS3/SYS	A-E5	C-05	Maske: 0000 0000
4F98	CLS	SYS1/SYS	A-E3	C-09	Maske: 0001 0000
4F9E	*CONT	SYS9/SYS	A-EB	C-05	Maske: 0000 0000
4FA5	*COPY	SYS6/SYS	A-48	C-00	Maske: 0000 0000
4FAC	CREATE	SYS14/SYS	A-F0	C-02	Maske: 0100 0000
4FB5	DATUM	SYS7/SYS	A-E9	C-0B	Maske: 0000 0000
4FBD	DIR	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FC3	DISK	SYS29/SYS	A-FF	C-03	Maske: 0000 0000
4FCA	*DO	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
4FCF	DR	SYS28/SYS	A-FE	C-02	Maske: 0000 0000
4FD4	DUMP	SYS7/SYS	A-E9	C-07	Maske: 1100 1000
4FDB	E	SYS14/SYS	A-F0	C-07	Maske: 0000 0000
4FDF	FORM	SYS28/SYS	A-FE	C-08	Maske: 0000 0000
4FE6	FREE	SYS8/SYS	A-4A	C-00	Maske: 0000 0000
4FED	HIMEM	SYS7/SYS	A-E9	C-02	Maske: 0000 0000
4FF5	I	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FF9	INFO	SYS29/SYS	A-FF	C-01	Maske: 0000 0000
5000	JKL	SYS3/SYS	A-A5	C-00	Maske: 0001 0000
5006	KILL	SYS3/SYS	A-45	C-00	Maske: 1001 0000
500D	LC	SYS3/SYS	A-E5	C-08	Maske: 0000 0000
5012	LF	SYS28/SYS	A-FE	C-01	Maske: 0000 0000
5017	LIB	SYS1/SYS	A-E3	C-02	Maske: 0000 0000
501D	LIST	SYS14/SYS	A-F0	C-05	Maske: 1000 1000
5024	LOAD	SYS2/SYS	A-A4	C-00	Maske: 0101 0000
502B	LWT	SYS23/SYS	A-F9	C-01	Maske: 0000 0000
5031	N	SYS2/SYS	A-E4	C-01	Maske: 1011 0000
5035	*NDF	SYS6/SYS	A-28	C-00	Maske: 0000 0000
503B	PAUSE	SYS9/SYS	A-EB	C-08	Maske: 0000 0000
5043	PD	SYS7/SYS	A-E9	C-03	Maske: 0000 0000
5048	PORT	SYS29/SYS	A-FF	C-02	Maske: 0000 0000
504F	PRINT	SYS14/SYS	A-F0	C-06	Maske: 1000 1000
5057	PROT	SYS7/SYS	A-E9	C-06	Maske: 0000 0000
505E	PURGE	SYS7/SYS	A-E9	C-09	Maske: 0000 0000
5066	R	SYS1/SYS	A-23	C-00	Maske: 0000 0000
506A	S	SYS7/SYS	A-E9	C-01	Maske: 0000 0000
506E	STMT	SYS9/SYS	A-EB	C-09	Maske: 0000 0000

5075	UHR	SYS3/SYS	A-E5	C-02	Maske: 0000 0000
507B	V+	SYS3/SYS	A-E5	C-04	Maske: 0000 0000
5080	V24	SYS24/SYS	A-FA	C-00	Maske: 0000 0000
5086	Z	SYS29/SYS	A-FF	C-06	Maske: 0000 0000
508A	ZEIT	SYS7/SYS	A-E9	C-0A	Maske: 0000 0000
5091	&	SYS3/SYS	A-E5	C-03	Maske: 0000 0000
5095	!	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
5099	;	SYS1/SYS	A-E3	C-06	Maske: 0000 0000
509D	/	SYS1/SYS	A-E3	C-05	Maske: 0000 0000
50A1	?	SYS1/SYS	A-E3	C-02	Maske: 0000 0000
50A5	*>	SYS6/SYS	A-48	C-00	Maske: 0000 0000
50A9	M>	SYS9/SYS	A-EB	C-02	Maske: 1011 0000
50AE	##	SYS23/SYS	A-F9	C-02	Maske: 0000 0000
50B3	*F#	SYS25/SYS	A-FB	C-00	Maske: 0000 0000
50B8	80	SYS23/SYS	A-F9	C-03	Maske: 0000 0000
50BD	64	SYS23/SYS	A-F9	C-04	Maske: 0000 0000
50C2	DDE	SYS15/SYS	A-F1	C-01	Maske: 0000 0000

5.5 Befehlstabelle GDOS für Genie IIIs

4F58	AIK	SYS17/SYS	A-53	C-00	Maske: 0000 0000
4F5E	*APPEND	SYS6/SYS	A-68	C-00	Maske: 0000 0000
4F67	ATTRIB	SYS7/SYS	A-E9	C-05	Maske: 1000 1000
4F70	AUTO	SYS7/SYS	A-E9	C-04	Maske: 0000 0000
4F77	B2	SYS9/SYS	A-EB	C-06	Maske: 0000 0000
4F7C	BL	SYS3/SYS	A-E5	C-01	Maske: 0000 0000
4F81	BOOT	SYS9/SYS	A-EB	C-0A	Maske: 0001 0000
4F88	BREAK	SYS3/SYS	A-E5	C-05	Maske: 0000 0000
4F90	CLS	SYS1/SYS	A-E3	C-09	Maske: 0001 0000
4F96	*CONT	SYS9/SYS	A-EB	C-05	Maske: 0000 0000
4F9D	*COPY	SYS6/SYS	A-48	C-00	Maske: 0000 0000
4FA4	CREATE	SYS14/SYS	A-F0	C-02	Maske: 0100 0000
4FAD	DATUM	SYS7/SYS	A-E9	C-0B	Maske: 0000 0000
4FB5	DDE	SYS15/SYS	A-F1	C-01	Maske: 0000 0000
4FBB	DIR	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FC1	DISK	SYS29/SYS	A-FF	C-03	Maske: 0000 0000
4FC8	*DO	SYS9/SYS	A-EB	C-03	Maske: 1000 1010
4FCD	DR	SYS28/SYS	A-FE	C-02	Maske: 0000 0000
4FD2	DUMP	SYS7/SYS	A-E9	C-07	Maske: 1100 1000
4FD9	E	SYS14/SYS	A-F0	C-07	Maske: 0000 0000
4FDD	FORM	SYS28/SYS	A-FE	C-08	Maske: 0000 0000
4FE4	FREE	SYS8/SYS	A-4A	C-00	Maske: 0000 0000
4FEB	*F#	SYS25/SYS	A-FB	C-00	Maske: 0000 0000
4FF0	HIMEM	SYS7/SYS	A-E9	C-02	Maske: 0000 0000
4FF8	I	SYS8/SYS	A-2A	C-00	Maske: 0000 0000
4FFC	INFO	SYS29/SYS	A-FF	C-01	Maske: 0000 0000
5003	JKL	SYS3/SYS	A-A5	C-00	Maske: 0001 0000
5009	KILL	SYS3/SYS	A-45	C-00	Maske: 1001 0000
5010	LC	SYS3/SYS	A-E5	C-08	Maske: 0000 0000
5015	LF	SYS28/SYS	A-FE	C-01	Maske: 0000 0000
501A	LIB	SYS1/SYS	A-E3	C-02	Maske: 0000 0000
5020	LIST	SYS14/SYS	A-F0	C-05	Maske: 1000 1000
5027	LOAD	SYS2/SYS	A-A4	C-00	Maske: 0101 0000

502E	M>	SYS9/SYS	A-EB	C-02	Maske:	1011 0000
5033	N	SYS2/SYS	A-E4	C-01	Maske:	1011 0000
5037	*NDF	SYS6/SYS	A-28	C-00	Maske:	0000 0000
503D	PAUSE	SYS9/SYS	A-EB	C-08	Maske:	0000 0000
5045	PD	SYS7/SYS	A-E9	C-03	Maske:	0000 0000
504A	PORT	SYS29/SYS	A-FF	C-02	Maske:	0000 0000
5051	PRINT	SYS14/SYS	A-F0	C-06	Maske:	1000 1000
5059	PROT	SYS7/SYS	A-E9	C-06	Maske:	0000 0000
5060	PURGE	SYS7/SYS	A-E9	C-09	Maske:	0000 0000
5068	R	SYS1/SYS	A-23	C-00	Maske:	0000 0000
506C	S	SYS7/SYS	A-E9	C-01	Maske:	0000 0000
5070	SIO	SYS24/SYS	A-FA	C-00	Maske:	0000 0000
5076	STMT	SYS9/SYS	A-EB	C-09	Maske:	0000 0000
507D	UHR	SYS3/SYS	A-E5	C-02	Maske:	0000 0000
5083	V+	SYS3/SYS	A-E5	C-04	Maske:	0000 0000
5088	Z	SYS22/SYS	A-F8	C-01	Maske:	0000 0000
508C	ZEIT	SYS7/SYS	A-E9	C-0A	Maske:	0000 0000
5093	ZL	SYS22/SYS	A-F8	C-02	Maske:	1000 1000
5098	0	SYS14/SYS	A-F0	C-04	Maske:	0000 0000
509C	64	SYS23/SYS	A-F9	C-04	Maske:	0000 0000
50A1	80	SYS23/SYS	A-F9	C-03	Maske:	0000 0000
50A6	↓	SYS9/SYS	A-EB	C-03	Maske:	1000 1010
50AA	##	SYS23/SYS	A-F9	C-02	Maske:	0000 0000
50AF	&	SYS3/SYS	A-E5	C-03	Maske:	0000 0000
50B3	§	SYS14/SYS	A-F0	C-01	Maske:	0000 0000
50B7	;	SYS1/SYS	A-E3	C-06	Maske:	0000 0000
50BB	/	SYS1/SYS	A-E3	C-05	Maske:	0000 0000
50BF	*>	SYS6/SYS	A-48	C-00	Maske:	0000 0000
50C3	?	SYS1/SYS	A-E3	C-02	Maske:	0000 0000

Kapitel 6: Die Parametertabelle in SYS6/SYS

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I+III
GDOS für Genie I,II,IIs,III,IIIs

Die Parametertabelle in SYS6/SYS enthält alle Parameter, die für die verschiedenen COPY- und FORMAT-Befehle in NEWDOS/80 und GDOS zulässig sind, sowie diverse Bits und Bytes variabler Länge, um spezielle Eigenschaften einiger Parameter erkennen und bearbeiten zu können.
Das Verständnis dieser Tabelle ist eine grundlegende Voraussetzung, um sich in SYS6 zurechtzufinden.

Die hier ausgedruckten Tabellen haben folgenden Aufbau:

- Tabellenadresse in SYS6/SYS, wo der Parameter steht

- Name des Parameters

- eine 8-Bit-Maske, die folgende Bedeutung hat:
  Bit 7: ist immer gesetzt
  Bit 6: wenn gesetzt: Parameter für FORMAT erlaubt
  Bit 5: wenn gesetzt: Parameter für COPY 5+6 erlaubt
  Bit 4: wenn gesetzt: es folgt eine 16-Bit-Adresse, die durch einen CALL aufgerufen werden muß
  Bit 3: wenn gesetzt: Parameter für COPY 1-4 erlaubt
  Bit 2: wenn gesetzt: es folgt eine 32-Bit-Maske, die angibt, welche der Bits im Bereich 5994H - 5997H nicht gesetzt sein dürfen, um unzulässige Kombinationen von Parametern auszuschließen.
  Bits 1+0: ergibt nach Addition von 5994H eine Adresse im Bereich 5994H - 5997H, in der das Bit gesetzt werden muß, das auch in dem folgenden Byte gesetzt ist

- eine Adresse und das Bit, welches bei der Eingabe dieses Parameters gesetzt wird

- ggf. eine 32-Bit-Maske gemäß Bit 2 der o.g. 8-Bit-Maske

- ggf. eine CALL-Adresse gemäß Bit 4 der o.g. 8-Bit-Maske

6.1 Parametertab. NEWDOS/80 Vers. 2 für TRS-80 Model I+III

Adr.	Param.	Maske	SET	Bit-Check	CALL	
505C	CFWO	1010 0001	5995,6	-- -- -- --	----	
5062	DDGA-	1111 0101	5995,1	00 00 00 0A	4F9E	
506F	ODPW-	1011 0101	5995,0	C0 00 00 00	4FED	
507C	NDPW-	1011 0101	5995,5	02 00 00 00	4FE5	
5089	DDSL-	1111 0101	5995,2	00 00 00 0A	4FC3	
5096	SPDN-	1011 1011	5997,7	-- -- -- --	4FFF	
509F	DPDN-	1111 1011	5997,6	-- -- -- --	4FF5	
50A8	PFST-	1101 0111	5997,1	B9 86 00 00	4FD6	(1)
50A8	PFST-	1101 0111	5997,1	F9 86 00 00	4FD6	(3)
50B5	PFTC-	1101 0011	5997,0	-- -- -- --	4FCE	
50BE	NDN-	1011 0100	5994,2	0A 00 00 00	4F95	
50CA	ODN-	1111 0100	5994,4	C0 00 00 02	4F8B	
50D6	DDND	1110 0100	5994,5	C0 80 00 02	----	
50E0	NDMW	1110 0101	5995,7	20 00 00 00	----	
50EA	SPW-	1011 0010	5996,6	-- -- -- --	4FDD	
50F2	NFMT	1010 0111	5997,3	00 06 00 04	----	
50FC	XLF-	1011 0111	5997,5	00 00 00 10	4FAC	
5108	ILF-	1011 0111	5997,4	00 00 00 20	4FAC	
5114	KDN	1110 0100	5994,3	C6 00 00 02	----	
511D	SN-	1011 0010	5996,7	-- -- -- --	4F90	
5124	KDD	1110 0100	5994,0	C2 10 00 02	----	
512D	USD	1010 0101	5995,4	03 00 00 00	----	
5136	BDU	1110 0100	5994,1	0C 30 08 00	----	
513F	UBB	1010 0010	5996,5	-- -- -- --	----	
5144	CBF	1010 0110	5996,3	02 00 00 00	----	
514D	UPD	1010 0010	5996,0	-- -- -- --	----	
5152	USR	1010 0010	5996,4	-- -- -- --	----	
5157	DFO	1010 0101	5995,3	00 00 00 04	----	
5160	RWF	1100 0110	5996,7	09 06 00 02	----	
5169	FMT	1010 0111	5997,2	00 08 00 08	----	
5172	/	1011 0010	5996,1	-- -- -- --	500D	
5177	Y	1110 0100	5994,6	B9 01 00 00	----	(1)
5177	Y	1110 0100	5994,6	B9 01 00 02	----	(3)
517E	N	1110 0100	5994,7	79 01 00 02	----	

(1) nur für TRS-80 Model I
(3) nur für TRS-80 Model III

6.2 Parametertabelle GDOS für Genie I,II,IIs,III,IIIs

Adr.	Param.	Maske	SET	Bit-Check	CALL	
505C	FRAG	1010 0001	5995,6	-- -- -- --	----	
5062	AEIV-	1111 0101	5995,1	00 00 00 0A	4F9E	
506F	AZKW-	1011 0101	5995,0	C0 00 00 00	4FED	
507C	NZKW-	1011 0101	5995,5	02 00 00 00	4FE5	
5089	SBIV-	1111 0101	5995,2	00 00 00 0A	4FC3	
5096	QPDN-	1011 1011	5997,7	-- -- -- --	4FFF	
509F	ZPDN-	1111 1011	5997,6	-- -- -- --	4FF5	
50A8	SPUR-	1101 0111	5997,1	B9 86 00 00	4FD6	
50B5	STOP-	1101 0011	5997,0	-- -- -- --	4FCE	
50BE	NZN-	1011 0100	5994,2	0A 00 00 00	4F95	
50CA	AZN-	1111 0100	5994,4	C0 00 00 02	4F8B	
50D6	ZZND	1110 0100	5994,5	C0 80 00 02	----	
50E0	KDWA	1110 0101	5995,7	20 00 00 00	----	
50EA	QKW-	1011 0010	5996,6	-- -- -- --	4FDD	
50F2	NFMT	1010 0111	5997,3	00 06 00 04	----	
50FC	XDL-	1011 0111	5997,5	00 00 00 10	4FAC	
5108	IDL-	1011 0111	5997,4	00 00 00 20	4FAC	
5114	BZN	1110 0100	5994,3	C6 00 00 02	----	
511D	QN-	1011 0010	5996,7	-- -- -- --	4F90	
5124	BZD	1110 0100	5994,0	C2 10 00 02	----	
512D	SQD	1010 0101	5995,4	03 00 00 00	----	
5136	IVU	1110 0100	5994,1	0C 30 08 00	----	
513F	JHL	1010 0010	5996,5	-- -- -- --	----	(*)
513F	TCS	1010 0010	5996,5	-- -- -- --	----	(S)
5144	EDK	1010 0110	5996,3	02 00 00 00	----	
514D	BEA	1010 0010	5996,0	-- -- -- --	----	
5152	FRD	1010 0010	5996,4	-- -- -- --	----	
5157	NVD	1010 0101	5995,3	00 00 00 04	----	
5160	MAG	1100 0110	5996,7	09 06 00 02	----	
5169	FMT	1010 0111	5997,2	00 08 00 08	----	
5172	/	1011 0010	5996,1	-- -- -- --	500D	
5177	J	1110 0100	5994,6	B9 01 00 00	----	
517E	N	1110 0100	5994,7	79 01 00 02	----	

(*) nur für Genie I,II,IIs,III
(S) nur für Genie IIIs

Kapitel 7: SYS-Files und Aufbau des Directory

Dieses Kapitel beschreibt Funktion und Einsprungbedingungen der SYS/Files, Aufbau und Struktur des Directory und von FCB's (File Control Blocks) sowie einige Fehler im DOS, die vielleicht noch nicht bekannt sind.

7.1 Die SYS-Files

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I+III
GDOS für Genie I,II,IIs,III,IIIs *)

*) Die meisten hier gemachten Angaben gelten auch für die SYS-Files von GDOS, es werden jedoch nicht diejenigen SYS-Files beschrieben, die es nur in GDOS gibt.

In den SYS-Files sind alle Befehle und Funktionen des DOS untergebracht. Die meisten benutzen den RAM-Bereich 4D00H bis 51FFH und wechseln sich dort so ab, wie sie gerade gebraucht werden. Für das Laden und Starten von SYS-Files ist UP GETSYS (4BC9H) zuständig, siehe Kapitel 3.

7.1.1 RAM-Bereiche der SYS-Files

Die folgende Tabelle gibt an, in welche RAM-Bereiche die einzelnen SYS-Files geladen und bei welcher Adresse sie gestartet werden sowie ihren EOF (End of File) - und zwar nicht den, der im Directory verzeichnet ist, sondern den wirklichen EOF:

SYS-File	EOF	von - bis	Startadresse
SYS0/SYS	15/0	400C-51DA *)	4D00
SYS1/SYS	4/248	4D00-51DF	4D00
SYS2/SYS	4/197	4D00-51AC	4D00
SYS3/SYS	4/248	4D00-51DF	4D00
SYS4/SYS	5/0	4D00-51E7	4D00
SYS5/SYS	5/0	4D00-51E7	4D00
SYS6/SYS	34/235	4D00-6FF9 *)	4D00
SYS7/SYS	5/0	4D00-51E7	4D00
SYS8/SYS	5/0	4D00-51E7	4D00
SYS9/SYS	4/248	4D00-51DF	4D00
SYS10/SYS	5/0	4D00-51E7	4D00
SYS11/SYS	5/0	4D0C-51F3	4D0C
SYS12/SYS	4/235	4D00-51D2	4D00

SYS-File	EOF	von - bis	Startadresse
SYS13/SYS	5/0	4D00-51E7	4D0C
SYS14/SYS	5/0	4D00-51E7	4D00
SYS15/SYS	5/0	4D00-51E7	4D00
SYS16/SYS	5/0	4D00-51E7	4D00
SYS17/SYS	5/0	4D00-51E7	4D00
SYS18/SYS	5/0	5200-56E7	5200
SYS19/SYS	5/0	5200-56E7	5200
SYS20/SYS	5/0	5200-56E7	5200
SYS21/SYS	5/0	4D00-51E7	4D00
BASIC/CMD	17/154	5700-684D *)	66BE

*) Diese Files laden nicht in den gesamten angegebenen Adressbereich, sondern lassen zwischendurch einige Lücken.

7.1.2 Einsprungbedingungen der SYS-Files

Die folgende Tabelle zeigt, welche DOS-Befehle (DB), Unterprogramme (UP) und Funktionen des DOS von welcher SYS-File ausgeführt werden und welche Werte (HEX) man dazu in Register A und ggf. in Register C laden muß:

Name	A=	C=	Funktion
SYS1	23		UP DOSRDY (402D)
	43		UP ERROR0 (4030)
	63		UP DOSCMD (4405)
	83		UP TFSPEC (441C)
	A3		UP INSEXT (4473)
	C3		UP DOSCAL (4419)
	E3	01	?
	E3	02	DB LIB
	E3	03	./. (ILLEGAL DOS FUNCTION)
	E3	04	Start MINI-DOS durch Tasten 'DFG'
	E3	05	DB MDBORT
	E3	06	DB MDRET
	E3	07	?
	E3	08	Re-Start von SYS1 nach UP LINPUT (0040H)
	E3	09	DB CLS
	E3	0A	./. (ILLEGAL DOS FUNCTION)
	E3	0B	ähnlich UP TFSPEC (441C)
	E3	0C	DEBUG-Befehl 'Q'
SYS2	24		UP OPEN (4424)
	44		UP INIT (4420)
	64		belegt neue GRANS für eine FILE
	84		berechnet den Code eines Passwortes
	A4		DB LOAD und UP LOAD (4430)
	C4		UP RUN (4433)
	E4	01	DB RENAME
	E4	02	freien FDE im HIT-Sector suchen + belegen
	E4	03	Laden und Starten von Benutzer-Programmen
	E4	04	legt eine neue File an, die vorher noch nicht existieren darf

Name	A=	C=	Funktion
SYS3	25		UP CLOSE (4428)
	45		DB KILL und UP KILL (442C)
	65		UP INTINS (4410) (Model III: 447B)
	85		UP INTDEL (4413)
	A5		DB JKL und Tasten 'JKL'
	C5		UP PUTEOF (4451)
	E5	01	DB BLINK
	E5	02	DB CLOCK
	E5	03	DB DEBUG
	E5	04	DB VERIFY
	E5	05	DB BREAK
	E5	06	?
	E5	07	?
	E5	08	DB LCDVR (entfällt bei Model III)
	E5	09	DB LC
SYS4	26		UP DOSERR (4409)
	46		arbeitet wie A=26, jedoch anschließend "DOS FATAL ERROR!! KEY 'R' FOR RESET"
	E6		(nur zum Laden von SYS4 in SYS6 für COPY)
SYS5	--		UP DEBUG (440D) und Tasten '123'
SYS6	28		DB FORMAT
	48		DB COPY
	68		DB APPEND
SYS7	E9	01	Anfang von DB SYSTEM (Rest in SYS17)
	E9	02	DB HIMEM
	E9	03	Anfang von DB PDRIVE (Rest in SYS16)
	E9	04	DB AUTO
	E9	05	DB ATTRIB
	E9	06	DB PROT
	E9	07	DB DUMP
	E9	08	?
	E9	09	DB PURGE
	E9	0A	DB TIME
	E9	0B	DB DATE
SYS8	2A		DB DIR
	4A		DB FREE
SYS9	2B		UP USRINS (4461)
	4B		UP USRDEL (4464)
	6B		./. (ILLEGAL DOS FUNCTION)
	8B		./. (ILLEGAL DOS FUNCTION)
	AB		./. (ILLEGAL DOS FUNCTION)
	CB		CHAINING-Aufruf in Tastatur-Routine
	EB	01	?
	EB	02	DB MDCOPY
	EB	03	DB CHAIN / DB DO
	EB	04	CHAINING-Aufruf in SYS1 (DOS READY)
	EB	05	DB CHNON
	EB	06	DB BASIC2 (entfällt bei Model III)
	EB	07	DB *name (Aufruf von Benutzer-Routinen)

Name	A=	C=	Funktion
SYS9	EB	08	DB PAUSE
	EB	09	DB STMT
	EB	0A	DB BOOT
	EB	0B	bewirkt "DOS FATAL ERROR!! KEY 'R' FOR RESET" und danach DB BOOT
	EB	0C	bewirkt "'RUN ONLY' STOPPED!! KEY 'R' FOR RESET" und danach DB BOOT
SYS10	2C		BASIC-Befehl GET (Z-Flag=1, Call 4D03) BASIC-Befehl PUT (Z-Flag=0, Call 4D03)
SYS11	--		BASIC-Befehl RENUM (Anfang ist in SYS13)
SYS12	--		BASIC-Befehl REF
SYS13	2F		BASIC-Fehlermeldung anzeigen (Register E)
	4F		Anfang von BASIC-Befehl RENUM
SYS14	F0	01	DB ROUTE
	F0	02	DB CREATE
	F0	03	?
	F0	04	DB CLEAR
	F0	05	DB LIST
	F0	06	DB PRINT
	F0	07	DB ERROR
SYS15	F1	01	DB FORMS (nur Model III)
	F1	02	DB SETCOM (nur Model III) - bei Model I enthält diese SYS-File gar nichts und kann gelöscht werden! -
SYS16	32		DB PDRIVE (der Anfang steht in SYS7)
SYS17	33		DB SYSTEM (der Anfang steht in SYS7)
	53		DB WRDIRP
	F3		unter GDOS für Genie I,II,IIs,III,IIIs steht in SYS17 bei 501CH eine Erweiterung für SYS6, die beim Aufruf des DOS-Befehls COPY 5 (kopieren einer ganzen Diskette ohne CBF) dazu dient, die GRAN Allocation Tabelle im GAT-Sector von Disketten mit mehr als 96D Lumps zu verlängern
SYS18	34		BASIC-Modul
SYS19	35		BASIC-Modul
SYS20	36		BASIC-Modul
SYS21	37		BASIC-Befehl CMD "O"
	57		wenn B=20H: BASIC-Befehl CMD "F=KEEP"
	57		wenn B=28H: BASIC-Befehl CMD "F=ERASE"

7.2 Der Directory

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I+III
GDOS für Genie I,II,IIs,III,IIIs
Colour DOS

Der Directory jeder Diskette besteht aus 1 GAT-Sector (GRAN Allocation Tabelle), 1 HIT-Sector (Hash Index Tabelle) und - in Abhängigkeit vom PDRIVE-Parameter DDGA (Default Directory GRAN Allocation) dieser Diskette - 8, 13D, 18D, 23D oder 28D weitere Sectoren, die jeder bis zu 8 FDE's (File Directory Entry) enthalten können.

Um die Position des Directory innerhalb der Diskette schnell und sicher finden zu können, ist in dem 3. Byte des 1. Sectors jeder Diskette die LUMP# eingetragen, in welcher der Directory auf dieser Diskette beginnt. Zur Sicherheit wird zusätzlich der Directory mit einem besonderen Data Adress Mark auf Diskette geschrieben, siehe Kapitel 1.4.5.

7.2.1 Der GAT-Sector

Der GAT-Sector ist der 1. Sector des Directory und enthält folgende Informationen:

Die ersten C0H (192D) Bytes bilden die GRAN Allocation Tabelle (GAT), die angibt, welches GRAN der Diskette noch frei ist und welches nicht. In Abhängigkeit vom PDRIVE-Parameter GPL (GRANS pro Lump) werden 2 bis 8 GRANS (aus Organisationsgründen) jeweils zu einem sogenannten LUMP zusammengefaßt. Für jedes der max. 192D möglichen Lumps einer Diskette gibt es 1 Byte in der GAT und für jedes GRAN eines Lumps gibt es 1 Bit in diesem Byte (Bit 0 für das 1. GRAN, Bit 1 für das nächste GRAN, usw.), welches gesetzt ist, wenn das GRAN durch eine File belegt ist bzw. gelöscht ist, wenn das GRAN frei ist.

In den relativen Bytes CEH und CFH ist der Code des Master-Passwortes der Diskette gespeichert (entfällt unter Colour DOS).

In den relativen Bytes D0H-D7H steht der Name der Diskette.

In den relativen Bytes D8H-DFH steht das Datum der Diskette.

In den relativen Bytes E0H-FFH kann ein AUTO-Befehl (mit 0DH als Endemarkierung) zum BOOTen dieser Diskette stehen (entfällt unter Colour DOS).

7.2.1.1 Belegung von freien GRANS

Bei der Suche nach einem freien GRAN auf einer Diskette könnte man ja einfach anhand der GAT (GRAN Allocation Tabelle im GAT-Sector) das erste freie GRAN auf der Diskette ermitteln und verwenden. Aber so einfach macht es sich das DOS nicht!

Während TRSDOS und NEWDOS 2.1 mehr zufällig bestimmt haben, welches freie GRAN auf einer Diskette als nächstes für eine File belegt werden sollte, um eine möglichst gleichmäßige Be- und Abnutzung der Disketten zu erreichen (die 3 niederwertigsten Bits vom DEC des FPDE dieser File ergaben nach der Multiplikation mit 4 eine Track# im Bereich 0,4,8,..,28D, ab der zuerst nach einem freien GRAN gesucht wurde) arbeiten NEWDOS/80 und GDOS nach einer anderen Philosophie:

Wenn ein neues GRAN für eine File belegt werden soll, die bereits 1 oder mehrere GRANS besitzt, dann wird zuerst versucht, die File an ihrem Ende zu erweitern, d. h. als nächstes GRAN genau das zu belegen, welches direkt hinter dem letzten belegten GRAN dieser File liegt (dadurch wird vermieden, Files auf der Diskette unnötig zu splitten).

Falls dieses GRAN jedoch nicht mehr frei ist, oder wenn ein neues GRAN für eine File belegt werden soll, die noch gar keine GRANS besitzt, dann wird das erste freie GRAN der Diskette (ab Track 0, Sector 0) genommen und belegt.

Durch dieses Verfahren werden die GRANS in den äußeren Tracks einer Diskette bevorzugt, was den Vorteil einer höheren Datensicherheit hat, da in den äußeren Tracks, aufgrund des größeren Umfangs, mehr Platz pro Bit zur Verfügung steht und der Disketten-Betrieb dort weniger störanfällig ist als in den inneren Tracks.

Die Routine zum Belegen von freien GRANS befindet sich unter NEWDOS/80 und GDOS in SYS2 bei 4F82H-50C9H und unter Colour DOS im ROM bei D7EFH-D937H.

7.2.2 Der HIT-Sector

Der HIT-Sector ist der 2. Sector des Directory und enthält folgende Informationen:

Das relative Byte 1FH gibt die um 10D verminderte Anzahl von Sectoren an, aus denen der Directory insgesamt besteht (inkl. GAT- und HIT-Sector).

Alle übrigen Bytes bilden die Hash Index Tabelle (HIT), die angibt, welche FDE's im Directory noch frei sind und welche nicht, und die außerdem dazu dient, eine File beim Öffnen (OPEN) möglichst schnell im Directory zu finden.

Zu jedem FDE des Directory gehört ein ganz bestimmtes Byte in der HIT. Wenn dieses Byte gelöscht ist (00H), dann ist der dazugehörige FDE unbenutzt oder existiert nicht. Wenn dieses Byte nicht gelöscht ist (ungleich 00H), dann wird der dazugehörige FDE von einer File benutzt (als FPDE oder FXDE) und der Wert dieses Bytes in der HIT gibt den sogenannten Hash-Code dieser File an (gilt sowohl für FPDE's als auch für FXDE's).

Der Hash-Code einer File ist immer ein Wert zwischen 01H und FFH und wird anhand des Filenamen und des Filetyps nach einer ganz bestimmten Formel berechnet. Um eine File beim Öffnen (OPEN) möglichst schnell im Directory zu finden, wird einfach der Hash-Code dieser File berechnet und der HIT-Sector geprüft, ob dieser Wert irgendwo steht:

Wenn nein, kann es die gesuchte File auf dieser Diskette nicht geben und die ganzen FDE-Sectoren brauchen überhaupt nicht mehr gelesen zu werden.

Wenn der berechnete Hash-Code jedoch einmal oder mehrmals in der HIT gefunden wird, brauchen nur die dazugehörigen FDE's von Diskette gelesen und überprüft zu werden, ob einer von Ihnen die gesuchte File enthält.

Dadurch wird die Anzahl der FDE-Sectoren, die von Diskette gelesen werden müssen, drastisch reduziert.

Die Position eines Bytes im HIT-Sector entspricht exakt dem DEC (Directory Entry Code) des dazugehörigen FDE's, sodaß zwischen beiden ein einfacher und eindeutiger Zusammenhang besteht.

7.2.2.1 Berechnung von Hash-Codes

Die Routine und Formel zur Berechnung von Hash-Codes ist sehr einfach und befindet sich unter NEWDOS/80 und GDOS in SYS2 bei 4E8DH bis 4E9DH und unter Colour DOS im ROM bei D70EH-D71EH.

7.2.3 Die FDE-Sectoren

Jeder Sector des Directory, mit Ausnahme des GAT-Sectors und des HIT-Sectors, kann 8 FDE's (File Directory Entry) enthalten. Während unter TRSDOS und NEWDOS 2.1 die ersten 2 FDE's jedes FDE-Sectors für SYS-Files reserviert waren, stehen unter NEWDOS/80, GDOS und Colour DOS alle 8 FDE's eines FDE-Sectors dem Anwender zur freien Verfügung.

Ein FDE besteht aus 32D Bytes und enthält wichtige Informationen über eine File (z. B. den Namen der File, wo sie auf Diskette steht, usw.). Manchmal reicht jedoch 1 FDE nicht aus, um alle Informationen über eine File aufzunehmen. In so einem Fall werden dann mehrere FDE's für 1 File angelegt.

Der erste FDE einer File wird als FPDE (File Primary Directory Entry) bezeichnet und alle weiteren FDE's - sofern es welche gibt - als FXDE (File Extended Directory Entry).

Aus Organisationsgründen besitzen alle FDE's innerhalb des Directory eine Art Nummer, den sogenannten Directory Entry Code (DEC), der aus 8 Bits mit dem Format rrrssssss besteht:

ssssss+2	gibt an, im wievielten Sector des Directory der FDE steht (der GAT-Sector hat die Nummer 0)
rrr	gibt an, um den wievielten FDE in diesem Sector es sich handelt

7.2.3.1 Aufbau des FPDE

Die 32D Bytes innerhalb eines FPDE (File Primary Directory Entry) haben folgende Bedeutung:

Byte	Bedeutung im FPDE
0	Bit 7: gelöscht, da FPDE und kein FXDE Bit 6: gesetzt wenn es sich um eine SYS-File handelt Bit 5: ./. Bit 4: gesetzt wenn der FPDE zu einer File gehört, gelöscht wenn der FPDE unbenutzt (frei) ist Bit 3: gesetzt wenn die File unsichtbar (INV) ist Bits 2-0: Zugriffs-Level (0-7) aufgrund eines Passwortes
1	Bit 7: gesetzt wenn ASE=N (siehe DOS-Befehl ATTRIB) Bit 6: gesetzt wenn ASC=N (siehe DOS-Befehl ATTRIB) Bit 5: gesetzt wenn UDF=Y (siehe DOS-Befehl ATTRIB) Bits 4-0: ./.
2	./.
3	3. Byte des EOF-Wertes, gibt die relative Position innerhalb des letzten Sectors an
4	Logische Recordlänge (1-256D)
05-0C	Name der File, rechts mit Blanks aufgefüllt
0D-0F	Typ der File, rechts mit Blanks aufgefüllt
10H,11H	Code des Update-Passwortes (außer bei Colour DOS)
12H,13H	Code des Access-Passwortes (außer bei Colour DOS)
14H 15H	2. und 1. Byte des EOF-Wertes (End of File) Die 3 Bytes des EOF-Wertes geben im RBA-Format die aktuelle Länge der File an.

Byte	Bedeutung im FPDE
16H 17H	gibt an, wo der 1. Datenblock dieser File auf Diskette steht: a) 1. Byte = FF: Es steht kein Datenblock mehr auf Diskette. b) 1. Byte = FE: Im 2. Byte steht der DEC (Directory Entry Code) für den nächsten FXDE dieser File. c) 1. Byte = 00-FD: Gibt die LUMP# an, in der dieser Datenblock auf Diskette beginnt. In diesem Fall geben Bit 7-5 des 2. Bytes die Anzahl der GRANS an, die zwischen dem Beginn des LUMPS und dem Beginn des Datenblocks liegen, und Bits 4-0 des 2. Bytes die um 1 verminderte Anzahl der GRANS, aus denen dieser Datenblock besteht.
18H 19H	Angaben über den 2. Datenblock dieser File (Bedeutung der Bytes wie beim 1. Datenblock)
1AH 1BH	Angaben über den 3. Datenblock dieser File (Bedeutung der Bytes wie beim 1. Datenblock)
1CH 1DH	Angaben über den 4. Datenblock dieser File (Bedeutung der Bytes wie beim 1. Datenblock)
1EH 1FH	Angaben über weitere Datenblocks dieser File: a) 1. Byte = FF: Es steht kein weiterer Datenblock auf Diskette. b) 1. Byte = FE: Im 2. Byte steht der DEC (Directory Entry Code) für den nächsten FXDE dieser File.

7.2.3.2 Aufbau des FXDE

Wenn die Daten einer File auf mehr als 4 Datenblocks verteilt sind, reicht der Speicherplatz im FPDE nicht aus und es muß für jeweils 4 weitere Datenblocks ein FXDE angelegt werden.

Die 32D Bytes innerhalb eines FXDE (File Extended Directory Entry) haben folgende Bedeutung:

Byte	Bedeutung im FXDE
0	Bit 7: gesetzt, da FXDE und kein FPDE Bits 6+5: ./. Bit 4: gesetzt wenn FXDE von einer File benutzt wird Bits 3-0: ./.
1	Directory Entry Code (DEC) vom letzten FDE (FPDE oder FXDE) dieser File, der vor diesem FXDE liegt. Durch diese Rückwärts-Verkettung kann man von jedem FXDE aus leicht alle vorigen FDE's finden.
02H-15H	./. (Null)
16H-1FH	hat exakt die gleiche Bedeutung wie im FPDE

7.2.3.3 Belegung von freien FDE's

Wenn ein neuer FDE angelegt werden soll, könnte man ja einfach anhand der HIT (Hash Index Tabelle im HIT-Sector) den ersten freien FDE des Directory ermitteln und benutzen. Aber so einfach macht es sich das DOS auch hier nicht:

Während TRSDOS und NEWDOS 2.1 den Interrupt-Zähler 4040H, der bei jedem RTC-Interrupt (Real-Time Clock) um 1 erhöht wird, als Zufallsfunktion benutzt haben, um für eine gleichmäßige Auslastung aller FDE-Sectoren zu sorgen, wenden NEWDOS/80 und GDOS eine andere Methode an, um das gleiche Ziel zu erreichen:

Nachdem der Hash-Code der File berechnet worden ist, für die ein FPDE angelegt werden soll, wird anhand der niederwertigsten 5 Bits dieses Hash-Codes (die einen Wert zwischen 0 und 31D ergeben können), wie bei einem Abzählreim derjenige Sector des Directory bestimmt, in dem zuerst nach einem freien FDE gesucht werden soll (zur schnellen Erkennung von freien FDE's wird selbstverständlich die HIT herangezogen). Nur wenn in diesem Sector und in allen nachfolgenden Sectoren des Directory nichts mehr frei ist, erfolgt anschließend ein 2. Durchgang, der dann mit dem ersten FDE-Sector des Directory beginnt.

Bei der Belegung von FXDE's wird allerdings (wie auch schon in TRSDOS und NEWDOS 2.1) immer zuerst ab dem FDE-Sector nach einem freien FDE gesucht, in dem der FPDE der File steht, um zum Lesen aller FDE's einer File möglichst nur wenige verschiedene Sectoren des Directory laden zu müssen.

Die Routine zum Belegen von freien FDE's befindet sich unter NEWDOS/80 und GDOS in SYS2 bei 50CFH-5104H und unter Colour DOS im ROM bei D938H-D96DH.

7.2.3.4 Berechnung von Passwort-Codes

Die Routine mit der relativ komplizierten Formel zur Berechnung von Passwort-Codes befindet sich unter NEWDOS/80 und GDOS in SYS2 bei 5155H bis 5187H sowie zusätzlich in SYS6 bei 502BH bis 505BH und entfällt unter Colour DOS.

7.3 Der FCB (File Control Block)

Betriebssysteme: NEWDOS/80 Version 2 für TRS-80 Model I+III
GDOS für Genie I,II,IIs,III,IIIs
Colour DOS

Beim Öffnen (OPEN) einer File wird vom DOS ein 32D Byte langer FCB angelegt, in dem bis zum Schließen (CLOSE) der File alle wichtigen Informationen über die File und ihren jeweiligen Zustand vom DOS eingetragen und ständig aktualisiert werden.

Der Aufbau des FCB wurde so konzipiert, daß der FCB möglichst viele (am besten alle) Informationen enthält, die man zum Arbeiten mit der File braucht, sodaß möglichst selten (am besten gar nicht) auf den Directory zurückgegriffen werden muß.

Wenn mehrere Files gleichzeitig geöffnet sind, besitzt jede File einen eigenen FCB. Wo dieser im RAM stehen soll, muß dem DOS jeweils beim Öffnen (OPEN) einer File angegeben werden.

