Datei: ‚API-B-DE.DOK‘ (59 Seiten) Version: 7. Juli, Jahr des CPC 2023

Datei zur Dokumentation aller Future Operating System OS Funktionen des B - ROMs Version 0.8 (Floppy und Hard-Disk ROM). Sämtliche für den Anwender oder Programmierer wichtige OS Funktionen werden erklärt und lokalisiert.

Alle OS Funktionen werden in folgender Form beschrieben:

1. Kurzbeschreibung: In einem Satz wird kurz die Funktion der OS Funktion beschrieben.

2. Label: Mit diesem Label wird die OS Funktion im Source Code bezeichnet. In der mitgelieferten Label-Bibliothek (siehe Datei #EQU-API.DEU) mit den jehweils aktuellen ROM Adressen findet ebenfalls dieses Label Verwendung. Aus Gründen der Kompatibilität sollte man in eigenen Programmen alle OS- / System-Funktionen (und System-Variablen) stets mit diesen Labels ansprechen.

3. ROM-Nummer: Hier ist die logische ROM - Nummer des FutureOS ROMs angegeben, in dem die entsprechende OS Funktion zu finden ist. Manche OS Funktionen kommen, wegen ihrer Kürze, in mehreren ROMs vor, dann sind hier auch mehrere ROM Nummern angegeben. Da die ROM Nummer logisch zu verstehen ist, sollte man sie nicht mit dem physikalischen ROM Select gleichsetzen. Wie man die physikalische ROM Nummer eines der FutureOS ROMs ermittelt wird im Handbuch erklärt.

4. Startadresse: Gibt die Einsprung-Adresse der OS Funktion an. Falls diese in mehreren ROMs vorkommt, wird hier auch eine entsprechende Anzahl von Adressen angegeben.

5. Einsprungbedingungen: Die Einsprungbedingungen der OS Funktion werden beschrieben und es wird sowohl auf Registerinhalte als auch auf RAM-Variablen eingegangen.

6. Aussprungbedingungen: Die Aussprungbedingungen der OS-Funktion werden beschrieben und es wird sowohl auf Registerinhalte als auch auf RAM-Variablen eingegangen.

7. Manipuliert: Es werden alle manipulierten oder zerstörten Register und RAM-Variablen wiedergegeben. Manchmal werden auch manipulierte Pheripheriebausteine wiedergegeben.

8. Beschreibung: Es folgt eine vollständige Erklärung der Anwendung und Wirkungsweise der beschriebenen OS Funktion.

9. Bitte Beachten: Es werden einige wichtige Details kurz erklärt. Dies ist besonders wichtig, da alle OS Funktionen kompromißlos auf Höchstgeschwindigkeit getrimmt worden sind. Bei falscher Handhabung kann es zu Problemen mit der Systemstabilität kommen.

Im ROM B sind alle OS Funktionen enthalten, die zum Betreib von Floppy Laufwerken am CPC benötigt werden (vor allem low level Funktionen). Dabei werden der interne (Amstrad) und der externe (Vortex) FDC765 unterstützt. Es sind alle Formate von SS 40 Spuren bis DS 80 Spuren möglich. Das HD Format ist noch nicht implementiert. Außerdem sind im ROM B alle grundlegenden OS Funktionen zur Verwaltung der Dobbertin HD20 Festplatte enthalten.

Die aktuelle Version dieser Datei API-B-DE.DOK ist via Email bei FutureSoft@gmx.de oder direkt im Internet erhältlich unter...

FutureOS Home-Page..: http://www.FutureOS.de

SEKTOR ID VON DISKETTE LESEN

Kurzbeschreibung: Eine beliebige Sekror - ID wird von der gerade eingelegten Diskette vom aktuelle Track gelesen.

Label: HOLE0ID (interner FDC) oder HOLE1ID (externer FDC)

ROM-Nummer: B

Startadresse: &C036 (HOLE0ID) bzw. &C03B (HOLE1ID)

Einsprungbedingungen: D = Laufwerksnummer 0..3(Bits 1,0) & Kopf(Bit 2)

Aussprungbedingungen: 7 Bytes ab FDC_RES (&B840) im System-RAM
0. Statusregister 0 des FDC
1. Statusregister 1 des FDC
2. Statusregister 2 des FDC
3. Aktuelle Spurnummer des Kopfes 0..39..79..xx
4. Aktuelle Kopfnummer 0 oder 1 (zwei Köpfe)
5. Aktuelle Sektornummer, WICHTIG zum Formattest
6. Aktuelle Sektorgröße 0,1,2..xx

Manipuliert: AF, BC, HL und FDC

Beschreibung: Lesen einer Sektor-ID. Diese OS Funktion ließt die Sektor-ID des nächsten verfügbaren Sektors, von der gerade eingelegten Diskette, vom aktuellen Track. Im D Register wird das gewünschte Laufwerk (D = DRIVE) in den Bits 0 und 1 übergeben. 00 = Laufwerk A, 01 = Laufwerk B usw. Weiterhin ist es möglich im Bit 2 die Kopfnummer anzugeben. Bei Laufwerken mit nur einem Kopf sollte man immer 0 verwenden. Es ist möglich den internen FDC oder den externen FDC anzusprechen. HOLE0ID spricht den internen FDC an, der auch das 3 Zoll Laufwerk steuert, den externen FDC spricht man mit HOLE1ID an, dieser ist z.B.: für die Vortex F1-D Floppy verantwortlich.