Die einzelnen Bytes in einem FCB haben folgende Bedeutung:

Byte	Bedeutung im FCB
0	Bit 7: gesetzt wenn der FCB geöffnet ist Bits 6-3: ./. Bit 2: muß gelöscht sein, wegen Abfrage in SYS0 bei 4823H (Model III: 47C8H, Colour DOS: D178H) Bit 1: gesetzt wenn der FCB nicht zu einer File gehört, sondern zu einer ganzen Diskette Bit 0: gesetzt wenn die Sectoren dieser File wie DIR-Sectoren lesegeschützt geschrieben werden sollen
1	Bit 7: gesetzt wenn die Logische Recordlänge (LRL) ungleich 256D ist Bit 6: (1) Der EOF-Wert soll nach dem Schreiben nur dann auf den NEXT-Wert gesetzt werden, wenn dieser größer als der EOF-Wert ist. (0) Der EOF-Wert soll nach jedem Schreiben auf den NEXT-Wert gesetzt werden. Bit 5: gelöscht wenn der aktuelle Sector im Buffer steht Bit 4: gesetzt wenn die Daten im Buffer noch auf Diskette geschrieben werden müssen Bit 3: gesetzt in NEWDOS/80 Version 2, GDOS und Colour DOS Bits 2-0: Zugriffs-Level (0-7) aufgrund eines Passwortes

Byte	Bedeutung im FCB
2	Bit 7: gesetzt wenn ASE=N (siehe DOS-Befehl ATTRIB) Bit 6: gesetzt wenn ASC=N (siehe DOS-Befehl ATTRIB) Bit 5: gesetzt wenn UDF=Y (siehe DOS-Befehl ATTRIB) Bits 4-0: ./.
3 4	Zeiger auf den Buffer, der für diese File benutzt werden soll
5	3. Byte des NEXT-Wertes, gibt die relative Position innerhalb des aktuellen Sectors an
6	Drive# der File bzw. der Diskette
7	DEC (Directory Entry Code) vom FPDE der File
8	3. Byte des EOF-Wertes, gibt die relative Position innerhalb des letzten Sectors an
9	Logische Recordlänge (1-256D)
0A 0B	2. und 1. Byte des NEXT-Wertes. Die 3 Bytes des NEXT-Wertes geben im RBA-Format an, wohin/woher der nächste Record zu schreiben/zu lesen ist, wonach der NEXT-Wert jeweils entsprechend erhöht wird.
0C 0D	2. und 1. Byte des EOF-Wertes (End of File) Die 3 Bytes des EOF-Wertes geben im RBA-Format die aktuelle Länge der File an.
0E 0F	gibt an, wo der 1. Datenblock dieser File auf Diskette steht: a) 1. Byte = FF: Es steht kein Datenblock mehr auf Diskette. b) 1. Byte = FE: Im 2. Byte steht der DEC (Directory Entry Code) für den nächsten FXDE dieser File. c) 1. Byte = 00-FD: Gibt die LUMP# an, in der dieser Datenblock auf Diskette beginnt. In diesem Fall geben Bit 7-5 des 2. Bytes die Anzahl der GRANS an, die zwischen dem Beginn des LUMPS und dem Beginn des Datenblocks liegen, und Bits 4-0 des 2. Bytes die um 1 verminderte Anzahl der GRANS, aus denen dieser Datenblock besteht.
10H 11H	Angaben über den 2. Datenblock dieser File (Bedeutung der Bytes wie beim 1. Datenblock)
12H 13H	Angaben über den 3. Datenblock dieser File (Bedeutung der Bytes wie beim 1. Datenblock)
14H 15H	Angaben über den 4. Datenblock dieser File (Bedeutung der Bytes wie beim 1. Datenblock)

Byte	Bedeutung im FCB
16H 17H	wenn <> FFFF: in den folgenden 8 Bytes sind Angaben über die 4 Datenblöcke eines FXDE, der zu dieser File gehört, enthalten und hier steht die Anzahl der GRANS, die zwischen dem Beginn der File und dem Beginn des 1. Erweiterungs-Datenblocks dieses FXDE's liegen.
18H 19H	gibt an, wo der 1. Erweiterungs-Datenblock eines FXDE's auf Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)
1AH 1BH	gibt an, wo der 2. Erweiterungs-Datenblock eines FXDE's auf Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)
1CH 1DH	gibt an, wo der 3. Erweiterungs-Datenblock eines FXDE's auf Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)
1EH 1FH	gibt an, wo der 4. Erweiterungs-Datenblock eines FXDE's auf Diskette steht (Bedeutung der Bytes wie beim 1. Datenblock)

7.4 Fehler im DOS

Die folgenden Fehler sind mir aufgefallen, ohne daß ich etwas über ihr Bekanntsein oder über ZAPS, wie man sie umgehen kann, in Erfahrung bringen konnte.

Sich selber ZAPS auszudenken, ist nicht ungefährlich, weil man sehr leicht eine Eigenheit des DOS übersehen und den Fehler unter Umständen sogar noch schlimmer machen kann.

Daher erfolgt hier die Angabe von ZAPS, sofern welche vorgeschlagen werden, unter Vorbehalt und OHNE JEDE GEWÄHR!

Die ZAPS werden in folgendem Format angegeben:
- Filename der betreffenden SYS-File
- RAM-Adresse, wo die Änderung erfolgt
- relativer Sector der File (beginnend ab 0) und
- relatives Byte innerhalb dieses Sectors, wo der Inhalt für diese RAM-Adresse steht
- der bisherige (falsche) Inhalt an dieser Stelle und
- der neue (hoffentlich richtige) Inhalt

1) SYS0 aus NEWDOS/80 Version 2 und GDOS

Wenn innerhalb der Initialisierung von SYS0 der PDRIVE-Sector von Diskette gelesen wird, stehen die Zeitkonstanten für UP DELAY1 (47E3H, Model III: 4789H) und UP DELAY2 (4CEDH, Model III: 4C92H) noch auf dem Wert für SYSTEM BJ=1. Je schneller die CPU läuft, besonders wenn SYS0 mehrere Tracks lang ist und zum Laden des PDRIVE-Sectors über mehrere Tracks hinweg gesteppt werden muß, umso eher führt dies zu einem Absturz des Systems.

ZAP-Vorschlag:
a) NEWDOS/80 Version 2 für TRS-80 Model I und GDOS für Genie I,II,IIs,III,IIIs:

SYS0/SYS, 47E4H, 04H, DDH, von 06H nach 18H,
4CEFH, 0AH, 00H, von 01H nach 04H

b) NEWDOS/80 Version 2 für TRS-80 Model III:

SYS0/SYS, 478AH, 04H, 59H, von 08H nach 20H,
4C94H, 09H, 77H, von 01H nach 04H

2) SYS0 aus NEWDOS/80 Version 2, GDOS und Colour DOS

In UP READ (4436H, Colour DOS: CE36H) wird bei 4A29H (Model III: 49CEH, Colour DOS: D36FH) auch nach einem fehlerhaften Versuch, einen Sector von Diskette zu lesen, im FCB vermerkt, der zu lesende Sector stünde nun im Buffer zur Verfügung.

3) SYS0 aus NEWDOS/80 Version 2 und GDOS

In UP GETSYS (4BC9H, Model III: 4B6EH) werden bei 4BF5H (Model III: 4B9AH) bereits gelöschte SYS-Files nicht als gelöscht erkannt und trotzdem geladen und gestartet, was so lange gut geht, bis die betreffenden GRANS der gelöschten SYS-File eines Tages mit neuem Inhalt belegt werden.

Am besten setzt man nach dem Löschen einer SYS-File mit SUPERZAP (NEWDOS/80) oder DDE (GDOS) im Directory das 1. Byte des FPDE's der gelöschten SYS-File auf 00H, denn dann erkennt das DOS beim Versuch, diese SYS-File zu laden, daß es sich um keine SYS-File mehr handeln kann und gibt korrekterweise die entsprechende Fehlermeldung "SYSTEM PROGRAM NOT FOUND" aus.

4) SYS0 aus GDOS für Genie IIIs und GDOS 2.4

Wenn in UP GETSYS (4BC9H) beim Laden einer SYS-File (ausgenommen SYS4) ein Error auftritt, wird in jedem Fall zum Aufrufer von GETSYS zurückgekehrt, auch wenn dieser evtl. gar nicht darauf eingestellt ist. So kann das ganze System hübsch abstürzen.

5) SYS0 aus GDOS für Genie IIIs

Falscher Sprung am Ende der Initialisierung von SYS0 bei 4F3AH.

ZAP-Vorschlag:

SYS0/SYS, 4F3BH, 0CH, 54H, von B4H nach B5H

6) SYS1 aus NEWDOS/80 Version 2 und GDOS

UP DOSCMD (4405H) und UP DOSCAL (4419H) werden wegen 4E4BH (Model III: 4E62H) in SYS1 überhaupt nicht ausgeführt, wenn am Anfang des DOS-Buffers bei 4318H (Model III: 4225H) zufällig 0DH steht.

Entweder sorgt man jeweils vor dem Aufruf von DOSCMD bzw. DOSCAL durch einen entsprechenden POKE 4318H,00 (Model III: 4225H,00) für Abhilfe oder macht folgenden ZAP (dann kommt auch jedesmal "BAD FILESPEC", wenn man bei der Eingabe von DOS-Befehlen stattdessen nur ENTER drückt).

ZAP-Vorschlag:

a) NEWDOS/80 Version 2 für TRS-80 Model I und
GDOS für Genie I,II,IIs,III,IIIs:

SYS1/SYS, 4E4BH, 01H, 53H, von C8H nach 00H

b) NEWDOS/80 Version 2 für TRS-80 Model III:

SYS1/SYS, 4E62H, 01H, 6AH, von C8H nach 00H

7) SYS6 aus NEWDOS/80 Version 2 und GDOS

In der Fehlerbehandlungs-Routine von SYS6 bei 523FH ist ein Byte falsch. Hier muß RST 28H aufgerufen werden und nicht RST 18H.

ZAP-Vorschlag:
NEWDOS/80 Version 2 für TRS-80 Model I+III,
GDOS für Genie I,II,IIs,III,IIIs:

SYS6/SYS, 523FH, 04H, DDH, von DFH nach EFH

8) SYS8 aus NEWDOS/80 Version 2

Falscher Sprung im DOS-Befehl "DIR $" bei 4FACH. Wenn man zum Wechseln der Disketten aus Versehen eine andere Taste als ENTER drückt, hängt sich das DOS auf.

ZAP-Vorschlag:
NEWDOS/80 Version 2 für TRS-80 Model I+III:

SYS8/SYS, 4FADH, 02H, B9H, von FCH nach F9H

```
*******************************
* Kapitel 8: Hardwarezugriffe *
*******************************
```

Betriebssystem: NEWDOS/80 Version 2 für TRS-80 Model I

Dieses Kapitel enthält alle bekannten Stellen im DOS, die bei Hardware-Änderungen überprüft und ggf. geändert werden müssen.

8.1 Tastaturverwaltung

SYS0/SYS:	4545-455A	Abfrage der Tastatur-Matrix
	457B-457C	Tasten-Entprellung
	4599	Groß/Kleinschrift ohne Umlaute
	45A0-45A5	SHIFT-Tasten
	45B9	Groß/Kleinschrift ohne Umlaute
	45C7-45CB	'DFG'
	45D3-45D7	'123'
	45E8-45EC	'JKL'
	4E35-4E39	'Hochpfeil'
	4E3C-4E41	neue Tastatur-Routine
	4F2A-4F2D	'ENTER'
SYS1/SYS	518E	Groß/Kleinschrift ohne Umlaute
	5193	Groß/Kleinschrift ohne Umlaute
SYS3/SYS:	5185-5189	'BREAK' für DOS-Befehl JKL
SYS6/SYS:	573B-5749	'Hochpfeil', 'Rechtspfeil', 'ENTER' für DOS-Befehle COPY und FORMAT
SYS8/SYS:	507B-5082	'ENTER' und 'BREAK' für DIR-Befehl
SYS9/SYS:	4E67-4E6B	'Rechtspfeil' für CHAINING
	4EDC-4EE0	'Hochpfeil' für CHAINING
	4F15-4F19	'ENTER' für CHAINING
SYS12/SYS:	4DB6-4DC8	'Hochpfeil', 'BREAK' und 'ENTER' für BASIC-Befehl REF
SYS14/SYS:	5115-512C	'Hochpfeil', 'Rechtspfeil', 'ENTER' für DOS-BEFEHLE LIST und PRINT
BASIC/CMD:	57BE	Groß/Kleinschrift ohne Umlaute

8.2 Bildschirmverwaltung

SYS0/SYS:	4020-4021	Cursor-Position
	4079	HEX-Ausgabe auf Bildschirm-RAM durch DEBUG
	44A4	Uhrzeit auf Bildschirm anzeigen
	44FC-4503	(blinkendes) Cursorzeichen
	4E42-4E47	neue Bildschirm-Routine
	4E48-4E50	Test, ob Kleinbuchstaben möglich
SYS3/SYS:	5182-519D	DOS-Befehl 'JKL'
SYS5/SYS:		DEBUG
SYS6/SYS:	571B-5728	"*" (blinkend) bei COPY
SYS11/SYS:	4F70-4F76	Berechnung der Cursor-Spalten#
	4F7A-4F84	Dezimalausgabe auf Bildschirm-RAM für BASIC-Befehl RENUM
SYS12/SYS:	51A1-51B1	Berechnung der Cursor-Spalten# für BASIC-Befehl REF
SYS18/SYS:	5394-53A6	Berechnung der Cursor-Zeilen# und Bildschirm-RAM auslesen
BASIC/CMD:	5D2A-5D30	Dezimalausgabe auf Bildschirm-RAM für RENUM, REF und CMD"F-SS"
	63F9-6416	CMD"F-SS": Anzeige der Zeilen#
	642E-6436	CMD"F-SS": Rest der Zeile löschen

8.3 Diskettenbetrieb

SYS0/SYS:	4309	Bitmuster für Drive-Select
	4630-4744	Sector I/O
	4745-476D	FDC-Kommandos und Status
	4776-4808	Drive-Select, Diskette eingelegt ?
	4DFD-4E01	Data Adress Marker für SD
SYS6/SYS:	64BA-65B9	BOOT-Sector
	68C4-68CB	SEEK-Kommando
	6A73-6A86	Side-Select
	6A90-6C1A	Formatieren inkl. Verify
	6CBD-6CC1	STEP-IN-Kommando
SYS9/SYS:	4DC6-4DC9	DOS-Befehl BOOT

8.4 Schreibversuche ins BASIC-ROM

Folgende Stellen mißbrauchen das BASIC-ROM als Disk-Buffer, wenn entweder Sectoren auf ihre fehlerfreie Lesefähigkeit geprüft (Verify) oder Kommentar-Blocks in LOAD-Files überlesen werden sollen:

BOOT/SYS:	4234-4241	ROM-Adressen 0000-01FE + 0300-20FE für Kommentar-Blocks in SYS0/SYS
SYS0/SYS:	4636	ROM-Adressen 0100-02FE für Verify in UP TESTS (4634H)
	4C53-4C62	ROM-Adressen 0000-01FE + 0300-20FE für Kommentar-Blocks in LOAD-Files
SYS6/SYS:	6C07-6C0C	ROM-Adressen 0000-00FF für Verify nach dem Formatieren
SYS19/SYS:	5616	ROM-Adressen 0000-01FE für Verify nach BASIC-Befehl SAVE (IX steht auf FCB-0DH !)

8.5 Sonstiges

SYS0/SYS:	45F6-45FD	Interrupt-Status
	47E4	CPU-Speed: 6 * SYSTEM BJ
	47F4-47F5	CPU-Speed: 2400H * SYSTEM BJ
	4CEF	CPU-Speed: 1 * SYSTEM BJ
	4D01	Interrupt-Modus
	4D03-4D0F	Test auf vorhandenes RAM
	4D93	CPU-Speed: SYSTEM BJ * 6
	4D95-4D96	CPU-Speed: SYSTEM BJ * 2400H

Kapitel 9: Andere Betriebssysteme

9.1 NEWDOS/80 Version 2 für TRS-80 Model III

9.1.1 Initialisierung des DOS

Als erstes wird aus dem BASIC-ROM der sogenannte Urlader aufgerufen, der den BOOT-Sector von Diskette (Drive 0, Track 0, Sector 1) in Double Density ins RAM ab 4300H liest und startet.

> Der Urlader steht in den verschiedenen Versionen des BASIC-ROMS (Deutsche und Amerikanische Version) an unterschiedlichen Adressen, arbeitet jedoch völlig identisch. Nachfolgend sind beide Versionen ausgedruckt.

Das Programm im BOOT-Sector bewirkt nun, daß SYS0/SYS, welches ab Track 0, Sector 5 auf Diskette stehen muß, ins RAM zwischen 400CH und 51EBH geladen und bei 4D01H gestartet wird.

Das Programm im BOOT-Sector kann in Abhängigkeit der folgenden PDRIVE-Parameter für Drive 0 leicht variieren:

- 8 Zoll oder 5.25 Zoll
- Single oder Double Density
- Single oder Double Sided
- Anzahl Sectoren / Track
- Track# ab 0 oder ab 1
- Sector# ab 0 oder ab 1

Nachfolgend wird der BOOT-Sector für die Standardversion (5.25 Zoll, Double Density, Single Sided, 18 Sectoren pro Track) beschrieben. Die Stellen, wo sich sonst Abweichungen ergeben können, sind durch (*) gekennzeichnet.

Für das Format von SYS0 und dessen Initialisierung gilt das bereits in Kapitel 2 gesagte.

9.1.1.1 Listing des Urladers (Deutsche Version)

```
3473 DB F0        IN      F0          ;Disk angeschlossen ?
3475 3C           INC     A
3476 CA 75 00     JP      Z,0075      ;nein, BASIC-Initialisierung
                  ----------------------
                  Track 00 gefunden ?
3479 01 00 00     LD      BC,0000     ;Schleifenzähler mit 65536D laden
347C 0B           DEC     BC          ;Zähler-1
347D 3E 81        LD      A,81        ;Disk-Motor on (Drive 0, Side 0)
347F D3 F4        OUT     F4
3481 78           LD      A,B         ;Zähler =0 ?
3482 B1           OR      C
3483 CA 75 00     JP      Z,0075      ;ja, BASIC-Initialisierung
3486 DB F0        IN      F0          ;nein, Disk-Status lesen
3488 CB 57        BIT     2,A         ;Track 00 gefunden ?
348A 28 F0        JR      Z,347C      ;nein, weiter warten
                  ----------------------
                  Disk-Initialisierung
348C 1E 05        LD      E,05        ;Schleifenzähler1 =5
348E 01 00 00     LD      BC,0000     ;Schleifenzähler2 =65536D
3491 DB F0        IN      F0          ;Disk-Status lesen
3493 CB 4F        BIT     1,A         ;Index-Marker ?
3495 20 11        JR      NZ,34A8     ;ja
3497 0B           DEC     BC          ;nein, Zähler2 -1
3498 3E 81        LD      A,81        ;Disk-Motor on (Drive 0, Side 0)
349A D3 F4        OUT     F4
349C 78           LD      A,B         ;Zähler2 =0 ?
349D B1           OR      C
349E 20 F1        JR      NZ,3491     ;nein, weiter warten
34A0 21 77 02     LD      HL,0277     ;ja, Zeiger auf Text 'Diskette?'
34A3 CD 1B 02     CALL    021B        ;Text ausgeben
34A6 18 E4        JR      348C        ;weiter warten
34A8 1D           DEC     E           ;wenn Index-Marker, Zähler1 -1
34A9 20 E3        JR      NZ,348E     ;wenn >0, weiter warten
34AB 3E 81        LD      A,81        ;Disk-Motor on (Drive 0, Side 0)
34AD D3 F4        OUT     F4
34AF 21 E3 34     LD      HL,34E3     ;NMI (nicht maskierbarer Interrupt)
34B2 22 4A 40     LD      (404A),HL   ;auf 34E3 verlegen
34B5 3E C3        LD      A,C3
34B7 32 49 40     LD      (4049),A
34BA 3E 80        LD      A,80        ;erlaubt Auslösen eines NMI
34BC D3 E4        OUT     E4          ;durch Disk-INTRQ
34BE 01 F3 00     LD      BC,00F3     ;Zähler B auf 256D, Register C
                                      ;auf Port F3 (FDC-Datenregister)
34C1 21 00 43     LD      HL,4300     ;Zeiger auf Buffer für BOOT-Sector
34C4 3E 01        LD      A,01        ;Sector 1 anwählen
34C6 D3 F2        OUT     F2
34C8 3E 80        LD      A,80        ;READ-SECTOR Befehl an
34CA D3 F0        OUT     F0          ;Disk-Controller
34CC CD 06 31     CALL    3106        ;ca. 26 us Verzögerung
34CF DB F0        IN      F0          ;Disk-Status lesen
34D1 E6 02        AND     02          ;DRQ (Data Request) ?
34D3 CA CF 34     JP      Z,34CF      ;nein, warten
34D6 ED A2        INI                 ;Datenbyte lesen, Zeiger +1, Zähler -1
34D8 3E 81        LD      A,81        ;Disk-Motor on (Drive 0, Side 0)
34DA F6 40        OR      40          ;WAIT-Status auslösen
34DC D3 F4        OUT     F4
```

```
34DE ED A2        INI                 ;Datenbyte lesen, Zeiger +1, Zähler -1
34E0 C3 D8 34     JP       34D8       ;weiter
                  ----------------------
                  NMI-Service-Routine
34E3 AF           XOR      A          ;verbietet Auslösen eines NMI
34E4 D3 E4        OUT      E4         ;durch Disk-INTRQ
34E6 21 ED 45     LD       HL,45ED    ;NMI auf Befehl 'RETN' verlegen
34E9 22 49 40     LD       (4049),HL
34EC CD 06 31     CALL     3106       ;ca. 26 us Verzögerung
34EF DB F0        IN       F0         ;Disk-Status lesen
34F1 E1           POP      HL         ;RETURN-Adresse nach HL
34F2 E6 1C        AND      1C         ;Fehler-Bits maskieren
34F4 CA 00 43     JP       Z,4300     ;wenn kein Fehler, BOOT-Sector starten
34F7 18 B2        JR       34AB       ;sonst nochmals versuchen
```

9.1.1.2 Listing des Urladers (Amerikanische Version)

```
3492 DB F0        IN       F0         ;Disk angeschlossen ?
3494 3C           INC      A
3495 CA AF 37     JP       Z,37AF     ;nein, BASIC-Initialisierung
                  ----------------------
                  Track 00 gefunden ?
3498 01 00 00     LD       BC,0000    ;Schleifenzähler mit 65536D laden
349B 0B           DEC      BC         ;Zähler -1
349C 3E 81        LD       A,81       ;Disk-Motor on (Drive 0, Side 0)
349E D3 F4        OUT      F4
34A0 78           LD       A,B        ;Zähler =0 ?
34A1 B1           OR       C
34A2 CA AF 37     JP       Z,37AF     ;ja, BASIC-Initialisierung
34A5 DB F0        IN       F0         ;nein, Disk-Status lesen
34A7 CB 57        BIT      2,A        ;Track 00 gefunden ?
34A9 28 F0        JR       Z,349B     ;nein, weiter warten
                  ----------------------
                  Disk-Initialisierung
34AB 1E 05        LD       E,05       ;Schleifenzähler1 =5
34AD 01 00 00     LD       BC,0000    ;Schleifenzähler2 =65536D
34B0 DB F0        IN       F0         ;Disk-Status lesen
34B2 CB 4F        BIT      1,A        ;Index-Marker ?
34B4 20 11        JR       NZ,34C7    ;ja
34B6 0B           DEC      BC         ;nein, Zähler2 -1
34B7 3E 81        LD       A,81       ;Disk-Motor on (Drive 0, Side 0)
34B9 D3 F4        OUT      F4
34BB 78           LD       A,B        ;Zähler2 =0 ?
34BC B1           OR       C
34BD 20 F1        JR       NZ,34B0    ;nein, weiter warten
34BF 21 77 02     LD       HL,0277    ;ja, Zeiger auf Text 'Diskette?'
34C2 CD 1B 02     CALL     021B       ;Text ausgeben
34C5 18 E4        JR       34AB       ;weiter warten
34C7 1D           DEC      E          ;wenn Index-Marker, Zähler1 -1
34C8 20 E3        JR       NZ,34AD    ;wenn >0, weiter warten
34CA 3E 81        LD       A,81       ;Disk-Motor on (Drive 0, Side 0)
34CC D3 F4        OUT      F4
34CE 21 02 35     LD       HL,3502    ;NMI (nicht maskierbarer Interrupt)
34D1 22 4A 40     LD       (404A),HL  ;auf 3502 verlegen
34D4 3E C3        LD       A,C3
34D6 32 49 40     LD       (4049),A
```

```
34D9 3E 80        LD      A,80          ;erlaubt Auslösen eines NMI
34DB D3 E4        OUT     E4            ;durch Disk-INTRQ
34DD 01 F3 00     LD      BC,00F3       ;Zähler B auf 256D, Register C
                                        ;auf Port F3 (FDC-Datenregister)
34E0 21 00 43     LD      HL,4300       ;Zeiger auf Buffer für BOOT-Sector
34E3 3E 01        LD      A,01          ;Sector 1 anwählen
34E5 D3 F2        OUT     F2
34E7 3E 80        LD      A,80          ;READ-SECTOR Befehl an
34E9 D3 F0        OUT     F0            ;Disk-Controller
34EB CD 18 35     CALL    3518          ;ca. 26 us Verzögerung
34EE DB F0        IN      F0            ;Disk-Status lesen
34F0 E6 02        AND     02            ;DRQ (Data Request) ?
34F2 CA EE 34     JP      Z,34EE        ;nein, warten
34F5 ED A2        INI                   ;Datenbyte lesen, Zeiger +1, Zähler -1
34F7 3E 81        LD      A,81          ;Disk-Motor on (Drive 0, Side 0)
34F9 F6 40        OR      40            ;WAIT-Status auslösen
34FB D3 F4        OUT     F4
34FD ED A2        INI                   ;Datenbyte lesen, Zeiger +1, Zähler -1
34FF C3 F7 34     JP      34F7          ;weiter
                  ----------------------
                  NMI-Service-Routine
3502 AF           XOR     A             ;verbietet Auslösen eines NMI
3503 D3 E4        OUT     E4            ;durch Disk-INTRQ
3505 21 ED 45     LD      HL,45ED       ;NMI auf Befehl 'RETN' verlegen
3508 22 49 40     LD      (4049),HL
350B CD 18 35     CALL    3518          ;ca. 26 us Verzögerung
350E DB F0        IN      F0            ;Disk-Status lesen
3510 E1           POP     HL            ;RETURN-Adresse nach HL
3511 E6 1C        AND     1C            ;Fehler-Bits maskieren
3513 CA 00 43     JP      Z,4300        ;wenn kein Fehler, BOOT-Sector starten
3516 18 B2        JR      34CA          ;sonst nochmals versuchen
```

9.1.1.3 Listing des BOOT-Sectors

```
4300 00           ;./.
4301 FE           ;./.
4302*11           ;LUMP#, in welcher der Directory beginnt
4303 F3           DI                    ;Interrupts sperren
4304 11*05*00     LD      DE,0005       ;D=Track# (TI=J oder TI=K => D=01)
                                        ;E=Sector# (TI=I oder TI=M => E=06)
4307 D9           EXX                   ;gewünschte Track# und Sector# nach DE'
4308 31 E0 41     LD      SP,41E0       ;Stackpointer auf 41E0
430B 21 FF 51     LD      HL,51FF       ;Zeiger auf Ende des Sector-Buffers
                  ----------------------
                  SYS0 einlesen
430E CD 45 43     CALL    4345          ;Steuer-Code lesen
4311 FE 20        CP      20            ;größer als 1FH ?
4313 47           LD      B,A           ;B = 1. Byte
4314 30 29        JR      NC,433F       ;wenn ja, Error
4316 57           LD      D,A           ;D = 1. Byte
4317 CD 45 43     CALL    4345          ;nächstes Byte lesen
431A 4F           LD      C,A           ;C = 2. Byte
431B CD 45 43     CALL    4345          ;nächstes Byte lesen
431E 5F           LD      E,A           ;E = 3. Byte
431F 10 12        DJNZ    4333          ;wenn Steuer-Code <> 01H, dann 4333
                  ----------------------
```

```
                    Steuer-Code 01H:
4321 CD 45 43       CALL    4345        ;4. Byte nach D lesen,
4324 57             LD      D,A         ;DE zeigt auf zu ladenden Speicher
4325 0D             DEC     C           ;Längen-Byte
4326 0D             DEC     C           ;um 2 vermindern
4327 2C             INC     L           ;Zeiger auf Buffer +1, Buffer zu Ende ?
4328 CC 48 43       CALL    Z,4348      ;wenn ja: nächsten Sector lesen
432B 7E             LD      A,(HL)      ;nächstes Byte aus Buffer holen
432C 12             LD      (DE),A      ;im Speicher ablegen
432D 13             INC     DE          ;Zeiger auf Speicher +1
432E 0D             DEC     C           ;Längen-Byte -1, Block zu Ende ?
432F 20 F6          JR      NZ,4327     ;wenn nein
4331 18 DB          JR      430E        ;wenn ja: nächsten Block bearbeiten
                    ----------------------
4333 10 F9          DJNZ    432E        ;wenn Steuer-Code <> 02H, dann 432E:
                                        ;D hat einen Wert zwischen 00 und 1F,
                                        ;somit werden Kommentare über das
                                        ;BASIC-ROM geladen!
                    ----------------------
                    Steuer-Code 02H:
4335 CD 45 43       CALL    4345        ;4. Byte nach D lesen,
4338 57             LD      D,A         ;DE = Startadresse
4339 1A             LD      A,(DE)      ;ist dort die Kennung
433A FE A5          CP      A5          ;für NEWDOS/80 vorhanden ?
433C 13             INC     DE          ;Startadresse +1
433D D5             PUSH    DE          ;und in den Stack
433E C8             RET     Z           ;wenn ja, starten !
                    ----------------------
                    Fehlermeldung
433F 21 F0 43       LD      HL,43F0     ;Text CLS + "NO SYS"
4342 C3 CC 43       JP      43CC        ;auf Bildschirm ausgeben
```

```
*********************************************************
* Name:       BYREAD                                    *
* Funktion:   nächstes Byte von SYS0 holen              *
* Input:      HL: zeigt auf nächstes Byte im Buffer     *
*             D': zeigt auf nächste Track#              *
*             E': zeigt auf nächste Sector#             *
* Verändert:  HL=HL+1, ggf. DE' erhöhen, HL', B', F     *
* Output:     A: nächstes Byte von SYS0                 *
*********************************************************
```

Wenn beim Lesen eines neuen Sectors ein Disk-Error auftritt, wird eine Meldung ausgegeben und BYREAD kehrt nicht zurück (Endlos-Schleife).

```
4345 2C             INC     L           ;Zeiger auf Buffer +1
4346 7E             LD      A,(HL)      ;nächstes Byte lesen
4347 C0             RET     NZ          ;wenn Buffer nicht zu Ende: RET
4348 D9             EXX                 ;Zweit-Registersatz einschalten
4349 06 0A          LD      B,0A        ;Zähler für max. 10D Versuche
434B 2E*01          LD      L,81        ;Drive 0, Seite 0, SD oder DD
434D D5             PUSH    DE          ;DE retten (Track# und Sector#)
434E C5             PUSH    BC          ;B retten (Zähler Anzahl Versuche)
                    ----------------------
434F 7B             LD      A,E         ;bei doppelseitigen Laufwerken:
4350 D6*12          SUB     12          ;wenn "Anzahl der Sectoren / Seite"
4352 38 03          JR      C,4357      ;kleiner als E ist:
4354 5F             LD      E,A         ;Sector# für die Rückseite berechnen
4355 2E*11          LD      L,91        ;und Seite 1 (SD oder DD) wählen
                    ----------------------
```

```
4357 CD D7 43     CALL    43D7          ;warten, bis FDC nicht mehr busy ist
435A 7D           LD      A,L           ;Drive 0, Single oder Double Density,
435B D3 F4        OUT     F4            ;Seite 0 oder 1 anwählen
435D 7A           LD      A,D           ;Track# an FDC senden
435E D3 F3        OUT     F3
4360 7B           LD      A,E           ;Sector# an FDC senden
4361 D3 F2        OUT     F2
4363 3E 1B        LD      A,1B          ;SEEK-Kommando an FDC senden
4365 D3 F0        OUT     F0
4367 CD D7 43     CALL    43D7          ;warten, bis FDC nicht mehr busy ist
436A 7D           LD      A,L           ;im Drive-Status
436B F6 40        OR      40            ;das Bit für CPU-WAIT setzen
436D 5F           LD      E,A           ;und in E merken
436E 16 02        LD      D,02          ;D = Maske für DRQ (Data Request)
4370 DB F0        IN      F0            ;FDC-Status lesen, wozu ?
4372 ED 73 A6 43  LD      (43A6),SP     ;SP nach 43A6 retten
4376 21 A2 43     LD      HL,43A2       ;NMI (nicht maskierbarer Interrupt)
4379 22 4A 40     LD      (404A),HL     ;auf 43A2 verlegen
437C 3E C3        LD      A,C3
437E 32 49 40     LD      (4049),A
4381 3E C0        LD      A,C0          ;Auslösen eines NMI durch
4383 D3 E4        OUT     E4            ;Disk-INTRQ erlauben
4385 3E 88        LD      A,88          ;READ-Kommando an FDC senden
4387 D3 F0        OUT     F0
4389 21 00 51     LD      HL,5100       ;Zeiger auf Buffer für nächsten Sector
438C 01 F3 00     LD      BC,00F3       ;Zähler B auf 256D, Register C
                                        ;auf Port F3 (FDC-Datenregister)
438F CD E2 43     CALL    43E2          ;ca. 69 us warten
4392 DB F0        IN      F0            ;FDC-Status lesen
4394 A2           AND     D             ;Data Request ?
4395 28 FB        JR      Z,4392        ;wenn nein, warten
4397 ED A2        INI                   ;1. Byte vom FDC holen
4399 7B           LD      A,E           ;CPU-WAIT-Status auslösen, der
439A D3 F4        OUT     F4            ;durch DRQ wieder gelöscht wird
439C ED A2        INI                   ;nächstes Byte vom FDC holen
439E 20 FA        JR      NZ,439A       ;wenn noch kein Sector-Ende
43A0 18 FE        JR      43A0          ;auf NMI warten
                  ----------------------
                  NMI-Service-Routine
43A2 AF           XOR     A             ;verbietet Auslösen eines NMI
43A3 D3 E4        OUT     E4            ;durch Disk-INTRQ
43A5 31*00*00     LD      SP,0000       ;SP zurück
43A8 DB F0        IN      F0            ;FDC-Status lesen
43AA E6 FC        AND     FC            ;Fehler-Bits maskieren
43AC 3E D0        LD      A,D0          ;FORCE-INTERRUPT-Kommando
43AE D3 F0        OUT     F0            ;an FDC senden
43B0 C1           POP     BC            ;Zähler für Anzahl Versuche zurück
43B1 D1           POP     DE            ;Track# und Sector# zurück
43B2 20 0C        JR      NZ,43C0       ;wenn Error
                  ----------------------
43B4 1C           INC     E             ;Sector# +1
43B5 7B           LD      A,E           ;ist die nächste Sector# größer als die
43B6 D6*12        SUB     12            ;"Anzahl Sectoren pro Track" ?
43B8 20 03        JR      NZ,43BD       ;wenn nein
43BA 14           INC     D             ;Track# +1
43BB 1E*00        LD      E,00          ;E = 1. Sector# im Track (TI=I/M => E=1)
                  ----------------------
43BD D9           EXX                   ;Zweit-Registersatz zurück
43BE 7E           LD      A,(HL)        ;1. Byte aus dem Buffer holen
43BF C9           RET                   ;fertig
```

```
                    ----------------------
                    Fehlerbehandlung nach Sector lesen
43C0 CD E2 43       CALL    43E2        ;69 us warten
43C3 3E 0B          LD      A,0B        ;RESTORE-Kommando an FDC
43C5 D3 F0          OUT     F0
43C7 10 82          DJNZ    434B        ;wenn noch keine 10D Versuche gemacht
43C9 21 E8 43       LD      HL,43E8     ;Text CLS + "ERROR"
                    ----------------------
                    Text (HL) anzeigen und Endlos-Schleife
43CC 7E             LD      A,(HL)      ;nächstes Zeichen holen
43CD FE 03          CP      03          ;Ende-Markierung ?
43CF 28 FB          JR      Z,43CC      ;wenn ja
43D1 23             INC     HL          ;Zeiger +1
43D2 CD 33 00       CALL    0033        ;Zeichen auf Bildschirm ausgeben
43D5 18 F5          JR      43CC        ;weiter
                    ----------------------
                    Warten, bis FDC nicht mehr busy ist
43D7 CD E2 43       CALL    43E2        ;69 us warten
43DA DB F0          IN      F0          ;ist FDC noch busy ?
43DC 0F             RRCA
43DD 38 FB          JR      C,43DA      ;wenn ja
43DF DB F0          IN      F0          ;Status-Register lesen
43E1 C9             RET                 ;fertig
                    ----------------------
                    Verzögerung 69 us (bei 2.02752 MHz)
43E2 3E 08          LD      A,08        ;Zähler setzen
43E4 3D             DEC     A           ;Zähler -1
43E5 20 FD          JR      NZ,43E4     ;wenn <> 0
43E7 C9             RET                 ;fertig
                    ----------------------
                    Text CLS + "ERROR"
43E8 1C 1F 45 52 52 4F 52 03                        ..ERROR.
                    ----------------------
                    Text CLS + "NO SYS"
43F0 1C 1F 4E 4F 20 53 59 53 03                     ..NO SYS.
                    ----------------------
43F9 00 00 00 00    ;./.
                    ----------------------
                    Wichtige PDRIVE-Parameter für Drive 0
                    (werden von SYS0 zum Lesen des PDRIVE-Sectors benötigt)
                    Erläuterungen siehe Kapitel 4.2
43FD*FE             ;PDRIVE+6
43FE*05             ;PDRIVE+2
43FF*01             ;PDRIVE+7
```

9.1.2 Unterschiede in SYS0/SYS

Die Unterprogramme in SYS0 für Model III sind zwar gegenüber denen für Model I im Adressbereich verschoben, ansonsten aber so gut wie identisch.

Eine Ausnahme bilden aufgrund der unterschiedlichen Hardware nur die Unterprogramme zum Lesen und Schreiben von Sectoren, sie sind daher hier noch einmal komplett disassembliert und kommentiert ausgedruckt.

9.1.2.1 Lesen und Schreiben von Sectoren

```
*************************************************************
* Name:       READS                                         *
* Funktion:   liest einen Sector von Diskette               *
* Input:      DE: gewünschte Sector# (Disk-relativ)         *
*             HL: Zeiger auf zu benutzenden Buffer          *
*             (427EH): gewünschte Drive# (0-3)              *
* Verändert:  --                                            *
* Output:     AF: A=Fehlercode, wenn Z=0                    *
*************************************************************
```

READS liest einen physikalischen Sector von Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 427EH die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
45DB 3E 88        LD      A,88          ;READ-SECTOR-Kommando für FDC
45DD 18 0E        JR      45ED          ;weiter bei 45ED
```

```
*************************************************************
* Name:       TESTS                                         *
* Funktion:   testet einen Sector auf Fehler (Verify)       *
* Input:      DE: gewünschte Sector# (Disk-relativ)         *
*             (427EH): gewünschte Drive# (0-3)              *
* Verändert:  --                                            *
* Output:     AF: A=Fehlercode, wenn Z=0                    *
*************************************************************
```

TESTS testet einen physikalischen Sector auf Diskette, ob er ohne Fehler lesbar ist. Dazu wird versucht, diesen Sector von Diskette zu lesen, wobei als Buffer einfach das BASIC-ROM im Bereich 0100H-02FEH benutzt wird. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 427EH die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
45DF E5           PUSH    HL            ;HL retten
45E0 26 01        LD      H,01          ;Buffer = 01XX
45E2 CD DB 45     CALL    45DB          ;Sector lesen
45E5 E1           POP     HL            ;HL zurück
45E6 C9           RET
```

```
*********************************************************
* Name:       WRITDS                                    *
* Funktion:   schreibt einen Sector des Directory auf   *
*             Diskette                                  *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             HL: Zeiger auf zu benutzenden Buffer      *
*             (427EH): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

WRITDS schreibt einen physikalischen Sector mit dem Data Adress Mark für Directory-Sectoren auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 427EH die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
45E7 3E A9        LD      A,A9          ;WRITE-SECTOR-Kommando für FDC mit
                                        ;Data Adress Marker F8H für Directory
45E9 18 02        JR      45ED          ;weiter bei 45ED
```

```
*********************************************************
* Name:       WRITES                                    *
* Funktion:   schreibt einen Sector auf Diskette        *
* Input:      DE: gewünschte Sector# (Disk-relativ)     *
*             HL: Zeiger auf zu benutzenden Buffer      *
*             (427EH): gewünschte Drive# (0-3)          *
* Verändert:  --                                        *
* Output:     AF: A=Fehlercode, wenn Z=0                *
*********************************************************
```

WRITES schreibt einen physikalischen Sector auf Diskette. DE gibt an, um den wievielten Sector innerhalb der Diskette (beginnend ab 0) es sich dabei handelt. Zuvor muß bei 427EH die gewünschte Drive# eingetragen worden sein, z. B. durch DRVSEL (445BH) oder TSTDSK (445EH).

```
45EB 3E A8        LD      A,A8          ;WRITE-SECTOR-Kommando für FDC mit
                                        ;"normalem" Data Adress Marker FBH.
                  ----------------------
45ED 32 87 46     LD      (4687),A      ;FDC-Kommando nach 4687
45F0 E6 20        AND     20            ;READ oder WRITE ?
45F2 C5           PUSH    BC            ;BC retten
45F3 06 A2        LD      B,A2          ;"INI"
45F5 28 03        JR      Z,45FA        ;wenn READ
45F7 04           INC     B             ;"OUTI"
45F8 3E 08        LD      A,08          ;Offset für Fehlercode bei WRITE = 08H
45FA 32 E1 46     LD      (46E1),A      ;Offset für Fehlercode (READ=0/WRITE=8)
45FD 78           LD      A,B           ; READ / WRITE:
45FE 32 AC 46     LD      (46AC),A      ;"INI" / "OUTI" nach 46AC
4601 32 B1 46     LD      (46B1),A      ;"INI" / "OUTI" nach 46B1
4604 06*0A        LD      B,0A          ;SYSTEM AM: Anzahl Versuche bei Errors
4606 CD 20 47     CALL    4720          ;Select Drive (427EH) und Motor starten
4609 20 18        JR      NZ,4623       ;wenn Error
460B C5           PUSH    BC            ;B retten (Zähler für Anzahl Versuche)
460C D5           PUSH    DE            ;DE retten (Disk-relative Sector#)
460D E5           PUSH    HL            ;HL retten (Zeiger auf Buffer)
460E 3A 84 42     LD      A,(4284)      ;A = PDRIVE+4: SPT (Sectors pro Track)
4611 EB           EX      DE,HL         ;=> HL = gewünschte Sector#
4612 CD 59 4C     CALL    4C59          ;1) C=A (Sectors pro Track)
```

```
                                        ;2) DIV: HL=INT(HL/A), A=Rest
4615 55          LD      D,L            ;D = gesuchte Track#
4616 5F          LD      E,A            ;E = gesuchte Sector#
4617 21 83 42    LD      HL,4283        ;HL zeigt auf PDRIVE+3: TC (Track Count)
461A 7A          LD      A,D            ;ist die berechnete Track# größer als
461B BE          CP      (HL)           ;die Anzahl der vorhandenen Tracks ?
461C 38 09       JR      C,4627         ;wenn nein
                 ----------------------
461E 3E 14       LD      A,14           ;Fehlercode "TRACK # TOO HIGH"
4620 E1          POP     HL             ;HL zurück (Zeiger auf Buffer)
4621 D1          POP     DE             ;DE zurück (Disk-relative Sector#)
4622 C1          POP     BC             ;B zurück (Zähler für Anzahl Versuche)
4623 B7          OR      A              ;Z-Flag löschen, A=Fehlercode
4624 C3 E7 46    JP      46E7           ;weiter bei 46E7
                 ----------------------
4627 3A 87 42    LD      A,(4287)       ;PDRIVE+7: TI und TD
462A 47          LD      B,A            ;nach B
462B CB 40       BIT     0,B            ;Double Density (DD) ?
462D 21 7F 42    LD      HL,427F        ;Zeiger auf Bit-Maske für aktuelle Drive
4630 28 02       JR      Z,4634         ;wenn nein
4632 CB FE       SET     7,(HL)         ;wenn ja: DD vermerken
                 ----------------------
4634 3A 83 42    LD      A,(4283)       ;PDRIVE +3: Track Count
4637 0F          RRCA                   ;liegt die gewünschte Track# in
4638 BA          CP      D              ;der inneren Hälfte der Diskette ?
4639 30 02       JR      NC,463D        ;wenn nein
463B CB EE       SET     5,(HL)         ;Write-Precompensation enable
                 ----------------------
463D CB 48       BIT     1,B            ;TI=J oder TI=K (Track# ab 1) ?
463F 28 01       JR      Z,4642         ;wenn nein
4641 14          INC     D              ;wenn ja: Track# +1
                 ----------------------
4642 D5          PUSH    DE             ;D retten (gewünschte Track#)
4643 CB 50       BIT     2,B            ;TI=L (doppelte Step-Impulse) ?
4645 28 02       JR      Z,4649         ;wenn nein
4647 CB 02       RLC     D              ;Pseudo-Track# für SEEK verdoppeln
                 ----------------------
4649 CB 70       BIT     6,B            ;Diskette doppelseitig (DS) ?
464B 28 09       JR      Z,4656         ;wenn nein
464D CB 09       RRC     C              ;=> C = Sectors pro Track pro Seite
464F 7B          LD      A,E            ;ist die gewünschte Sector#
4650 91          SUB     C              ;auf der Rückseite der Diskette ?
4651 38 03       JR      C,4656         ;wenn nein
4653 5F          LD      E,A            ;E = gesuchte Sector# auf der Rückseite
4654 CB E6       SET     4,(HL)         ;Bit für Rückseite (Seite 1) setzen
4656 CD 15 47    CALL    4715           ;gewünschte Disk-Seite anwählen
                 ----------------------
4659 CB 60       BIT     4,B            ;TI=I oder TI=M (Sector# ab 1) ?
465B 28 01       JR      Z,465E         ;wenn nein
465D 1C          INC     E              ;wenn ja: Sector# +1
                 ----------------------
465E 7B          LD      A,E            ;gewünschte Sector# an FDC senden
465F D3 F2       OUT     F2
4661 7A          LD      A,D            ;gewünschte SEEK-Track# an FDC senden
4662 D3 F3       OUT     F3
4664 0E 18       LD      C,18           ;SEEK-Kommando an FDC senden
4666 CD F7 46    CALL    46F7           ;und warten, bis FDC nicht mehr busy ist
4669 F1          POP     AF             ;A = gewünschte Track#
466A E1          POP     HL             ;HL = Zeiger auf Buffer
466B E5          PUSH    HL             ;HL retten (Zeiger auf Buffer)
```

```
466C D3 F1        OUT     F1          ;gewünschte Track# ins Track-Register
466E D5           PUSH    DE          ;D retten (Pseudo-Track# für SEEK)
466F F3           DI                  ;Interrupts sperren
4670 DB F0        IN      F0          ;FDC-Status lesen - wozu ?
4672 ED 73 BA 46  LD      (46BA),SP   ;SP nach 46BA retten
4676 11 B6 46     LD      DE,46B6     ;NMI (nicht maskierbarer Interrupt)
4679 ED 53 4A 40  LD      (404A),DE   ;auf 46B6 verlegen
467D 3E C3        LD      A,C3
467F 32 49 40     LD      (4049),A
4682 3E C0        LD      A,C0        ;Auslösen eines NMI durch
4684 D3 E4        OUT     E4          ;Disk-INTRQ erlauben
4686 3E*00        LD      A,00        ;Kommando (READ / WRITE)
4688 D3 F0        OUT     F0          ;an FDC senden
468A CD 89 47     CALL    4789        ;64 us warten und FDC-Status lesen
468D 01 F3 00     LD      BC,00F3     ;Zähler B auf 256D, Register C
                                      ;auf Port F3 (FDC-Datenregister)
4690 3A 7F 42     LD      A,(427F)    ;Bit-Maske für aktuelle Drive holen
4693 F6 40        OR      40          ;Bit für CPU-WAIT-Status setzen
4695 5F           LD      E,A         ;und in E merken
4696 16 02        LD      D,02        ;D = Maske für DRQ (Data Request)
4698 3A 87 46     LD      A,(4687)    ;ist das auszuführende Kommando
469B CB 6F        BIT     5,A         ;das READ-Kommando ?
469D 28 07        JR      Z,46A6      ;wenn ja
                  ----------------------
                  WRITE-SECTOR:
469F DB F0        IN      F0          ;FDC-Status lesen
46A1 A2           AND     D           ;Data Request ?
46A2 28 FB        JR      Z,469F      ;wenn nein, warten
46A4 ED A3        OUTI                ;1. Byte an FDC senden
                  ----------------------
                  READ + WRITE:
46A6 DB F0        IN      F0          ;FDC-Status lesen
46A8 A2           AND     D           ;Data Request ?
46A9 28 FB        JR      Z,46A6      ;wenn nein, warten
46AB ED*A2        INI                 ;READ: 1. Byte vom FDC holen
                                      ;WRITE: nächstes Byte an FDC senden
46AD 7B           LD      A,E         ;CPU-WAIT-Status auslösen, der
46AE D3 F4        OUT     F4          ;durch DRQ wieder gelöscht wird
46B0 ED*A2        INI                 ;READ: nächstes Byte vom FDC holen
                                      ;WRITE: nächstes Byte an FDC senden
46B2 20 FA        JR      NZ,46AE     ;wenn noch kein Sector-Ende
46B4 18 FE        JR      46B4        ;auf NMI warten
                  ----------------------
                  NMI-Service-Routine
46B6 AF           XOR     A           ;verbietet Auslösen eines NMI
46B7 D3 E4        OUT     E4          ;durch Disk-INTRQ
46B9 31*00*00     LD      SP,0000     ;SP zurück
46BC 21 ED 45     LD      HL,45ED     ;NMI auf Befehl 'RETN' verlegen
46BF 22 49 40     LD      (4049),HL
46C2 DB F0        IN      F0          ;FDC-Status lesen
46C4 47           LD      B,A         ;in B merken
46C5 3E D0        LD      A,D0        ;FORCE-INTERRUPT-Kommando
46C7 D3 F0        OUT     F0          ;an FDC senden
46C9 F1           POP     AF          ;A = Pseudo-Track# für SEEK
46CA D3 F1        OUT     F1          ;ins Track-Register des FDC schreiben
46CC 78           LD      A,B         ;FDC-Status nach A zurück
46CD E1           POP     HL          ;HL zurück (Zeiger auf Buffer)
46CE D1           POP     DE          ;DE zurück (Disk-relative Sector#)
46CF C1           POP     BC          ;B zurück (Zähler für Anzahl Versuche)
46D0 FB           EI                  ;Interrupts wieder erlauben
```

```
46D1 E6 FC        AND     FC          ;Error-Bits maskieren
46D3 28 12        JR      Z,46E7      ;wenn kein Error
                  ----------------------
                  Fehlerbehandlung
46D5 4F           LD      C,A         ;Error-Bits retten
46D6 E6 9C        AND     9C          ;wenn "Write Protected", "Write Fault"
46D8 28 06        JR      Z,46E0      ;oder "Data Adress Mark": abbrechen
46DA 4F           LD      C,A         ;maskierte Error-Bits
46DB 87           ADD     A           ;wenn "Drive not Ready":
46DC 28 02        JR      Z,46E0      ;abbrechen (kein weiterer Versuch)
46DE 10 0A        DJNZ    46EA        ;schon SYSTEM AM Versuche gemacht ?
                  ----------------------
                  Fehlercode berechnen
46E0 3E*00        LD      A,00        ;Offset für Fehlercode (READ=0, WRITE=8)
46E2 3C           INC     A           ;Fehlercode +1
46E3 CB 09        RRC     C           ;Error-Bit (C) nach rechts verschieben
46E5 30 FB        JR      NC,46E2     ;solange, bis Error-Bit im Carry ist
46E7 FB           EI                  ;Interrupts wieder erlauben
46E8 C1           POP     BC          ;BC zurück (ursprünglicher Wert)
46E9 C9           RET                 ;READS/TESTS/WRITDS/WRITES fertig
                  ----------------------
                  Fehlerbehandlung: neuen Versuch starten
46EA CB 61        BIT     4,C         ;"Record Not Found" Error ?
46EC C4 F2 46     CALL    NZ,46F2     ;wenn ja: alle 2 Versuche RESTORE an FDC
46EF C3 0B 46     JP      460B        ;nächster Versuch
```

9.1.2.2 Unterprogramme und Ansprungadressen

In Klammern sind jeweils die Adressen für Model I angegeben.

Nach Adressen sortiert:

0013	READB	(0013)	liest ein Byte aus einer File (siehe 4A98H)
001B	WRITEB	(001B)	schreibt ein Byte in eine File (siehe 4A3DH)
402D	DOSRDY	(402D)	Sprung nach DOS READY (ohne Rückkehr)
4030	ERROR0	(4030)	nach einem Fehler: Sprung nach DOS READY
4405	DOSCMD	(4405)	DOS-Befehl ausführen (ohne Rückkehr)
4409	DOSERR	(4409)	Fehlermeldung des DOS ausgeben
440D	DEBUG	(440D)	DEBUG aufrufen
4413	INTDEL	(4413)	Benutzer-Interrupt-Routine löschen
4416	MOTONX	(4416)	Drive-Motoren weiterlaufen lassen
4419	DOSCAL	(4419)	DOS-Befehl ausführen und zurück
441C	TFSPEC	(441C)	Filespec (HL) nach FCB (DE) übertragen
4420	INIT	(4420)	File öffnen, ggf. neue File anlegen
4424	OPEN	(4424)	File öffnen, ggf. keine neue File anlegen
4428	CLOSE	(4428)	File schließen
442C	KILL	(442C)	File löschen
4430	LOAD	(4430)	Programm laden
4433	RUN	(4433)	Programm laden und starten (keine Rückkehr)
4436	READ	(4436)	nächsten Sector/Record aus einer File lesen
4439	WRITE	(4439)	nächsten Sector/Record in eine File schreiben
443C	VERIFY	(443C)	Sector/Record in eine File schreiben + testen
443F	POS0	(443F)	FCB auf Beginn einer File positionieren
4442	POSBC	(4442)	FCB auf Logische Record# (BC) positionieren
4445	POSDEC	(4445)	FCB um 1 Record zurückpositionieren
4448	POSEOF	(4448)	FCB auf Ende der File (EOF) positionieren
444B	EXPAND	(444B)	File erweitern, wenn File zu kurz ist
444E	POSRBA	(444E)	FCB auf RBA-Format (HLC) positionieren
4451	PUTEOF	(4451)	EOF im FPDE auf den neuesten Stand bringen
445B	DRVSEL	(445B)	Drive auswählen und Motor starten
445E	TSTDSK	(445E)	Drive auswählen, prüfen ob Diskette eingelegt
4461	USRINS	(4461)	Benutzer-Routine (*name) einfügen
4464	USRDEL	(4464)	Benutzer-Routine (*name) löschen
4467	TEXTTV	(4467)	Text auf Bildschirm ausgeben
446A	TEXTLP	(446A)	Text auf Drucker ausgeben
446D	TIME	(446D)	Uhrzeit im Format HH:MM:SS in Buffer ablegen
4470	DATE	(4470)	Datum im Format MM/TT/JJ in Buffer ablegen
4473	INSEXT	(4473)	File-Typ (/EXT) in Filespec einfügen
447B	INTINS	(4410)	Benutzer-Interrupt-Routine einfügen
44D2	HEXDE	(4063)	DE hexadezimal nach (HL) ausgeben
44D7	HEXA	(4068)	A hexadezimal nach (HL) ausgeben
4548	UPCASE	(45B5)	Kleinbuchstaben in Großbuchstaben umwandeln
45DB	READS	(4630)	Sector von Diskette lesen
45DF	TESTS	(4634)	Sector auf Lesbarkeit testen
45E7	WRITDS	(463C)	Directory-Sector auf Diskette schreiben
45EB	WRITES	(4640)	normalen Sector auf Diskette schreiben
46F5	RESTOR	(4745)	RESTORE-Kommando an FDC senden
46F7	FDCCMD	(4747)	Kommando mit richtiger TSR an FDC senden
46FF	WNBUSY	(4750)	warten, bis der FDC nicht mehr busy ist
470D	FBREAK	(475E)	FORCE-INTERRUPT-Kommando an FDC senden

4715	MOTON	(4767)	Drive-Motoren starten
4789	DELAY1	(47E3)	ca. 64 us warten und Status des FDC lesen
47B5	FILPOS	(4810)	Disk-Position eines File-Sectors berechnen
48AF	DIRR	(490A)	liest einen Sector des Directory
48C4	DIRW	(491F)	schreibt einen Sector des Directory
48DB	GETFDE	(4936)	holt einen FDE aus dem Directory
490D	NXTEOF	(4968)	NEXT-Wert aus FCB holen + mit EOF vergleichen
4925	PUSHR	(4980)	Push Register, IX=FCB+0, IY=4280H, A=00
4A5D	WRITXV	(4AB8)	Sector oder DIR-Sector schreiben, ggf. Verify
4A62	WRITEV	(4ABD)	normalen Sector schreiben, ggf. Verify
4A6F	WRITDV	(4ACA)	Directory-Sector schreiben, ggf. Verify
4B6E	GETSYS	(4BC9)	SYS-File laden und starten
4BCD	LOADP	(4C28)	Programm laden
4C19	DIRPOS	(4C74)	Disk-Position eines DIR-Sectors berechnen
4C37	MULTL	(4C92)	Multipliziere AHL = L*A
4C39	MULTHL	(4C94)	Multipliziere AHL = HL*A
4C42	MULTC	(4C9D)	Multipliziere HLC = HL*A
4C57	DIV05	(4CB2)	Dividiere HL=INT(HL/5), A=Rest
4C59	DIVA	(4CB4)	Dividiere HL=INT(HL/A), A=Rest
4C6A	CPBCHL	(4CC5)	Vergleiche Text (BC) mit Text (HL)
4C7A	NEXTC1	(4CD5)	Nächstes Zeichen holen und Flags setzen
4C7E	NEXTC2	(4CD9)	Nächstes Zeichen holen und Flags setzen
4C92	DELAY2	(4CED)	ca. B * 3.79 ms warten

Alphabetisch sortiert:

CLOSE	4428	(4428)	File schließen
CPBCHL	4C6A	(4CC5)	Vergleiche Text (BC) mit Text (HL)
DATE	4470	(4470)	Datum im Format MM/TT/JJ in Buffer ablegen
DEBUG	440D	(440D)	DEBUG aufrufen
DELAY1	4789	(47E3)	ca. 64 us warten und Status des FDC lesen
DELAY2	4C92	(4CED)	ca. B * 3.79 ms warten
DIRPOS	4C19	(4C74)	Disk-Position eines DIR-Sectors berechnen
DIRR	48AF	(490A)	liest einen Sector des Directory
DIRW	48C4	(491F)	schreibt einen Sector des Directory
DIV05	4C57	(4CB2)	Dividiere HL=INT(HL/5), A=Rest
DIVA	4C59	(4CB4)	Dividiere HL=INT(HL/A), A=Rest
DOSCAL	4419	(4419)	DOS-Befehl ausführen und zurück
DOSCMD	4405	(4405)	DOS-Befehl ausführen (ohne Rückkehr)
DOSERR	4409	(4409)	Fehlermeldung des DOS ausgeben
DOSRDY	402D	(402D)	Sprung nach DOS READY (ohne Rückkehr)
DRVSEL	445B	(445B)	Drive auswählen und Motor starten
ERROR0	4030	(4030)	nach einem Fehler: Sprung nach DOS READY
EXPAND	444B	(444B)	File erweitern, wenn File zu kurz ist
FBREAK	470D	(475E)	FORCE-INTERRUPT-Kommando an FDC senden
FDCCMD	46F7	(4747)	Kommando mit richtiger TSR an FDC senden
FILPOS	47B5	(4810)	Disk-Position eines File-Sectors berechnen
GETFDE	48DB	(4936)	holt einen FDE aus dem Directory
GETSYS	4B6E	(4BC9)	SYS-File laden und starten
HEXA	44D7	(4068)	A hexadezimal nach (HL) ausgeben
HEXDE	44D2	(4063)	DE hexadezimal nach (HL) ausgeben
INIT	4420	(4420)	File öffnen, ggf. neue File anlegen
INSEXT	4473	(4473)	File-Typ (/EXT) in Filespec einfügen
INTDEL	4413	(4413)	Benutzer-Interrupt-Routine löschen
INTINS	447B	(4410)	Benutzer-Interrupt-Routine einfügen
KILL	442C	(442C)	File löschen

LOAD	4430	(4430)	Programm laden
LOADP	4BCD	(4C28)	Programm laden
MOTON	4715	(4767)	Drive-Motoren starten
MOTONX	4416	(4416)	Drive-Motoren weiterlaufen lassen
MULTC	4C42	(4C9D)	Multipliziere HLC = HL*A
MULTHL	4C39	(4C94)	Multipliziere AHL = HL*A
MULTL	4C37	(4C92)	Multipliziere AHL = L*A
NEXTC1	4C7A	(4CD5)	Nächstes Zeichen holen und Flags setzen
NEXTC2	4C7E	(4CD9)	Nächstes Zeichen holen und Flags setzen
NXTEOF	490D	(4968)	NEXT-Wert aus FCB holen + mit EOF vergleichen
OPEN	4424	(4424)	File öffnen, ggf. keine neue File anlegen
POS0	443F	(443F)	FCB auf Beginn einer File positionieren
POSBC	4442	(4442)	FCB auf Logische Record# (BC) positionieren
POSDEC	4445	(4445)	FCB um 1 Record zurückpositionieren
POSEOF	4448	(4448)	FCB auf Ende der File (EOF) positionieren
POSRBA	444E	(444E)	FCB auf RBA-Format (HLC) positionieren
PUSHR	4925	(4980)	Push Register, IX=FCB+0, IY=4280H, A=00
PUTEOF	4451	(4451)	EOF im FPDE auf den neuesten Stand bringen
READ	4436	(4436)	nächsten Sector/Record aus einer File lesen
READB	0013	(0013)	liest ein Byte aus einer File (siehe 4A98H)
READS	45DB	(4630)	Sector von Diskette lesen
RESTOR	46F5	(4745)	RESTORE-Kommando an FDC senden
RUN	4433	(4433)	Programm laden und starten (keine Rückkehr)
TESTS	45DF	(4634)	Sector auf Lesbarkeit testen
TEXTLP	446A	(446A)	Text auf Drucker ausgeben
TEXTTV	4467	(4467)	Text auf Bildschirm ausgeben
TFSPEC	441C	(441C)	Filespec (HL) nach FCB (DE) übertragen
TIME	446D	(446D)	Uhrzeit im Format HH:MM:SS in Buffer ablegen
TSTDSK	445E	(445E)	Drive auswählen, prüfen ob Diskette eingelegt
UPCASE	4548	(45B5)	Kleinbuchstaben in Großbuchstaben umwandeln
USRDEL	4464	(4464)	Benutzer-Routine (*name) löschen
USRINS	4461	(4461)	Benutzer-Routine (*name) einfügen
VERIFY	443C	(443C)	Sector/Record in eine File schreiben + testen
WNBUSY	46FF	(4750)	warten, bis der FDC nicht mehr busy ist
WRITDS	45E7	(463C)	Directory-Sector auf Diskette schreiben
WRITDV	4A6F	(4ACA)	Directory-Sector schreiben, ggf. Verify
WRITE	4439	(4439)	nächsten Sector/Record in eine File schreiben
WRITEB	001B	(001B)	schreibt ein Byte in eine File (siehe 4A3DH)
WRITES	45EB	(4640)	normalen Sector auf Diskette schreiben
WRITEV	4A62	(4ABD)	normalen Sector schreiben, ggf. Verify
WRITXV	4A5D	(4AB8)	Sector oder DIR-Sector schreiben, ggf. Verify

9.2 GDOS für Genie I+II

9.2.1 Initialisierung des DOS

Die Initialisierung des DOS ist weitgehend identisch mit der von NEWDOS/80 Version 2 für TRS-80 Model I. Im BOOT-Sector bestehen geringe Unterschiede, die weitgehend daher kommen, daß auf dem Genie I+II standardmäßig mit Double Density und daher mit 18 Sectoren pro Track gearbeitet wird. Die geänderten Stellen sind durch ">" markiert:

```
4202>18          ;LUMP#, in welcher der Directory beginnt

4207 36>FF       LD      (HL),FF      ;Double Density wählen

4211 11 05>01    LD      DE,0105      ;D=Track#, E=Sector# vom Beginn SYS0

4260 D6>12       SUB     12           ;Anzahl Sectoren pro Track pro Seite

42AF D6>12       SUB     12           ;Anzahl Sectoren pro Track

                 ----------------------
                 Text CLS + "RESET"
42DD 1C 1F>52>45>53>45>54 03                          ..RESET.
                 ----------------------
                 Text CLS + "G-DOS?"
42E5 1C 1F>47>2D>44>4F>53>3F 03                       ..G-DOS?.
                 ----------------------
                 Copyright Meldung
42EE>00>00>40>27>38>32>20>54>43>47>26>4A>48>4C>00     ..§'82 TCG&JHL.
                 ----------------------
                 PDRIVE-Parameter für Drive 0
42FD 00          ;PDRIVE+6
42FE>53          ;PDRIVE+2
42FF>03          ;PDRIVE+7
```

9.2.2 Unterschiede in SYS0/SYS

Ans Ende von SYS0 wurde ein kleiner ZAP angehängt, der nach 4028H-4029H (Drucker-DCB) lädt. Dadurch wurde SYS0 auf 51D4H statt 51DAH verkürzt. Die Stellen, die sich gegenüber NEWDOS/80 Version 2 für TRS-80 Model I unterscheiden, sind durch ">" markiert:

```
4028>48          ;Drucker-DCB: 72D Zeilen / Seite
4029>00          ;Drucker-DCB: Zeilenzähler

4041>32          ;Uhrzeit: Sekunden (50D)
4042>1E          ;         Minuten  (30D)
4043>08          ;         Stunden  (08D)
4044>53          ;Datum:   Jahr     (83D)
4045>06          ;         Monat    (06D)
4046>16          ;         Tag      (22D)

44C5 06>2E       LD      B,2E         ;Trennzeichen "." für Datum
```

```
4500 36>5F        LD      (HL),5F      ;Cursorzeichen auf Bildschirm

4598 FE>1F        CP      1F           ;Buchstaben inkl. Umlaute

45B8 FE>7F        CP      7F           ;Buchstaben inkl. Umlaute

4F49 3E>2E        LD      A,2E         ;Trennzeichen "." für Datum

                  ----------------------
                  Text "GENIE DOS 2.2 .. JHL-Köln .. 1.April 1983 .. Genie I"
4FAB 1B 1B 1B 1B 1B 1B BE 8F 8F BD 20 20 80 80 20 20  ..........  ..
4FBB BF 8F 8F BD 20 20 BE 8F 8F BD 20 20 BE 8F 8F BD  ....  ....  ....
4FCB 20 20 20 20 BE 8F 8F BD 20 20 80 20 BE 8F 8F BD      ....  . ....
4FDB C7 4A 48 4C 20 2D 20 4B 7C 6C 6E 0A BF 80 8C BC  .JHL - Köln.....
4FEB 20 65 6E 69 65 20 BF 80 80 BF 20 20 BF 80 80 BF   enie ....  ....
4FFB 20 20 8B 8C 8C B4 20 20 20 20 80 A0 9E 81 C2 80    ....    . ....
500B 20 80 A0 9E 81 C3 20 20 20 31 2E 41 70 72 69 6C   . ...   1.April
501B 20 31 39 38 33 20 20 0A AF BC BC 9F 20 20 80 80   1983  .....  ..
502B 20 20 BF BC BC 9F 20 20 AF BC BC 9F 20 20 AF BC    ....  ....  ..
503B BC 9F 20 20 20 20 B8 BF BC BC 20 A8 94 20 B8 BF  ..    .... .. ..
504B BC BC C6 20 20 47 65 6E 69 65 20 20 49 20 20 0D  ...  Genie  I  .

                  ----------------------
                  Texte für Datum und Uhrzeit
5063 44>41>54>55>4D>3F 20 28>54>54>2E>4D>4D>2E>4A>4A  DATUM? (TT.MM.JJ
5073 29 20 03                                         ) .
                  ----------------------
5076>5A>45>49>54 3F 20 20 28 48 48 3A 4D 4D 3A 53 53  ZEIT?  (HH:MM:SS
5086 29 20 03                                         ) .
                  ----------------------
5089>54>54>2E>4D>4D>2E>4A>4A 20 20 48 48 3A 4D 4D 3A  TT.MM.JJ  HH:MM:
5099 53 53 0D                                         SS.
                  ----------------------
                  Tabelle für die zulässige Eingabe des Datums
509C 01>1F        ;Tag    (1-31)
509E 01>0C        ;Monat  (1-12)
50A0>53 64        ;Jahr   (83-99)
```

9.3 GDOS für Genie III

9.3.1 Initialisierung des DOS

Als erstes wird aus dem BOOT-EPROM der sogenannte Urlader aufgerufen, der den BOOT-Sector von Diskette (Drive 0, Track 0, Sector 0) in Single Density ins RAM ab 4200H liest. Anschließend wird der Urlader für eine spätere Verwendung ins RAM ab 3D00H kopiert und so modifiziert, daß er dort läuft. Dann wird der BOOT-Sector bei 4200H gestartet.

Das Programm im BOOT-Sector bewirkt nun, daß erstmal der BASIC-Interpreter, der an das eigentliche SYS0 angehängt ist und auf Double Density Disketten ab Track 1, Sector 20D steht, ins RAM von 0000H bis 2FFFH und einige Erweiterungs-Routinen ins RAM zwischen 3500H und 4F15H geladen und bei 4F00H gestartet werden.

Von dort wird der BASIC-Interpreter bei 0000H gestartet, der innerhalb seiner Initialisierung, wenn die BREAK-Taste nicht gedrückt ist, das Video-RAM (3C00H - 3FFFH) ausblendet und den nach 3D00H kopierten Urlader bei 3D13H erneut aufruft, um den BOOT-Sector nochmals nach 4200H zu laden.

Danach wird im BASIC-Interpreter bei 06ADH das Memory-Banking auf Normal-Betrieb gesetzt (EPROM aus, Video-RAM 3C00H-3FFFH an, Video-RAM 4000H-47FFH aus, FDC und Tastatur bei 37E0H-3BFFH an) und der BOOT-Sector zum 2. Mal bei 4200H gestartet.

Dieses Mal bewirkt das Programm im BOOT-Sector, daß das eigentliche SYS0, welches auf Double Density Disketten ab Track 1, Sector 5 steht, ins RAM zwischen 400CH und 51DEH geladen und bei 4D01H gestartet wird.

Für das Format von SYS0 und dessen Initialisierung gilt das bereits in Kapitel 2 gesagte.

9.3.1.1 Listing des Urladers im BOOT-EPROM

```
000F 11 0F 01     LD      DE,010F       ;Flag: zuerst den BASIC-Interpreter
0012 D5           PUSH    DE            ;und dann SYS0/SYS laden, Flag retten
0013 3E C0        LD      A,C0          ;Umschalter 5.25 Zoll / 8 Zoll
0015 08           EX      AF,AF'        ;Umschalter retten
                  ----------------------
                  Lese-Versuch BOOT-Sector abwechselnd 5.25 Zoll / 8 Zoll
0016 21 EC 37     LD      HL,37EC       ;Zeiger auf 37EC
0019 36 FC        LD      (HL),FC       ;Single Density wählen
001B 3E 01        LD      A,01          ;Drive 0 anwählen und Motor starten
001D 32 E1 37     LD      (37E1),A
0020 77           LD      (HL),A        ;REASTORE-Kommando an FDC senden
0021 3C           INC     A             ;ca. 1.2 ms warten
0022 20 FD        JR      NZ,0021
0024 CB 46        BIT     0,(HL)        ;ist der FDC noch busy ?
0026 20 F9        JR      NZ,0021       ;wenn ja, warten
0028 08           EX      AF,AF'        ;Umschalter 5.25 Zoll / 8 Zoll
0029 EE 40        XOR     40            ;umschalten
002B 11 EE 37     LD      DE,37EE       ;Zeiger auf 37EE
002E 12           LD      (DE),A        ;abwechselnd 5.25 Zoll / 8 Zoll wählen
002F 08           EX      AF,AF'        ;Umschalter retten
0030 12           LD      (DE),A        ;Sector# 00 ins Sector-Register des FDC
0031 36 8C        LD      (HL),8C       ;READ-SECTOR-Kommando an FDC senden
0033 13           INC     DE            ;Zeiger auf Daten-Register des FDC
0034 06 09        LD      B,09          ;ca. 35 us warten
0036 10 FE        DJNZ    0036
0038 01 00 42     LD      BC,4200       ;Zeiger auf Buffer für BOOT-Sector
003B 7E           LD      A,(HL)        ;Status-Register des FDC lesen
003C E6 03        AND     03            ;Data Request und FDC busy ?
003E E2 3B*00     JP      PO,003B       ;wenn nein, warten
0041 1A           LD      A,(DE)        ;nächstes Byte vom FDC holen
0042 02           LD      (BC),A        ;und in den Buffer schreiben
0043 03           INC     BC            ;Zeiger auf Buffer +1
0044 CB 4E        BIT     1,(HL)        ;Data Request ?
0046 20 F9        JR      NZ,0041       ;wenn ja
0048 CB 4E        BIT     1,(HL)        ;Data Request ?
004A 20 F5        JR      NZ,0041       ;wenn ja
004C CB 46        BIT     0,(HL)        ;ist FDC noch busy ?
004E 20 F4        JR      NZ,0044       ;wenn ja
0050 7E           LD      A,(HL)        ;Status des FDC lesen
0051 E6 9F        AND     9F            ;Error-Bits maskieren
0053 20 C1        JR      NZ,0016       ;wenn Fehler: neuer Versuch
0055*18*03        JR      005A          ;wenn kein Fehler: weiter bei 005A
                  ----------------------
0057 C3 AD 06     JP      06AD          ;beim 2. Mal weiter bei 06AD
                  ----------------------
                  Urlader nach 3D00-3D7F kopieren
005A 47           LD      B,A
005B 67           LD      H,A           ;HL=0000
005C 6F           LD      L,A
005D 5F           LD      E,A           ;DE=3D00
005E 16 3D        LD      D,3D
0060 0E 80        LD      C,80          ;BC=0080
0062 3E F6        LD      A,F6          ;neuer Wert für Memory-Banking:
0064 18 01        JR      0067          ;weiter bei 0067
                  ----------------------
0066 C7           RST     00            ;NMI-Vektor
```

```
                        ----------------------
0067 D3 FA              OUT     FA              ;Video-RAM (3C00-3FFF) ausblenden
0069 ED B0              LDIR                    ;ROM 0000-007F nach RAM 3D00 kopieren
006B ED 43 55 3D        LD      (3D55),BC       ;Urlader modifizieren, daß er bei 3D00
006F 7A                 LD      A,D             ;läuft und daß es beim nächsten Mal nach
0070 32 40 3D           LD      (3D40),A        ;dem Laden bei 06AD weiter geht
                        ----------------------
                        BOOT-Sector starten
0073 3E F4              LD      A,F4            ;Video-RAM (3C00-3FFF) wieder einblenden
0075 D3 FA              OUT     FA
0077 21 EC 37           LD      HL,37EC         ;Zeiger auf 37EC
007A C3 00 42           JP      4200            ;BOOT-Sector starten
```

9.3.1.2 Unterschiede im BOOT-Sector

Die Stellen, die sich gegenüber dem BOOT-Sector aus NEWDOS/80 Version 2 für TRS-80 Model I unterscheiden, sind durch ">" markiert:

```
4200 00           ;./.
4201 FE           ;./.
4202>30           ;LUMP#, in welcher der Directory beginnt
4203>36>FF        LD      (HL),FF       ;Double Density wählen
4205>23           INC     HL            ;Zeiger auf Track-Register des FDC
4206>36>00        LD      (HL),00       ;Track# 00 eintragen
4208>D1           POP     DE            ;DE zurück (1: 010FH, 2: 0012H)
4209>15           DEC     D             ;D=01 ?
420A>11>14>01     LD      DE,0114       ;DE=Track#,Sector# vom Beginn des BASIC
420D>28>03        JR      Z,4212        ;wenn ja
420F>11>05>01     LD      DE,0105       ;DE=Track#,Sector# vom Beginn von SYS0
4212>3E>05        LD      A,05          ;Memory-Banking auf Normal-Betrieb
4214>D3>FA        OUT     FA
4216>D9           EXX                   ;gewünschte Track# und Sector# nach DE'
4217>00           NOP

4249>D5           PUSH    DE            ;wenn keine SYS0-Kennung:
424A>C0           RET     NZ            ;bei Startadresse+0 starten
424B>13           INC     DE            ;wenn SYS0-Kennung:
424C>D5           PUSH    DE            ;bei Startadresse+1 starten
424D>C9           RET
424E>00           NOP

4260 D6>12        SUB     12            ;Anzahl Sectoren pro Track pro Seite

4265 36>11        LD      (HL),11       ;Seite 1 anwählen

42AF D6>24        SUB     24            ;Anzahl Sectoren pro Track

                  ----------------------
                  Text CLS + "RESET"
42DD 1C 1F>52>45>53>45>54 03                        ..RESET.
                  ----------------------
                  Text CLS + "GDOS ?"
42E5 1C 1F>47>44>4F 53>20>3F 03                     ..GDOS ?.
                  ----------------------
                  Unbenutzt
42EE FF>04 FF>04 FF>04 FF>04 FF>04 FF>04 FF>04 FF
                  ----------------------
                  PDRIVE-Parameter für Drive 0
42FD 00           ;PDRIVE+6
42FE>D0           ;PDRIVE+2
42FF>43           ;PDRIVE+7
```

9.3.2 Unterschiede in SYS0/SYS

Die Stellen, die sich gegenüber NEWDOS/80 Version 2 für TRS-80 Model I unterscheiden, sind durch ">" markiert:

```
44C5 06>2E        LD      B,2E          ;Trennzeichen "." für Datum

                  ----------------------
                  Interrupt-Routine zum weitersetzen der Uhrzeit
44C9 EB 44        ;Zeiger auf Interrupt-Kette
44CB 28           ;40D Aufrufe pro Sekunde
44CC 01           ;Interrupt-Zähler
44CD 21 41 40     LD      HL,4041       ;Zeiger auf Uhrzeit und Datum
44D0>11>AC>43     LD      DE,43AC       ;nach 43AC-43B1 übertragen
44D3>01>06>00     LD      BC,0006
44D6>ED>B0        LDIR
44D8>C3>00>37     JP      3700          ;Uhren-IC nach 4041-4046 auslesen
                  ----------------------
44DB>00>00>00>00>00>00>00>00>00>00>00>00>00>00>00>00>00   ;gelöscht

4500 36>5F        LD      (HL),5F       ;Cursorzeichen auf Bildschirm

4510 DA>7B 04     JP      C,047B        ;Fortsetzung der Bildschirm-Routine

454C>C3>90>35     JP      3590          ;Fortsetzung der Tastatur-Routine

455B>C3>0E>01     JP      010E          ;Fortsetzung der Tastatur-Routine

                  ----------------------
                  Toggle Flag für Umschaltung Groß/Kleinschrift
4594>3A>E0>38     LD      A,(38E0)      ;Taste LOCK gedrückt ?
4597>E6>08        AND     08
4599>3E>C9        LD      A,C9
459B>28>02        JR      Z,459F        ;wenn nein
459D>EE>C9        XOR     C9            ;Toggle Flag
459F>32>B4>45     LD      (45B4),A      ;neues Flag speichern
45A2>79           LD      A,C           ;gedrückte Taste
45A3>E6>DF        AND     DF            ;Buchstabe ?
45A5>FE>41        CP      41
45A7>38>05        JR      C,45AE        ;wenn nein
45A9>FE>5F        CP      5F
45AB>79           LD      A,C           ;gedrückte Taste
45AC>38>04        JR      C,45B2        ;wenn ja: LOCK-Status berücksichtigen
45AE>79           LD      A,C
45AF>C9           RET

45B8 FE>7F        CP      7F            ;Buchstaben inkl. Umlaute

4635 26>39        LD      H,39          ;Verify-Buffer = 3900-3AFE
```

```
                    ----------------------
                    Behandlung von doppelseitigen Disketten
4698 CB 70          BIT     6,B         ;Diskette doppelseitig ?
469A 28>02          JR      Z,469E      ;wenn nein
469C CB 09          RRC     C           ;=> C = Sectors pro Track pro Seite
469E 7B             LD      A,E         ;ist die gewünschte Sector#
469F 91             SUB     C           ;auf der Rückseite der Diskette ?
46A0>CD>58>35       CALL    3558        ;Erweiterung: ggf. Bit für Rückseite
46A3>00             NOP                 ;setzen und bei einem Wechsel der
46A4>00             NOP                 ;Disk-Seite ca. 90 ms warten
46A5 CD 67 47       CALL    4767        ;gewünschte Disk-Seite anwählen

4798>CD>72>35       CALL    3572        ;Erweiterung: errechnetes Bit-Muster
                                        ;nach 4309 speichern, ohne das Bit für
                                        ;die gewählte Disk-Seite zu verändern

47CA 06>FF          LD      B,FF        ;Verzögerung nach dem Starten der
                                        ;Drive-Motoren ca. 960 ms

47D8 06>18          LD      B,18        ;Verzögerung für TI=H ca. 90 ms

                    ----------------------
                    Laden von Programm-Files
4C2E CD 65 4C       CALL    4C65        ;nächstes Byte lesen
4C31>FE>1F          CP      1F          ;ist das 1. Byte größer als 1EH ?
4C33>4F             LD      C,A         ;C = 1. Byte
4C34>3E>22          LD      A,22        ;Fehlercode "LOAD FILE FORMAT ERROR"
4C36>D0             RET     NC          ;wenn ja
4C37>CD>65>4C       CALL    4C65        ;nächstes Byte lesen
4C3A>47             LD      B,A         ;B = 2. Byte
4C3B>CD>65>4C       CALL    4C65        ;nächstes Byte lesen
4C3E>6F             LD      L,A         ;L = 3. Byte
4C3F>26>38          LD      H,38        ;HL = Buffer für Kommentare 3800-39FE

4C5B 3E>23          LD      A,23        ;Fehlercode "MEMORY FAULT"

4D92 2E>08          LD      L,08        ;Multiplikations-Faktor für SYSTEM BJ

                    ----------------------
                    Test, ob Bildschirm-RAM auf Kleinbuchstaben umgerüstet ist
4E48>00 00>00>00>00>00>00>00>00>00>00  ;gelöscht

                    ----------------------
                    CRT-Controller initialisieren
4ECD>01>10>00       LD      BC,0010     ;Initialisierungs-Werte für
4ED0>21>C0>42       LD      HL,42C0     ;den CRT-Controller aus dem
4ED3>11>F0>37       LD      DE,37F0     ;PDRIVE-Sector 42C0-42CF
4ED6>ED>B0          LDIR                ;nach 37F0 übertragen
4ED8>CD>B8>37       CALL    37B8        ;CRT-Controller damit initialisieren
4EDB>21>AB>4F       LD      HL,4FAB     ;Text "GENIE DOS 2.1 ... JHL-Köln ..."
4EDE>CD>67>44       CALL    4467        ;auf Bildschirm ausgeben
4EE1>CD>CD>44       CALL    44CD        ;Uhren-IC nach 4041-4046 auslesen
```

```
                    ----------------------
                    Text "GENIE DOS 2.1 .. JHL-Köln .. 1.Mai 1984 .. GENIE III"
4FAB 1B 1B 1B 1B 1B 1B BE 8F 8F BD 20 20 80 80 20 20  ..........  ..
4FBB BF 8F 8F BD 20 20 BE 8F 8F BD 20 20 BE 8F 8F BD  ....  ....  ....
4FCB 20 20 20 20 BE 8F 8F BD 20 20 80 20 80 A0 BE BF      ....  . . ..
4FDB C7 4A 48 4C 20 2D 20 4B 7C 6C 6E 0A BF 88 8C BC  .JHL - Köln.....
4FEB 20 65 6E 69 65 20 BF 80 80 BF 20 20 BF 80 80 BF   enie ....  ....
4FFB 20 20 8B 8C 8C B4 20 20 20 20 80 A0 9E 81 C2 80    ....    . ....
500B 20 80 80 80 BF C2 20 20 20 20 31 2E 4D 61 69 20   .....    1.Mai
501B 20 20 20 31 39 38 34 0A AF BC BC 9F 20 20 80 80     1984.....  ..
502B 20 20 BF BC BC 9F 20 20 AF BC BC 9F 20 20 AF BC    ....  ....  ..
503B BC 9F 20 20 20 20 B8 BF BC BC 20 A8 94 20 80 80  ..    .... .. ..
504B BC BF BC C6 2E 47 45 4E 49 45 C2 49 49 49 2E 0D  .....GENIE.III..

                    ----------------------
                    Texte für Datum und Uhrzeit
5063 44>41>54>55>4D>3F 20 28 4D 4D>2E>54>54>2E>4A>4A  DATUM? (MM.TT.JJ
5073 29 20 03                                         ) .
                    ----------------------
5076>5A>45>49>54 3F 20 20 28 48 48 3A 4D 4D 3A 53 53  ZEIT?  (HH:MM:SS
5086 29 20 03                                         ) .
                    ----------------------
5089 4D 4D>2E>54>54>2E>4A>4A 20 20 48 48 3A 4D 4D 3A  MM.TT.JJ  HH:MM:
5099 53 53 0D                                         SS.
```

9.4 GDOS für Genie IIIs

9.4.1 Initialisierung des DOS

Als erstes wird aus dem BOOT-EPROM der sogenannte Urlader von 007CH-0199H ins RAM nach 3800H-391DH kopiert und dort gestartet, um den BOOT-Sector von Diskette (Drive 0, Track 0, Sector 0) in Single oder Double Density (wird abwechselnd probiert) zu lesen. Der BOOT-Sector wird dabei jedoch nirgendwo im RAM gespeichert, sondern es geht lediglich darum, das relative Byte E0H des BOOT-Sectors zu erfahren, welches den Typ der eingelegten Diskette angibt.