Die OS Funktion liest sieben Bytes ab FDC_RES (&B840) ein, diese sind (s.o.) wie eine ganz normale Result Phase aufgebaut. Diese OS Funktion dient z.B. dazu um das Format einer eingelegten aber unbekannten Diskette zu ermitteln.

Bitte Beachten: Ist keine Diskette eingelegt, so wird dies zu korrupten Daten führen, und die Systemstabilität gefährden. Also zuvor testen ob Disk eingelegt.

Physikalisch nicht vorhandene Laufwerke keinesfalls ansprechen, dies kann bei ungenügender Dekodierung ein anderes Laufwerk beeinflussen.

ANFAHREN EINER BESTIMMTEN SPUR (SEEK)

Kurzbeschreibung: Diese OS Funktion bewegt den Kopf eines bestimmten Laufwerks über eine bestimmte Spur.

Label: SEEK0 (interner FDC), SEEK1 (externer FDC)

ROM-Nummer: B

Startadresse: &C079 (SEEK0) oder &C07E (SEEK1)

Einsprungbedingungen: D = Laufwerk 0..3 // E = neue Spurnummer Dem FDC sollte zuvor die korrekte Step-Rate mitgeteilt werden.

Aussprungbedingungen: Der Kopf der selektierten Floppy ist über die gewünschte Spur bewegt worden.
A = Status Register 0 des FDC.
L = aktuelle Spurnummer, über der sich der Kopf nun befindet.

Manipuliert: AF, BC, L, AF' und der selektierte FDC.

Beschreibung: Mit der OS Funktion SEEK0 wird der Schreib-Lese-Kopf des durch D bestimmten Laufwerks des internen FDCs über die durch E bestimmte Spur bewegt. Will man den externen FDC verwenden so benutzt man einfach SEEK1 anstatt von seek0. Hat man es sehr eilig so sollte man SINI0 und STER0 verwenden. Ein Spurwechselkommande dauert u.U. mehrere Millisekunden.

Bitte Beachten: Laufwerk muss bereit, Diskette eingelegt sein, die korrekte Step-Rate-Time = Spurwechselzeit muss an den FDC geschickt werden. Für FDC 0 und FDC 1 existieren auch hier wieder zwei eigene OS Funktionen.

KOMMANDO 'SPUR ANFAHREN (SEEK)' ABSCHICKEN

Kurzbeschreibung: Das Kommando 'Spur anfahren' wird abgeschickt, und sofort zurückgekehrt. Diese Funktion kann mit Hilfe von STER0/1 die OS Funktion SEEK0/1 ersetzen.

Label: SINI0 (interner FDC) oder SINI1 (externer FDC)

ROM-Nummer: B

Startadresse: &C0C5 (SINI0) bzw. &C0CA (SINI1)

Einsprungbedingungen:
D = Laufwerk, dessen Kopf bewegt werden soll.
E = Spurnummer, die der Kopf anfahren soll.
Dem FDC muss die korrekte Spurwechselzeit bekannt sein.

Aussprungbedingungen: Der selbe Kopf des selben Laufwerks ist zur selben Spur unterwegs.

Manipuliert: AF, BC und der FDC

Beschreibung: Diese OS Funktion ähnelt dem SEEK Kommando (s.o.). Allerdings wird diesmal der Kopf nur auf den Weg geschickt, und nicht auf dessen Ankunft gewartet. Anschließend kann der entsprechende FDC kein weiteres Kommando entgegennehmen, abgesehen von weiteren SEEK Kommandos, die ANDERE Laufwerke betreffen.

Um den Spurwechsel korrekt zu beenden MUSS man im Anschluss an diese OS Funktion ein andere OS Funktion aufrufen, diese heisst STER0 bzw. STER1. Zwischen den Aufrufen der beiden OS Funktionen können andere Aufgaben erledigt werden.

Wie immer hat jeder FDC seine eigene OS Funktion. SINI0 spricht den internen FDC an, und SINI1 den Externen.

Bitte Beachten: Im Anschluß muss irgendwann STER0/1 aufgerufen werden, sonst kann die Arbeit mit dem sel. FDC nicht fortgesetzt werden.

Ansonsten siehe SEEK0/1

BEENDEN DES KOMMANDOS 'SPUR ANFAHREN (SEEK)'

Kurzbeschreibung: OS Funktion wartet bis alle Köpfe des entsprechenden FDC ihre Spur erreicht haben.

Label: STER0 (interner FDC) oder STER1 (externer FDC)

ROM-Nummer: B

Startadresse: &C0EE (STER0) bzw. &C0F3 (STER1)

Einsprungbedingungen: Es muss irgendein Kopf auf die Wanderschaft geschickt worden sein, ob mit SINI0/1 oder wie auch immer.
Auswahl des FDC wie immer über das Label.

Aussprungbedingungen: L = neue Spur nach SEEK.
Alle Köpfe des akt. FDC haben ihre Zielspur erreicht.

Manipuliert: AF, BC, HL, AF' und der FDC

Beschreibung: Diese OS Funktion dient dazu ein vorrangegangenes SINI0 oder SINI1 Kommando zu beenden. Bitte entsprechend STER0 oder STER1 verwenden. Nach Rückkehre können an den akt. FDC wieder alle Befehle geschickt werden, oder aller Floppy OS Funktionen benutzt werden. Die Zeit zwischen dem Aufruf von SINI0/1 und STER0/1 steht dem Anwender zur freien Verfügung. Beite OS Funktionen-Teile stellen einen vollwertigen Ersatz der OS Funktion SEEK0/1 dar.

Bitte Beachten: Falls bei Aufruf der OS Funktion gar kein Kopf unterwegs war, so wird das System INAKTIV (Absturtz!)

STATUS REGISTER 3 DES FDC EINLESEN

Kurzbeschreibung: Das Status Register 3 des FDC wird eingelesen, dies ist nur durch diesen Befehlsablauf möglich.

Label: HOLE_S3 (interner FDC) oder HOL1_S3 (externer FDC)

ROM-Nummer: B

Startadresse: &C0F8 (HOLE_S3) bzw. &C0FD (HOL1_S3)

Einsprungbedingungen: D = Laufwerksnummer

Aussprungbedingungen: A = Status Register 3 int./ext. FDC

Manipuliert: AF, BC

Beschreibung: Will man das Status Register 3 des FDC auslesen, so muß man dafür ein extra Kommande an den FDC schicken. Dafür dient diese OS Funktion. HOLE_S3 ließt beim internen FDC das Status Reg. 3 und HOL1_S3 beim externen FDC. Für jedes Laufwerk kann dieser Status 3 extra ermittelt werden, deshalb ist es nötig beim Einsprung die Laufwerksnummer von 0 bis 3 im D Register anzugeben. Wie immer wird im 3. Bit (dem Bit 2) die Kopfnummer angegeben.

Die OS Funktion liefert dann den Inhalt des Status Register 3 im A Register zurück. Dabei haben die Bits folgende Bedeutung:
UNIT SELECT Bit 0 und 1: geben das gewählte Laufwerk wieder.
HEAD ADRESS Bit 2: gibt den gewählten Kopf des LWs wieder.
TWO SIDE Bit 3: Einseitige Laufwerke legen dieses Bit normlerweise auf 1, und zweiseitige Laufwerke auf 0, beim 3 Zoll LW des CPC ist diese Bit unbestimmt. Man sollte dieses Bit NICHT benutzen.
TRACK 0 Bit 4: Dieses Bit wird auf 1 gesetzt, wenn der Kopf des LW gerade über der Spur 0 steht, sonst ist es auf 0 gesetzt.
READY Bit 5: Ist dieses Bit auf 1 gesetzt, dann meldet das LW daß es READY ist, d.h. Diskette ist eingelegt und der Motor läuft mit korrekter Drehzahl. Das Laufwerk kann also benutzt werden. Ist irgend-eine Bedingung nicht erfüllt, dann ist das Bit gelöscht, das LW kann nicht benutzt werden.
WRITE PROTECT Bit 6: Ist dieses Bit gesetzt, dann ist die Diskette schreibgeschützt. Ist das Bit dagegen gelöscht, dann kann auf das LW geschrieben werden.
FAULT Bit 7: Dieses Bit wird bei einem Fehler im Laufwerk auf 1 gesetzt. Ist dies Bit gelöscht, dann meldet das Laufwerk auch keinen Fehler. Achtung nicht alle Laufwerke haben eine FAULT Leitung.

Bitte Beachten: Die Laufwerksmotoren müssen nicht aktiviert sein. Auch bei ausgeschaltetem Motor kann so z.B. abgefragt werden ob eine Disk eingelegt ist oder nicht, vorrausgesetzt daß die Disk nicht schreibgeschützt ist (denn dieses Bit wird getestet).

ZUGRIFFSZEITEN FüR DIE LAUFWERKE ANGEBEN

Kurzbeschreibung: Mit diesem Kommando werden die Zugriffszeiten für alle Laufwerke eines FDC angegeben.