GDOS-Disketten haben dort die Kennung 01H, CP/M-Disketten die Kennung 02H, Genie Service-Disketten die Kennung 03H und alle übrigen Werte kennzeichnen Disketten für NEWDOS/80 und GDOS auf TRS-80 Model I und Genie I,II,IIs.

Wenn es sich um eine GDOS-Diskette handelt (Kennung 01H) wird der gleiche BOOT-Sector nochmals von Diskette gelesen, dieses Mal aber im RAM ab 4200H gespeichert. Vor dem Starten des BOOT-Sectors bei 4200H wird dann noch das RAM im Bereich 4000H-405CH initialisiert.

Das Programm im BOOT-Sector bewirkt nun, daß erstmal der BASIC-Interpreter, der an das eigentliche SYS0 angehängt ist und auf Double Density Disketten ab Track 1, Sector 20D steht, ins RAM von 0000H bis 2FFFH und einige Erweiterungs-Routinen ins RAM zwischen 3500H und 4F15H geladen und bei 4F00H gestartet werden.

Von dort wird das Programm im BOOT-Sector erneut aufgerufen, dieses Mal aber bei 420FH, sodaß nun das eigentliche SYS0, welches auf Double Density Disketten ab Track 1, Sector 5 steht, ins RAM zwischen 400CH und 51DEH geladen und bei 50E3H gestartet wird, um für den Fall, daß im Genie IIIs ein Uhren-IC eingebaut ist, in SYS0 noch einen kleinen ZAP zu machen.

Für das Format von SYS0 und dessen weitere Initialisierung bei 4D00H gilt das bereits in Kapitel 2 gesagte.

9.4.1.1 Listing des Urladers aus dem BOOT-EPROM

```
3804 AF           XOR     A           ;"NOP" nach 3895: den nächsten Sector
3805 32 95 38     LD      (3895),A    ;nur lesen, aber nicht ins RAM schreiben
3808 3A FF FF     LD      A,(FFFF)    ;geretteten Status von 38A0 lesen
380B CB 47        BIT     0,A         ;wurde die "F1"-Taste gedrückt ?
380D 20 52        JR      NZ,3861     ;wenn ja, Monitor starten
380F CD 68 38     CALL    3868        ;BOOT-Sector lesen, aber nicht ins RAM
                                      ;schreiben, E = relatives Byte E0H
                                      ;(Kennung der eingelegten Diskette)
3812 DD 21 00 00  LD      IX,0000     ;Bufferadresse für BOOT-Sector
3816 7B           LD      A,E         ;ermitteln:
3817 FE 03        CP      03
3819 28 0C        JR      Z,3827      ;E=03 => IX=0000
381B DD 21 00 FC  LD      IX,FC00     ;        (Genie Service-Diskette)
381F FE 02        CP      02
3821 28 04        JR      Z,3827      ;E=02 => IX=FC00 (CP/M-Diskette)
3823 DD 21 00 42  LD      IX,4200     ;Rest => IX=4200 (GDOS-Diskette)
3827 3E 02        LD      A,02        ;"LD  (BC),A": nächsten Sector
3829 32 95 38     LD      (3895),A    ;normal ins RAM schreiben
382C CD 68 38     CALL    3868        ;BOOT-Sector in Buffer (IX) einlesen
382F 31 FE FF     LD      SP,FFFE     ;SP auf FFFE
3832 DD E5        PUSH    IX          ;Startadresse des Buffers in den Stack
3834 7B           LD      A,E
3835 FE 03        CP      03          ;wenn E=03:
3837 C8           RET     Z           ;BOOT-Sector starten (Service-Diskette)
3838 FE 02        CP      02          ;wenn E=02:
383A C8           RET     Z           ;BOOT-Sector starten (CP/M-Diskette)
383B F5           PUSH    AF          ;Register retten (GDOS-Diskette):
383C E5           PUSH    HL
383D 21 E8 38     LD      HL,38E8     ;RAM 4000-4035 initialisieren
3840 11 00 40     LD      DE,4000
3843 01 36 00     LD      BC,0036
3846 ED B0        LDIR
3848 AF           XOR     A           ;RAM 4036-405C löschen
3849 06 27        LD      B,27
384B 12           LD      (DE),A
384C 13           INC     DE
384D 10 FC        DJNZ    384B
384F E1           POP     HL          ;Register zurück
3850 F1           POP     AF
3851 FE 01        CP      01          ;wenn E=01 (Kennung für GDOS):
3853 C8           RET     Z           ;BOOT-Sector bei 4200 starten

                  ----------------------
                  BOOT-Sector ggf. nach Buffer (IX) lesen
3868 21 ED 37     LD      HL,37ED     ;Zeiger auf Track-Register des FDC
386B 36 00        LD      (HL),00     ;Track# 00 eintragen
386D 2B           DEC     HL          ;Zeiger auf Kommando/Status-Register
386E 06 0A        LD      B,0A        ;Zähler für Anzahl Versuche
3870 36 D0        LD      (HL),D0     ;FORCE-INTERRUPT-Kommando an FDC senden
3872 CD DF 38     CALL    38DF        ;Drive 0 wählen und Motor starten
3875 C5           PUSH    BC          ;B retten (Zähler für Anzahl Versuche)
3876 CD D0 38     CALL    38D0        ;warten, bis der FDC nicht mehr busy ist
3879 AF           XOR     A           ;Null
387A 32 EE 37     LD      (37EE),A    ;ins Sector-Register
387D 32 EF 37     LD      (37EF),A    ;und ins Daten-Register schreiben
3880 36 0B        LD      (HL),0B     ;RESTORE-Kommando an FDC senden
```

```
3882 CD D0 38     CALL    38D0        ;warten, bis der FDC nicht mehr busy ist
3885 36 88        LD      (HL),88     ;READ-SECTOR-Kommando an FDC senden
3887 DD E5        PUSH    IX          ;Bufferadresse
3889 C1           POP     BC          ;nach BC
388A 11 00 E1     LD      DE,E100     ;D=Zähler für E=relatives Byte E0H
388D CD D9 38     CALL    38D9        ;ca. 20 us warten
3890 18 0A        JR      389C        ;weiter bei 389C
                  ----------------------
3892 3A EF 37     LD      A,(37EF)    ;nächstes Byte vom FDC holen
3895 02           LD      (BC),A      ;ggf. in den Buffer schreiben
3896 15           DEC     D           ;Zähler -1
3897 C2 9B 38     JP      NZ,389B     ;wenn Zähler <> 0
389A 5F           LD      E,A         ;relatives Byte E0H in E merken
389B 03           INC     BC          ;Zeiger auf Buffer +1
389C CB 4E        BIT     1,(HL)      ;Data Request ?
389E C2 92 38     JP      NZ,3892     ;wenn ja
38A1 CB 4E        BIT     1,(HL)      ;Data Request ?
38A3 C2 92 38     JP      NZ,3892     ;wenn ja
38A6 CB 4E        BIT     1,(HL)      ;Data Request ?
38A8 C2 92 38     JP      NZ,3892     ;wenn ja
38AB CB 46        BIT     0,(HL)      ;ist der FDC noch busy ?
38AD CA BA 38     JP      Z,38BA      ;wenn nein
38B0 CB 4E        BIT     1,(HL)      ;Data Request ?
38B2 C2 92 38     JP      NZ,3892     ;wenn ja
38B5 CB 7E        BIT     7,(HL)      ;sind die Drive-Motoren noch an ?
38B7 CA 9C 38     JP      Z,389C      ;wenn ja
38BA 7E           LD      A,(HL)      ;Status des FDC lesen
38BB 36 D0        LD      (HL),D0     ;FORCE-INTERRUPT-Kommando an FDC
38BD C1           POP     BC          ;B zurück (Zähler für Anzahl Versuche)
38BE E6 FC        AND     FC          ;Error-Bits maskieren
38C0 C8           RET     Z           ;wenn kein Error
                  ----------------------
                  Fehlerbehandlung
38C1 3E FE        LD      A,FE        ;bei jedem neuen Versuch
38C3 B0           OR      B           ;zwischen Single Density und
38C4 77           LD      (HL),A      ;Double Density abwechseln
38C5 10 A9        DJNZ    3870        ;noch keine 10D Versuche gemacht ?
38C7 3E 90        LD      A,90        ;System-Port:
38C9 D3 FA        OUT     FA          ;EPROM einschalten
38CB 3E 01        LD      A,01        ;Fehlercode "BOOT ERROR"
38CD C3 6B 00     JP      006B        ;weiter im Monitor
                  ----------------------
                  Warten, bis der FDC nicht mehr busy ist
38D0 CD D9 38     CALL    38D9        ;ca. 20 us warten
38D3 CB 46        BIT     0,(HL)      ;Status des FDC lesen
38D5 20 FC        JR      NZ,38D3     ;wenn busy: warten
38D7 7E           LD      A,(HL)      ;Status des FDC lesen
38D8 C9           RET
                  ----------------------
                  ca. 20 us warten (bei 7.2 MHz)
38D9 3E 08        LD      A,08        ;Zähler setzen
38DB 3D           DEC     A           ;Zähler -1
38DC 20 FD        JR      NZ,38DB     ;wenn Zähler <> 0
38DE C9           RET
                  ----------------------
                  Drive 0 wählen und Motor starten
38DF 3E 01        LD      A,01        ;Drive 0 wählen
38E1 32 E0 37     LD      (37E0),A    ;und Motor starten
38E4 CD D9 38     CALL    38D9        ;ca. 20 us warten
38E7 C9           RET
```

9.4.1.2 Unterschiede im BOOT-Sector

Die Stellen, die sich gegenüber dem BOOT-Sector aus NEWDOS/80 Version 2 für TRS-80 Model I unterscheiden, sind durch ">" markiert:

```
4200 00           ;./.
4201 FE           ;./.
4202>30           ;LUMP#, in welcher der Directory beginnt
4203>36>FF        LD      (HL),FF       ;Double Density wählen
4205>23           INC     HL            ;Zeiger auf Track-Register des FDC
4206>36>00        LD      (HL),00       ;Track# 00 eintragen
4208>AF           XOR     A
4209>AF           XOR     A
420A>11>14>01     LD      DE,0114       ;DE=Track#,Sector# vom Beginn des BASIC
420D>18>03        JR      4212          ;BASIC-Interpreter von Diskette laden
                  ----------------------
420F>11>05>01     LD      DE,0105       ;DE=Track#,Sector# vom Beginn von SYS0
4212>3E>C4        LD      A,C4          ;System-Port auf Normal-Betrieb
4214>D3>FA        OUT     FA
4216>D9           EXX                   ;gewünschte Track# und Sector# nach DE'
4217>00           NOP

4249>D5           PUSH    DE            ;wenn keine SYS0-Kennung:
424A>C0           RET     NZ            ;bei Startadresse+0 starten
424B>13           INC     DE            ;wenn SYS0-Kennung:
424C>D5           PUSH    DE            ;bei Startadresse+1 starten
424D>C9           RET
424E>00           NOP

4260 D6>12        SUB     12            ;Anzahl Sectoren pro Track pro Seite

4265 36>11        LD      (HL),11       ;Seite 1 anwählen

42AF D6>24        SUB     24            ;Anzahl Sectoren pro Track

                  ----------------------
                  Text CLS + "????"
42DD 1C 1F>3F>01>3F>3F>3F 03                          ..?.???.
                  ;bei 42E0 steht die Kennung für GDOS: 01H
                  ----------------------
                  Text CLS + "GDOS ?"
42E5 1C 1F>47>44>4F 53>20>3F 03                       ..GDOS ?.
                  ----------------------
                  Unbenutzt
42EE>64>7C>32>39>7C>FD>75>35>FD>CB>34>C6>D5>CD>24
                  ----------------------
                  PDRIVE-Parameter für Drive 0
42FD 00           ;PDRIVE+6
42FE>D4           ;PDRIVE+2
42FF>43           ;PDRIVE+7
```

9.4.2 Unterschiede in SYS0/SYS

Die Stellen, die sich gegenüber NEWDOS/80 Version 2 für TRS-80 Model I unterscheiden, sind durch ">" markiert:

```
44C5 06>2E        LD      B,2E          ;Trennzeichen "." für Datum

4500 36>5F        LD      (HL),5F       ;Cursorzeichen auf Bildschirm

4510 DA>7B 04     JP      C,047B        ;Fortsetzung der Bildschirm-Routine

454C>C3>90>35     JP      3590          ;Fortsetzung der Tastatur-Routine

455B>C3>0E>01     JP      010E          ;Fortsetzung der Tastatur-Routine

                  ----------------------
                  Toggle Flag für Umschaltung Groß/Kleinschrift
4594>3A>E0>38     LD      A,(38E0)      ;Taste LOCK gedrückt ?
4597>E6>08        AND     08
4599>3E>C9        LD      A,C9
459B>28>02        JR      Z,459F        ;wenn nein
459D>EE>C9        XOR     C9            ;Toggle Flag
459F>32>B4>45     LD      (45B4),A      ;neues Flag speichern
45A2>79           LD      A,C           ;gedrückte Taste
45A3>E6>DF        AND     DF            ;Buchstabe ?
45A5>FE>41        CP      41
45A7>38>05        JR      C,45AE        ;wenn nein
45A9>FE>5F        CP      5F
45AB>79           LD      A,C           ;gedrückte Taste
45AC>38>04        JR      C,45B2        ;wenn ja: LOCK-Status berücksichtigen
45AE>79           LD      A,C
45AF>C9           RET

45B8 FE>7F        CP      7F            ;Buchstaben inkl. Umlaute

45FD DC>80>37     CALL    C,3780        ;bei RTC-Interrupts alle definierten
                                        ;Interrupt-Routinen bearbeiten

4635 26>39        LD      H,39          ;Verify-Buffer = 3900-3AFE

                  ----------------------
                  Behandlung von doppelseitigen Disketten
4698>00           NOP
4699>CD>83>35     CALL    3583          ;Diskette doppelseitig ?
469C CB 09        RRC     C             ;=> C = Sectors pro Track pro Seite
469E 7B           LD      A,E           ;ist die gewünschte Sector#
469F 91           SUB     C             ;auf der Rückseite der Diskette ?
46A0>CD>58>35     CALL    3558          ;Erweiterung: ggf. Bit für Rückseite
46A3>00           NOP                   ;setzen und bei einem Wechsel der
46A4>00           NOP                   ;Disk-Seite ca. 90 ms warten
46A5 CD 67 47     CALL    4767          ;gewünschte Disk-Seite anwählen

4798>CD>72>35     CALL    3572          ;Erweiterung: errechnetes Bit-Muster
                                        ;nach 4309 speichern, ohne das Bit für
                                        ;die gewählte Disk-Seite zu verändern

47CA 06>FF        LD      B,FF          ;Verzögerung nach dem Starten der
                                        ;Drive-Motoren ca. 960 ms
```

```
47D8 06>18        LD      B,18          ;Verzögerung für TI=H ca. 90 ms

4BDE 28>38        JR      Z,4C18        ;wenn die benötigte SYS-File bereits
                                        ;im Speicher steht

                  ----------------------
                  SYS-Files laden und starten
4C06 CD 28 4C     CALL    4C28          ;SYS-File laden
4C09 22 1E 4C     LD      (4C1E),HL     ;Startadresse der SYS-File nach 4C1E
4C0C 28>0A        JR      Z,4C18        ;wenn kein Error
                  ----------------------
                  Fehlerbehandlung
4C0E 3A 17 43     LD      A,(4317)      ;sollte SYS4/SYS
4C11 FE 06        CP      06            ;geladen werden ?
4C13>28>FE        JR      Z,4C13        ;wenn ja: Endlosschleife
4C15>26>2E        LD      H,2E          ;Fehlercode "SYSTEM PROGRAM NOT FOUND"
4C17>E3           EX      (SP),HL       ;in den Stack statt gerettetem AF-Wert
                  ----------------------
                  SYS-File starten
4C18>C1           POP     BC            ;BC = ursprünglicher Wert von AF
4C19>78           LD      A,B           ;A zurück (oder Fehlercode 2E)
4C1A C1           POP     BC            ;BC zurück
4C1B D1           POP     DE            ;DE zurück
4C1C E1           POP     HL            ;HL zurück
4C1D>CC*00*00     CALL    Z,0000        ;wenn kein Error: SYS-File starten
     ********** FEHLER! Nach einem Error beim Laden von SYS-Files  **********
     ********** wird in jedem Fall zum Aufrufer von GETSYS zurück- **********
     ********** gekehrt, obwohl dieser vielleicht nach DOSRDY      **********
     ********** (402DH), ERROR0 (4030H) oder DOSCMD (4405H) wollte **********
     ********** und daher gar nicht mit einer Rückkehr rechnet !   **********

                  ----------------------
                  Laden von Programm-Files
4C2E CD 65 4C     CALL    4C65          ;nächstes Byte lesen
4C31>FE>1F        CP      1F            ;ist das 1. Byte größer als 1EH ?
4C33>4F           LD      C,A           ;C = 1. Byte
4C34>3E>22        LD      A,22          ;Fehlercode "LOAD FILE FORMAT ERROR"
4C36>D0           RET     NC            ;wenn ja
4C37>CD>65>4C     CALL    4C65          ;nächstes Byte lesen
4C3A>47           LD      B,A           ;B = 2. Byte
4C3B>CD>65>4C     CALL    4C65          ;nächstes Byte lesen
4C3E>6F           LD      L,A           ;L = 3. Byte
4C3F>26>38        LD      H,38          ;HL = Buffer für Kommentare 3800-39FE

4C5B 3E>23        LD      A,23          ;Fehlercode "MEMORY FAULT"

                  ----------------------
                  UP DELAY2
4CF1 CD>4B>37     CALL    374B          ;Verzögerung B * 3.75 ms
4CF4>5F           LD      E,A           ;E=00
4CF5>C9           RET
4CF6>00           NOP
4CF7>00           NOP

4D92 2E>08        LD      L,08          ;Multiplikations-Faktor für SYSTEM BJ

                  ----------------------
                  Test, ob Bildschirm-RAM auf Kleinbuchstaben umgerüstet ist
4E48>00 00>00>00>00>00>00>00>00>00>00  ;gelöscht
```

```
                    ----------------------
                    CRT-Controller initialisieren
4ECD>01>10>00       LD      BC,0010       ;Initialisierungs-Werte für
4ED0>21>C0>42       LD      HL,42C0       ;den CRT-Controller aus dem
4ED3>11>F0>37       LD      DE,37F0       ;PDRIVE-Sector 42C0-42CF
4ED6>ED>B0          LDIR                  ;nach 37F0 übertragen
4ED8>CD>B8>37       CALL    37B8          ;CRT-Controller damit initialisieren
4EDB>21>AB>4F       LD      HL,4FAB       ;"GENIE DOS 2.1c .. TCS Computer GmbH"
4EDE>CD>67>44       CALL    4467          ;auf Bildschirm ausgeben
4EE1>CD>CD>44       CALL    44CD          ;ggf. Uhren-IC nach 4041-4046 auslesen

                    ----------------------
                    Ende der Initialisierung von SYS0
4F2A 3A 40 38       LD      A,(3840)      ;Tastatur abfragen
4F2D 0F             RRCA                  ;ENTER gedrückt ?
4F2E DA>B0>50       JP      C,50B0        ;wenn ja, weiter bei 50B0
4F31 7E             LD      A,(HL)        ;Zeiger auf AUTO-Befehl
4F32 FE 0D          CP      0D            ;ist ein AUTO-Befehl vorhanden ?
4F34 CA>B0>50       JP      Z,50B0        ;wenn nein, weiter bei 50B0
4F37 CD 67 44       CALL    4467          ;AUTO-Befehl auf Bildschirm anzeigen
4F3A C3>B4>50       JP      50B4          ;weiter bei 50B4
     ********** FEHLER! Es muß 50B5 heißen ! **********

                    ----------------------
                    Text "GENIE DOS 2.1c .. TCS Computer GmbH .. GENIE IIIs
4FAB 1B 1B 1B 1B 1B 1B BE 8F 8F BD C6 BF 8F 8F BD C2  ................
4FBB BE 8F 8F BD C2 BE 8F 8F BD C3 BE 8F 8F BD C4 A0  ................
4FCB BE BF C3 20 54 43 53 20 43 6F 6D 70 75 74 65 72  ... TCS Computer
4FDB 20 47 6D 62 48 20 20 0A 00 00 00 00 BF 88 8C BC   GmbH  .........
4FEB 20 45 4E 49 45 20 BF C2 BF C2 BF C2 BF C2 8B 8C   ENIE ..........
4FFB 8C B4 C4 00 00 00 00 00 00 00 A0 9E 81 C2 80 20  .......... ....
500B 80 80 BF 20 C0 20 20 20 96 83 83 83 84 20 20 20  ... .   .....
501B 20 C3 20 31 39 38 34 0A AF BC BC 9F C6 BF BC BC   . 1984.........
502B 9F 20 20 AF BC BC 9F C2 AF BC BC 9F C3 B8 BF BC  .  ..............
503B BC 20 A8 94 20 80 BC BF BC 20 C1 20 A5 B0 B0 B0  . .. .... . ....
504B 84 80 47 45 4E 49 45 20 49 49 49 20 73 0D 00 00  ..GENIE III s...

                    ----------------------
                    Texte für Datum und Uhrzeit
5063 44>41>54>55>4D>3F 20 28 4D 4D>2E>54>54>2E>4A>4A  DATUM? (MM.TT.JJ
5073 29 20 03                                         ) .
                    ----------------------
5076>5A>45>49>54 3F 20 20 28 48 48 3A 4D 4D 3A 53 53  ZEIT?  (HH:MM:SS
5086 29 20 03                                         ) .
                    ----------------------
5089 4D 4D>2E>54>54>2E>4A>4A 20 20 48 48 3A 4D 4D 3A  MM.TT.JJ  HH:MM:
5099 53 53 0D                                         SS.

                    ----------------------
                    Neu: Zeichensatz auswählen
50B0 00             NOP                   ;
50B1 3E 78          LD      A,78          ;Code für SYS22/SYS
50B3 18 02          JR      50B7          ;weiter bei 50B7
50B5 3E 38          LD      A,38          ;Code für SYS22/SYS
50B7 0E 00          LD      C,00          ;=> DOS-Befehl "ZL X" ausführen
50B9 F5             PUSH    AF            ;A retten
50BA CD 02 44       CALL    4402          ;SYS22 laden und starten
50BD F1             POP     AF            ;A zurück
50BE FE 78          CP      78            ;war ein AUTO-Befehl vorhanden ?
50C0 CA 00 44       JP      Z,4400        ;wenn nein, Sprung nach DOS READY
```

```
50C3 C3 05 44     JP      4405          ;wenn ja, AUTO-Befehl ausführen

                  ----------------------
                  Neu: Start der Initialisierung von SYS0/SYS
50E3 DB FA        IN      FA            ;Status des System-Ports
50E5 47           LD      B,A           ;nach B retten
50E6 CB B7        RES     6,A           ;auf 1.78 MHz umschalten
50E8 D3 FA        OUT     FA
50EA 3E 04        LD      A,04          ;Prüfen,
50EC D3 5B        OUT     5B            ;ob ein
50EE DB 5A        IN      5A            ;Uhren-IC
50F0 3C           INC     A             ;eingebaut ist
50F1 78           LD      A,B           ;alten Zustand des
50F2 D3 FA        OUT     FA            ;System-Ports zurück
50F4 28 1B        JR      Z,5111        ;wenn kein Uhren-IC vorhanden
                  ----------------------
                  ZAP in SYS0 eintragen, wenn Uhren-IC vorhanden
50F6 21 14 51     LD      HL,5114       ;5114-512C
50F9 11 CD 44     LD      DE,44CD       ;nach
50FC 01 19 00     LD      BC,0019       ;44CD-44E5
50FF ED B0        LDIR                  ;kopieren
5101 CB C7        SET     0,A           ;Video-RAM ausblenden
5103 D3 FA        OUT     FA
5105 11 00 3C     LD      DE,3C00       ;512D-5178 nach
5108 01 4C 00     LD      BC,004C       ;3C00-3C4B
510B ED B0        LDIR                  ;kopieren
510D CB 87        RES     0,A           ;Video-RAM wieder einblenden
510F D3 FA        OUT     FA
5111 C3 00 4D     JP      4D00          ;zur weiteren Initialisierung von SYS0
                  ----------------------
                  ZAP für 44CD-44E5: Interrupt-Routine zum weitersetzen der Uhr
5114 F3           DI                    ;Interrupts sperren
5115 21 41 40     LD      HL,4041       ;Zeiger auf Datum und Uhrzeit
5118 11 AC 43     LD      DE,43AC       ;nach 43AC-43B1 kopieren
511B 01 06 00     LD      BC,0006
511E ED B0        LDIR
5120 3E C5        LD      A,C5          ;Video-RAM ausblenden
5122 D3 FA        OUT     FA
5124 CD 00 3C     CALL    3C00          ;Uhren-IC nach 4041-4046 lesen
5127 3E C4        LD      A,C4          ;Video-RAM wieder einblenden
5129 D3 FA        OUT     FA
512B FB           EI                    ;Interrrupts wieder erlauben
512C C9           RET
                  ----------------------
                  ZAP für 3C00H zum Auslesen des Uhren-IC's
512D DD E5 DD 21 3F 3C 2E 44 0E 02 16 CC 06 03 CD 31
513D 3C 87 5F 87 87 83 5F CD 31 3C 83 77 23 10 EF 16
514D 5C 3E 2B 32 1C 3C 2E 43 0D 20 E1 DD E1 AF D3 5B
515D C9 7A D3 5B 16 10 92 57 DD 23 DB 5A DD A6 00 C9
516D 0F 0F 01 0F 03 0F 03 0F 07 0F 07 0F
```

9.5 Colour DOS

Das gesamte Disketten-Betriebssystem des Colour Genie befindet sich in einem ROM, welches die Speicherplätze C000H bis DFFFH belegt. In diesem ROM sind auch Erweiterungen für den BASIC-Interpreter enthalten, damit dieser auch Dateien auf Diskette bearbeiten kann.

Der Floppy Disk Controller wird über die Adressen FFE0-FFEF angesprochen, die in Kapitel 1 erklärt sind.

9.5.1 DOS-Befehl CMD ".."

Dieser Befehl arbeitet als Verteiler für einige wichtige Funktionen des DOS und ist daher hier disassembliert und kommentiert abgedruckt.

Es gibt auch noch ältere ROM-Versionen, in denen dieser Befehl die Speicherplätze C317H bis C355H belegt, ansonsten aber genauso arbeitet (abgesehen davon, daß die Befehlstabelle nicht aus dem RAM, sondern aus dem ROM bei C178H bis C1B7H benutzt wird).

```
                        ----------------------
                        DOS-Befehl CMD ".."
C2FB CF                 RST     08              ;Syntaxcheck: Pointer auf '"' ?
C2FC 22
C2FD D6 3B              SUB     3B              ;";" abziehen
C2FF CB AF              RES     5,A             ;Kleinschrift in Großschrift umwandeln
C301 FE 20              CP      20              ;war Zeichen > "Z" ?
C303 D2 97 19           JP      NC,1997         ;wenn ja: SYNTAX ERROR
C306 E5                 PUSH    HL              ;Pointer auf BASIC-Programm retten
C307 21 08 5B           LD      HL,5B08         ;Zeiger auf Beginn der Befehlstabelle
C30A 87                 ADD     A               ;entsprechenden Eintrag
C30B 85                 ADD     L               ;in der Tabelle berechnen
C30C 6F                 LD      L,A
C30D 5E                 LD      E,(HL)          ;Startadresse von CMD ".."
C30E 23                 INC     HL              ;aus der Tabelle entnehmen
C30F 56                 LD      D,(HL)
C310 E1                 POP     HL              ;Pointer auf BASIC-Programm zurück
C311 D5                 PUSH    DE              ;Startadresse retten
C312 01 18 5A           LD      BC,5A18         ;Buffer für Übergabe-Parameter
C315 C5                 PUSH    BC              ;Bufferadresse retten
C316 23                 INC     HL              ;ist das nächste Zeichen
C317 7E                 LD      A,(HL)          ;im BASIC-Programm
C318 FE 22              CP      22              ;ein '"' ?
C31A 28 07              JR      Z,C323          ;wenn ja
C31C B7                 OR      A               ;oder Ende der BASIC-Zeile ?
C31D 28 04              JR      Z,C323          ;wenn ja
C31F 02                 LD      (BC),A          ;nächstes Zeichen in den Buffer
C320 03                 INC     BC              ;Zeiger auf Buffer +1
C321 18 F3              JR      C316            ;weiter
                        ----------------------
C323 3E 0D              LD      A,0D            ;Endemarkierung
C325 02                 LD      (BC),A          ;in den Buffer
C326 D1                 POP     DE              ;DE = Zeiger auf Buffer
C327 C1                 POP     BC              ;BC = Startadresse
C328 E5                 PUSH    HL              ;Pointer auf BASIC-Programm retten
C329 21 30 C3           LD      HL,C330         ;Rücksprungadresse C330
C32C E5                 PUSH    HL              ;in den Stack schreiben
C32D C5                 PUSH    BC              ;Startadresse in den Stack
C32E EB                 EX      DE,HL           ;=> HL zeigt auf Buffer
C32F C9                 RET                     ;Befehl CMD ".." aufrufen
                        ----------------------
C330 E1                 POP     HL              ;HL = Pointer auf BASIC-Programm
C331 7E                 LD      A,(HL)          ;Ende der BASIC-Zeile ?
C332 B7                 OR      A
C333 28 02              JR      Z,C337          ;wenn ja
C335 CF                 RST     08              ;Syntaxcheck: Pointer auf '"' ?
C336 22
C337 C9                 RET
```

9.5.1.1 Befehlstabelle für CMD ".."

Diese Tabelle wird in der Initialisierung des DOS aus dem ROM von C15CH-C19BH ins RAM kopiert. Sie enthält die Startadressen sämtlicher CMD ".." Befehle.

Es gibt auch noch ältere ROM-Versionen, in denen diese Tabelle im ROM bei C178H-C1B7H steht und nicht ins RAM kopiert wird.

```
5B08 94 41      ;CMD ";" = 4194
5B0A 17 CC      ;CMD "<" = CC17
5B0C 94 41      ;CMD "=" = 4194
5B0E 94 41      ;CMD ">" = 4194
5B10 48 CC      ;CMD "?" = CC48
5B12 94 41      ;CMD "§" = 4194
5B14 94 41      ;CMD "A" = 4194
5B16 94 41      ;CMD "B" = 4194
5B18 61 C2      ;CMD "C" = C261
5B1A 59 C6      ;CMD "D" = C659
5B1C 46 C6      ;CMD "E" = C646
5B1E 88 DD      ;CMD "F" = DD88
5B20 5E CC      ;CMD "G" = CC5E
5B22 A9 C2      ;CMD "H" = C2A9
5B24 8B CC      ;CMD "I" = CC8B
5B26 17 CD      ;CMD "J" = CD17
5B28 94 41      ;CMD "K" = 4194
5B2A 58 CD      ;CMD "L" = CD58
5B2C 94 41      ;CMD "M" = 4194
5B2E 9D D5      ;CMD "N" = D59D
5B30 94 41      ;CMD "O" = 4194
5B32 94 41      ;CMD "P" = 4194
5B34 94 41      ;CMD "Q" = 4194
5B36 36 C6      ;CMD "R" = C636
5B38 62 CD      ;CMD "S" = CD62
5B3A 3D C6      ;CMD "T" = C63D
5B3C 74 CD      ;CMD "U" = CD74
5B3E 69 CD      ;CMD "V" = CD69
5B40 8B DC      ;CMD "W" = DC8B
5B42 85 CD      ;CMD "X" = CD85
5B44 68 CD      ;CMD "Y" = CD68
5B46 94 CD      ;CMD "Z" = CD94
```

9.5.2 Laufwerks-Parameter "A" - "L"

Die Tabelle mit den Laufwerks-Parametern "A" - "L" steht im ROM bei CEA0H bis CF17H. Sie enthält für jeden Laufwerkstyp 10D Bytes, die durch den Befehl CMD "< lw#-typ" in die folgenden RAM-Adressen kopiert werden:

Drive 0: 5A71 - 5A7A
Drive 1: 5A7B - 5A84
Drive 2: 5A85 - 5A8E
Drive 3: 5A8F - 5A98

Die Bedeutung dieser 10D Bytes ist mit den PDRIVE-Parametern in NEWDOS/80 und GDOS identisch und wurde bereits in Kapitel 4 erklärt.

```
                    ----------------------
                    Laufwerks-Parameter Typ "A"
CEA0 14 28 07 28 0A 02 00 00 05 02
                    ----------------------
                    Laufwerks-Parameter Typ "B"
CEAA 14 28 07 28 14 04 00 40 05 04
                    ----------------------
                    Laufwerks-Parameter Typ "C"
CEB4 18 30 53 28 12 03 00 03 05 03
                    ----------------------
                    Laufwerks-Parameter Typ "D"
CEBE 18 30 53 28 24 06 00 43 05 06
                    ----------------------
                    Laufwerks-Parameter Typ "E"
CEC8 14 28 07 28 0A 02 00 04 05 02
                    ----------------------
                    Laufwerks-Parameter Typ "F"
CED2 14 28 07 28 14 04 00 44 05 04
                    ----------------------
                    Laufwerks-Parameter Typ "G"
CEDC 18 30 53 28 12 03 00 07 05 03
                    ----------------------
                    Laufwerks-Parameter Typ "H"
CEE6 18 30 53 28 24 06 00 47 05 06
                    ----------------------
                    Laufwerks-Parameter Typ "I"
CEF0 28 50 07 50 0A 02 00 00 05 02
                    ----------------------
                    Laufwerks-Parameter Typ "J"
CEFA 28 50 07 50 14 04 00 40 05 04
                    ----------------------
                    Laufwerks-Parameter Typ "K"
CF04 30 60 53 50 12 03 00 03 05 03
                    ----------------------
                    Laufwerks-Parameter Typ "L"
CF0E 30 60 53 50 24 06 00 43 05 06
```

9.5.3 Belegung des Disk-RAM

Hier sind die wichtigsten RAM-Adressen, die das DOS benutzt, angegeben. Mit NEWDOS/80 Version 2 für Model I und GDOS besteht insofern eine Übereinstimmung, als der Bereich 4200H bis 439AH, den NEWDOS/80 und GDOS benutzen, unter Colour DOS nach 5900H bis 5A9AH verschoben ist.

In Klammern sind jeweils die Adressen aus NEWDOS/80 Version 2 für Model I angegeben.

4040-4046	(4040-4046)	Buffer für Datum und Uhrzeit
4050,4051	(44C9,44CA)	Zeiger auf Interrupt-Kette
405E	(4930)	zuletzt geladene DIR-Sector#
405F-4061	(49A6,49A7)	Rücksprungadresse für PUSHR (D2D9H)
4062	(4731)	Offset für Fehlercode in Sector I/O
4064	(46E8)	Buffer für 2. Zeichen in Sector I/O
4066	(486A)	DEC des letzten FDE's in UP EXPAND
4068,4069	(49CE,49CF)	geretteter SP in PUSHR (D2D9H)
406E,406F	(4C6A,4C6B)	Zeiger auf FCB bei LOADP (D4E5H)
4075	(46C4)	FDC-Kommando in Sector I/O
407B		Schalter KILL (01) oder CLOSE (00)
407C,407D	(4403,4404)	Startadresse von gelad. Programmen
43B6-43BB	(43AC-43B1)	Speicher für Datum und Uhrzeit
5900-59FF	(4200-42FF)	DOS-Buffer
5A00-5A03	(4300-4303)	aktuelle Track# der Drives 0-3
5A08	(4308)	aktuelle Drive# (0-3)
5A09	(4309)	Bitmuster für aktuelle Drive#
5A0A-5A11	(430A-4311)	PDRIVE-Parameter für aktuelle Drive
5A12	(4CB3)	Anzahl Sectoren pro GRAN
5A18-5A67	(4318-4367)	Buffer für CMD ".." Parameter
5A71-5A7A	(4371-437A)	PDRIVE-Parameter für Drive 0
5A7B-5A84	(437B-4384)	PDRIVE-Parameter für Drive 1
5A85-5A8E	(4385-438E)	PDRIVE-Parameter für Drive 2
5A8F-5A98	(438F-4398)	PDRIVE-Parameter für Drive 3
5A99,5A9A	(4399,439A)	Zeiger auf aktuellen PDRIVE-Block
5ACD-5AD7		Filespec-Buffer für OPEN (CE24H)
5AE0-5AFF		FCB für CMD "N" (Rename)
5B00-5B06	(43B2-43B8)	1. ROUTE Control Block
5B08-5B47		Befehlstabelle CMD ".."
5B48-5B67	(4480-449F)	FCB zum Laden von Programmen

9.5.4 Unterprogramme und Ansprungadressen

Die meisten Unterprogramme aus SYS0/SYS in NEWDOS/80 Version 2 für TRS-80 Model I, die auch im Colour DOS vorhanden sind, arbeiten zwar dort in einem anderen Adressbereich, sind aber sonst im wesentlichen gleich. Die folgende Tabelle gibt über die unterschiedlichen Adressen Aufschluß.

In Klammern sind jeweils die Adressen aus NEWDOS/80 Version 2 für Model I angegeben.