Label: ZEIT0 (interner FDC) oder ZEIT1 (externer FDC)

ROM-Nummer: B

Startadresse: &C12F (ZEIT0) bzw. &C134 (ZEIT1)

Einsprungbedingungen: D = Steprate (in den oberen 4 Bits) = &X0
Die unteren 4 Bit von D geben die Kopfladezeit an. Da diese nur bei 8 Zoll LWs wichtig ist, sollte Sie auf 0 gesetzt werden.
Siehe Tabelle der Spurwechselzeiten.

Aussprungbedingungen: Laufwerk ist mit einer Kopfentladezeit von 4 ms versehen worden. (kleiner gehts nicht, ist beim CPC unnötig). Der FDC arbeitet im Nicht-DMA Modus.
Die Spurwechselzeit (und die Kopfladezeit) aus Register D sind nun für alle Laufwerke des FDC gültig.

Manipuliert: AF, BC und Zugriffszeiten aller LWs eines FDC

Beschreibung: Vor jedem Zugriff auf ein spezielles Laufwerk muß dem FDC mitgeteilt werden welche Zugriffszeiten dieses hat. Dazu wird im D Register die Steprate (obere 4 Bit) und die Kopfladezeit/HEAD LOAD TIME (untere 4 Bit) angegeben. Da die Kopfladezeit nur bei 8 Zoll Laufwerken von Bedeutung ist, sollte man die unteren 4 Bit des D Registers auf 0 setzen.

Wird die Spurwechselzeit (Steprate) zu klein gewählt, dann kann das Laufwerk den Kopf nicht schnell genug bewegen. ==> Fehlfunktion. Wird die Steprate zu groß gwewählt, dann wird das Laufwerk unnötig aufgehalten. Beide Fehler sind durchaus kritisch.

Bitte Beachten: Für ALLE vier Laufwerke die an einem FDC angeschlossen sind existieren die Register der Zugriffszeiten nur einmal. Es ist also unerläßlich vor jedem Zugriff auf ein Laufwerk die entsprechende Steprate an den FDC zu senden. 3 Zoll und 5.25 Zoll Laufwerke unterscheiden sich teils erheblich in ihrer Steprate, und damit in ihrer Geschwindigkeit.

Tabelle der Werte für D und der zugehörigen Spurwechselzeiten:

D = &00 ==> 32 ms Spurwechselzeit	//	D = &10 ==> 30 ms Spurwechselzeit
D = &20 ==> 28 ms Spurwechselzeit	//	D = &30 ==> 26 ms Spurwechselzeit
D = &40 ==> 24 ms Spurwechselzeit	//	D = &50 ==> 22 ms Spurwechselzeit
D = &60 ==> 20 ms Spurwechselzeit	//	D = &70 ==> 18 ms Spurwechselzeit
D = &80 ==> 16 ms Spurwechselzeit	//	D = &90 ==> 14 ms Spurwechselzeit
D = &A0 ==> 12 ms Spurwechselzeit	//	D = &B0 ==> 10 ms Spurwechselzeit
D = &C0 ==> 8 ms Spurwechselzeit	//	D = &D0 ==> 6 ms Spurwechselzeit
D = &E0 ==> 4 ms Spurwechselzeit	//	D = &F0 ==> 2 ms Spurwechselzeit

Die Spurwechselzeit (Steprate) ist für ALLE Laufwerke eines FDC gültig.

STANDARTBEFEHLSPHASE EINLEITEN - LESEN, SCHREIBEN, VERFIZIEREN

Kurzbeschreibung: Diese OS Funktionen erlauben es die Befehlsphase des FDC einzuleiten, d.h. ein Kommando abzuschicken. Man kann Sektor(en) oder Spuren lesen, schreiben bzw. verifizieren. Fehler sind erkennbar.

Labels: LSV0 (interner FDC) oder LSV1 (externer FDC / Vortex).
LSV0X / LSV1X entsprechen LSV0 / LSV1 mit mehreren L-S-V-Versuchen.
LSV0XE / LSV1XE entsprechen LSV0X / LSV1X mit Fehlerbehandler.

ROM-Nummer: B

Startadresse: &C15A (LSV0) bzw. &C15F (LSV1)
&E6B2 (LSV0X) bzw. &E6BA (LSV1X)
&E709 (LSV0XE) bzw. &E717 (LSV1XE)

Einsprungbedingungen:
D = Laufwerk (Bits 1,0) & Kopf (Bit 2) mit dem gearbeitet werden soll.
E = Spurnummer &00..&28..&50 (&FF), muss schon angefahren sein.
H = erster Sektor, der bearbeitet werden soll.
L = Sektorgröße: 2 = 512 Byte, 3 = 1024 Byte.
Zweitregister:
A' = Kommandokode für den FDC, gibt an, was der FDC machen soll.
H' = letzter Sektor, der bearbeitet werden soll.
L' = GAP#3, Lücke zwischen ID und Daten.
DE'= Quell- oder Ziel- Adresse des Datenblocks, den es zu bearbeiten gilt, z.B. Sektor(en) oder Spur.
Das Laufwerk muß READY melden, die Spur muß angefahren sein.
LSV0X(E) / LSV1X(E): RAM Variable FDCLSV enthält Anzahl der Lese/Schreib-Versuche.
LSV0XE / LSV1XE: Variablen FDC_ERR und FDE_RAM zeigen auf Fehler-Behandler.

Aussprungbedingungen: DE = Zeiger auf Byte nach dem Datenblock, der bearbeitet wurde. FDC_RES und noch 6 Bytes enthalten die Status- Informationen die der FDC geliefert hat. Bitte diese 7 Bytes nachschlagen.

Manipuliert: AF, BC, D und AF', BC', HL' dann EXX! Interrupts AUS!
LSV0X(E) / LSV1X(E): außerdem FDCLSA, E, HL, DE'

Beschreibung: Diese OS Funktion leitet die Befehlsphase eines jeden Datentransfers mit dem FDC ein. Es kann gelesen oder geschrieben werden. Auch kann man die Daten mit dem Verify Kommando testen. Da man immer einen ERSTEN und einen LETZTEN Sektor angibt, ist es nicht nur möglich einzelne Sektoren zu bearbeiten. Es ist auch möglich mehrere aufeinanderfolgende Sektoren oder eine ganze Spur zu bearbeiten.

Was genau der FDC nun tut hängt vom Kommando ab, daß im A' Register übergeben wird.

Vor Aufruf dieser OS Funktion muß die korrekte Spur angefahren werden, und dieses SEEK Kommande ORDNUNGSGEMäß abgeschlossen werden. Das Laufwerk muß bereit sein, d.h. im Status Register 3 muß das READY Bit gesetzt sein. Bei LSV0X(E) bzw. LSV1X(E) befindet

sich in FDCLSV die Anzahl der Versuche. FDCLSV gibt an wie oft ein Kommando wiederholt werden soll, wenn der FDC einen Fehler meldet. Bei LSV0XE / LSV1XE wird nach erfolgloser Abarbeitung aller Versuche ein FehlerBehandler aufgerufen, dieser wird durch FDC_ERR & FDE_RAM spezifiziert.

Bei Aufruf wird nun der Datenblock transferiert. Ab FDC_RES stehen 7 Bytes im RAM die über den Erfolg der Aktion Auskunft geben. Das DE Register zeigt auf das Byte nach dem bearbeiteten Datenblock.

Bitte Beachten: Wurde ein SEEK nicht abgeschlossen, d.h. ist im Hauptstatusregister des FDC noch eins der unteren 4 Bits gesetzt, dann wird das System abstürzen! Jedes Spurwechselkommando muß ordnungsgemäß beendet worden sein, z.B. mit einen STER falls NÖTIG!

EINE SPUR IM FUTURE-OS SPEZIALFORMAT FORMATIEREN

Kurzbeschreibung: Eine gerade angefahrene Spur auf der Diskette wird im Future-OS Spezialformat formatiert.

Label: FOR0F (interner FDC) oder FOR1F (externer FDC)

ROM-Nummer: B

Startadresse: &C214 (FOR0F) oder &C219 (FOR1F)

Einsprungbedingungen:
D = Kopf(Bit 2), LW-Nr.(0..3)
E = Spur-Nummer (für Sektor ID).
Die zu formatierende Spur muss angefahren sein. Das Laufwerk muss 'DRIVE READY' melden.

Aussprungbedingungen: Spur wurde formatiert.

Manipuliert: AF, BC, D und L

Beschreibung: Diese OS Funktion dient dazu um eine Spur einer Diskette im FutureOS Spezialformat zu formatieren. Es werden fünf Sektoren mit 1024 Bytes geschrieben. Folgendes Sektormuster kommt zum Einsatz: &80, &81, &82, &83, &84. Die Spur hat also Platz für 5 KB Daten.

Bei Aufruf der OS Funktion muss die korrekte Spur bereits angefahren sein. Der Laufwerksmotor muss laufen, und das Laufwerk muss bereit sein (DRIVE READY).

Die in E übergebene Spurnummer muss nicht mit der physikalischen Spurnummer übereinstimmen, wird aber beim Lesen oder Schreiben eines Sektors vom FDC verlangt.

Bitte Beachten: Spur muss angefahren sein, LW muss bereit sein. Dieses Format kann Ein- und Zwei- seitig verwendet werden.

EINE SPUR IM VORTEX FORMAT FORMATIEREN

Kurzbeschreibung: Eine gerade angefahrene Spur auf der Diskette wird mit Vortex Format formatiert.