Nach Adressen sortiert:

0013	READB	(0013)	liest ein Byte aus einer File (siehe D430H)
001B	WRITEB	(001B)	schreibt ein Byte in eine File (siehe D3D5H)
CDF7	UPCASE	(45B5)	Kleinbuchstaben in Großbuchstaben umwandeln
CE09	DOSERR	(4409)	Fehlermeldung des DOS ausgeben
CE10	INTINS	(4410)	Benutzer-Interrupt-Routine einfügen
CE13	INTDEL	(4413)	Benutzer-Interrupt-Routine löschen
CE16	MOTONX	(4416)	Drive-Motoren weiterlaufen lassen
CE1C	TFSPEC	(441C)	Filespec (HL) nach FCB (DE) übertragen
CE20	INIT	(4420)	File öffnen, ggf. neue File anlegen
CE24	OPEN	(4424)	File öffnen, ggf. keine neue File anlegen
CE28	CLOSE	(4428)	File schließen
CE2C	KILL	(442C)	File löschen
CE30	LOAD	(4430)	Programm laden
CE33	RUN	(4433)	Programm laden und starten (keine Rückkehr)
CE36	READ	(4436)	nächsten Sector/Record aus einer File lesen
CE39	WRITE	(4439)	nächsten Sector/Record in eine File schreiben
CE3C	VERIFY	(443C)	Sector/Record in eine File schreiben + testen
CE3F	POS0	(443F)	FCB auf Beginn einer File positionieren
CE42	POSBC	(4442)	FCB auf Logische Record# (BC) positionieren
CE45	POSDEC	(4445)	FCB um 1 Record zurückpositionieren
CE48	POSEOF	(4448)	FCB auf Ende der File (EOF) positionieren
CE4B	EXPAND	(444B)	File erweitern, wenn File zu kurz ist
CE4E	POSRBA	(444E)	FCB auf RBA-Format (HLC) positionieren
CE51	PUTEOF	(4451)	EOF im FPDE auf den neuesten Stand bringen
CE5B	DRVSEL	(445B)	Drive auswählen und Motor starten
CE5E	TSTDSK	(445E)	Drive auswählen, prüfen ob Diskette eingelegt
CE67	TEXTTV	(4467)	Text auf Bildschirm ausgeben
CE6A	TEXTLP	(446A)	Text auf Drucker ausgeben
CE6D	TIME	(446D)	Uhrzeit im Format HH:MM:SS in Buffer ablegen
CE70	DATE	(4470)	Datum im Format MM/TT/JJ in Buffer ablegen
CE76	MULTHL	(4C94)	Multipliziere AHL = HL*A
CE79	DIVA	(4CB4)	Dividiere HL=INT(HL/A), A=Rest
CE7C	HEXDE	(4063)	DE hexadezimal nach (HL) ausgeben
CF6F	READS	(4630)	Sector von Diskette lesen
CF73	TESTS	(4634)	Sector auf Lesbarkeit testen
CF7B	WRITDS	(463C)	Directory-Sector auf Diskette schreiben
CF7F	WRITES	(4640)	normalen Sector auf Diskette schreiben
D099	RESTOR	(4745)	RESTORE-Kommando an FDC senden
D09B	FDCCMD	(4747)	Kommando mit richtiger TSR an FDC senden
D0A2	WNBUSY	(4750)	warten, bis der FDC nicht mehr busy ist
D0B2	FBREAK	(475E)	FORCE-INTERRUPT-Kommando an FDC senden
D0BB	MOTON	(4767)	Drive-Motoren starten

D137	DELAY1	(47E3)	ca. 60 us warten und Status des FDC lesen
D168	FILPOS	(4810)	Disk-Position eines File-Sectors berechnen
D25F	DIRR	(490A)	liest einen Sector des Directory
D274	DIRW	(491F)	schreibt einen Sector des Directory
D28F	GETFDE	(4936)	holt einen FDE aus dem Directory
D2C1	NXTEOF	(4968)	NEXT-Wert aus FCB holen + mit EOF vergleichen
D2D9	PUSHR	(4980)	Push Register, IX=FCB+0, IY=5A80H, A=00
D3F5	WRITXV	(4AB8)	Sector oder DIR-Sector schreiben, ggf. Verify
D3FA	WRITEV	(4ABD)	normalen Sector schreiben, ggf. Verify
D407	WRITDV	(4ACA)	Directory-Sector schreiben, ggf. Verify
D4E5	LOADP	(4C28)	Programm laden
D532	DIRPOS	(4C74)	Disk-Position eines DIR-Sectors berechnen
D551	MULTL	(4C92)	Multipliziere AHL = L*A
D55C	MULTC	(4C9D)	Multipliziere HLC = HL*A
D585	NEXTC1	(4CD5)	Nächstes Zeichen holen und Flags setzen
D589	NEXTC2	(4CD9)	Nächstes Zeichen holen und Flags setzen

Alphabetisch sortiert:

CLOSE	CE28	(4428)	File schließen
DATE	CE70	(4470)	Datum im Format MM/TT/JJ in Buffer ablegen
DELAY1	D137	(47E3)	ca. 60 us warten und Status des FDC lesen
DIRPOS	D532	(4C74)	Disk-Position eines DIR-Sectors berechnen
DIRR	D25F	(490A)	liest einen Sector des Directory
DIRW	D274	(491F)	schreibt einen Sector des Directory
DIVA	CE79	(4CB4)	Dividiere HL=INT(HL/A), A=Rest
DOSERR	CE09	(4409)	Fehlermeldung des DOS ausgeben
DRVSEL	CE5B	(445B)	Drive auswählen und Motor starten
EXPAND	CE4B	(444B)	File erweitern, wenn File zu kurz ist
FBREAK	D0B2	(475E)	FORCE-INTERRUPT-Kommando an FDC senden
FDCCMD	D09B	(4747)	Kommando mit richtiger TSR an FDC senden
FILPOS	D168	(4810)	Disk-Position eines File-Sectors berechnen
GETFDE	D28F	(4936)	holt einen FDE aus dem Directory
HEXDE	CE7C	(4063)	DE hexadezimal nach (HL) ausgeben
INIT	CE20	(4420)	File öffnen, ggf. neue File anlegen
INTDEL	CE13	(4413)	Benutzer-Interrupt-Routine löschen
INTINS	CE10	(4410)	Benutzer-Interrupt-Routine einfügen
KILL	CE2C	(442C)	File löschen
LOAD	CE30	(4430)	Programm laden
LOADP	D4E5	(4C28)	Programm laden
MOTON	D0BB	(4767)	Drive-Motoren starten
MOTONX	CE16	(4416)	Drive-Motoren weiterlaufen lassen
MULTC	D55C	(4C9D)	Multipliziere HLC = HL*A
MULTHL	CE76	(4C94)	Multipliziere AHL = HL*A
MULTL	D551	(4C92)	Multipliziere AHL = L*A
NEXTC1	D585	(4CD5)	Nächstes Zeichen holen und Flags setzen
NEXTC2	D589	(4CD9)	Nächstes Zeichen holen und Flags setzen
NXTEOF	D2C1	(4968)	NEXT-Wert aus FCB holen + mit EOF vergleichen
OPEN	CE24	(4424)	File öffnen, ggf. keine neue File anlegen
POS0	CE3F	(443F)	FCB auf Beginn einer File positionieren
POSBC	CE42	(4442)	FCB auf Logische Record# (BC) positionieren
POSDEC	CE45	(4445)	FCB um 1 Record zurückpositionieren
POSEOF	CE48	(4448)	FCB auf Ende der File (EOF) positionieren
POSRBA	CE4E	(444E)	FCB auf RBA-Format (HLC) positionieren
PUSHR	D2D9	(4980)	Push Register, IX=FCB+0, IY=5A80H, A=00
PUTEOF	CE51	(4451)	EOF im FPDE auf den neuesten Stand bringen

READ	CE36	(4436)	nächsten Sector/Record aus einer File lesen
READB	0013	(0013)	liest ein Byte aus einer File (siehe D430H)
READS	CF6F	(4630)	Sector von Diskette lesen
RESTOR	D099	(4745)	RESTORE-Kommando an FDC senden
RUN	CE33	(4433)	Programm laden und starten (keine Rückkehr)
TESTS	CF73	(4634)	Sector auf Lesbarkeit testen
TEXTLP	CE6A	(446A)	Text auf Drucker ausgeben
TEXTTV	CE67	(4467)	Text auf Bildschirm ausgeben
TFSPEC	CE1C	(441C)	Filespec (HL) nach FCB (DE) übertragen
TIME	CE6D	(446D)	Uhrzeit im Format HH:MM:SS in Buffer ablegen
TSTDSK	CE5E	(445E)	Drive auswählen, prüfen ob Diskette eingelegt
UPCASE	CDF7	(45B5)	Kleinbuchstaben in Großbuchstaben umwandeln
VERIFY	CE3C	(443C)	Sector/Record in eine File schreiben + testen
WNBUSY	D0A2	(4750)	warten, bis der FDC nicht mehr busy ist
WRITDS	CF7B	(463C)	Directory-Sector auf Diskette schreiben
WRITDV	D407	(4ACA)	Directory-Sector schreiben, ggf. Verify
WRITE	CE39	(4439)	nächsten Sector/Record in eine File schreiben
WRITEB	001B	(001B)	schreibt ein Byte in eine File (siehe D3D5H)
WRITES	CF7F	(4640)	normalen Sector auf Diskette schreiben
WRITEV	D3FA	(4ABD)	normalen Sector schreiben, ggf. Verify
WRITXV	D3F5	(4AB8)	Sector oder DIR-Sector schreiben, ggf. Verify

Literaturverzeichnis

WESTERN DIGITAL Corporation, California 92663:

- FD 1771B Product Guide
- FD 1771-01 Disk Formatter / Controller
- FD 1771 Application Note
- FD 179X-01 Floppy Disk Formatter / Controller Family
- 179X-01 Application Notes

NATIONAL SEMICONDUCTORS Corporation, California 95051:

- INS 1771-1 Floppy Disk Formatter / Controller

Luidger Röckrath, 5100 Aachen:

- TRS-80 Model I ROM-Listing
- Assemblertricks auf dem Colour-Genie

Hartmut Grosser & Luidger Röckrath, 5100 Aachen:

- TRS-80 Model III ROM-Listing

Norbert Heicke & Luidger Röckrath, 5100 Aachen:

- Colour-Genie ROM-Listing

TANDY Corporation, Fort Worth, Texas 76102:

- TRS-80 Expansion Interface Hardware Manual
- TRS-80 Model III Technical Reference Manual

APPARAT Incorporated, Denver, Colorado 80237:

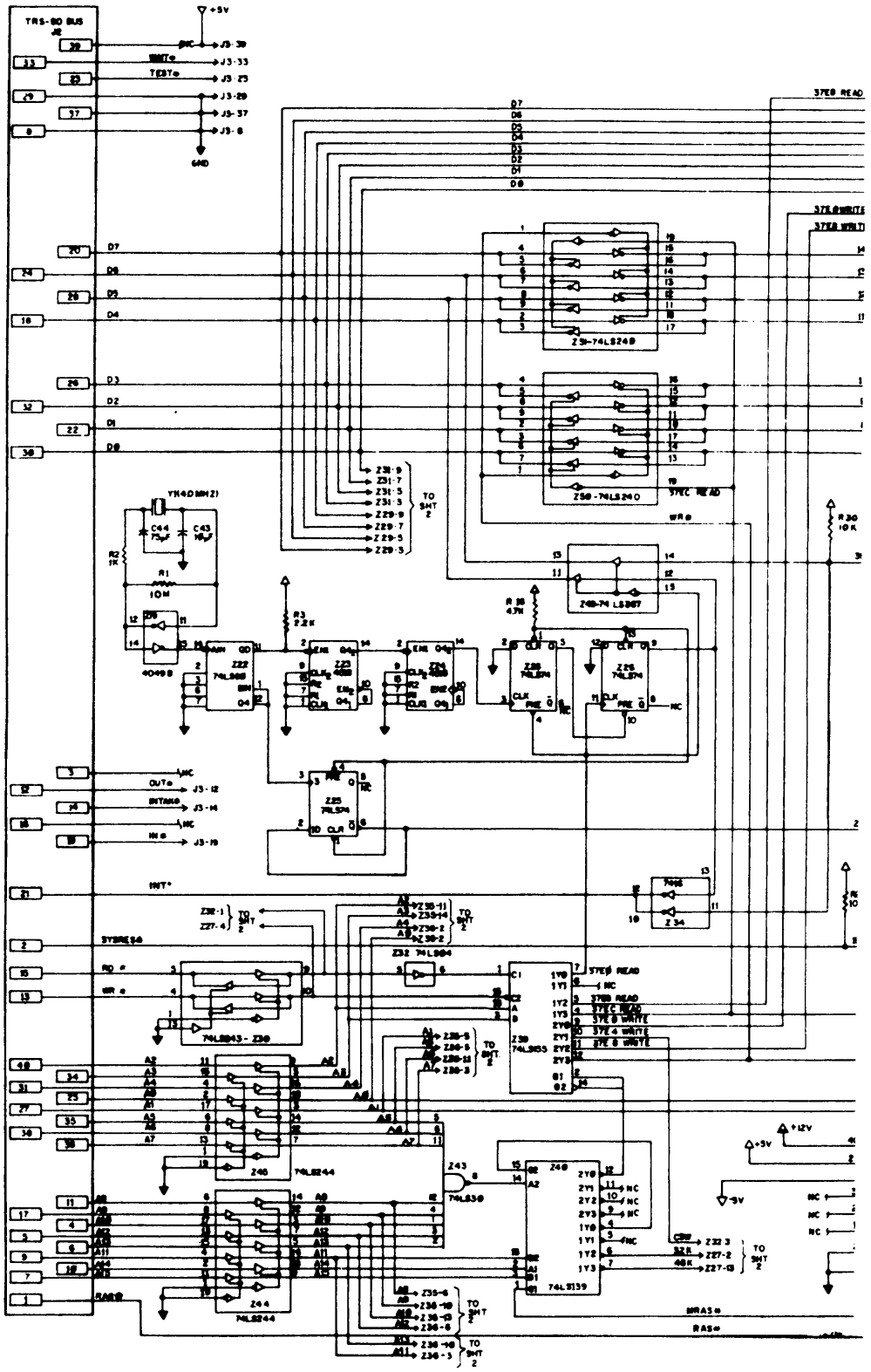
- NEWDOS/80 Operation Manual

TCS GmbH, St. Augustin:

- Colour-Genie Disk-BASIC
- Technische Beschreibung zum Genie IIIs

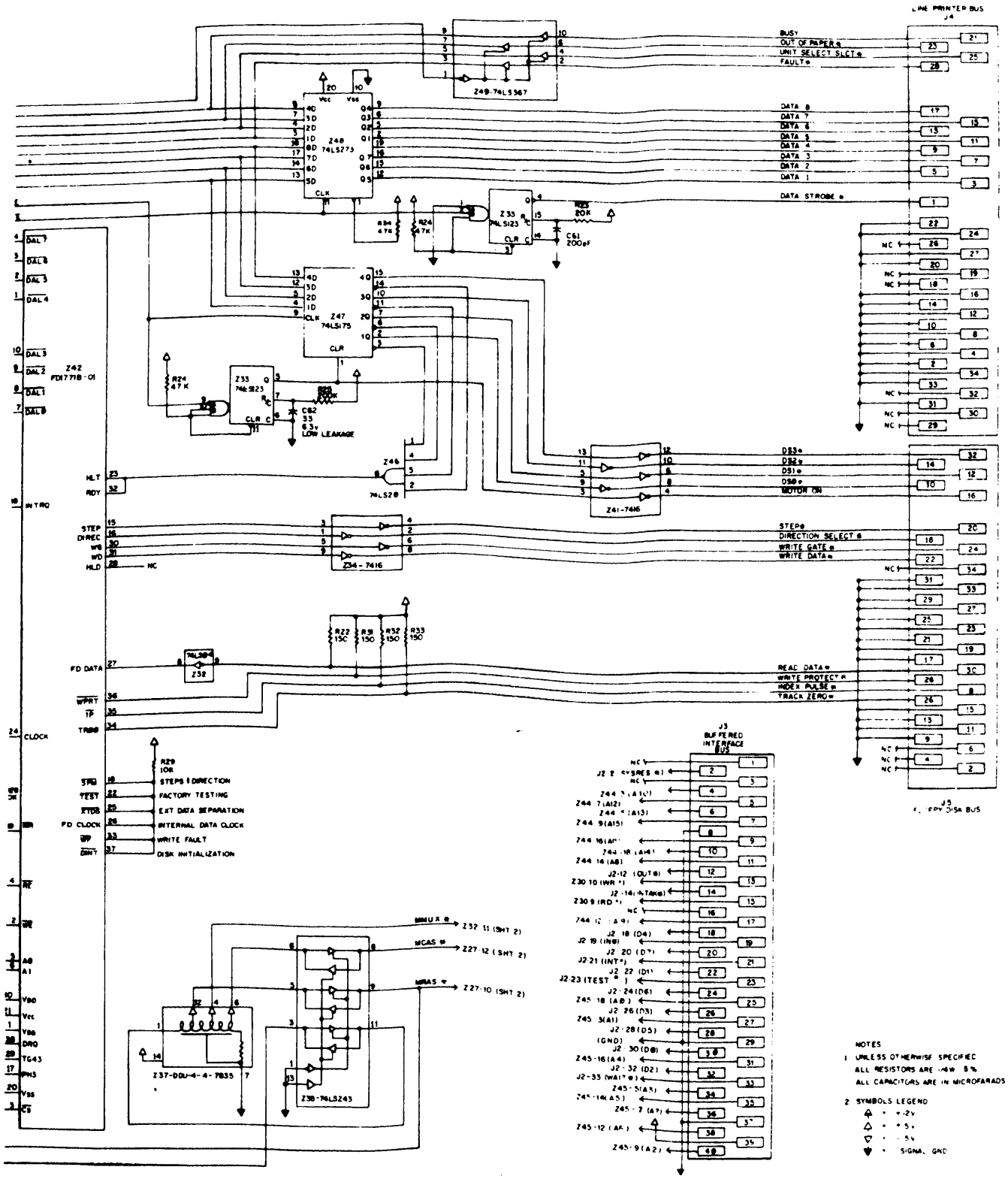
EACA Computer Ltd., Hong Kong:

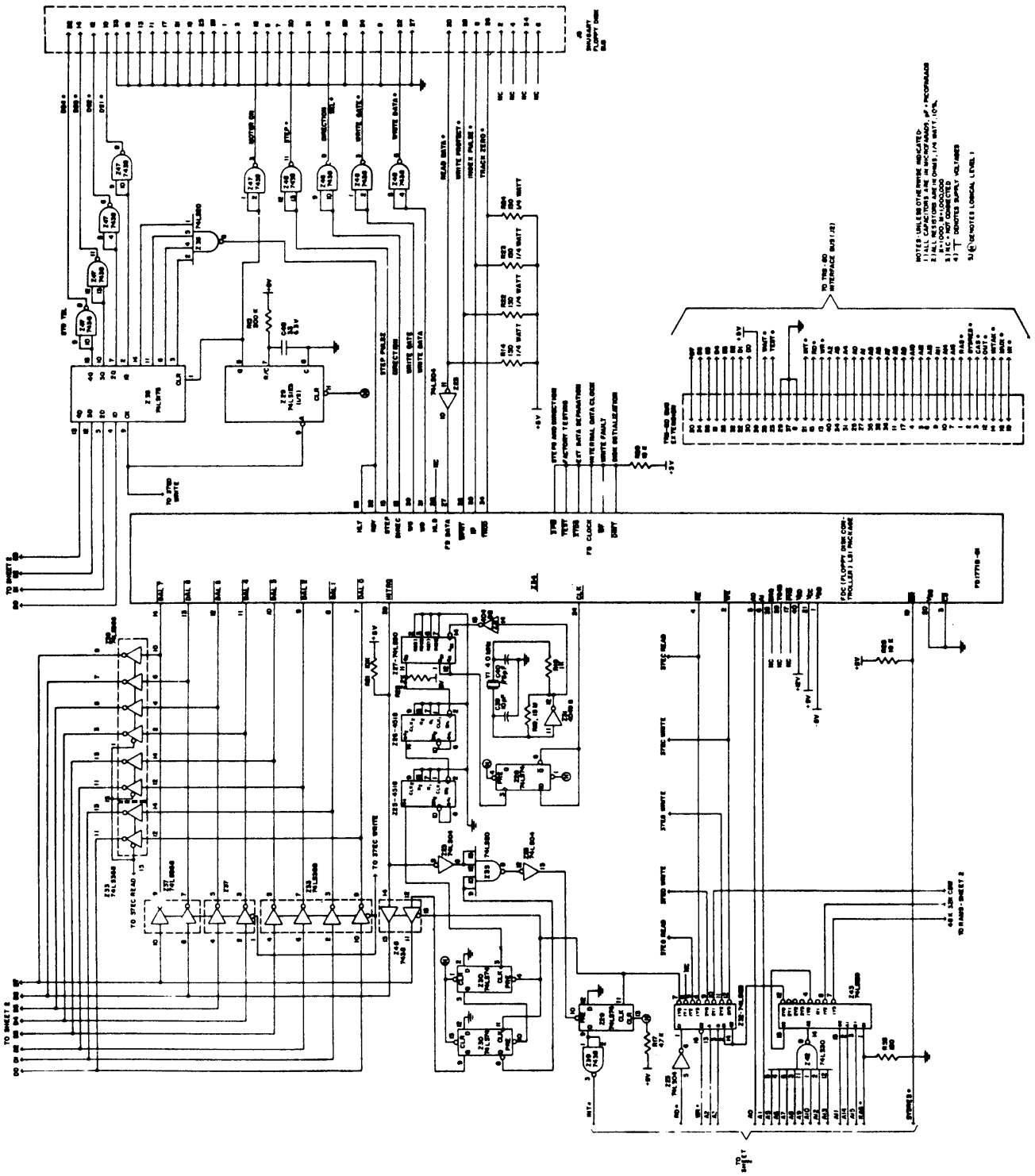
- Genie III User's Manual



Expansion Interface schematic diagram

Schematic Diagrams





Expansion Interface schematic diagram

WESTERN DIGITAL MOS/LSI

FD1771B

PRODUCT GUIDE FLOPPY DISK FORMATTER/CONTROLLER

GENERAL DESCRIPTION

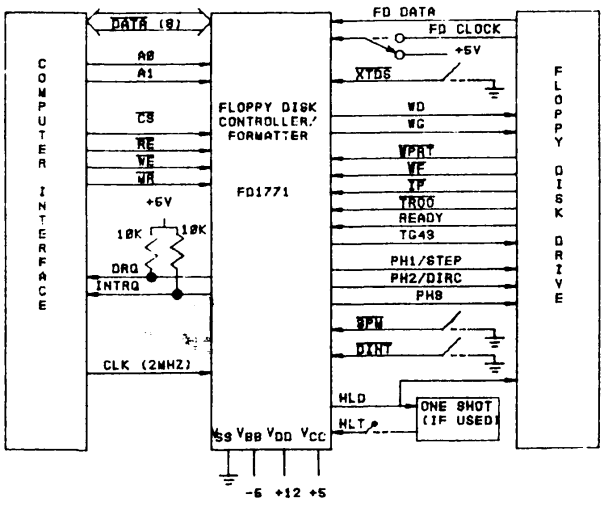
The FD1771B is a MOS/LSI device that performs the functions of a Floppy Disk Controller/Formatter. The device is designed to be included in the disk drive electronics, and contains a flexible interface organization that accomodates the interface signals from most drive manufacturers. The FD1771B is compatible with the IBM 3740 data entry system format.

The processor interface consists of an 8-bit bi-directional bus for data, status, and control word transfers. The FD1771B is set up to operate on a multiplexed bus with other bus-oriented devices.

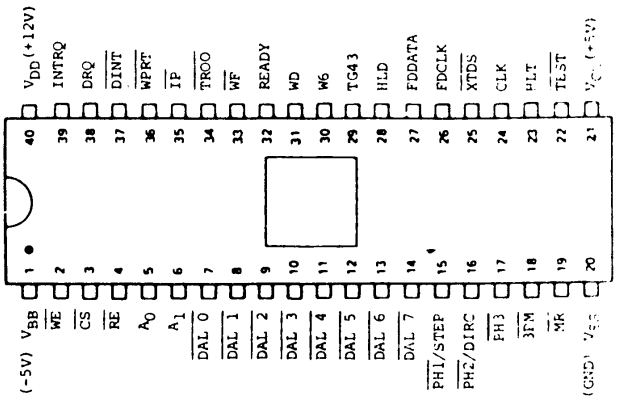
The FD1771B is fabricated in N-channel Silicon Gate MOS technology and is TTL compatible on all inputs and outputs.

FEATURES

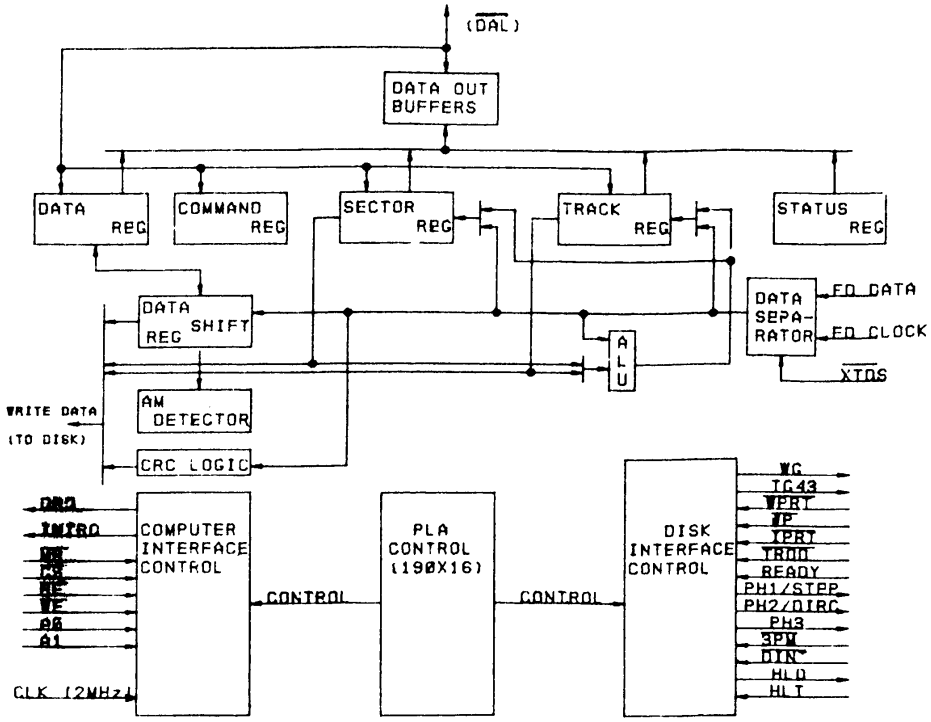
- o SOFT SECTOR FORMAT COMPATIBILITY
- o AUTOMATIC TRACK SEEK WITH VERIFICATION
- o READ MODE
  - Single/Multiple Record Read with Automatic Sector Search or Entire Track Read
  - Selectable 128 Byte or Variable Length Record
- o WRITE MODE
  - Single/Multiple Record Write with Automatic Sector Search
  - Entire Track Write for Diskette Initialization
- o PROGRAMMABLE CONTROLS
  - Selectable Track to Track Stepping Time
  - Selectable Head Settling and Head Engage Times
  - Selectable Three Phase or Step and Direction Motor Controls
- o SYSTEM COMPATIBILITY
  - Double Buffering of Data 8 Bit Bi-Directional Bus for Data, Control and Status
  - DMA or Programmed Data Transfers
  - All Inputs and Outputs are TTL Compatible



PIN CONNECTIONS



WESTERN DIGITAL FD1771B



FD1771 BLOCK DIAGRAM

ORGANIZATION

The Floppy Disk Formatter block diagram is illustrated above. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register - This 8-bit register assembles serial data from the Read Data input (FDDATA) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register - This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command the Data Register holds the address of the desired Track position. This register can be loaded from the DAL and gated onto the DAL under processor control.

Track Register - This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in (towards track 76) and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL.

WESTERN DIGITAL FD1771B

ORGANIZATION (continued)

Sector Register (SR) - This 8 bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL.

Command Register (CR) - This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the execution of the current command is to be over-ridden. This latter action results in an interrupt. The command register can be loaded from the DAL, but not read onto the DAL.

Status Register (STR) - This 8-bit register holds device Status information. The meaning of the Status bits are a function of the contents of the Command Register. This register can be read onto the DAL, but not loaded from the DAL.

CRC Logic - This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is: $G(x) = x^{16} + x^{12} + x^5 + 1$.

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) - The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

AM Detector - The Address Mark detector is used to detect ID, Data, and Index address marks during Read and Write operations.

Timing and Control - All computer and Floppy Disk interface controls are generated through this logic. The internal device timing is generated from a 2.0 MHz external crystal clock.

PROCESSOR INTERFACE (SEE BLOCK DIAGRAM)

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer Data, Status, and Control words out of, or into the FD1771. The DAL are three state buffers that are enabled as output drivers when Chip Select (CS) and Read Enable ($\overline{RE}$) are active (low logic state) or act as input receivers when CS and Write Enable ($\overline{WE}$) are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and CS is made low. The least-significant address bits A1 and AØ, combined with the signals $\overline{RE}$ during a Read operation or $\overline{WE}$ during a Write operation are interpreted as selecting the following registers:

A1-AØ	READ ($\overline{RE}$)	WRITE ($\overline{WE}$)
Ø Ø	Status Register	Command Register
Ø 1	Track Register	Track Register
1 Ø	Sector Register	Sector Register
1 1	Data Register	Data Register

WESTERN DIGITAL FD1771B

PROCESSOR INTERFACE (continued)

During Direct Memory Access (DMA) types of data transfers between the Data Register of the FD1771 and the processor, the Data Request (DRQ) output is used in Data Transfer control.

Read

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operation continues until the end of sector is reached.

Write

On Disk Write operations the Data Request is activated when the Data Register transfers its contents to the Data Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit is set in the Status Register.

The Lost Data bit and certain other bits in the Status Register will activate the interrupt request (INTRQ). The interrupt line is also activated with normal completion or abnormal termination of all controller operations. The INTRQ signal remains active until reset by reading the Status Register to the processor or by the loading of the Command Register. In addition, the INTRQ is generated if a Force Interrupt command condition is met.

2

FLOPPY DISK INTERFACE

The Floppy Disk interface consists of head positioning controls, write gate controls, and data transfers. A 2.0 MHz ± 1% square wave clock is required at the CLK input for internal control timing.

HEAD POSITIONING

Four commands cause positioning of the Read-Write head (see Command Section). The period of each positioning step is specified by the r field in bits 1 and Ø of the command word. After the last directional step an additional 10 milliseconds of head settling time takes place. The three programmable stepping rates are tabulated below. The rates can be applied to a Three Phase Motor or a Step-Direction Motor through the device interface. When the 3PM input is connected to ground the device operates with a three-phase motor control interface with one active low signal per Phase on the three output signals PH1, PH2 and PH3. The stepping sequence, when stepping in, is Phases 1-2-3-1, and when stepping out, Phases 1-3-2-1. Phase 1 is active low after Master Reset.

r1 (bit 1)	rØ (bit Ø)	Period (ms)	Rate (steps per second)
Ø	Ø	6	166
Ø	1	6	166
1	Ø	8	125
1	1	10	100

HEAD POSITIONING (continued)

The Step-Direction Motor Control interface is activated by leaving input 3PM open or connecting it to +5V. The Phase 1 pin PH1 becomes a Step pulse of 4 microseconds width. The Phase 2 pin PH2 becomes a direction control with a high voltage on this pin indicating a Step In, and a low voltage indicating a Step Out. The Direction output is valid a minimum of 24 µs prior to the activation of the Step pulse.

When a Seek, Step or Restore command is executed an optional verification of Read-Write head position can be performed by setting bit 2 in the command word to a logic 1. The verification operation begins at the end of the 10 millisecond settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete. If track comparison is not made but the CRC checks, an interrupt is generated, the Seek Error status (Bit 4) is set and the Busy status bit is reset. If no verification is received, a Seek error (Bit 4) is set in the status word.

The Head Load (HDL) output controls the movement of the read/write head against the disk for data recording or retireval. It is activated at the beginning of a Read, Write (E Flag On) or Verify Operation, or a Seek or Step operation with the head load bit, h, a logic one and remains activated until the third index pulse following the last operation which uses the Read, Write head. Reading or Writing does not occur until a minimum of 10 msec delay after the HDL signal is made active. If executing the type 2 commands with the E flag off, there is no 10 msec delay and the head is assumed to be engaged. The delay is determined by sampling of the Head Load Timing (HLT) input after 10 msec. A high logic state input, generated from the Head Load output transition and delayed externally, identifies engagement of the head against the disk. In the Seek and Step commands, the head is loaded at the start of the command execution when the h bit is a logic one. In a verify command the head is loaded before stepping to the last track on the disk whenever the h bit is a logic zero.

DISK READ OPERATION

The 2.0 MHz external clock provided to the device is internally divided by 4 to form the 500 KHz clock rate for data transfer. When reading data from a diskette this divider is synchronized to transitions of the Read Data (FDDATA) input and the FDCLK input.
The 500 KHz data clock is further divided by 2 internally to separate the clock and information bits. The divider is phased to the information by the detection of the address mark.

In the internal data read and separation mode the Read Data input toggles from one state to the opposite state for each logic one bit of clock or information. This signal can be derived from the amplified, differentiated, and sliced Read Head signal, or by the output of a flip-flop toggling on the Read Data pulses. This input is sampled by the 2 MHz clock to detect transitions.

The chip can also operate on externally separated data, as supplied by methods such as Phase Lock loop, One Shots, or variable frequency oscillators. This is accomplished by grounding the External Data Separator (XTDS) input. When the Read Data input makes a high-to-low transition, the information input to

the FDDATA line is clocked into the Data Shift Register. The assembled 8 bit data from the Data Shift Register are then transferred to the Data Register.

The normal sector length for Read or Write operations with the IBM 3740 format is 128 bytes. This format or binary multiples of 128 bytes will be adopted by setting a logic 1 in Bit 3 of the Read(Track) and Write (Track) commands. Additionally, a variable sector length feature is provided which allows an indicator recorded in the ID Field to control the length of the sector. Variable sector lengths can be read or written in Read or Write commands respectively by setting a logic 0 in Bit 3 of the command word. The sector length indicator specifies the number of 16 byte groups or 16 X N, where N is equal to 1 to 256 groups. An indicator of all zeroes is interpreted as 256 sixteen byte groups.

DISK WRITE OPERATION

After data is loaded from the processor into the Data Register, and is transferred to the Data Shift Register, data will be shifted serially through the Write Data (WD) output. Interlaced with each bit of data is a positive clock pulse of 0.5 usec duration. This signal may be used to externally toggle a flip-flop to control the direction of Write Current flow.

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the FD1771 before the Write Gate signal can be activated.

Writing is inhibited when the $\overline{\text{Write Protect}}$ input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set. The Write Fault input, when activated, signifies a writing fault condition detected in disk drive electronics such as failure to detect write current flow when the Write Gate is activated. On detection of this fault the FD1771 terminates the current command, and sets the Write Fault bit (bit5) In the Status Word. The Write Fault input should be made inactive when the Write Gate output becomes inactive.

Whenever a Read or Write command is received the FD1771 samples the READY input. If this input is logic low the command is not executed and an interrupt is generated. The Seek or Step commands are performed regardless of the state of the READY input.

COMMAND DESCRIPTION

The FD1771 will accept and execute eleven commands. Command words whould only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrupt command. Whenever a command

COMMAND SUMMARY

		BITS							
TYPE	COMMAND	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step In	0	1	0	u	h	V	r_1	r_0
I	Step Out	0	1	1	u	h	V	r_1	r_0
II	Read Command	1	0	0	m	b	E	0	0
II	Write Command	1	0	1	m	b	E	a_1	a_0
III	Read Address	1	1	0	0	0	1	0	0
III	Read Track	1	1	1	0	0	1	0	$\overline{s}$
III	Write Track	1	1	1	1	0	1	0	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

WESTERN DIGITAL FD1771B

COMMAND DESCRIPTION (continued)

is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in the Command Summary Table.

COMMAND FLAG SUMMARY

TYPE I

h = Head Load flag (Bit 3)

h=1, Load head at beginning
h=0, Do not load head at beginning

V = Verify flag (Bit 2)

V=1, Verify on last track
V=0, No verify

r_1r_0 = Stepping motor rate (Bits 1-0)

r_1r_0=00, 6ms between steps
r_1r_0=01, 6ms between steps
r_1r_0=10, 8ms between steps
r_1r_0=11, 10ms between steps

u = Update flag (Bit 4)

u=1, Update Track register
u=0, No update

TYPE II

m = Multiple Record flag (Bit 4)

m=0, Single Record
m=1, Multiple Records

b = Block length flag (Bit 3)

b=1, IBM format (128 to 1024 bytes)
b=0, Non-IBM format (16 to 4096 bytes)

a_1 a_0 = Data Address Mark (Bits 1-0)

a_1 a_0 = 00, FB (Data Mark)
a_1 a_0 = 01, FA (Data Mark)
a_1 a_0 = 10, F9 (Data Mark)
a_1 a_0 = 11, F8 (Data Mark)

TYPE III

s = Synchronize flag (Bit 0)

$\overline{s}$=0, Synchronize to AM
$\overline{s}$=1, Do Not Synchronize to AM

TYPE IV

Ii = Interrupt Condition flags (Bits 3-0)

I_0=1, Not Ready to Ready Transition
I_1=1, Ready to Not Ready Transition
I_2=1, Index Pulse
I_3=1, Every 10 ms

E = Enable HLD and 10 msec Delay

E=1, Enable HLD, HLT and 10 msec Delay
E=0, Head is assumed Engaged and there is no 10 msec Delay

STATUS DESCRIPTION

Upon receipt of any command, except the Force Interrupt command, the BUSY Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the BUSY status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, the BUSY Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I Commands.

WESTERN DIGITAL FD1771B

The format of the Status Register is shown below:

7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Status varies according to the type of command executed as shown in Table below.

YOUR LOCAL WDC REPRESENTATIVE IS:

STATUS REGISTER SUMMARY

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ	READ TRACK	WRITE	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	RECORD TYPE	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD ENGAGED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	ID NOT FOUND	RECORD NOT FOUND	0	RECORD NOT FOUND	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	NOT VALID	CRC ERROR	NOT VALID
S2	TRACK 0	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX	0	0	0	0	0
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

V_{DD} With Respect to V_{BB} (Ground)	+20 to -0.3V
Max Voltage to Any Input With Respect to V_{BB}	+20 to -0.3V
Operating Temperature	$0^{\circ}C$ to $70^{\circ}C$
Storage Temperature	$-55^{\circ}C$ to $+125^{\circ}C$

OPERATING CHARACTERISTICS (DC)

$T_A = \emptyset^{\circ}C$ to $7\emptyset^{\circ}C$, $V_{DD} = +12.\emptyset V \pm .6V$, $V_{BB} = -5.\emptyset \pm .5V$,

$V_{SS} = \emptyset V$, $V_{CC} = +5V \pm .25V$

V_{DD} = 8ma Nominal, V_{CC} = 28ma Nominal, V_{BB} = 0.4 ua Nominal

WESTERN DIGITAL
CORPORATION

FD 179X-01 Floppy Disk Formatter/Controller Family

FEATURES

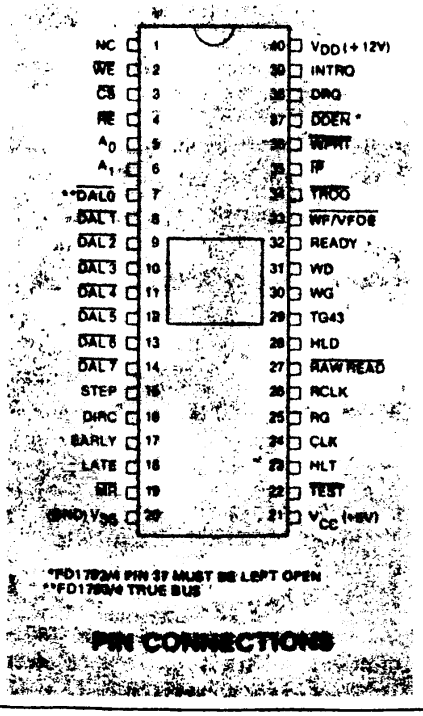
- TWO VFO CONTROL SIGNALS
- SOFT SECTOR FORMAT COMPATIBILITY
- AUTOMATIC TRACK SEEK WITH VERIFICATION
- ACCOMMODATES SINGLE AND DOUBLE DENSITY FORMATS
  - IBM 3740 Single Density (FM)
  - IBM System 34 Double Density (MFM)
- READ MODE
  - Single/Multiple Sector Read with Automatic Search or Entire Track Read
  - Selectable 128 Byte or Variable length Sector
- WRITE MODE
  - Single/Multiple Sector Write with Automatic Sector Search
  - Entire Track Write for Diskette Formatting
- SYSTEM COMPATIBILITY
  - Double Buffering of Data 8 Bit Bi-Directional Bus for Data, Control and Status
  - DMA or Programmed Data Transfers
  - All Inputs and Outputs are TTL Compatible
  - On-Chip Track and Sector Registers/Comprehensive Status Information
- PROGRAMMABLE CONTROLS
  - Selectable Track to Track Stepping Time
  - Side Select Compare
- WRITE PRECOMPENSATION (MFM AND FM)
- WINDOW EXTENSION
- INCORPORATES ENCODING/DECODING AND ADDRESS MARK CIRCUITRY
- FD1792/4 IS SINGLE DENSITY ONLY
- FD1793/4 HAS TRUE DAL LINES

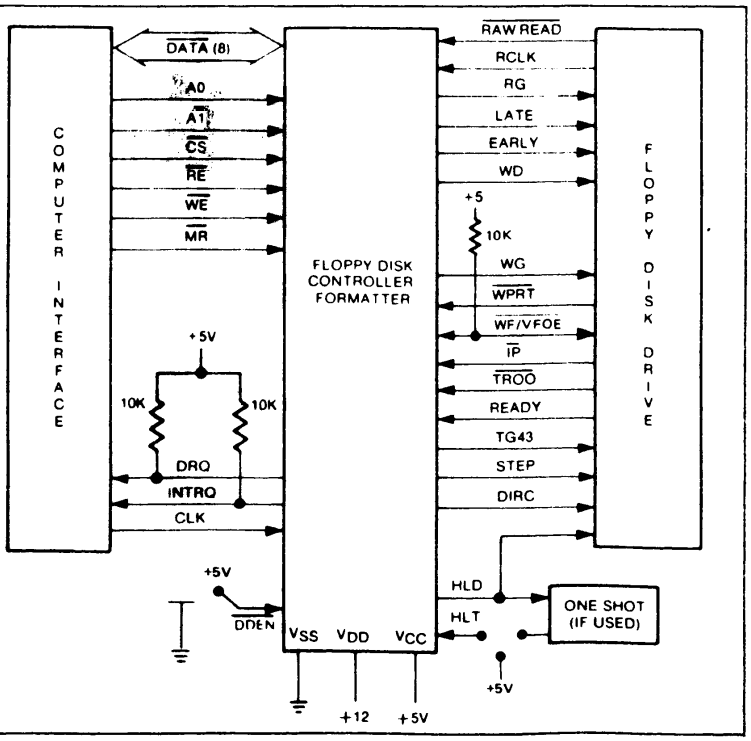
179X-01 FAMILY CHARACTERISTICS

FEATURES	1791-01	1792-01	1793-01	1794-01
Single Density (FM)	X	X	X	X
Double Density (MFM)	X		X	
True Data Bus			X	X
Inverted Data Bus	X	X		
Write Precomp	X	X	X	X
Window Extension	X	X	X	X

APPLICATIONS

FLOPPY DISK DRIVE INTERFACE
SINGLE OR MULTIPLE DRIVE CONTROLLER/ FORMATTER
NEW MINI-FLOPPY CONTROLLER





FD179X SYSTEM BLOCK DIAGRAM

GENERAL DESCRIPTION

The FD179X are MOS LSI devices which perform the functions of a Floppy Disk Formatter/Controller in a single chip implementation. The FD179X, which can be considered the end result of both the FD1771 and FD1781 designs, is IBM 3740 compatible in single density mode (FM) and System 34 compatible in Double Density Mode (MFM). The FD179X contains all the features of its predecessor the FD1771, plus the added features necessary to read/write and format a double density diskette. These include address mark detection, FM and MFM encode and decode logic, window extension, and write precompensation. In order to maintain compatibility, the FD1771, FD1781, and FD179X designs were made as close as possible with the computer interface, instruction set, and I/O registers being identical. Also, head load control is identical. In each case, the actual pin assignments vary by only a few pins from any one to another.

The processor interface consists of an 8-bit bidirectional bus for data, status, and control word transfers. The FD179X is set up to operate on a multiplexed bus with other bus-oriented devices.