Label: FOR0V (interner FDC) oder FOR1V (externer FDC)

ROM-Nummer: B

Startadresse: &C295 (FOR0V) oder &C29A (FOR1V)

Einsprungbedingungen:
D = Kopf (Bit 2), LW-Nr. (0..3).
E = Spur-Nummer (für Sektor ID).
Die zu formatierende Spur muss angefahren sein.

Aussprungbedingungen: Spur wurde formatiert.

Manipuliert: AF, BC, D ... L

Beschreibung: Eine bereits mittels SEEK angefahrene Spur wird mit neun Sektoren mit je 512 Bytes formatiert. Folgendes Sektormuster wird verwendet: 1,6,2,7,3,8,4,9,5. Die Spur hat also für 4.5 KB Daten Platz.

Das Vortex Format ist ein zweiseitiges Format das auf 80-spurigen Laufwerken Verwendung findet.

Bitte Beachten: Die korrekte Spur muss angefahren sein. Das Laufwerk muss DRIVE READY melden.

EINE SPUR IM DATA FORMAT FORMATIEREN

Kurzbeschreibung: Eine gerade angefahrene Spur auf der Diskette wird mit Data Format formatiert.

Label: FOR0D (interner FDC) oder FOR1D (externer FDC)

ROM-Nummer: B

Startadresse: &C315 (FOR0D) oder &C31A (FOR1D)

Einsprungbedingungen:
D = Kopf(Bit 2), LW-Nr.(0..3) // E = Spur-Nummer (für Sektor ID).
Die zu formatierende Spur muss angefahren sein.

Aussprungbedingungen: Die zuvor angefahrene Spur wurde im Data Format formatiert.

Manipuliert: AF, BC, D ... L

Beschreibung: Eine bereits mittels SEEK angefahrene Spur wird mit neun Sektoren mit je 512 Bytes formatiert. Folgendes Sektormuster wird verwendet: &C1, &C6, &C2, &C7, &C3, &C8, &C4, &C9, &C5. Die Spur hat also für 4.5 KB Daten Platz.

Das DATA Format wird normalerweise auf einseitigen 40 Spur Laufwerken verwendet.

Bitte Beachten: Vor Aufruf der OS Funktion muss das Laufwerk bereit sein d.h. Motor läuft, Diskette eingelegt. Ausserdem muss die korrekte Spur bereits angefahren sein.

EINE SPUR IM SYSTEM FORMAT FORMATIEREN

Kurzbeschreibung: Eine gerade angefahrene Spur auf der Diskette wird im System Format formatiert.

Label: FOR0S (interner FDC) oder FOR1S (externer FDC)

ROM-Nummer: B

Startadresse: &C395 (FOR0S) oder &C39A (FOR1S)

Einsprungbedingungen:
D = Kopf(Bit 2), LW-Nr.(0..3)
E = Spur-Nummer (für Sektor ID).
Die zu formatierende Spur muss angefahren sein und das Laufwerk muss 'DRIVE READY' melden.

Aussprungbedingungen: Die aktuelle Spur ist im System Format formatiert worden.

Manipuliert: AF, BC, D ... L

Beschreibung: Eine bereits mittels SEEK angefahrene Spur wird mit neun Sektoren mit je 512 Bytes formatiert. Folgendes Sektormuster wird verwendet: &41, &46, &42, &47, &43, &48, &44, &49, &45. Die Spur hat also für 4.5 KB Daten Platz.

Das SYSTEM Format wird normalerweise auf einseitigen 40 Spur Laufwerken verwendet.

Bitte Beachten: Vor Aufruf der OS Funktion muss das Laufwerk bereit sein d.h. Motor läuft, Diskette eingelegt. Ausserdem muss die korrekte Spur bereits angefahren sein.

EINE SPUR IM IBM FORMAT FORMATIEREN

Kurzbeschreibung: Eine gerade angefahrene Spur wird IBM formatiert.

Label: FOR0I (interner FDC) oder FOR1I (externer FDC)

ROM-Nummer: B

Startadresse: &C415 (FOR0I) oder &C41A (FOR1I)

Einsprungbedingungen:
D = Kopf(Bit 2), LW-Nr.(0..3) /// E = Spur-Nummer (für Sektor ID).
Die Spur muss angefahren und das Laufwerk muss 'DRIVE READY' sein.

Aussprungbedingungen: Die aktuelle Spur ist IBM formatiert worden.

Manipuliert: AF, BC, D und L

Beschreibung: Eine bereits mittels SEEK angefahrene Spur wird mit acht Sektoren mit je 512 Bytes formatiert. Folgendes Sektormuster wird verwendet: &01, &05, &02, &06, &03, &07, &04, &08. Die Spur hat also für 4 KB Daten Platz.

Das IBM Format wird normalerweise auf einseitigen 40 Spur Laufwerken verwendet.

Bitte Beachten: Vor Aufruf der OS Funktion muss das Laufwerk bereit sein d.h. Motor läuft, Diskette eingelegt. Ausserdem muss die korrekte Spur bereits angefahren sein.

EINE SPUR LESEN ODER SCHREIBEN FDC 0/1

Kurzbeschreibung: Eine Spur lesen oder schreiben egal welches Laufwerk, Formate: DATA, IBM, VORTEX oder SYSTEM.

Labels:

S0SR (System schreiben)	///	L0SR (System lesen)	- jehweils FDC 0
S0VR (Vortex schreiben)	///	L0VR (Vortex lesen)	- jehweils FDC 0
S0DR (Data schreiben)	///	L0DR (Data lesen)	- jehweils FDC 0
S0IR (IBM schreiben)	///	L0IR (IBM lesen)	- jehweils FDC 0
S1SR (System schreiben)	///	L1SR (System lesen)	- jehweils FDC 1
S1VR (Vortex schreiben)	///	L1VR (Vortex lesen)	- jehweils FDC 1
S1DR (Data schreiben)	///	L1DR (Data lesen)	- jehweils FDC 1
S1IR (IBM schreiben)	///	L1IR (IBM lesen)	- jehweils FDC 1

ROM-Nummer: B

Startadressen:

(S0SR) &C813 // (L0SR) &C819 // (S1SR) &C9CF // (L1SR) &C9D5
(S0VR) &C81F // (L0VR) &C825 // (S1VR) &C9DB // (L1VR) &C9E1
(S0DR) &C82B // (L0DR) &C831 // (S1DR) &C9E7 // (L1DR) &C9ED
(S0IR) &CB8B // (L0IR) &CB91 // (S1IR) &CD02 // (L1IR) &CD08

Einsprungbedingungen:
DE = Adresse des 4 KB bzw 4.5 KB Datenblocks
REG08_0 = Spur (muss schon angefahren sein)
REG08_1 = Laufwerk 0..3 (Bits 0,1), Kopf (Bit 2)

Aussprungbedingungen: DE = Byte nach dem transferierten Datenblock

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL und REG_PC low

Beschreibung: Dieser Programmkomplex wird dazu verwendet um eine Spur von Diskette zu lesen oder auf sie zu schreiben. Es kann der interne oder der externe Floppy-Kontroller verwendet werden. Man kann DATA-, IBM-, VORTEX- oder SYSTEM- Format verwenden.
Jeh nachdem welches der acht Laufwerke, welches der vier Formate oder welche Datenrichtung man verwenden will, springt man zum entsprechenden Label. Es existieren 16 Labels (8 LW * 4 Fm * 2 DR = 16).
Nachdem der Datenblock transferiert wurde steht DE auf dem darauf folgenden Byte, so dass man nach dem Anfahren einer neuen Spur sofort mit den Daten weiterarbeiten kann. Diese OS Funktion arbeitet mit Interleave "0", d.h. eine Spur wird in nur einer Umdrehung der Disk komplett gelesen, obwohl einige Formate mit einem Interleave formatiert sind.

Bitte Beachten: Die Spur die man bearbeiten will muss bereits angefahren sein. Das entsprechende Laufwerk muss zum Datentransfer bereit sein, d.h. DRIVE READY melden.

Wie bei allen Diskettenoperationen dürfen wärend der Ausführung keine Interrupts auftreten.

LADEN IN HAUPTSPEICHER MAXIMAL 64 KB SYSTEM, DATA

Kurzbeschreibung: Eine Datei wird direkt in die aktuellen 64K geladen.

Label: L0DS (interner FDC) bzw. L1DS (externer FDC)

ROM-Nummer: B

Startadresse: &CE79 (L0DS) oder &CF97 (L1DS)

Einsprungbedingungen:
DE = Zieladresse der zu ladenden Datei bzw.Daten
HL = Zeiger auf Anfang Track, Sektor Tabelle
REG08_0 = Spur die ersten Sektor enthält
REG08_1 = Kopf, Laufwerk von dem geladen wird
SINI0/1 muss abgeschickt sein und Ladetabelle muss generiert sein.

Aussprungbedingungen: Es wurde geladen, wenn Laufwerk zuvor READY war.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen REG08_0, REG_PC(low), FDC_RES und der FDC. Außerdem u.U. der Headerbereich für Vordergrund-Programme (BC00 bis BC7F).

Beschreibung: Diese OS Funktion dient dazu um eine Datei oder bestimmte Tracks, Sektoren an eine beliebige Adresse im Speicher zu laden. Es kann jedes der acht Laufwerke verwendet werden, aber nur System oder Data Format. Andere Formate siehe unten. Man sollte keine Daten oberhalb von &A000 laden, da sonst wichtige Systemvariablen überschrieben werden. Die Adresse, ab der geladen werden soll gibt man in DE an. In HL wird die Adresse der Ladetabelle angegeben.