The FD179X is fabricated in N-channel Silicon Gate MOS technology and is TTL compatible on all inputs and outputs. The 1793 is identical to the 1791 except the DAL lines are TRUE for systems that utilize true data busses.

PIN OUTS

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
1	NO CONNECTION	NC	Pin 1 is internally connected to a back bias generator and must be left open by the user.
19	MASTER RESET	$\overline{MR}$	A logic low on this input resets the device and loads HEX 03 into the command register. The Not Ready (Status Bit 7) is reset during $\overline{MR}$ ACTIVE. When $\overline{MR}$ is brought to a logic high a RESTORE Command is executed, regardless of the state of the Ready signal from the drive. Also, HEX 01 is loaded into sector register.
20	POWER SUPPLIES	V_{SS}	Ground
21		V_{CC}	+5V ±5%
40		V_{DD}	+12V ±5%
COMPUTER INTERFACE:			
2	WRITE ENABLE	$\overline{WE}$	A logic low on this input gates data on the DAL into the selected register when $\overline{CS}$ is low.
3	CHIP SELECT	$\overline{CS}$	A logic low on this input selects the chip and enables computer communication with the device.
4	READ ENABLE	$\overline{RE}$	A logic low on this input controls the placement of data from a selected register on the DAL when $\overline{CS}$ is low.
5,6	REGISTER SELECT LINES	A0, A1	These inputs select the register to receive/transfer data on the DAL lines under $\overline{RE}$ and $\overline{WE}$ control: A1 A0 $\overline{RE}$ $\overline{WE}$ 0 0 Status Reg Command Reg 0 1 Track Reg Track Reg 1 0 Sector Reg Sector Reg 1 1 Data Reg Data Reg
7-14	DATA ACCESS LINES	$\overline{DAL0}$-$\overline{DAL7}$	Eight bit inverted Bidirectional bus used for transfer of data, control, and status. This bus is receiver enabled by $\overline{WE}$ or transmitter enabled by $\overline{RE}$.
24	CLOCK	CLK	This input requires a free-running square wave clock for internal timing reference, 2 MHz for 8" drives, 1 MHz for mini-drives.

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
38	DATA REQUEST	DRQ	This open drain output indicates that the DR contains assembled data in Read operations, or the DR is empty in Write operations. This signal is reset when serviced by the computer through reading or loading the DR in Read or Write operations, respectively. Use 10K pull-up resistor to +5.
39	INTERRUPT REQUEST	INTRQ	This open drain output is set at the completion of any command and is reset when the STATUS register is read or the command register is written to. Use 10K pull-up resistor to +5.
FLOPPY DISK INTERFACE:			
15	STEP	STEP	The step output contains a pulse for each step.
16	DIRECTION	DIRC	Direction Output is active high when stepping in, active low when stepping out.
17	EARLY	EARLY	Indicates that the WRITE DATA pulse occurring while Early is active (high) should be shifted early for write precompensation.
18	LATE	LATE	Indicates that the write data pulse occurring while Late is active (high) should be shifted late for write precompensation.
22	$\overline{\text{TEST}}$	$\overline{\text{TEST}}$	This input is used for testing purposes only and should be tied to +5V or left open by the user unless interfacing to voice coil actuated motors.
23	HEAD LOAD TIMING	HLT	When a logic high is found on the HLT input the head is assumed to be engaged.
25	READ GATE	RG	A high level on this output indicates to the data separator circuitry that a field of zeros (or ones) has been encountered, and is used for synchronization.
26	READ CLOCK	RCLK	A nominal square-wave clock signal derived from the data stream must be provided to this input. Phasing (i.e. RCLK transitions) relative to RAW READ is important but polarity (RCLK high or low) is not.
27	$\overline{\text{RAW READ}}$	$\overline{\text{RAW READ}}$	The data input signal directly from the drive. This input shall be a negative pulse for each recorded flux transition.
28	HEAD LOAD	HLD	The HLD output controls the loading of the Read-Write head against the media.
29	TRACK GREATER THAN 43	TG43	This output informs the drive that the Read/Write head is positioned between tracks 44-76. This output is valid only during Read and Write Commands.
30	WRITE GATE	WG	This output is made valid before writing is to be performed on the diskette.
31	WRITE DATA	WD	A 250 ns (MFM) or 500 ns (FM) pulse per flux transition. WD contains the unique Address marks as well as data and clock in both FM and MFM formats.

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
32	READY	READY	This input indicates disk readiness and is sampled for a logic high before Read or Write commands are performed. If Ready is low the Read or Write operation is not performed and an interrupt is generated. Type I operations are performed regardless of the state of Ready. The Ready input appears in inverted format as Status Register bit 7.
33	$\overline{\text{WRITE FAULT}}$ VFO ENABLE	$\overline{\text{WF}}$/$\overline{\text{VFOE}}$	This input detects writing fault indications from the drive. When WG = 1 and $\overline{\text{WF}}$ goes low the current Write command is terminated and the Write Fault status bit is set. The $\overline{\text{WF}}$ input should be made inactive (high) when WG becomes inactive. When WG = 0, this pin functions as a VFO enable output. $\overline{\text{VFOE}}$ is made active when the head is fully engaged and data is being inspected off of the diskette.
34	$\overline{\text{TRACK 00}}$	$\overline{\text{TR00}}$	This input informs the FD179X that the Read/Write head is positioned over Track 00.
35	$\overline{\text{INDEX PULSE}}$	$\overline{\text{IP}}$	This input informs the FD179X when the index hole is encountered on the diskette.
36	$\overline{\text{WRITE PROTECT}}$	$\overline{\text{WPRT}}$	This input is sampled whenever a Write Command is received. A logic low terminates the command and sets the Write Protect Status bit.
37	$\overline{\text{DOUBLE DENSITY}}$	$\overline{\text{DDEN}}$	This pin selects either single or double density operation. When $\overline{\text{DDEN}}$ = 0, double density is selected. When $\overline{\text{DDEN}}$ = 1, single density is selected. This line must be left open on the 1792/4

ORGANIZATION

The Floppy Disk Formatter block diagram is illustrated on page 5. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register—This 8-bit register assembles serial data from the Read Data input ($\overline{\text{RAW READ}}$) during Read operations and transfers serial data to the Write Data output during Write operations.

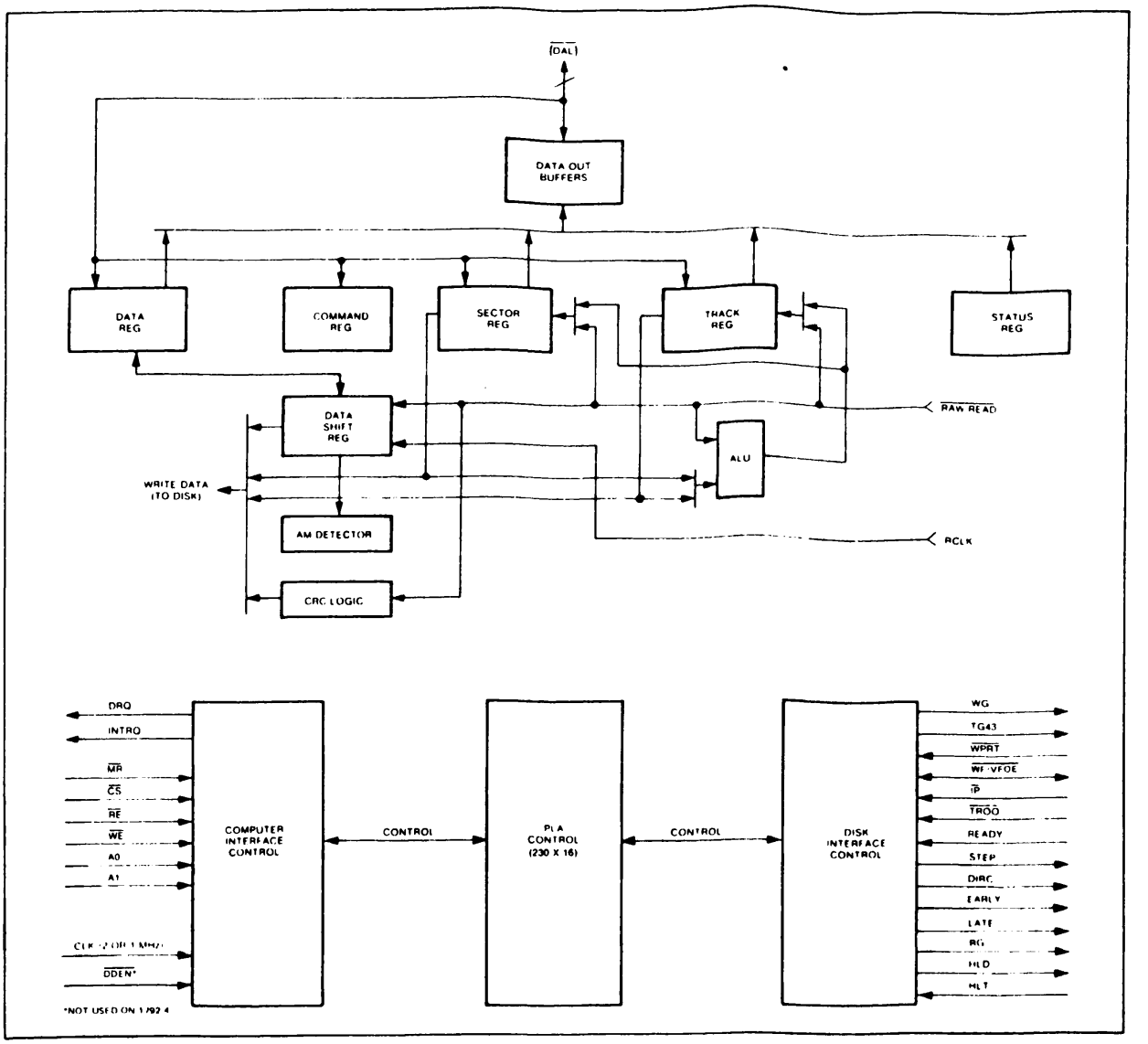
Data Register—This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command the Data Register holds the address of the desired Track position. This register is loaded from the DAL and gated onto the DAL under processor control.

Track Register—This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in (towards track 76) and decremented by one when the head is stepped out (towards track 00) if the verify flag is on. The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register should not be loaded when the device is busy.

Sector Register (SR)—This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

Command Register (CR)—This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the new command is a force interrupt. The command register can be loaded from the DAL, but not read onto the DAL.



FD179X BLOCK DIAGRAM

Status Register (STR)—This 8-bit register holds device Status information. The meaning of the Status bits is a function of the type of command previously executed. This register can be read onto the DAL, but not loaded from the DAL.

CRC Logic—This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is: $G(x) = x^{16} + x^{12} + x^5 + 1$.

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU)—The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

Timing and Control—All computer and Floppy Disk Interface controls are generated through this logic. The internal device timing is generated from an external crystal clock.

The FD1791/3 has two different modes of operation according to the state of $\overline{DDEN}$. When $\overline{DDEN}$ = 0 double density (MFM) is assumed. When $\overline{DDEN}$ = 1, single density (FM) is assumed.

AM Detector—The address mark detector detects ID, data and index address marks during read and write operations.

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines ($\overline{DAL}$) and associated control signals. The $\overline{DAL}$ are used to transfer Data, Status, and Control words out of, or into the FD179X. The $\overline{DAL}$ are three state buffers that are enabled as output drivers when Chip Select ($\overline{CS}$) and

Read Enable ($\overline{RE}$) are active (low logic state) or act as input receivers when $\overline{CS}$ and Write Enable ($\overline{WE}$) are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and $\overline{CS}$ is made low. The address bits A1 and A0, combined with the signals $\overline{RE}$ during a Read operation or $\overline{WE}$ during a Write operation are interpreted as selecting the following registers:

A1-A0	READ ($\overline{RE}$)	WRITE ($\overline{WE}$)
0 0	Status Register	Command Register
0 1	Track Register	Track Register
1 0	Sector Register	Sector Register
1 1	Data Register	Data Register

During Direct Memory Access (DMA) types of data transfers between the Data Register of the FD179X and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operation continues until the end of sector is reached.

On Disk Write operations the data Request is activated when the Data Register transfers its contents to the Data Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit is set in the Status Register.

At the completion of every command an INTRQ is generated. INTRQ is reset by either reading the status register or by loading the command register with a new command. In addition, INTRQ is generated if a Force Interrupt command condition is met.

FLOPPY DISK INTERFACE

The 1791 and 1793 have two modes of operation according to the state of $\overline{DDEN}$ (Pin 37). When $\overline{DDEN}$ = 1, single density is selected. In either case, the CLK input (Pin 24) is at 2 MHz. However, when interfacing with the mini-floppy, the CLK input is set at 1 MHz for both single density and double density. When the clock is at 2 MHz, the stepping rates of 3, 6, 10, and 15 ms are obtainable. When CLK equals 1 MHz these times are doubled. The 1792/4 operates in the single density mode only, with Pin 37 left open by the user.

HEAD POSITIONING

Five commands cause positioning of the Read-Write head (see Command Section). The period of each positioning step is specified by the r field in bits 1 and 0 of the command word. After the last directional step an additional 15 milliseconds of head settling time takes place if the Verify flag is set in Type I commands. Note that this time doubles to 30 ms for a 1 MHz clock. If $\overline{TEST}$ = 0, there is zero settling time. There is also a 15 ms head settling time if the E flag is set in any Type II or III command.

The rates (shown in Table 1) can be applied to a Step-Direction Motor through the device interface.

Step—A 2 μs (MFM) or 4 μs (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output.

Direction (DIRC)—The Direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 12 μs before the first stepping pulse is generated.

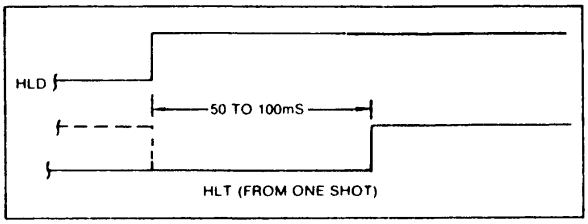
When a Seek, Step or Restore command is executed an optional verification of Read-Write head position can be performed by setting bit 2 (V = 1) in the command word to a logic 1. The verification operation begins at the end of the 15 millisecond settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete and an INTRQ is generated with no errors. The FD179X must find an ID field with correct track number and correct CRC within 5 revolutions of the media; otherwise the seek error is set and an INTRQ is generated.

Table 1. STEPPING RATES

CLK	2 MHz	2 MHz	1 MHz	1 MHz	2 MHz	1 MHz
$\overline{DDEN}$	0	1	0	1	x	x
R1 R0	$\overline{TEST}$=1	$\overline{TEST}$=1	$\overline{TEST}$=1	$\overline{TEST}$=1	$\overline{TEST}$=0	$\overline{TEST}$=0
0 0	3 ms	3 ms	6 ms	6 ms	200 μs	400 μs
0 1	6 ms	6 ms	12 ms	12 ms	200 μs	400 μs
1 0	10 ms	10 ms	20 ms	20 ms	200 μs	400 μs
1 1	15 ms	15 ms	30 ms	30 ms	200 μs	400 μs

The Head Load (HLD) output controls the movement of the read/write head against the media. HLD is activated at the beginning of a Type I command if the h flag is set (h = 1), at the end of the Type I command if the verify flag (V = 1), or upon receipt of any Type II or III command. Once HLD is active it remains active until either a Type I command is received with (h = 0 and V = 0); or if the FD179X is in an idle state (non-busy) and 15 index pulses have occurred.

Head Load Timing (HLT) is an input to the FD179X which is used for the head engage time. When HLT = 1, the FD179X assumes the head is completely engaged. The head engage time is typically 30 to 100 ms depending on drive. The low to high transition on HLD is typically used to fire a one shot. The output of the one shot is then used for HLT and supplied as an input to the FD179X.



HEAD LOAD TIMING

When both HLD and HLT are true, the FD179X will then read from or write to the media. The "and" of HLD and HLT appears as a status bit in Type I status.

In summary for the Type I commands: if h = 0 and V = 0, HLD is reset. If h = 1 and V = 0, HLD is set at the beginning of the command and HLT is not sampled nor is there an internal 15 ms delay. If h = 0 and V = 1, HLD is set near the end of the command, an internal 15 ms occurs, and the FD179X waits for HLT to be true. If h = 1 and V = 1, HLD is set at the beginning of the command. Near the end of the command, after all the steps have been issued, an internal 15 ms delay occurs and the FD179X then waits for HLT to occur.

For Type II and III commands with E flag off, HLD is made active and HLT is sampled until true. With E flag on, HLD is made active, an internal 15 ms delay occurs and then HLT is sampled until true.

DISK READ OPERATIONS

Sector lengths of 128, 256, 512 or 1024 are obtainable in either FM or MFM formats. For FM, $\overline{\text{DDEN}}$ should be placed to logical "1." For MFM formats, $\overline{\text{DDEN}}$ should be placed to a logical "0." Sector lengths are determined at format time by a special byte in the "ID" field. If this Sector length byte in the ID field is zero, then the sector length is 128 bytes. If 01 then 256 bytes. If 02, then 512 bytes. If 03, then the sector length is 1024 bytes. The number of sectors per track as far as the FD179X is concerned can be from 1 to 255 sectors. The number of tracks as far as the FD179X is concerned is from 0 to 255 tracks. For IBM 3740 compatibility, sector lengths are 128 bytes with 26 sectors per track. For System 34 compatibility (MFM), sector lengths are 256 bytes/sector with 26 sectors/track; or lengths of 1024 bytes/sector with 8 sectors/track. (See Sector Length Table.)

For read operations, the FD179X requires $\overline{\text{RAW READ}}$ Data (Pin 27) signal which is a 250 ns pulse per flux transition and a Read clock (RCLK) signal to indicate flux transition spacings. The RCLK (Pin 26) signal is provided by some drives but if not it may be derived externally by Phase lock loops, one shots, or counter techniques. In addition, a Read Gate Signal is provided as an output (Pin 25) which can be used to inform phase lock loops when to acquire synchronization. When reading from the media in FM. RG is made true when 2 bytes of zeroes are detected. The FD179X must find an address mark within the next 10 bytes; otherwise RG is reset and the search for 2 bytes of zeroes begins all over again. If an address mark is found within 10 bytes, RG remains true as long as the FD179X is deriving any useful information from the data stream. Similarly for MFM, RG is made active when 4 bytes of "00" or "FF" are detected. The FD179X must find an address mark within the next 16 bytes, otherwise RG is reset and search resumes.

During read operations (WG = 0), the $\overline{\text{VFOE}}$ (Pin 33) is provided for phase lock loop synchronization. $\overline{\text{VFOE}}$ will go active when:

a) Both HLT and HLD are True
b) Settling Time, if programmed, has expired
c) The 179X is inspecting data off the disk

If $\overline{\text{WF}}/\overline{\text{VFOE}}$ is not used, leave open or tie to a 10K resistor to +5.

DISK WRITE OPERATION

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the FD179X before the Write Gate signal can be activated.

Writing is inhibited when the $\overline{\text{Write Protect}}$ input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set. The Write Fault input, when activated, signifies a writing fault condition detected in disk drive electronics such as failure to detect write current flow when the Write Gate is activated. On detection of this fault the FD179X terminates the current command, and sets the $\overline{\text{Write Fault}}$ bit (bit 5) in the Status Word. The $\overline{\text{Write Fault}}$ input should be made inactive when the Write Gate output becomes inactive.

For write operations, the FD179X provides Write Gate (Pin 30) and Write Data (Pin 31) outputs. Write data consists of a series of 500 ns pulses in FM ($\overline{\text{DDEN}}$ = 1) and 250 ns pulses in MFM ($\overline{\text{DDEN}}$ = 0). Write Data provides the unique address marks in both formats.

Also during write, two additional signals are provided for write precompensation. These are EARLY (Pin 17) and LATE (Pin 18). EARLY is active true when the WD pulse appearing on (Pin 30) is to be written early. LATE is active true when the WD pulse is to be written LATE. If both EARLY and LATE are low when the WD pulse is present, the WD pulse is to be written at nominal. Since write precompensation values vary from disk manufacturer to disk manufacturer, the actual value is determined by several one shots or delay lines which are located external to the FD179X. The write precompensation signals EARLY and LATE are valid for the duration of WD in both FM and MFM formats.

Whenever a Read or Write command (Type II or III) is received the FD179X samples the Ready input. If this input is logic low the command is not executed and an interrupt is generated. All Type I commands are performed regardless of the state of the Ready input. Also, whenever a Type II or III command is received, the TG43 signal output is updated.

COMMAND DESCRIPTION

The FD179X will accept eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in Table 2.

Table 2. COMMAND SUMMARY

		BITS							
TYPE	COMMAND	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step In	0	1	0	u	h	V	r_1	r_0
I	Step Out	0	1	1	u	h	V	r_1	r_0
II	Read Sector	1	0	0	m	S	E	C	0
II	Write Sector	1	0	1	m	S	E	C	a_0
III	Read Address	1	1	0	0	0	E	0	0
III	Read Track	1	1	1	0	0	E	0	0
III	Write Track	1	1	1	1	0	E	0	0
IV	Force Interrrupt	1	1	0	1	I_3	I_2	I_1	I_0

Note: Bits shown in TRUE form.

Table 3. FLAG SUMMARY

TYPE I COMMANDS

h = Head Load Flag (Bit 3)
- h = 1, Load head at beginning
- h = 0, Unload head at beginning

V = Verify flag (Bit 2)
- V = 1, Verify on destination track
- V = 0, No verify

r_1r_0 = Stepping motor rate (Bits 1-0)
- Refer to Table 1 for rate summary

u = Update flag (Bit 4)
- u = 1, Update Track register
- u = 0, No update

Table 4. FLAG SUMMARY

TYPE II & III COMMANDS

m = Multiple Record flag (Bit 4)
- m = 0, Single Record
- m = 1, Multiple Records

a_0 = Data Address Mark (Bit 0)
- a_0 = 0, FB (Data Mark)
- a_0 = 1, F8 (Deleted Data Mark)

E = 15 ms Delay (2MHz)
- E = 1, 15 ms delay
- E = 0, no 15 ms delay

S = Side Select Flag
- S = 0, Compare for Side 0
- S = 1, Compare for Side 1

C = Side Compare Flag
- C = 0, disable side select compare
- C = 1, enable side select compare

Table 5. FLAG SUMMARY

TYPE IV COMMAND

Ii = Interrupt Condition flags (Bits 3-0)
- I0 = 1, Not-Ready to Ready Transition
- I1 = 1, Ready to Not-Ready Transition
- I2 = 1, Index Pulse
- I3 = 1, Immediate Interrupt
- I_3 -I_0 = 0, Terminate with no Interrupt

TYPE I COMMANDS

The Type I Commands include the Restore, Seek, Step, Step-In, and Step-Out commands. Each of the Type I Commands contains a rate field (r_0r_1), which determines the stepping motor rate as defined in Table 1.

The Type I Commands contain a head load flag (h) which determines if the head is to be loaded at the beginning of the command. If h = 1, the head is loaded at the beginning of the command (HLD output is made active). If h = 0, HLD is deactivated. Once the head is loaded, the head will remain engaged until the FD179X receives a command that specifically disengages the head. If the FD179X is idle (busy = 0) for 15 revolutions of the disk, the head will be automatically disengaged (HLD made inactive).

The Type I Commands also contain a verification (V) flag which determines if a verification operation is to take place on the destination track. If V = 1, a verification is performed, if V = 0, no verification is performed.

During verification, the head is loaded and after an internal 15 ms delay, the HLT input is sampled. When HLT is active (logic true), the first encountered ID field is read off the disk. The track address of the

ID field is then compared to the Track Register; if there is a match and a valid ID CRC, the verification is complete, an interrupt is generated and the Busy status bit is reset. If there is not a match but there is valid ID CRC, an interrupt is generated, and Seek Error Status bit (Status bit 4) is set and the Busy status bit is reset. If there is a match but not a valid CRC, the CRC error status bit is set (Status bit 3), and the next encountered ID field is read from the disk for the verification operation. If an ID field with a valid CRC cannot be found after four revolutions of the disk, the FD179X terminates the operation and sends an interrupt, (INTRQ).

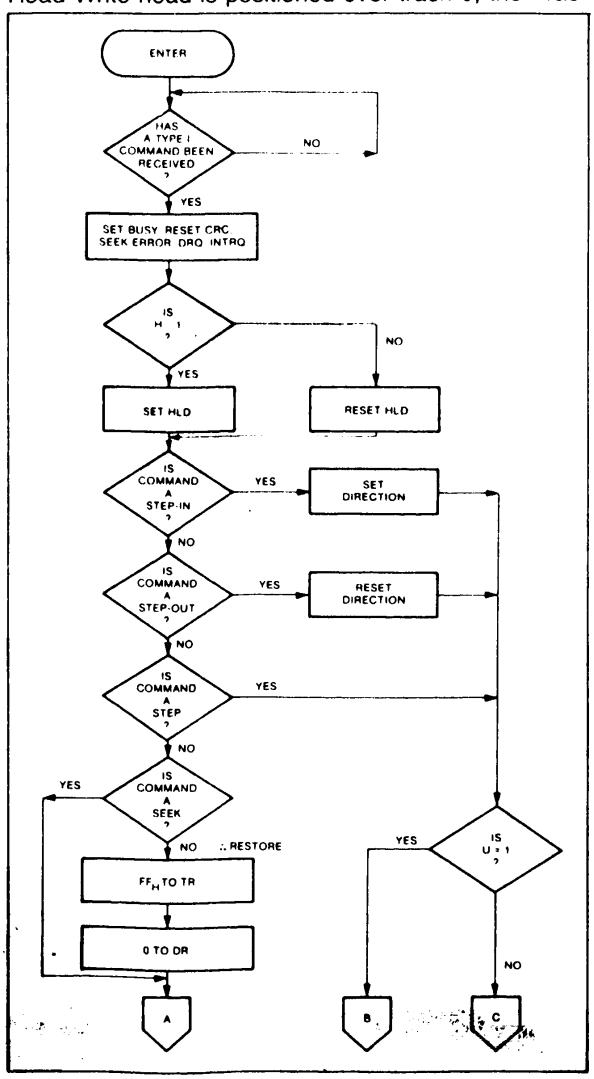
The Step, Step-In, and Step-Out commands contain an Update flag (U). When U = 1, the track register is updated by one for each step. When U = 0, the track register is not updated.

RESTORE (SEEK TRACK 0)

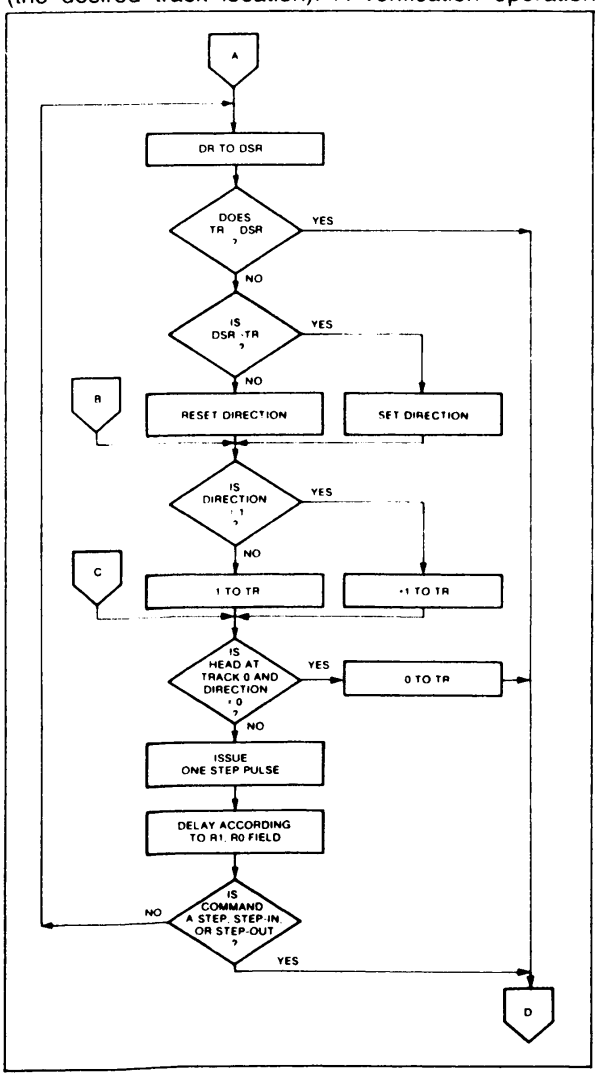
Upon receipt of this command the Track 00 ($\overline{TROO}$) input is sampled. If $\overline{TROO}$ is active low indicating the Read-Write head is positioned over track 0, the Track Register is loaded with zeroes and an interrupt is generated. If $\overline{TROO}$ is not active low, stepping pulses (pins 15 to 16) at a rate specified by the rr0 field are issued until the $\overline{TROO}$ input is activated. At this time the Track Register is loaded with zeroes and an interrupt is generated. If the $\overline{TROO}$ input does not go active low after 255 stepping pulses, the FD179X terminates operation, interrupts, and sets the Seek error status bit. A verification operation takes place if the V flag is set. The h bit allows the head to be loaded at the start of command. Note that the Restore command is executed when $\overline{MR}$ goes from an active to an inactive state.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read-Write head and the Data Register contains the desired track number. The FD179X will update the Track register and issue stepping pulses in the appropriate direction until the contents of the Track register are equal to the contents of the Data Register (the desired track location). A verification operation



TYPE I COMMAND FLOW



TYPE I COMMAND FLOW

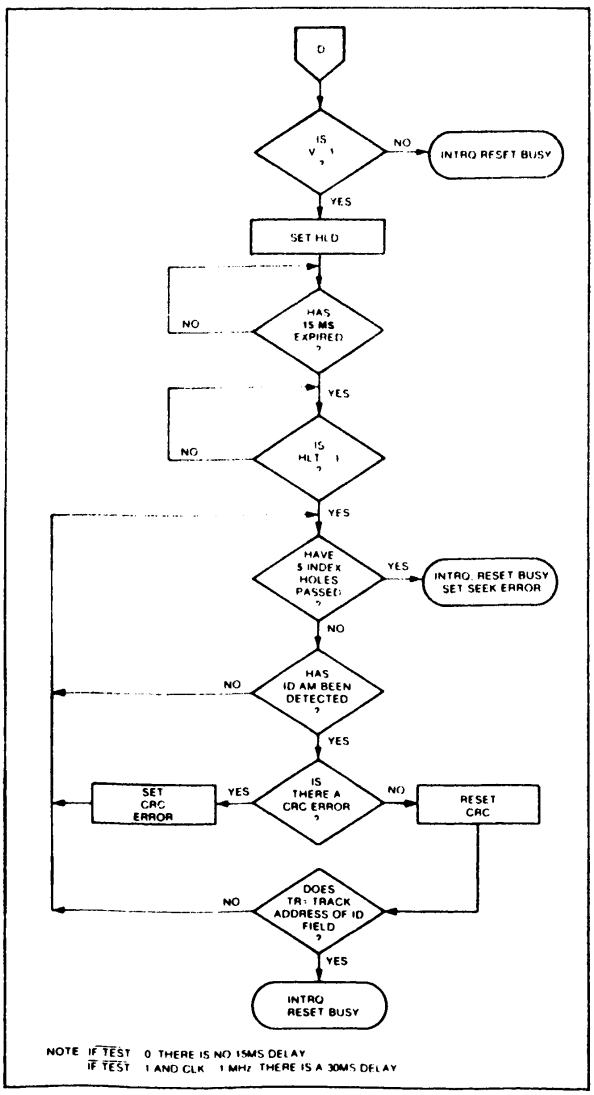
takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP

Upon receipt of this command, the FD179X issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the r1r0 field, a verification takes place if the V flag is on. If the u flag is on, the Track Register is updated. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP-IN

Upon receipt of this command, the FD179X issues one stepping pulse in the direction towards track 76. If the u flag is on, the Track Register is incremented by one. After a delay determined by the r1r0 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.



TYPE I COMMAND FLOW

STEP-OUT

Upon receipt of this command, the FD179X issues one stepping pulse in the direction towards track 0. If the u flag is on, the Track Register is decremented by one. After a delay determined by the r1r0 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

TYPE II COMMANDS

The Type II Commands are the Read Sector and Write Sector commands. Prior to loading the Type II Command into the Command Register, the computer must load the Sector Register with the desired sector number. Upon receipt of the Type II command, the busy status Bit is set. If the E flag = 1 (this is the normal case) HLD is made active and HLT is sampled after a 15 msec delay. If the E flag is 0, the head is loaded and HLT sampled with no 15 msec delay. The ID field and Data Field format are shown on page 12.

When an ID field is located on the disk, the FD179X compares the Track Number on the ID field with the Track Register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is then located and will be either written into, or read from depending upon the command. The FD179X must find an ID field with a Track number, Sector number, side number, and CRC within four revolutions of the disk; otherwise, the Record not found status bit is set (Status bit 3) and the command is terminated with an interrupt.

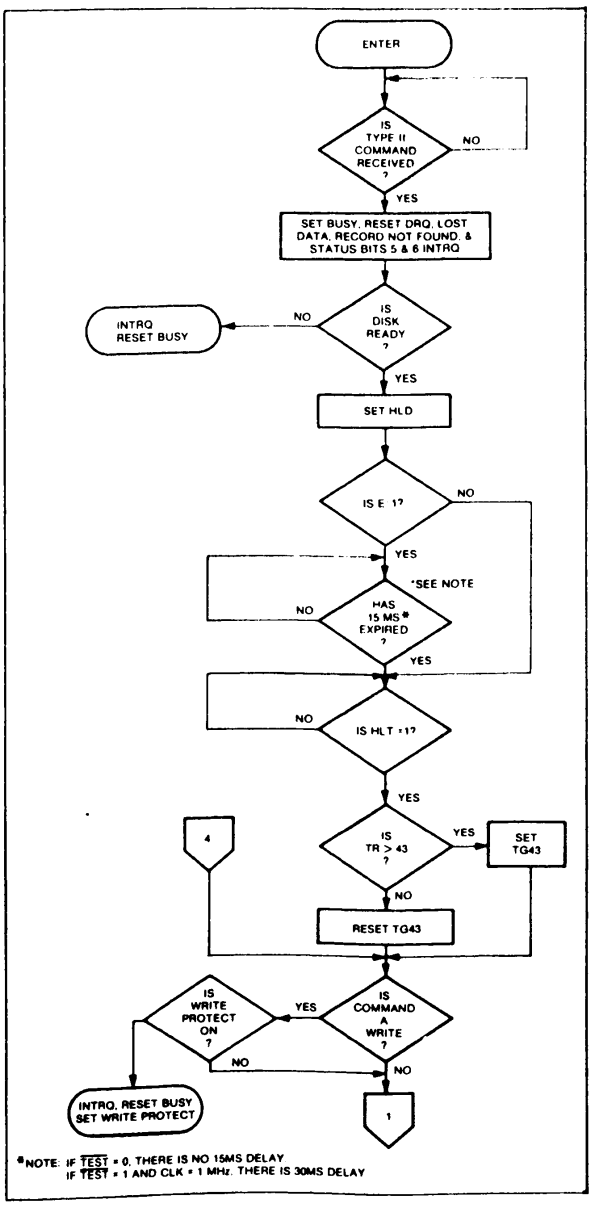
Sector Length Table	
Sector Length Field (hex)	**Number of Bytes in Sector (decimal)**
00	128
01	256
02	512
03	1024

Each of the Type II Commands contains an (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If m = 0, a single sector is read or written and an interrupt is generated at the completion of the command. If m = 1, multiple records are read or written with the sector register internally updated so that an address verification can occur on the next record. The FD179X will continue to read or write multiple records and update the sector register until the sector regis-

ter exceeds the number of sectors on the track or until the Force Interrupt command is loaded into the Command Register, which terminates the command and generates an interrupt.

If the Sector Register exceeds the number of sectors on the track, the Record-Not-Found status bit will be set.

The Type II commands also contain side select compare flags. When C = 0, no side comparison is made. When C = 1, the LSB of the side number is read off the ID Field of the disk and compared with the contents of the (S) flag. If the S flag compares with the side number recorded in the ID field, the 179X continues with the ID search. If a comparison is not made within 5 index pulses, the interrupt line is made active and the Record-Not-Found status bit is set.

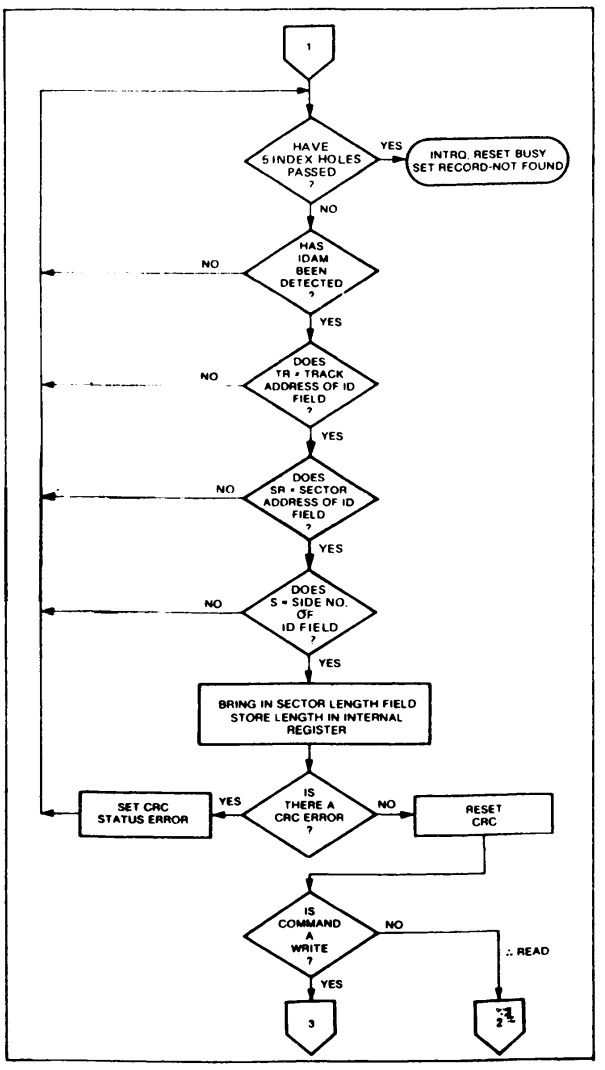


TYPE II COMMAND

READ SECTOR

Upon receipt of the Read Sector command, the head is loaded, the Busy status bit set, and when an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, the data field is presented to the computer. The Data Address Mark of the data field must be found within 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte; if not, the Record Not Found status bit is set and the operation is terminated.

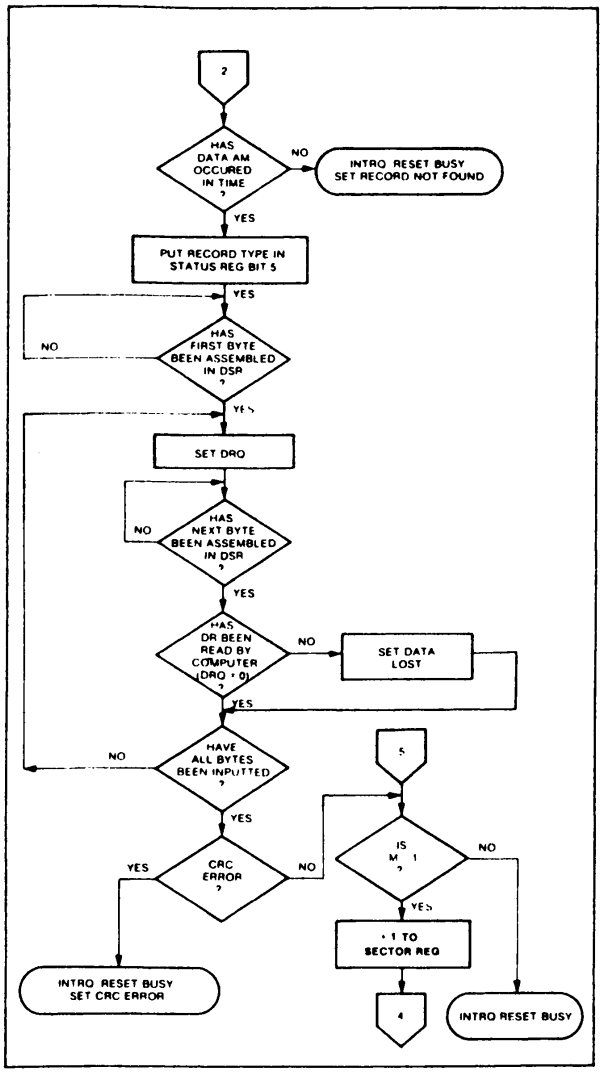
When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the Computer has not read the previous contents of the DR before a new character is transferred that character is lost and



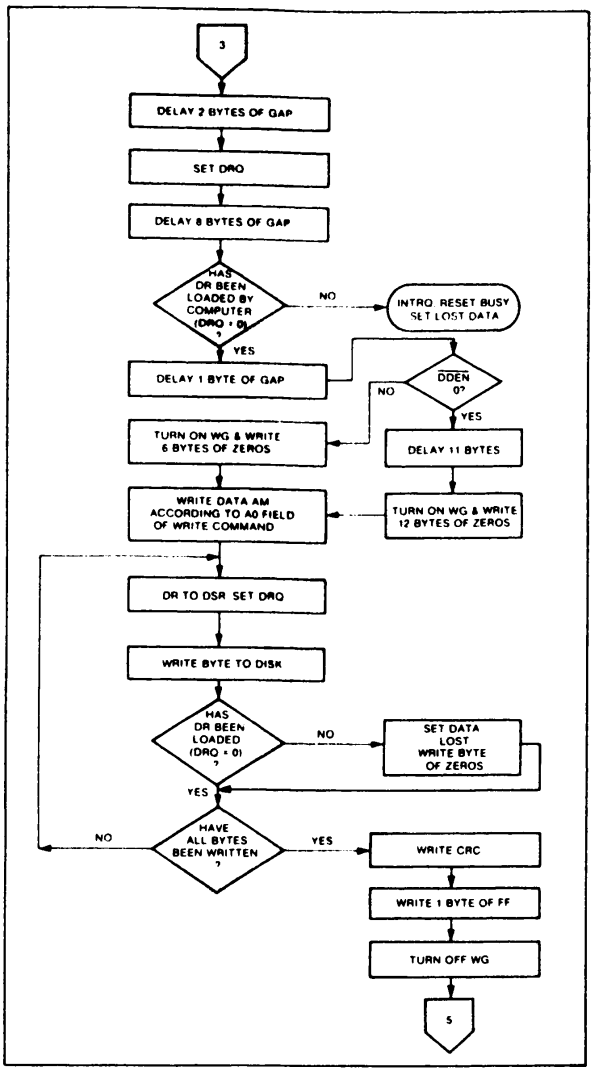
TYPE II COMMAND

GAP III	ID AM	TRACK NUMBER	SIDE NUMBER	SECTOR NUMBER	SECTOR LENGTH	CRC 1	CRC 2	GAP II	DATA AM	DATA FIELD	CRC 1	CRC 2
ID FIELD										DATA FIELD		

In MFM only, IDAM and DATA AM are preceded by three bytes of A1 with clock transition between bits 4 and 5 missing.



TYPE II COMMAND



TYPE II COMMAND

the Lost Data Status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bit 5) as shown below:

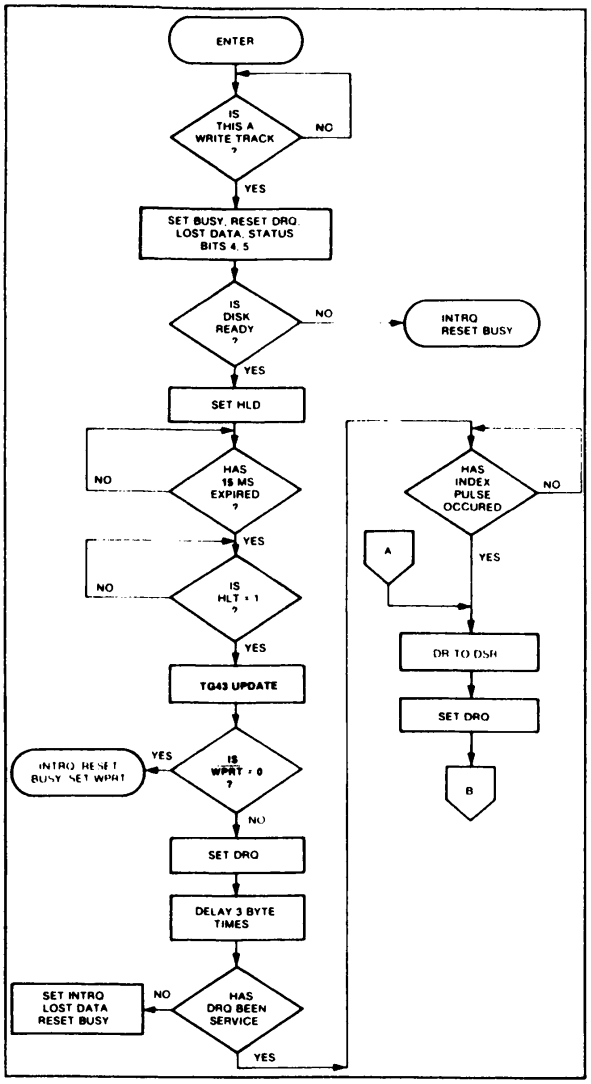
STATUS BIT 5	
1	Deleted Data Mark
0	Data Mark

WRITE SECTOR

Upon receipt of the Write Sector command, the head is loaded (HLD active) and the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, a DRQ is generated. The FD179X counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e., the DR has been loaded by the computer). If DRQ has not been serviced, the command is terminated and the Lost Data status bit is set. If the DRQ has been serviced, the WG is made active and six bytes of zeros in single density and 12 bytes in double density are

then written on the disk. At this time the Data Address Mark is then written on the disk as determined by the a_0 field of the command as shown below:

a_0	Data Address Mark (Bit 0)
1	Deleted Data Mark
0	Data Mark



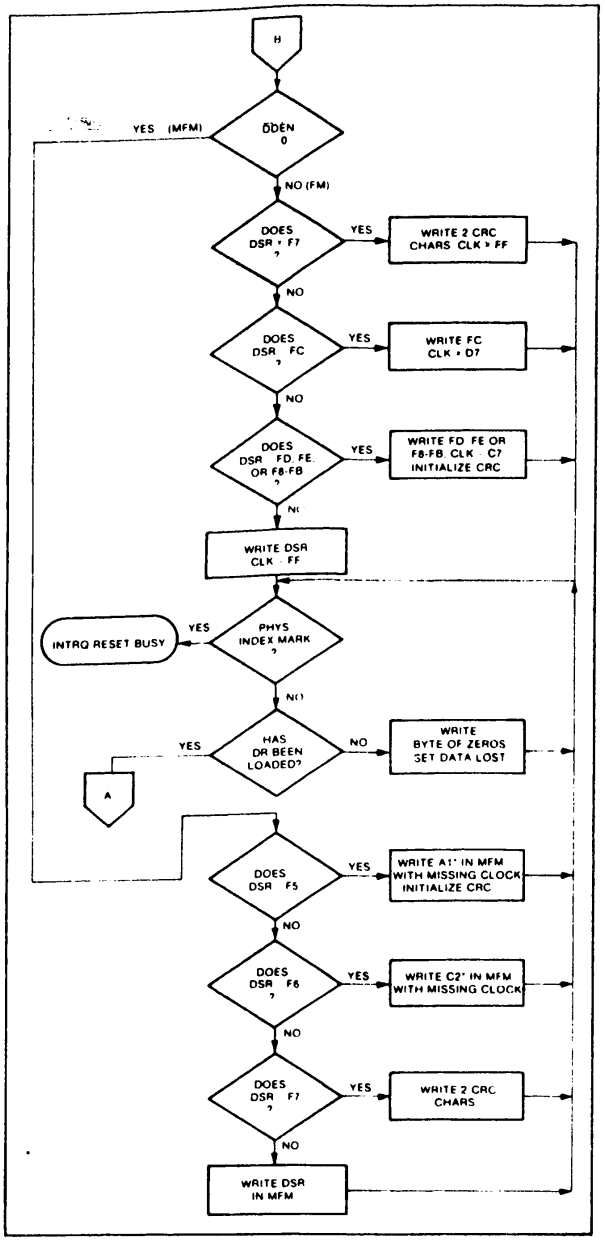
TYPE III COMMAND WRITE TRACK

The FD179X then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the Lost Data Status Bit is set and a byte of zeros is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two-byte CRC is computed internally and written on the disk followed by one byte of logic ones in FM or in MFM. The WG output is then deactivated.

TYPE III COMMANDS

READ ADDRESS

Upon receipt of the Read Address command, the head is loaded and the Busy Status Bit is set. The



TYPE III COMMAND WRITE TRACK

next encountered ID field is then read in from the disk, and the six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below:

TRACK ADDR	SIDE NUMBER	SECTOR ADDRESS	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6

Although the CRC characters are transferred to the computer, the FD179X checks for validity and the CRC error status bit is set if there is a CRC error. The Track Address of the ID field is written into the sector register. At the end of the operation an interrupt is generated and the Busy Status is reset.

READ TRACK

Upon receipt of the Read Track command, the head is loaded and the Busy Status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. As each byte is assembled it is transferred to the Data Register and the Data Request is generated for each byte. No CRC checking is performed. Gaps are included in the input data stream. The accumulation of bytes is synchronized to each Address Mark encountered. Upon completion of the command, the interrupt is activated. RG is not activated during the Read Track command.

WRITE TRACK

Upon receipt of the Write Track command, the head is loaded and the Busy Status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data Request is activated immediately upon receiving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded by the time the index pulse is encountered the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the Interrupt is activated. If a byte is not present in the DR when needed, a byte of zeros is substituted. Address Marks and CRC characters are written on the disk by detecting certain data byte patterns in the outgoing data stream as shown in the table below. The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR in FM or by receipt of F5 in MFM.

CONTROL BYTES FOR INITIALIZATION

DATA PATTERN IN DR (HEX)	FD179X INTERPRETATION IN FM ($\overline{DDEN}$ = 1)	FD1791/3 INTERPRETATION IN MFM ($\overline{DDEN}$ = 0)
00 thru F4	Write 00 thru F4 with CLK = FF	Write 00 thru F4, in MFM
F5	Not Allowed	Write A1* in MFM, Preset CRC
F6	Not Allowed	Write C2** in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Write F8 thru FB, Clk = C7, Preset CRC	Write F8 thru FB, in MFM
FC	Write FC with Clk = D7	Write FC in MFM
FD	Write FD with Clk = FF	Write FD in MFM
FE	Write FE, Clk = C7, Preset CRC	Write FE in MFM
FF	Write FF with Clk = FF	Write FF in MFM

*Missing clock transition between bits 4 and 5
**Missing clock transition between bits 3 & 4

TYPE IV COMMAND

FORCE INTERRUPT

This command can be loaded into the command register at any time. If there is a current command under execution (Busy Status Bit set), the command will be terminated and an interrupt will be generated when the condition specified in the I_0 through I_3 field is detected. The interrupt conditions are shown below:

I_0 = Not-Ready-To-Ready Transition
I_1 = Ready-To-Not-Ready Transition
I_2 = Every Index Pulse
I_3 = Immediate Interrupt (requires reset, see Note)

NOTE: If $I_0 - I_3 = 0$, there is no interrupt generated but the current command is terminated and busy is reset. *This is the only command that will enable the immediate interrupt to clear on a subsequent Load Command Register or Read Status Register.*

STATUS DESCRIPTION

Upon receipt of any command, except the Force Interrupt command, the Busy Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, the Busy Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The format of the Status Register is shown below:

(BITS)							
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Status varies according to the type of command executed as shown in Table 6.

Table 6. STATUS REGISTER SUMMARY

BIT	ALL TYPE I COMMANDS	READ ADDRESS	READ SECTOR	READ TRACK	WRITE SECTOR	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	0	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD LOADED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	RNF	RNF	0	RNF	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	0	CRC ERROR	0
S2	TRACK 0	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

STATUS FOR TYPE I COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset it indicates that the drive is ready. This bit is an inverted copy of the Ready input and logically 'ored' with MR.
S6 PROTECTED	When set, indicates Write Protect is activated. This bit is an inverted copy of WRPT input.
S5 HEAD LOADED	When set, it indicates the head is loaded and engaged. This bit is a logical "and" of HLD and HLT signals.
S4 SEEK ERROR	When set, the desired track was not verified. This bit is reset to 0 when updated.
S3 CRC ERROR	CRC encountered in ID field.
S2 TRACK 00	When set, indicates Read/Write head is positioned to Track 0. This bit is an inverted copy of the $\overline{TR00}$ input.
S1 INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the $\overline{IP}$ input.
S0 BUSY	When set command is in progress. When reset no command is in progress.

STATUS BITS FOR TYPE II AND III COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset, it indicates that the drive is ready. This bit is an inverted copy of the Ready input and 'ored' with MR. The Type II and III Commands will not execute unless the drive is ready.
S6 WRITE PROTECT	On Read Record: Not Used. On Read Track: Not Used. On any Write: It indicates a Write Protect. This bit is reset when updated.
S5 RECORD TYPE/ WRITE FAULT	On Read Record: It indicates the record-type code from data field address mark. 1 = Deleted Data Mark. 0 = Data Mark. On any Write: It indicates a Write Fault. This bit is reset when updated.
S4 RECORD NOT FOUND (RNF)	When set, it indicates that the desired track, sector, or side were not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields; otherwise it indicates error in data field. This bit is reset when updated.
S2 LOST DATA	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated.
S1 DATA REQUEST	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a Write operation. This bit is reset to zero when updated.
S0 BUSY	When set, command is under execution. When reset, no command is under execution.

FORMATTING THE DISK

(Refer to section on Type III commands for flow diagrams.)

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA control with a large amount of memory. When operating under DMA with limited amount of memory, formatting is a more difficult task. This is because gaps as well as data must be provided at the computer interface.

Formatting the disk is accomplished by positioning the R/W head over the desired track number and issuing the Write Track command. Upon receipt of the Write Track command, the FD179X raises the Data Request signal. At this point in time, the user loads the data register with desired data to be written on the disk. For every byte of information to be written on the disk, a data request is generated. This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a normal clock pattern. However, if the FD179X detects a data pattern of F5 thru FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation. For instance, in FM an FE pattern will be interpreted as an ID address mark (DATA-FE, CLK-C7) and the CRC will be initialized. An F7 pattern will generate two CRC characters in FM or MFM. As a consequence, the patterns F5 thru FE must not appear in the gaps, data fields, or ID fields. Also, CRC's must be generated by an F7 pattern.

Disks may be formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, or 1024 bytes.

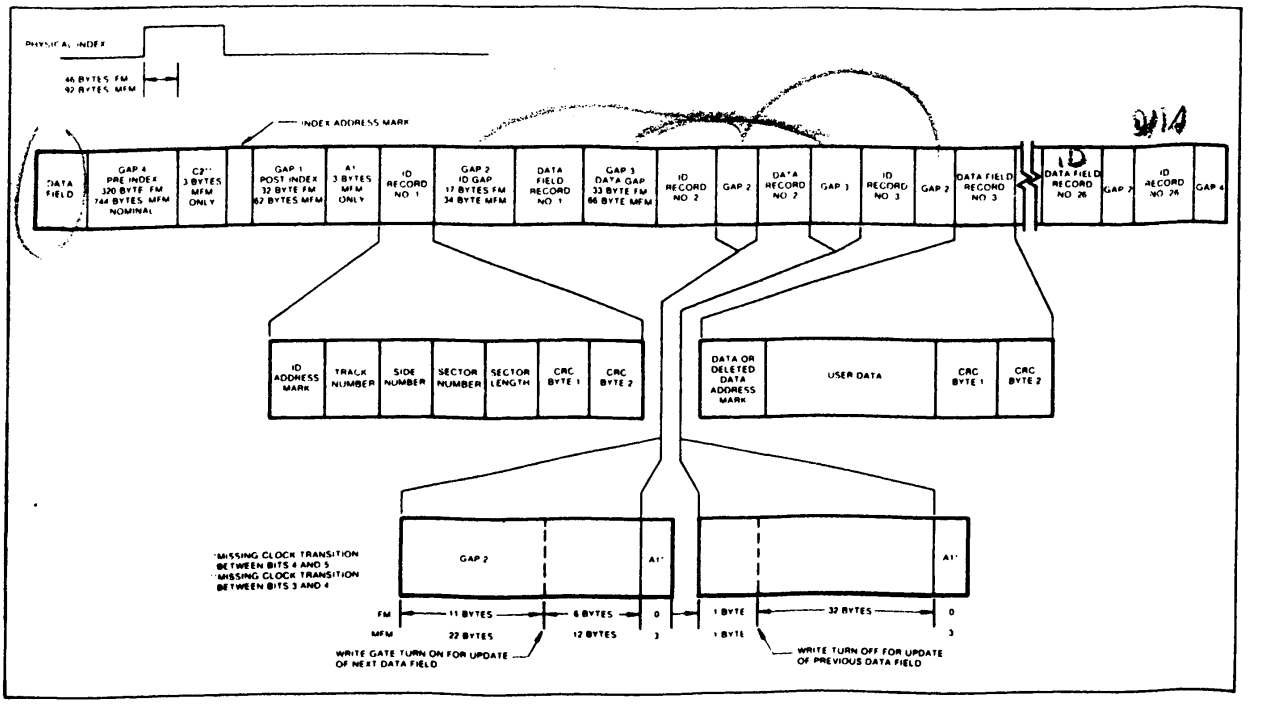
IBM 3740 FORMAT—128 BYTES/SECTOR

Shown below is the IBM single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one data request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF
6	00
1	FC (Index Mark)
26*	FF
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00
1	F7 (2 CRC's written)
11	FF
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	FF
247**	FF

*Write bracketed field 26 times

**Continue writing until FD1791 interrupts out. Approx. 247 bytes.



IBM TRACK FORMAT

IBM SYSTEM 34 FORMAT- 256 BYTES/SECTOR

Shown below is the IBM dual-density format with 256 bytes/sector. In order to format a diskette the user must issue the Write Track command and load the data register with the following values. For every byte to be written, there is one data request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F6
1	FC (Index Mark)
50*	4E
12	00
3	F5
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01
1	F7 (2 CRCs written)
22	4E
12	00
3	F5
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRCs written)
54	4E
598**	4E

* Write bracketed field 26 times
**Continue writing until FD179X interrupts out. Approx. 598 bytes.

NON-IBM FORMATS

Variations in the IBM format are possible to a limited extent if the following requirements are met: sector size must be a choice of 128, 512 or 1024 bytes; gap size must be according to the following table. Note that the Index Mark is not required by the FD179X. All gap sizes shown are the minimum values required by the 179X.

	FM	MFM
Gap I	16 bytes FF	16 bytes 4E
Gap II	11 bytes FF	22 bytes 4E
*	6 bytes 00	12 bytes 00
		3 bytes A1
Gap III	10 bytes FF	16 bytes 4E
**	4 bytes 00	8 bytes 00
		3 bytes A1
Gap IV	16 bytes FF	16 bytes 4E

*Byte counts must be exact.
**Byte counts are minimum, except exactly 3 bytes of A1 must be written.

ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

V_{DD} With Respect to V_{SS} (Ground) =15 to −0.3V
Max. Voltage to Any Input With Respect to V_{SS} =15 to −0.3V
Operating Temperature 0°C to 70°C
Storage Temperature −55°C to +125°C

OPERATING CHARACTERISTICS (DC)

T_A = 0°C to 70°C, V_{DD} = +12.0V ±.6V, V_{SS} = OV, V_{CC} =+5V ±.25V
V_{DD} = 10 ma Nominal, V_{CC} = 35 ma Nominal

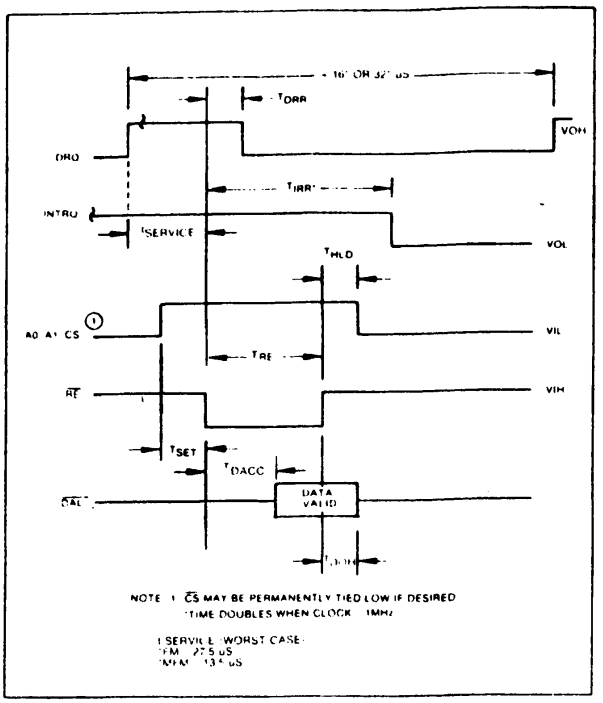
SYMBOL	CHARACTERISTIC	MIN.	TYPE.	MAX.	UNITS	CONDITIONS
I_{IL}	Input Leakage			10	μA	V_{IN} = V_{DD}
I_{OL}	Output Leakage			10	μA	V_{OUT} = V_{DD}
V_{IH}	Input High Voltage	2.6			V	
V_{IL}	Input Low Voltage			0.8	V	
V_{OH}	Output High Voltage	2.8			V	I_O = 100 μA
V_{OL}	Output Low Voltage			0.45	V	I_O = 1.6 mA
P_D	Power Dissipation			0.5	W	

TIMING CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = + 12V ± .6V, V_{SS} = 0V, V_{CC} =+5V ± .25V

READ ENABLE TIMING

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TSET	Setup ADDR & CS to $\overline{RE}$	0			nsec	
THLD	Hold ADDR & CS from $\overline{RE}$	10			nsec	
TRE	$\overline{RE}$ Pulse Width	400			nsec	C_L = 50 pf
TDRR	DRQ Reset from $\overline{RE}$		400	500	nsec	
TIRR	INTRQ Reset from $\overline{RE}$		500	3000	nsec	See Note 6
TDACC	Data Access from $\overline{RE}$			300	nsec	C_L = 50 pf
TDOH	Data Hold From $\overline{RE}$	50		150	nsec	C_L = 50 pf



READ ENABLE TIMING

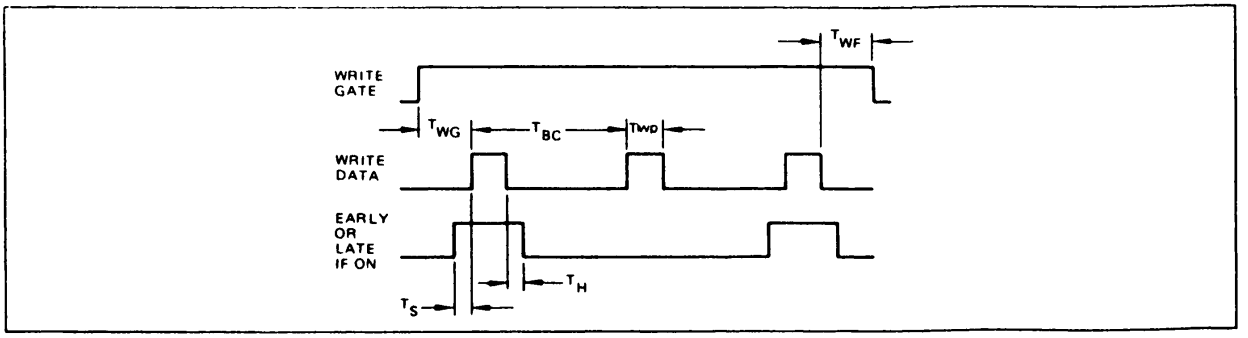
WRITE ENABLE TIMING

WRITE ENABLE TIMING

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TSET	Setup ADDR & CS to $\overline{WE}$	50			nsec	
THLD	Hold ADDR & CS from $\overline{WE}$	10			nsec	
TWE	$\overline{WE}$ Pulse Width	350			nsec	
TDRR	DRQ Reset from $\overline{WE}$		400	500	nsec	
TIRR	INTRQ Reset from $\overline{WE}$		500	3000	nsec	See Note 6
TDS	Data Setup to $\overline{WE}$	250			nsec	
TDH	Data Hold from $\overline{WE}$	20			nsec	

INPUT DATA TIMING:

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Tpw	$\overline{Raw\ Read}$ Pulse Width	100	200		nsec	See Note 1,2
tbc	$\overline{Raw\ Read}$ Cycle Time	1600	2000		nsec	See Note 3
Ta	RCLK Duty (High)	800			nsec	See Note 4, 5
Tb	RCLK Duty (Low)	800			nsec	
Tc	RCLK Cycle Time	1600			nsec	
Tx_1	RCLK hold to $\overline{Raw\ Read}$	40			nsec	
Tx_2	$\overline{Raw\ Read}$ hold to RCLK	40			nsec	See Note 1



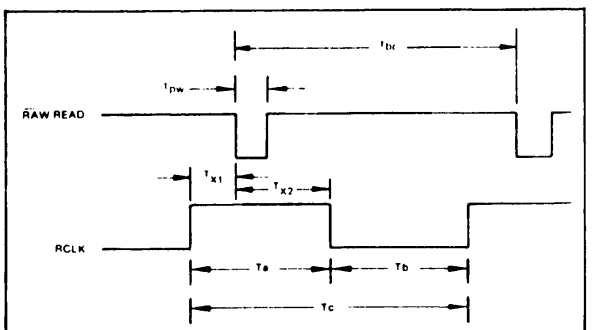
WRITE DATA TIMING

WRITE DATA TIMING: (ALL TIMES DOUBLE WHEN CLK = 1 MHz)

SYMBOL	CHARACTERISTICS	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Twp	Write Data Pulse Width	450	500	550	nsec	FM
		150	200	250	nsec	MFM
Twg	Write Gate to Write Data		2		µsec	FM
			1		µsec	MFM
Tbc	Write data cycle Time		2,3, or 4		µsec	±CLK Error
Ts	Early (Late) to Write Data	125			nsec	MFM
Th	Early (Late) From Write Data	125			nsec	MFM
Twf	Write Gate off from WD		2		µsec	FM
			1		µsec	MFM

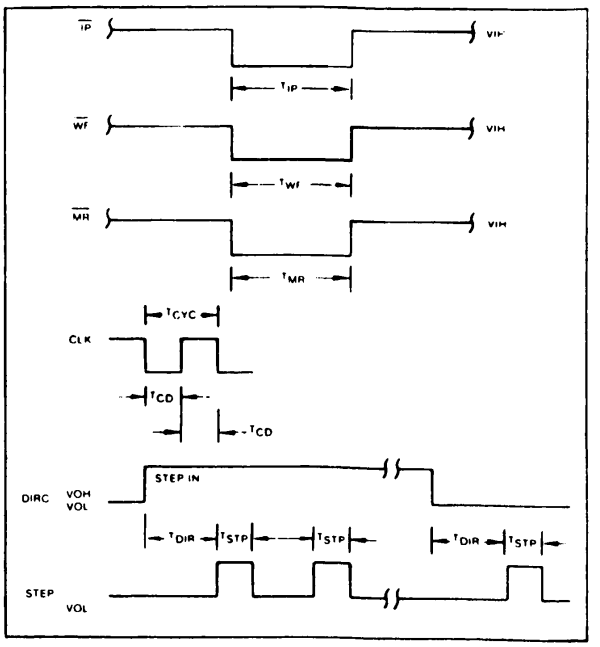
MISCELLANEOUS TIMING:

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TCD_1	Clock Duty (low)	230	250	20000	nsec	
TCD_2	Clock Duty (high)	200	250	20000	nsec	
TSTP	Step Pulse Output	2 or 4			µsec	See Note 6
TDIR	Dir Setup to Step	12			µsec	
TMR	Master Reset Pulse Width	50			µsec	
TIP	Index Pulse Width	10			µsec	See Note 6
TWF	Write Fault Pulse Width	10			µsec	



				NOMINAL		
DISKETTE	MODE	$\overline{DDEN}$	CLK	T_a	T_b	T_c
8"	MFM	0	2 MHz	1 µs	1 µs	2 µs
8"	FM	1	2 MHz	2 µs	2µs	4 µs
5"	MFM	0	1 MHz	2 µs	2µs	4 µs
5"	FM	1	1 MHz	4 µs	4 µs	8 µs

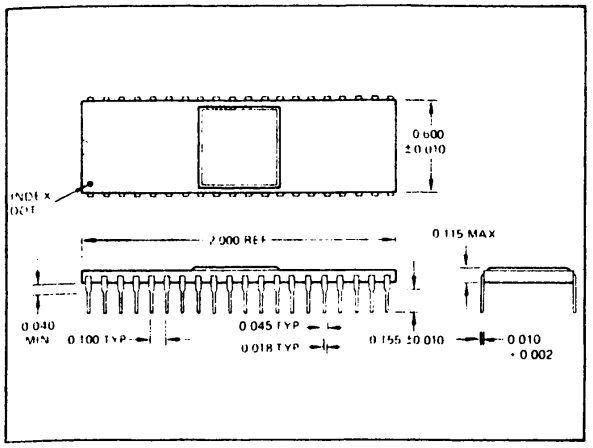
INPUT DATA TIMING



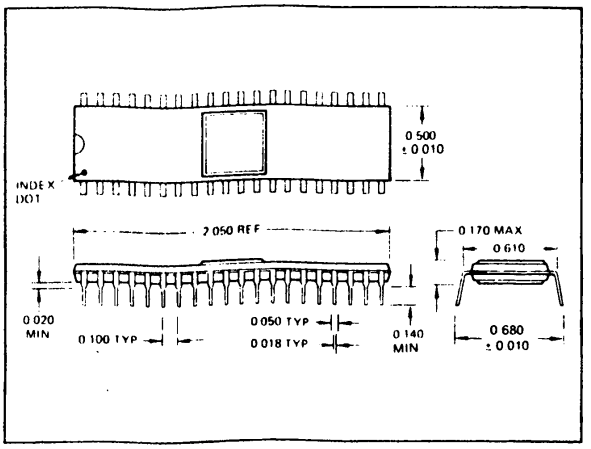
MISCELLANEOUS TIMING

NOTES:

1. Pulse width on RAW READ (Pin 27) is normally 100-300 ns. However, pulse may be any width if pulse is entirely within window. If pulse occurs in both windows, then pulse width must be less than 300 ns for MFM at CLK = 2 MHz and 600 ns for FM at 2 MHz. Times double for 1 MHz.
2. 100 ns. pulses are recommended for 8" MFM.
3. tbc should be 2 µs, nominal in MFM and 4 µs nominal in FM. Times double when CLK = 1 MHz.
4. RCLK may be high or low during $\overline{RAW\ READ}$ (Polarity is unimportant).
5. RCLK should be forced low when $\overline{VFOE}$ = 1 and free-running when $\overline{VFOE}$ = 0.
6. Times double when clock = 1 MHz.



FD179XA-01 CERAMIC PACKAGE



FD179XB-01 PLASTIC PACKAGE

This is a preliminary specification with tentative device parameters and may be subject to change after final product characterization is completed.

Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use; nor any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change said circuitry at anytime without notice.

Anmerkung des Herausgebers zum folgenden Beitrag "ID"

Der folgende Beitrag zum DOS-Buch "ID - Eine Idee zum NEWDOS80" von Ulrich Heidenreich wurde kurzfristig in das DOS-Buch aufgenommen und ist völlig unabhängig davon enstanden. Daher lassen sich einige Überschneidungen und Wiederholungen nicht vermeiden. Wir wollten Ihnen diesen Beitrag aber nicht vorenthalten, da er zum einen ein interessantes Programm beinhaltet, zum anderen einige der in diesem Buch aufgezeigten Zusammenhänge in einer konkreten Anwendung verdeutlicht.

Die Abbildungen 5 + 6 sollten nur im Zusammenhang mit diesem Beitrag gesehen werden. Insbesondere sind diese, da auf den Gegenstand dieses Beitrag abgestimmt, nicht vollständig. Vollständige Darstellungen zum Nachschlagen finden Sie im Hauptteil des Buches in den Kapiteln 3 und 4. Die Abbildung 3 wurde gestrichen, da der Overlay-Loader im SYS0-Listing (Kapitel 3) enthalten ist.

Luidger Röckrath

Ulrich Heidenreich:

ID - Eine Idee zum NEWDOS80

Formate, Formate, Formate!

Von Tandy's altem (?) TRS80 Modell 1 bis hin zu TCS's GenieIII findet das Betriebssystem NEWDOS80.2 mit seinem deutschem Gegenstück GDOS 2.x weite Verbreitung. Das verwendete Diskettenformat (ein für 5 1/4" modifiziertes IBM 3740-Format) ist allen Konfigurationen identisch, die Disketten-Organisation jedoch nicht: So existieren viele mögliche Kombinationen von 35 oder 40 Spur einseitig mit einfacher Schreibdichte bis hin zu 80 Spur doppelseitig mit doppelter Schreibdichte. Hinzu kommen noch unterschiedliche Sektorzahlen und voneinander abweichende Lagen des Inhaltsverzeichnis.

NEWDOS80 kann zwar mit entsprechenden Laufwerken dank seiner PDRIVE-Möglichkeiten fast alle dieser Disketten lesen, doch wird mancher Benutzer nach mehreren erfolglosen Kombinationsversuchen resigniert aufgegeben haben, die geeigneten Parameter zu finden. Und selbst wenn die Laufwerks-Spezifikationen bekannt sind, wäre statt des PDRIVE-Befehls ein entsprechendes Hilfsprogramm zur automatischen Einstellung recht nützlich. Wie ein solches Programm arbeitet, soll dieser Beitrag zeigen; ganz 'nebenbei' erfährt der Leser hier aber auch, wie dieses als Betriebssystem-Erweiterung realisiert wird und NEWDOS80 einige seiner Aufgaben prinzipiell löst.

'ID' arbeitet nämlich nicht als unabhängiges Hilfsprogramm am RAM-Ende oder im Utility-Bereich ab 5200H, sondern stellt eine vollwertige Ergänzung des Betriebssystems dar. Unter GDOS residiert es deshalb in der - hoffentlich noch - freien Datei SYS22/SYS, im NEWDOS80 kann SYS15/SYS hierfür herhalten; FORMS und SETCOM fehlen ja im Modell-1-NEWDOS.

Die Systemdatei SYS1/SYS ...

... enthält den NEWDOS-'CCP', also das System-Programm, welches Benutzer-Eingaben entgegennimmt und die Routinen des Betriebssystems ausführt, zumindest aber aufruft. SYS1/SYS besitzt hierfür eine Tabelle (Bild 1), in der alle System-Befehle eingetragen und mit Informationen über die aufzurufenden Systemdateien versehen sind. Diese Tabelle ist nun zunächst um einen zusätzlichen Befehl zu erweitern. Im GDOS 2.x müßte 'ID' gerade noch an das Ende dieser Tabelle passen; im NEWDOS80 ersetzt man sinvollerweise den überflüssigen Befehl 'CHAIN' (er ist ja völlig identisch mit 'DO') durch 'IDENT'. So umgeht man die sonst durch Verlängerung der Tabelle entstehende Adressverschiebung.

Tabellen-Aufbau

Jeder einzelne Eintrag in dieser Tabelle besteht aus der ASCII-Zeichenfolge des entsprechenden Befehlsworts, der jeweils drei Byte folgen:

1. Byte:

Bit 7 immer gesetzt: Endmarkierung des vorangehenden Befehlworts
Bit 6 gesetzt: Befehl ist unter Mini-DOS nicht erlaubt.
Bit 6 nicht gesetzt: Befehl ist unter Mini-DOS erlaubt.
Bits 5...0: Entry-Zähler:
Eine System-Datei kann durchaus mehrere Aufgaben übernehmen; der Entry-Zähler bestimmt neben den Bits 7...5 des Request-Codes (s.d.), welche Teilroutine der System-Datei den gewünschten Befehl ausführt. Er wird im Register C der System-Datei übergeben (**Bild 2** zeigt den Beginn von SYS7/SYS als Beispiel für die Auswertung des Entry-Zählers)

2. Byte:

Bits 0..4 minus 2: Request-Code:
Wird dieser Code im Register A dem Overlay-Loader (Anm. des Hrsg.: Hier bezieht sich der Autor auf Bild 3. Dieses Bild wurde ersatzlos gestrichen, da das Kapitel 3 unter anderem auch ein Listing des Overlay-Loaders enthält.) übergeben, so wird geprüft, ob die entsprechende System-Datei bereits geladen ist, diese andernfalls geladen und anschließend ausgeführt.

3. Byte:

Bit 0 gesetzt: Besitzt der anzugebende Dateiname keinen /TYP, so ist /CMD anzufügen.
Bit 1 gesetzt: dto. /JOB bzw. /JCL
Bit 3 gesetzt: Die aufgerufene System-Datei erwartet auf dem Stack den Zeiger auf die weiteren Parameter.
Bit 3 nicht gesetzt: Das Registerpaar HL dient als Zeiger auf die weiteren Parameter.
Bit 4 gesetzt: Weitere Parameter folgen dem Befehl.
Bit 4 nicht gesetzt: Keine weiteren Parameter dürfen folgen.
Bit 5 gesetzt: Der Befehl erwartet einen zweiten Dateinamen.
Bit 6 gesetzt: Eine neue Datei ist zu öffnen.
Bit 6 nicht gesetzt: Nur eine bestehende Datei ist zu öffnen. (vgl. Bit 7)
Bit 7 gesetzt: SYS1 öffnet eine neue oder bestehende Datei (vgl. Bit 6) zur weiteren Bearbeitung durch die aufgerufene System-Datei.

Als Beispiele hierzu seien einmal die Befehle 'KILL' und 'FORMAT' herausgegriffen:

KILL 80H,45H,90H

1.Byte Bit 6 nicht gesetzt => unter Mini-DOS erlaubt
1.Byte sonst 0 => Entry-Zähler ist 0
2.Byte Bits 0...4 -2 = 3 => Routine steht in SYS3/SYS
3.Byte Bit 7 gesetzt => SYS1/SYS öffnet die Datei
3.Byte Bit 4 gesetzt => Keine weiteren Parameter erlaubt

FORMAT C0H,28H,00H

1.Byte Bit 6 gesetzt -> Unter Mini-DOS verboten
1.Byte sonst 0 -> Entry-Zähler ist 0
2.Byte Bits 0...4 -2 - 6 -> Routine steht in SYS6/SYS
3.Byte Bit 7 nicht gesetzt -> Es ist keine Datei zu öffnen
3.Byte Bit 4 nicht gesetzt -> Parameter folgen

ID unter SYS22/SYS, IDENT unter SYS15/SYS einbinden

Zur Erweiterung der Befehls-Tabelle ist 'ID' also wie folgt zu codieren

ID 80H,F8H,00H

1.Byte Bit 6 nicht setzen -> Unter Mini-DOS erlaubt
1.Byte sonst 0 -> Entry-Zähler ist 0
2.Byte Bits 0...4 -2 - 22 -> Routine steht in SYS22/SYS
3.Byte Bit 7 nicht setzen -> Keine Datei öffnen
3.Byte Bit 4 nicht setzen -> Parameter erlauben

und an den in Bild 1 mit 'xx' bezeichneten Stellen als 49,44,80,F8,00 einzutragen. Sinngemäß steht dann unter NEWDOS80 an Stelle von 'CHAIN': 49,44,45,4E,54,80,F1,00. Obwohl die Bits 7, 6 und 5 des Request-Codes offensichtlich nicht vom Overlay-Loader ausgewertet werden, zeigt die Erfahrung, daß diese für ID dennoch gesetzt sein müssen; deshalb also als Request-Codes F8H bzw. F1H.

SYS22/SYS zu füllen ...

... gilt es nun. Bild 4 zeigt das Assembler-Listing des ID. Wer von Ihnen, liebe Leser, nicht an der näheren Arbeitsweise des Disketten-Identifizierers interessiert ist, mag die folgenden Ausführungen überlesen, seine Tabelle in SYS1/SYS modifizieren, das Listing abtippen, unter SYS22/SYS bzw. SYS15/SYS assemblieren und die Gebrauchsanleitung zur Kenntnis nehmen. Wer doch, lese hier weiter!

PDRIVE-Parameter und wie sie ausgewertet werden

Gerade die Fülle der verschiedenen möglichen (und noch mehr der unmöglichen!) Kombinationen macht das korrekte Zusammenstellen der Laufwerksparameter mühsam und die Aufgabe für ID scheinbar unüberschaubar groß, wollte man nicht den Floppy-Disk-Controller direkt programmieren. So elegant wie ein Read-Track-Befehl also zur Plattenidentifizierung auch sein mag, steht dem entgegen, daß nicht alle TRS-80-Kompatiblen den gleichen Floppy-Disk-Controller besitzen und so ID nicht universell auf allen NEWDOS/GDOS-Systemen einsetzbar wäre. Um diese Universalität zu erhalten, sollte man doch eine 'Umweg' ber das Betriebssystem wählen. Folgende überlegungen bestätigen, daß das Problem mit Hilfe des DOS relativ einfach lösbar sein muß:

Wollte das Betriebssystem selbst nämlich für jeden Diskettenzugriff die Laufwerks-Parameter erneut auswerten, wäre das eine zu zeitintensive Aufgabe. Tatsächlich findet deshalb diese Auswertung einmalig nur nach dem Erstellen einer neuen Parameterfolge mit dem PDRIVE-Befehl statt. Diese ergibt je logischem Laufwerk drei zusätzliche Bytes in der Systemtabelle, die es buchstäblich in sich haben und auch vom ID nutzbar sein müßten!

Die Systemtabelle im dritten BOOT-Sektor

Bild 5 zeigt eine solche Systemtabelle, in der neben den PD-Spezifikationen die SYSTEM-Optionen, ein evtl. Copyright-Vermerk und für's Genie III die Initialisierungsdaten für den Video-Controller HD 46505 Platz finden. Die PDRIVE-Spezifikationen werden nach jedem System-Warstart oder bei gegbener ',A'-Option des PDRIVE-Befehls in die System-'Zero-'Page ab 4300H kopiert und stehen dort zur Verfügung (Bild 6). Dank zwei dieser Bytes (Info-Byte 1 und Info- Byte 2) entfällt für das DOS die zeitraubende Auswertung der Daten für SWZ (NEWDOS:'TSR') TI und TD. Unser Diskettenidentifizierer begnügt sich sogar mit dem Info-Byte 2, um seine Aufgabe zu lösen: Wohl eine deutliche Rationalisierung gegenüber der Alternative, alle möglichen Kombination der Laufwerks-Parameter durchzuspielen!

Wann ist eine Diskette lesbar?

Tatsächlich genügt es also zum - zunächst nur probeweisen - Lesen einer unbekannten Diskette, das Info-Byte 2 (Testbyte) in Schreibdichte, Step-Impuls und Beginn der Sektornummerierung zu variieren und Leseversuche mindestens von Spur 1 zu starten. Nur über Spur 1 kann ID in Verbindung mit 80-Spur-Laufwerken feststellen, ob ein doppelter Step-Impuls nötig, also eine 40-Spur-Diskette vorhanden ist (Spur 0 von 40 Spuren ist logisch auch ohne einen solchen lesbar!). Weiter könnte Spur 0 in anderer Schreibdichte als die restliche Platte geschrieben sein; auch das wird sinnvollerweise zunächst ausgeschlossen und ein Sektor von einer 'normalen' Spur versucht zu lesen.

Besitzer von 8-Zoll-Laufwerken mögen auch Bit 7 im Testbyte entsprechend ändern und ID somit für ihre Belange erweitern (mir fehlte es leider an 8 Zoll-Laufwerken, sodaß ich diese Prüfung mangels Testmöglichkeiten ausließ). Andererseits entfällt bei 40-Spur-Laufwerken die Prüfung mit doppeltem Step-Impuls. Für diese ersten Leseversuche sind die Bits 1 und 6 (Spurnummerierung und SS/DS) irrelevant, sodaß mit einer dieser acht möglichen Prüfbyte-Kombinationen jede entsprechend formatierte Diskette lesbar gemacht werden kann!

Man suche die Systemtabelle!

Als nächster Schritt folgt nun die Aufgabe, die noch fehlenden Informationen - Sektor-Anzahl, einseitig/doppelseitig, 40 oder 35 Spuren, Lage des Inhaltsverzeichnis, usw. - zu beschaffen. Handelt es sich um keine NEWDOS/GDOS-Diskette, so wird man zu entsprechenden Disk-Zap-Programmen greifen müssen, mit denen die weitere Identifizierung kein großes Problem darstellen dürfte. Andernfalls sind weitere Leseversuche zur Prüfung überflüssig, da die fehlenden Informationen in der System-Tabelle im dritten BOOT-Sektor der Diskette zu finden sind. ID sucht diese Tabelle -hier spielt jetzt auch Bit 1 und 6 des Testbytes und evtl. andere Schreibdichte der Spur 0 eine Rolle - und liest hieraus die Laufwerksparameter der unbekannten Diskette.

Woran erkennt ID nun aber 'seine' Systemtabelle? Wegen der doch recht unterschiedlichen Tabellen-Inhalte ist eine genaue Identifizierung der Systemtabelle unmöglich, sodaß sich ID auf zwei Plausibilitätsprüfungen beschränken muß:

1.) Im Info-Byte 1 muß mindestens eins der Bits 2...4 gesetzt sein; dies ist auch die Prüfung, die der PDRIVE-Befehl durchführt, um im Zweifelsfalle >>ACHTUNG<< zu melden.

2.) Im relativen Byte EF der System-Tabelle muß ein A5-Byte stehen; dies ist auch die Prüfung, die SYS0/SYS durchführt, um im Zweifelssfalle 'Schlechte Initialisierungsdaten auf Systemdiskette' zu melden.

ID erkennt also eine korrekte System-Tabelle ebenso (un-) sicher wie das Betriebssystem selbst; dies dürfte für alle Fälle ausreichend sein!

ID-Gebrauchsanleitung:

Die restlichen 'Feinheiten' auf dem Wege zur möglichst vollständigen Diskettenerkennung findet der interessierte Leser in den Kommentarzeilen des Assemblerlistings (Bild 4) erklärt; zum Abschluß folgt hier nun noch die zusätzliche Handbuch-Seite zum neuen Befehl 'ID':

ID LAUFWERKS-PARAMETER IDENTIFIZIEREN

ID,lw#<,pd#>

Dieser Befehl ermöglicht das automatische Erkennen der (unbekannten) Laufwerksparameter der Diskette im Laufwerk lw#:

Ist die Diskette im angegeben Laufwerk lesbar und enthält sie die Datei GDOS/SYS, werden analog zum "DISK"-Befehl die RAM-internen PD-Spezifikationen auf das erkannte Format eingestellt. Zur zusätzlichen Information führt ID dann einen "PD,lw#"-Befehl aus. Die PD-Spezifikationen auf der System-Diskette bleiben unverändert.

Wird zusätzlich der Parameter pd# gegeben, so erfolgt bei lesbarer Diskette ein Eintrag in die PD-Spezifikationen für das Laufwerk pd# auf der System-Diskette. Anschließend wird ein "PD,0,A"-Befehl ausgeführt.

Scheint die Diskette lesbar, aber fehlt die Datei GDOS/SYS auf dieser, erfolgt die Fehlermeldung "Lesefehler Inhaltsverzeichnis", sowie eine Angabe über Spuren und Schreibdichte der Diskette. Die RAM-internen PD-Parameter werden so weit wie möglich diesem vermuteten Format angepaßt, so daß eventuell ein Diskettenzugriff mit entsprechenden Hilfsprogrammen möglich wird. Wird unter diesen Umständen auf einen Diskettenzugriff verzichtet, sonst zur Herstellung des "Normal"-Zustandes - wie auch beim DISK-Befehl - ein "PD,0,A"-Befehl erforderlich.

Ist die Diskette unter GDOS nicht lesbar, wird ein "nicht kompatible Laufwerke oder Disketten"-Fehler gemeldet. Die PD-Spezifikationen sowohl im RAM als auch auf der System-Diskette bleiben unberührt.

ACHTUNG!

Wird für lw# das Systemlaufwerk (also 0) angegeben, fordert ID zu einem Diskettenwechsel auf. Weiter geht ID dann davon aus, daß es sich bei dieser Diskette um eine Systemdiskette handeln muß! Kann die Diskette einwandfrei identifiziert werden, so verbleibt sie als System-Diskette im Laufwerk und die PD-Spezifikationen werden entsprechend angepaßt. Eine evtl. gegebene pd#-Option bleibt in diesem Fall unberücksichtigt, da die PD-Spezifikationen ja bereits auf dieser Diskette korrekt vorliegen. Vorgesehen wurde diese Möglichkeit, um z.B. in einem 1-Disk-System GDOS-kompatible Betriebssysteme anderen Formats benutzen zu können.

```
DRV 00   D54C 20A9 CB59 2802 0102 004F E3E5 79E6 .L...Y(....O..y.
0   10   0728 0EE5 21BC 5123 2323 3D20 FACD 2A4F .(..!.Q###-...*O
0H  20   E179 01D3 49C5 CB7F C806 0021 0042 CB77 .y..I.. ...!.B.w
     30  CA24 44C3 2044 D5C5 011C 091A FE3A 280A .$D..D.......:(.
DRS 40   FE2F 3806 281B 0D13 10F1 2323 E5EB 0600 ./8.(.....##....
143250   0954 5D2B 1313 13ED B8E1 0E03 EDB8 3E2F .T]+..........>/
598H60   12C1 D1C9>3084 F000 4081 F000 4149 4B80 ....0...@...AIK.
     70  5300 4150 5045 4E44 C068 0041 5454 5249 S.APPEND.h.ATTRI
     80  4285 E988 4155 544F 84E9 0042 3286 EB00 B...AUTO...B2...
     90  424C 81E5 0042 4F4F 548A EB10 4252 4541 BL...BOOT...BREA
     A0  4B85 E500 434C 5389 E310 434F 4E54 C5EB K...CLS...CONT..
     B0  0043 4F50 59C0 4800 4352 4541 5445 82F0 .COPY.H.CREATE..
FRS  C0  4044 4154 554D 8BE9 0044 4952 802A 0044 @DATUM...DIR.*.D
2    D0  4953 4B83 FF00 444F C3EB 8A44 5282 FE00 ISK...DO...DR...
2H   E0  4455 4D50 87E9 C845 87F0 0046 4F52 4D88 DUMP...E...FORM.
     F0  FE00 4652 4545 804A 0048 494D 454D 82E9 ..FREE.J.HIMEM..
DRV 00   0049 802A 0049 4E46 4F81 FF00 0102 0050 .I.*.INFO......P
0   10   4A4B 4C80 A510 4B49 4C4C 8045 904C 4388 JKL...KILL.E.LC.
0H  20   E500 4C46 81FE 004C 4942 82E3 004C 4953 ..LF...LIB...LIS
     30  5485 F088 4C4F 4144 80A4 504C 5754 81F9 T...LOAD..PLWT..
DRS 40   004E 81E4 B04E 4446 C028 0050 4155 5345 .N...NDF.(.PAUSE
143350   88EB 0050 4483 E900 504F 5254 82FF 0050 ...PD...PORT...P
599H60   5249 4E54 86F0 8850 524F 5486 E900 5055 RINT...PROT...PU
     70  5247 4589 E900 5280 2300 5381 E900 5354 RGE...R.#.S...ST
     80  4D54 89EB 0055 4852 82E5 0056 2B84 E500 MT...UHR...V+...
     90  5632 3480 FA00 5A86 FF00 5A45 4954 8AE9 V24...Z...ZEIT..
     A0  0026 83E5 0021 83EB 8A3B 86E3 002F 85E3 .&...!...;.../..
     B0  003F 82E3 003E C048 004D 3E82 EBB0 2323 .?...>.H.M>...##
FRS  C0  82F9 0046 23C0 FB00 3830 83F9 0036 3484 ...F#...80...64.
3    D0  F900 4444 4581 F100 xxxx xxxx xx00<0021 ..DDE.........!
3H   E0  584F 0E40 0608 7ECB 7F23 2005 CDB7 5110 XO.@..~. #....Q.
     F0  F523 237E B7CA B551 0DCC B551 28E4 CDAD .##~...Q...Q(...
```

Bild 1: LIB-Tabelle des SYS1/SYS (Die mit 'xx' bezeichneten Byte sind im GDOS 2.1 frei zum Eintrag von 'ID')

```
00000 ;-----------------------------------------
00010 ;       Befehlsanwahl über Entry-
00020 ;       Counter C. (Labels entsprechen
00030 ;       weitgehend den LIB-CMDs!)
00040 ;-----------------------------------------
00050
00060         DEC     C
00070         JR      Z,SYSTEM        ; geht über SYS7, da
00080         DEC     C               ; Kennwortprüfung nötig!
00090         JP      Z,HIGMEM
00100         DEC     C
00110         JR      Z,PDRIVE        ; geht über SYS7, da
00120         DEC     C               ; Kennwortprüfung nötig!
00130         JP      Z,AUTO
00140         DEC     C
00150         JP      Z,ATTRIB
00160         DEC     C
00170         JP      Z,PROT
00180         DEC     C
00190         JP      Z,DUMP
00200         DEC     C
00210         JP      Z,REQ8
00220         DEC     C
00230         JR      Z,PURGE         ; geht über SYS7, da
00240                                 ; Kennwortprüfung nötig!
00250         END     SYS7
```

Bild 2: Ausschnitt aus der Entry-Zähler-Auswertung des SYS7/SYS

```
                00000 ;-------------------------------------------------------
                00010 ;       Automatische PDRIVE-Erkennung unter den
                00020 ;       Betriebssystemen GDOS 2.x und NEWDOS/80.2
                00030 ;       "ID,lw#<,pd#>"
                00040 ;-------------------------------------------------------
                00050
                00060         ;       SYS22:ID - SYS15:IDENT
                00061         ;       ABB 4 : QUELLPROGRAMM
                00070
4467            00080 MSGOUT  EQU     4467H           ; UP:Text (HL) ausgeben
0049            00090 INCHAR  EQU     0049H           ; UP:Tastendruck -> A
4CD5            00100 TERM1   EQU     4CD5H           ; UP:Terminator testen.
0033            00110 OUTCH   EQU     0033H           ; UP:A -> Bildschirm
402D            00120 DOS     EQU     402DH           ; Rückkehradresse GDOS
4318            00130 CMDBUF  EQU     4318H           ; Eingabepuffer
4436            00140 SECREA  EQU     4436H           ; UP:Sektor lesen
4439            00150 WRISEC  EQU     4439H           ; UP:Sektor schreiben
4378            00160 TBYTE   EQU     4378H           ; Zeiger auf Testbyte
437B            00170 PDRIVE  EQU     437BH           ; Zeiger auf PD-Tabelle
47EC            00180 DSKMNT  EQU     47ECH           ; UP:Diskette im Laufw.?
                00190
4D00            00200         ORG     4D00H           ; DOS-Overlay-Bereich
                00210
                00220 ;---------------------------------------
                00230 ;       Befehlsparameter einlesen
                00240 ;---------------------------------------
                00250
4D00 112A2A     00260 START:  LD      DE,2A2AH        ; 2 Sternchen in rechte
4D03 ED533E3C   00270         LD      (3C3EH),DE      ; obere Bildecke setzen.
4D07 7E         00280         LD      A,(HL)          ; Zeichen hinter ID,...
4D08 D630       00290         SUB     '0'             ; ASCII-Korrektur
4D0A 3804       00300         JR      C,NODN          ; Keine Ziffer
4D0C FE0A       00310         CP      10              ; Ziffer?
4D0E 3804       00320         JR      C,LWNUM         ; Jawoll!
4D10 3E20       00330 NODN:   LD      A,32            ; Fehlermeldung:
4D12 B7         00340         OR      A               ; Unzulässiges oder
4D13 C9         00350         RET                     ; fehlendes Laufwerk.
                00360
4D14 DD21824F   00370 LWNUM:  LD      IX,BUFFER       ; Zeiger auf Sektorpuffer
4D18 326A4F     00380         LD      (DRIVE),A       ; Laufwerk-Nr. -> FCB
4D1B B7         00390         OR      A               ; Systemlaufwerk gewählt?
4D1C 2004       00400         JR      NZ,NDCHG        ; Nein!
4D1E DDCBFFEE   00410         SET     5,(IX-1)        ; Setze Disk-Wechsel-Flag
4D22 23         00420 NDCHG:  INC     HL              ; Eingabezeiger weiter.
4D23 CDD54C     00430         CALL    TERM1           ; Befehl beendet?
4D26 281D       00440         JR      Z,NOWRT         ; Jawoll!
4D28 3003       00450         JR      NC,KOMMA        ; Parameter fogt!
4D2A 3E34       00460 ERROR:  LD      A,52            ; Sonst:
4D2C C9         00470         RET                     ; Fehlermeldung!
                00480
4D2D 7E         00490 KOMMA:  LD      A,(HL)          ; Wenn Komma gefunden,
4D2E D630       00500         SUB     '0'             ; ASCII-Korrektur pd#
4D30 38DE       00510         JR      C,NODN          ; Unter 0: Falsch!
4D32 28DC       00520         JR      Z,NODN          ; gleich 0: Nicht richtig
4D34 FE0A       00530         CP      10              ; über 10:
4D36 30D8       00540         JR      NC,NODN         ; auch falsch!
4D38 32604E     00550         LD      (PDEST),A       ; sonst Ziel-pd# merken.
4D3B 23         00560         INC     HL
```

```
4D3C 7E       00570          LD      A,(HL)
4D3D FE0D     00580          CP      0DH
4D3F 20E9     00590          JR      NZ,ERROR        ; Kein korr. Endzeichen!
              00600
4D41 DDCBFFF6 00610          SET     6,(IX-1)        ; Setze "pd#"-Optionsflag
              00620
4D45 DDCBFF6E 00630 NOWRT:   BIT     5,(IX-1)        ; Diskwechsel nötig?
4D49 2806     00640          JR      Z,NOWECH        ; Nein, nicht!
4D4B 21F34E   00650          LD      HL,TEXT1        ; Aufforderung zum selben
4D4E CD9F4E   00660          CALL    PROMPT          ; ausgeben; <ENTER> ?
              00670
              00680 ;----------------------------------------
              00690 ;        Auf vorhandenes Laufwerk bzw.
              00700 ;        vorhandene Diskette prüfen.
              00710 ;----------------------------------------
              00720
4D51 3A6A4F   00730 NOWECH:  LD      A,(DRIVE)       ; Diskette im Laufwerk?
4D54 CDEC47   00740          CALL    DSKMNT
4D57 C28B4E   00750          JP      NZ,EXIT         ; Nein, via EXIT 'raus!
              00760
              00770 ;-------------------------------------------------
              00780 ;        Laut gewähltem Laufwerk in der PD-
              00790 ;        Tabelle die Position des zuständigen
              00800 ;        "Testbytes" suchen.
              00810 ;-------------------------------------------------
              00820
4D5A 11644F   00830          LD      DE,FCB1         ; DE zeigt auf FCB
4D5D 216E43   00840          LD      HL,TBYTE-10     ; Testbyte für PD lw#
4D60 010A00   00850          LD      BC,10           ; errechnen.
4D63 3A6A4F   00860          LD      A,(DRIVE)
4D66 3C       00870          INC     A
4D67 3D       00880 GETBYT:  DEC     A
4D68 09       00890          ADD     HL,BC
4D69 20FC     00900          JR      NZ,GETBYT
              00910
              00920 ;-------------------------------------------------
              00930 ;        Leseversuche mit unterschiedlichen
              00940 ;        Testbytes unternehmen
              00950 ;-------------------------------------------------
              00960
4D6B FD21D54E 00970          LD      IY,DNSTAB-6     ; Zeiger Density-Texte.
4D6F 4E       00980          LD      C,(HL)          ; Testbyte in C retten
4D70 AF       00990          XOR     A               ; 80-Spur, Single Density
4D71 CDC74D   01000          CALL    TEST            ; prüfen.
4D74 3E03     01010          LD      A,3             ; 80-Spur, Double Density
4D76 CDC74D   01020          CALL    TEST            ; prüfen.
4D79 3E04     01030          LD      A,4             ; 40-Spur, Single Density
4D7B CDC74D   01040          CALL    TEST            ; prüfen.
4D7E 3E07     01050          LD      A,7             ; 40-Spur, Double Density
4D80 CDC74D   01060          CALL    TEST            ; prüfen.
4D83 FD21D54E 01070          LD      IY,DNSTAB-6     ; Textzeiger rücksetzen.
4D87 3E10     01080          LD      A,10H           ; gleiche Tests wie oben,
4D89 CDC74D   01090          CALL    TEST            ; jedoch Sektor-
4D8C 3E13     01100          LD      A,13H           ; Nummerierung ab 1 statt
4D8E CDC74D   01110          CALL    TEST            ; ab 0.
4D91 3E14     01120          LD      A,14H
4D93 CDC74D   01130          CALL    TEST
4D96 3E17     01140          LD      A,17H
```

```
4D98 CDC74D   01150          CALL    TEST
              01160
              01170 ;-----------------------------------------------
              01180 ;        Alle Tests waren fehlerhaft. Prüfen,
              01190 ;        ob völlig unlesbar oder nur fehlende
              01200 ;        PD-Tabelle festgestellt wurde.
              01210 ;-----------------------------------------------
              01220
4D9B DDCBFF7E 01230 TSTEND: BIT     7,(IX-1)        ; Keine PD-Tabelle?
4D9F 201F     01240         JR      NZ,READNO       ; Kein Sektor lesbar
4DA1 DDCBFF6E 01250         BIT     5,(IX-1)        ; Wechselflag gesetzt?
4DA5 2803     01260         JR      Z,ACTU          ; Nein, Testbyte o.k.
4DA7 71       01270         LD      (HL),C          ; Originaltestbyte
4DA8 1804     01280         JR      ORI             ; restaurieren.
4DAA 3A804F   01290 ACTU:   LD      A,(BUFFER-2)    ; Testbyte für letzten
4DAD 77       01300         LD      (HL),A          ; lesbaren Sektor.
4DAE FDE5     01310 ORI:    PUSH    IY              ; Density-Text
4DB0 E1       01320         POP     HL              ; ausgeben.
4DB1 CD6744   01330         CALL    MSGOUT
4DB4 211E4F   01340         LD      HL,TEXT2        ; 'Diskette nicht lesbar'
4DB7 CD6744   01350         CALL    MSGOUT          ; ausgeben.
4DBA 3E11     01360         LD      A,17            ; 'Lesefehler Inhalts-
4DBC B7       01370         OR      A               ;  verzeichnis'
4DBD C38B4E   01380         JP      EXIT            ; melden.
              01390
4DC0 71       01400 READNO: LD      (HL),C          ; Original-Testbyte
4DC1 3E3C     01410         LD      A,60            ; 'nicht kompatible Lauf-
4DC3 B7       01420         OR      A               ;  werke oder Disketten'
4DC4 C38B4E   01430         JP      EXIT            ; melden.
              01440
              01450 ;-----------------------------------------------------
              01460 ;        Leseversuche von Sektor 36, um
              01470 ;        festzustellen, ob:
              01480 ;        a) Doppelter Step-Impuls nötig ist
              01490 ;        b) Spur 0 in anderer Density geschrieben
              01500 ;-----------------------------------------------------
              01510
4DC7 77       01520 TEST:   LD      (HL),A          ; Testbyte -> PD-Tabelle
4DC8 79       01530         LD      A,C             ; Original-Testbyte -> A
4DC9 010600   01540         LD      BC,6            ; Zeiger in Density-
4DCC FD09     01550         ADD     IY,BC           ; Tabelle weiterstellen.
4DCE 4F       01560         LD      C,A             ; Testbyte wieder in C
4DCF 3E24     01570         LD      A,36            ; Mindestens Spur 1
4DD1 326E4F   01580         LD      (SECTOR),A      ; (wegen Doppelstep)
4DD4 11644F   01590         LD      DE,FCB1
4DD7 CDD04E   01600         CALL    READSEC         ; Leseversuch Sektor 36.
4DDA 2805     01610         JR      Z,SECOK         ; OK:Sektor lesbar!
4DDC DDCBFFFE 01620         SET     7,(IX-1)        ; Fehlerbit setzen und
4DE0 C9       01630         RET                     ; zurück!
              01640
              01650 ;-----------------------------------------------------
              01660 ;        Leseversuche von Sektor 2. Feststellen,
              01670 ;        ob dort die Systemtabelle (PDRIVE und SYSTEM)
              01680 ;        abgelegt ist.
              01690 ;-----------------------------------------------------
              01700
4DE1 DDCBFFBE 01710 SECOK:  RES     7,(IX-1)        ; Fehlerbit löschen
4DE5 7E       01720         LD      A,(HL)          ; Testbyte
```

```
4DE6 32804F   01730          LD      (BUFFER-2),A    ; merken.
4DE9 CDC84E   01740          CALL    RSEC2           ; Sektor 2 lesen.
4DEC 280C     01750          JR      Z,SEC2OK        ; Okay!
4DEE 7E       01760          LD      A,(HL)          ; Wenn 36, aber nicht 2
4DEF EE01     01770          XOR     1               ; lesbar, => Verdacht:
4DF1 77       01780          LD      (HL),A          ; Spur 0 in anderer
              01790                                  ; Density geschrieben!
              01800                                  ; So nochmals
4DF2 CDC84E   01810          CALL    RSEC2           ; Sektor 2 lesen.
4DF5 2803     01820          JR      Z,SEC2OK        ; Jawoll, es ging!
4DF7 D1       01830          POP     DE              ; Da stimmt 'was nicht!
4DF8 18A1     01840          JR      TSTEND          ; Test beenden.
              01850
4DFA CDAA4E   01860 SEC2OK:  CALL    PDCHK           ; Plausibilitätskontrolle
4DFD D1       01870          POP     DE              ; Rückkehradresse löschen
4DFE 2818     01880          JR      Z,PDOK          ; Kontrolle O.K.!
4E00 7E       01890          LD      A,(HL)          ; Nein, dann vielleicht
4E01 EE02     01900          XOR     2               ; andere Spurnummerierung
4E03 77       01910          LD      (HL),A
4E04 CDC84E   01920          CALL    RSEC2           ; Nochmals Leseveruch!
4E07 2809     01930          JR      Z,CHKPD         ; So geht's jetzt!
4E09 7E       01940          LD      A,(HL)          ; Density 'zurück'!
4E0A EE01     01950          XOR     1
4E0C 77       01960          LD      (HL),A
4E0D CDC84E   01970          CALL    RSEC2           ; wenn auch jetzt nicht
4E10 2079     01980          JR      NZ,EXIT         ; lesbar=> alles Falsch
              01990
4E12 CDAA4E   02000 CHKPD:   CALL    PDCHK           ; Sektor 2 = PD-Tabelle?
4E15 C29B4D   02010          JP      NZ,TSTEND       ; Nein, PD-Tabelle fehlt!
              02020
              02030 ;-------------------------------------------------------
              02040 ;       Ein Test war erfolgreich. Genaue PDRIVE-
              02050 ;       Daten aus dem Sektor 2 der geprüften Diskette
              02060 ;       in die RAM-PD-Tabelle kopieren
              02070 ;-------------------------------------------------------
              02080
4E18 7E       02090 PDOK:    LD      A,(HL)          ; Testbyte retten.
4E19 E5       02100          PUSH    HL
4E1A 11F9FF   02110          LD      DE,-7
4E1D 19       02120          ADD     HL,DE           ; 7 Byte vorher liegt
4E1E EB       02130          EX      DE,HL           ; Anfang der PD-Tabelle
4E1F 010A00   02140          LD      BC,10           ; PD-Parameter in diese
4E22 21824F   02150          LD      HL,BUFFER       ; vom Sektorpuffer
4E25 EDB0     02160          LDIR                    ; kopieren.
4E27 E1       02170          POP     HL
4E28 CB57     02180          BIT     2,A             ; War doppelter Step
4E2A 280A     02190          JR      Z,N2STEP        ; nötig? Nein!
4E2C CBD6     02200          SET     2,(HL)          ; Testbyte im RAM
4E2E DDCB07D6 02210          SET     2,(IX+7)        ; Testbyte im Sektor!
4E32 DDCB0EDE 02220          SET     3,(IX+14)       ; TI="L" setzen
              02230
              02240 ;-------------------------------------------------
              02250 ;       Wenn pd#-Option gesetzt und zulässig,
              02260 ;       die PD,0-Zeile der geprüften Diskette
              02270 ;       in die zugehörige PD-Zeile der System-
              02280 ;       Diskette kopieren.
              02290 ;-------------------------------------------------
              02300
```

```
4E36 DDCBFF76 02310 N2STEP: BIT     6,(IX-1)        ; Write-Flag gesetzt?
4E3A 2009     02320         JR      NZ,OPTPD        ; Ja, Option pd# gegeben!
4E3C 3A6A4F   02330         LD      A,(DRIVE)       ; Gewähltes lw -> A
4E3F 47       02340 RAUS:   LD      B,A
4E40 CD8B4E   02350         CALL    EXIT            ; evtl. Diskwechsel.
4E43 183E     02360         JR      DISPLY          ; PD-Tabelle ausgeben.
              02370
4E45 DDCBFF6E 02380 OPTPD:  BIT     5,(IX-1)        ; Diskwechselflag?
4E49 2803     02390         JR      Z,ERLAU         ; Nein, pd#-Option erl.
4E4B AF       02400         XOR     A
4E4C 18F1     02410         JR      RAUS
              02420
4E4E 11724F   02430 ERLAU:  LD      DE,FCB0         ; PDRIVE-Tabelle
4E51 CDD04E   02440         CALL    REASEC          ; Disk0 lesen
4E54 2035     02450         JR      NZ,EXIT         ; nanu, unlesbar?
4E56 21924F   02460         LD      HL,BUFFER+16    ; Zeiger System-PD's
4E59 11824F   02470         LD      DE,BUFFER       ; Zeiger Testdisk-PD's
4E5C 011000   02480         LD      BC,16           ; Länge einer PD-Zeile
4E5F 3E00     02490         LD      A,$-$           ; A:Ziel-pd# für
4E60          02500 PDEST   EQU     $-1             ; pd#-Option
4E61 3C       02510         INC     A               ; wegen DEC (s.u.)!
4E62 3D       02520 GETDST: DEC     A               ; Ziel-pd# zählen
4E63 2803     02530         JR      Z,FNDEST        ; Null: PD-Par. gefunden
4E65 09       02540         ADD     HL,BC           ; sonst: 16 Byte weiter
4E66 18FA     02550         JR      GETDST          ; zeigen!
              02560
4E68 EB       02570 FNDEST: EX      DE,HL           ; Quelle: Testdisk PD,0
4E69 EDB0     02580         LDIR                    ; Ziel: Systemdisk pd#
4E6B 3E02     02590         LD      A,2             ; PDRIVE-Sektor
4E6D 327C4F   02600         LD      (SEC0),A
4E70 11724F   02610         LD      DE,FCB0
4E73 CD3944   02620         CALL    WRISEC          ; zurückschreiben.
4E76 2013     02630         JR      NZ,EXIT         ; etwa Fehler?
4E78 CD8B4E   02640         CALL    EXIT            ; für evtl. Disk-Wechsel
              02650
              02660 ;-----------------------------------------------
              02670 ;       "PD,lw#" bzw. "PD,0,A" via SYS16/SYS
              02680 ;       ausführen.
              02690 ;-----------------------------------------------
              02700
4E7B 0600     02710         LD      B,0             ; B=Laufwerk-Nummer
4E7D 212C41   02720         LD      HL,412CH        ; ",A" im CMDBUF
4E80 221F43   02730         LD      (CMDBUF+7),HL   ; ablegen.
4E83 212043   02740 DISPLY: LD      HL,CMDBUF+8
4E86 3E32     02750         LD      A,32H           ; Requestcode SYS16/SYS
4E88 360D     02760         LD      (HL),13         ; <ENTER> für PD-Befehl
4E8A EF       02770         RST     28H             ; PDRIVE,0 ausführen
              02780
              02790 ;-------------------------------------------------
              02800 ;       Programm-Ausgang, wenn angenommen werden
              02810 ;       kann, daß Diskettenwechsel zur System-
              02820 ;       Diskette nötig wird.
              02830 ;-------------------------------------------------
              02840
4E8B F5       02850 EXIT:   PUSH    AF              ; Fehler-Flag retten.
4E8C DDCBFF6E 02860         BIT     5,(IX-1)        ; Disk-Wechsel-Flag
4E90 280B     02870         JR      Z,ENDE          ; Nicht gesetzt, Fertig!
4E92 F1       02880         POP     AF              ; Diskettenwechsel nur
```

```
4E93 F5       02890          PUSH    AF              ; dann erforderlich, wenn
4E94 B7       02900          OR      A               ; Fehler in ID,0 fest-
4E95 2806     02910          JR      Z,ENDE          ; gestellt!
4E97 21374F   02920          LD      HL,TEXT3        ; Aufforderung zum
4E9A CD9F4E   02930          CALL    PROMPT          ; Diskettenwechsel
4E9D F1       02940 ENDE:    POP     AF              ; Fehlerflag restaurieren
4E9E C9       02950          RET
              02960
              02970 ;-----------------------------------------------------
              02980 ;        UP: Meldung zum Diskettenwechsel
              02990 ;            ausgeben und Bestätigung abwarten.
              03000 ;-----------------------------------------------------
              03010
4E9F CD6744   03020 PROMPT:  CALL    MSGOUT          ; Disk-Wechsel-Text aus-
4EA2 CD4900   03030 WAITOD:  CALL    INCHAR          ; geben und auf Bestäti-
4EA5 FE0D     03040          CP      13              ; gung <ENTER>
4EA7 20F9     03050          JR      NZ,WAITOD       ; warten.
4EA9 C9       03060          RET
              03070
              03080 ;-------------------------------------------
              03090 ;        UP: Testet gelesenen Sektor auf
              03100 ;            gültige System-Tabelle
              03110 ;-------------------------------------------
              03120
4EAA 3A7150   03130 PDCHK:   LD      A,(BUFFER+0EFH) ; Kontrolle, ob gelesener
4EAD FEA5     03140          CP      0A5H            ; Sektor die PD-Tabelle
4EAF C0       03150          RET     NZ              ; enthält.
4EB0 E5       03160          PUSH    HL              ; Register retten.
4EB1 C5       03170          PUSH    BC
4EB2 21844F   03180          LD      HL,BUFFER+2     ; (HL) erstes Info-Byte 1
4EB5 111000   03190          LD      DE,10H          ; Abstand der Info-Bytes
4EB8 060A     03200          LD      B,10            ; Anzahl der Info-Bytes
4EBA 7E       03210 TBLOOP:  LD      A,(HL)          ; A:Info Byte 1
4EBB E61C     03220          AND     1CH             ; Bits 2,3,4 makieren
4EBD 2805     03230          JR      Z,NOTOK         ; keins gesetzt = Fehler
4EBF 19       03240          ADD     HL,DE           ; (HL) nächstes Info-Byte
4EC0 10F8     03250          DJNZ    TBLOOP          ; alle 10 testen.

4EC2 3EFF     03260          LD      A,-1
4EC4 3C       03270 NOTOK:   INC     A
4EC5 C1       03280 OKAY:    POP     BC              ; Register restaurieren
4EC6 E1       03290          POP     HL
4EC7 C9       03300          RET                     ; Zero=Test O.K.
              03310
              03320 ;-------------------------------------------
              03330 ;        UP: Sektor - meist 2 - lesen und
              03340 ;            Sternchen umschalten.
              03350 ;-------------------------------------------
              03360
4EC8 3E02     03370 RSEC2:   LD      A,2             ; FCB zum Lesen der PD-
4ECA 326E4F   03380          LD      (SECTOR),A      ; Tabelle von Sektor 2
4ECD 11644F   03390          LD      DE,FCB1         ; vorbereiten.
4ED0 3A3F3C   03400 REASEC:  LD      A,(3C3FH)       ; Vor Lesen eines jeden
4ED3 EE0A     03410          XOR     10              ; Sektors Sternchen in
4ED5 323F3C   03420          LD      (3C3FH),A       ; rechter oberer Bild-
4ED8 C33644   03430          JP      SECREA          ; Ecke umschalten.
              03440
              03450 ;-------------------------------------------
```

```
                03460 ;       Tabelle mit Spur/Density-Texten
                03470 ;-----------------------------------------
                03480
4EDB 38         03490 DNSTAB: DEFM    '80/SD'
4EE0 03         03500         DEFB    3
4EE1 38         03510         DEFM    '80/DD'
4EE6 03         03520         DEFB    3
4EE7 34         03530         DEFM    '40/SD'
4EEC 03         03540         DEFB    3
4EED 34         03550         DEFM    '40/DD'
4EF2 03         03560         DEFB    3
                03570
                03580 ;-------------------------------------------------
                03590 ;       Fehlermeldungs- und Diskwechseltexte
                03600 ;-------------------------------------------------
                03610
4EF3 3C         03620 TEXT1:  DEFM    '<ENTER>, wenn Testdiskette in '
4F11 4C         03630         DEFM    'Laufwerk 0 '
4F1C 070D       03640         DEFW    0D07H
                03650
4F1E 2D         03660 TEXT2:  DEFM    '-Diskette nicht lesbar: '
4F36 03         03670         DEFB    03
4F37 0A         03680 TEXT3:  DEFB    10
4F38 3C         03690         DEFM    '<ENTER>, wenn Systemdiskette in '
4F58 4C         03700         DEFM    'Laufwerk 0'
4F62 070D       03710         DEFW    0D07H
                03720
                03730 ;-------------------------------------------------
                03740 ;       File-Control-Blocks:
                03750 ;       FCB1 für Laufwerk der Testdiskette
                03760 ;       FCB0 für Systemlaufwerk
                03770 ;-------------------------------------------------
                03780
4F64 82         03790 FCB1:   DEFB    82H     ; Datei ist geöffnet
4F65 60         03800         DEFB    60H     ; Sektorpuffer (noch) leer
4F66 00         03810         DEFB    0       ; keine ATTRIBute
4F67 824F       03820 BUFPTR: DEFW    BUFFER  ; Sektorpuffer
4F69 00         03830         DEFB    0       ; relatives Sektorbyte
4F6A 01         03840 DRIVE:  DEFB    1       ; Laufwerk-Nummer
4F6B FF         03850         DEFB    255     ; DEC
4F6C 00         03860         DEFB    0       ; EoF-Byte
4F6D 00         03870         DEFB    0       ; Sektorlänge = 256 Bytes
4F6E 0000       03880 SECTOR: DEFW    0000H   ; Sektor
4F70 FFFF       03890         DEFW    0FFFFH  ; EoF-Sektor
                03900
4F72 82         03910 FCB0:   DEFB    82H
4F73 60         03920         DEFB    60H
4F74 00         03930         DEFB    0
4F75 924F       03940         DEFW    BUFFER+16
4F77 00         03950         DEFB    0
4F78 00         03960         DEFB    0
4F79 FF         03970         DEFB    255
4F7A 00         03980         DEFB    0
4F7B 00         03990         DEFB    0
4F7C 0200       04000 SEC0:   DEFW    2
4F7E FFFF       04010         DEFW    0FFFFH
                04020
4F80 0000       04030         DEFW    0       ; Platz für Fehlerbit/Testbyte
```

```
0110            04040 BUFFER: DEFS    256+16
                04050
0392            04060 XXXXXX  EQU     $-START
4D00            04070         END     START
0000  TOTAL ERRORS
22488  TEXT AREA BYTES LEFT
```

```
                                                    NLD
PDRIVE-Parameter:                                  LLD¦
                                        (2)         ¦ ¦
        GAT-Größe   SEK                 ¦ SBIV    ¦ ¦ SWZ
       SBIV ¦(1)SP ¦ EIB                ¦ ¦ AEIV¦ ¦ ¦ TI*) TD*)
         ¦  ¦ ¦  ¦ ¦ ¦                  ¦ ¦ ¦  ¦ ¦ ¦ ¦ ¦    ¦
         V  V V  V V V                  V V V  V V V V V    V
DRV 00   3060 5050 2406 0043 3006 0000 0004 0406 0`PP$..C0.......
0   10   3060 5050 2406 0043 3006 0000 0004 0406 0`PP$..C0.......
0H  20   1130 5228 1203 0003 1102 0000 0204 0404 .0R(............
    30   3048 D350 1204 0042 3006 0000 0384 0402 0H.P...B0.......
DRS 40   1128 0728 0A02 0004 1102 0000 0301 0800 .(.(............
2   50   1128 0728 1404 0044 1102 0000 0301 0802 .(.(...D........
2H  60   1830 5228 1203 0007 1803 0000 0204 0C04 .0R(............
    70   1130 5228 2406 0047 1102 0000 0204 0C06 .0R($..G........
TRK 80   1148 1328 1202 0005 1102 0000 0304 0804 .H.(............
0   90   1128 3328 1206 0011 1102 0000 0304 1004 .(3(............

SYSTEM-Optionen (erste 'Hälfte'):

0H  A0   0403 0000 005F 0314 FF02 0000 1111 0B52 ....._.........R
         AL ↑ AN ↑  ↑  AM ↑ AX ↑
            AW   AO   BI   AV   BJ

    B0   0000 0000 0000 0000 0000 0000 0000 0000 ................

CRTC-Initialisierungsdaten:

TRS C0   6240 5242 1908 1016 000B 2B0B 0000 0000 b@RB......+.....

System-Optionen (zweite 'Hälfte'):

2   D0   =AP= 0000 0000 0000 0000 0000 0000 0000 ................
2H  E0   5472 6F6D 6D65 7363 686C 7B67 6572 CCA5 Trommeschläger..
    F0   30F8 8000 0000 0000 F70F 8000 0000 0000 0...............
         + +                  + +  +
```

+) Codierung der restlichen SYSTEM-Optionen
(Bit gesetzt = "J", Bit nicht gesetzt = "N")

Byte	Bit: 7	6	5	4	3	2	1	0
F0	AA	AB	AG	AI			AR	AS
F1	AT	BC	BE	BK				
F8	AC	AD	AE	AF		AQ	AJ	AU
F9	AY	AZ	BA		BD	BF	BG	BH
FA	BN							

*)
TI-Codierung:

```
Bit:        7 6 5 4 3 2 1 0
Byte 1      H G F E D C B A
Byte 2      P O N M L K J I
```

TD-Codierung:

```
A     0 0 0 0       SD,SS,5 Zoll
B     0 0 0 1       SD,SS,8 Zoll
C     0 0 1 0       SD,DS,5 Zoll
D     0 0 1 1       SD,DS,8 Zoll
E     0 1 0 0       DD,SS,5 Zoll
F     0 1 0 1       DD,SS,8 Zoll
G     0 1 1 0       DD,DS,5 Zoll
H     0 1 1 1       DD,DS,8 Zoll

        ! ! !5/8-Zoll-Flag
        ! !Double/Single-Sided-Flag
        !Double/Single-Density-Flag
```

Info-Bytes (1) und (2):

```
(1)
Bit 0     SWZ
    1     SWZ
    2   !
    3   !  FDC-Informationen
    4   !
    5     TI-"M"
    6     TI-"K"
    7     TI-"H"

(2)
Bit 0     Density (1-DD)
    1     Beginn Spur-#
    2     Doppelter Step
    3
    4     Beginn Sek-#
    5
    6     Seite (1-DS)
    7     Größe (1-8 Zoll)
```

Bild 5: Abspeicherung der SYSTEM- und PDRIVE-Parameter

Adresse	Belegung
4300...4303	Memory-Map für Track-Register Laufwerk 0...3
4308	Memory-Map für Laufwerksnummer
4309	Memory-Map für Drive-Select
430A...4311	Erste acht Bytes der aktuellen PD-Parameter
4312...4314	BREAK-Vektor
4317	Letzter Request-Code
4318...4367	DOS-Eingabepuffer
4368	"System korrekt initialisiert"-Flag (A5)
4369...436D	FLAG0...FLAG4
4370	System-Option AX
4371...437A	PD-Parameter für Laufwerk 0 (Byte 1...10)
437B...4384	PD-Parameter für Laufwerk 1 (Byte 1...10)
4385...438E	PD-Parameter für Laufwerk 2 (Byte 1...10)
438F...4398	PD-Parameter für Laufwerk 3 (Byte 1...10)
4399...439A	Zeiger auf aktuellen PD-Parameter-Satz
439B...439E	Memory-Map für System-SP
439F	System-Option AL
43A0	System-Option AN
43A1	System-Option AO
43A2	System-Option BJ
43A5	Zeichenpuffer für /JOB-Dateien
43A7...43A8	Puffer für R-Befehl
43A9...43AA	System-Option AP
43AC...43AE	Uhrzeit (s,m,h)
43AF...43B1	Datum (J,M,T)
43B2...43CD	Routing-DCB-Tabelle
43CE...43DF	Overlay-Loader-FCB
43E0...43FF	/JOB-Datei-FCB

FLAG0:
- Bit 5 gesetzt = Chainig aktiv
- Bit 6 gesetzt = Overlay aktiv
- Bit 7 gesetzt = DEBUG aktiv

FLAG1:
- Bit 4 gesetzt = /JOB-Datei offen
- Bit 5,7 gesetzt = Mini-DOS
- Bit 5 gesetzt = DOS

FLAG2: ????

FLAG3:

Bit	S-Option	Bedeutung (Bit gesetzt)
0	AS	Kleinschrift verboten
1	AR	COPY ohne Kennworte
4	AI	Kleinschrift erlaubt
5	AG	BREAK-Taste aktiv
6	AB	Auto-Start
7	AA	Kennworte aktiv

FLAG4:

Bit	S-Option	Bedeutung (Bit gesetzt)
4	BK	Befehl AIK erlaubt
5	BE	Befehl R erlaubt
6	BC	JOB-Unterbrechung erlaubt
7	AT	JOB-"INKEY" erlaubt

Bild 6: Belegung der System-Zero-Page; in diese werden bei der A-Option des PDRIVE-Befehls die Laufwerksparameter kopiert.