Aufbau der Tabelle bei System oder Data Format:

Die Tabelle beginnt mit einer Tracknummer, dieser folgt die Anzahl der Sektoren, die von diesem Track zu laden sind. Der Sektoranzahl folgen die einzelnen Sektornummern.

Dannach folgt die nächste Tracknummer usw.

- Wenn die Sektoranzahl = &FF ist, dann ist der entsprechende Track komplett zu laden. In diesem Fall folgt auf die Tracknummer nur das Byte &FF, aber KEINE Sektornummern. Nach diesem &FF folgt direkt die nächste Tracknummer.

- Ist die Tracknummer = &FF, dann ist die Tabelle zuende. Das Tabellenende wird also durch eine Tracknummer von &FF signalisiert.

Bitte Beachten: WICHTIG!! Auf die Spur, die man in REG08_0 angibt, muss vor Aufruf dieser OS Funktion ein SINI0/1 abgegeben werden, dieser wird dann durch diese OS Funktion beendet, und dannach geladen.

LADEN IN HAUPTSPEICHER MAXIMAL 64 KB IBM-FORMAT

Kurzbeschreibung: Eine Datei wird direkt in die aktuellen 64K geladen.

Label: L0IB (interner FDC) bzw. L1IB (externer FDC)

ROM-Nummer: B

Startadresse: &D0B5 (L0IB) oder &D1D3 (L1IB)

Einsprungbedingungen:
DE = Zieladresse der zu ladenden Datei bzw.Daten
HL = Zeiger auf Anfang Track, Sektor Tabelle.
REG08_0 = Spur die ersten Sektor enthält.
REG08_1 = Kopf, Laufwerk von dem geladen wird.
SINI0/1 muss abgeschickt sein und Ladetabelle muss generiert sein.

Aussprungbedingungen: Es wurde geladen, wenn Laufwerk zuvor READY war.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen REG08_0, REG_PC(low), FDC_RES und der FDC. Außerdem u.U. der Headerbereich für Vordergrund-Programme (BC00 bis BC7F).

Beschreibung: Diese OS Funktion dient dazu um eine Datei oder bestimmte Tracks, Sektoren an eine beliebige Adresse im Speicher zu laden. Es kann jedes der acht Laufwerke verwendet werden, aber nur IBM Format. Man sollte keine Daten oberhalb von &A000 laden, da sonst wichtige Systemvariablen überschrieben werden. Die Adresse, ab der geladen werden soll gibt man in DE an. In HL wird die Adresse der Ladetabelle angegeben.

Die Ladetabelle bei IBM Format ist genauso aufgebaut wie bei Data bzw. System Format.

Bitte Beachten: WICHTIG!! Auf die Spur, die man in REG08_0 angibt, muss vor Aufruf dieser OS Funktion ein SINI0/1 abgegeben werden, dieser wird dann durch diese OS Funktion beendet, und dannach geladen.

LADEN IN HAUPTSPEICHER MAXIMAL 64 KB VORTEX

Kurzbeschreibung: Eine Datei wird direkt in die aktuellen 64K geladen.

Label: L0AV (interner FDC) bzw. L1AV (externer FDC)

ROM-Nummer: B

Startadresse: &D2F1 (L0AV) oder &D432 (L1AV)

Einsprungbedingungen:
DE = Zieladresse der zu ladenden Datei bzw.Daten
HL = Zeiger auf Anfang Track, Sektor Tabelle.
REG08_0 = Spur die ersten Sektor enthält.
REG08_1 = Kopf, Laufwerk von dem geladen wird.
SINI0/1 muss abgeschickt sein.
Ladetabelle muss generiert sein.

Aussprungbedingungen: Es wurde geladen, wenn Laufwerk zuvor READY war.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen REG08_0, REG08_1, REG_PC(low), FDC_RES und der FDC. Außerdem u.U. der Headerbereich für Vordergrund-Programme (BC00 bis BC7F).

Beschreibung: Diese OS Funktion dient dazu um eine Datei oder bestimmte Tracks, Sektoren an eine beliebige Adresse im Speicher zu laden. Es kann jedes der acht Laufwerke verwendet werden, aber nur das Vortex Format. Man sollte keine Daten oberhalb von &A000 laden, da sonst wichtige Systemvariablen überschrieben werden. Die Adresse, ab der geladen werden soll gibt man in DE an. In HL wird die Adresse der Ladetabelle angegeben. VORSICHT, die Ladetabelle bei Vortex Format ist anders aufgebaut, als die des System oder Data Formats (Kopfbit in Track-Byte).

Aufbau der Tabelle bei Vortex Format:
Die Tabelle zu Vortex Format hat an sich den selben Aufbau wie Die zum laden von System oder Data Format.
Der Unterschied liegt darin, daß beim Vortex Format zu jeder Spur auch immer eine Kopfnummer angegeben werden muss.
Die Kopfnummer ist im Bit 0 des jehweiligen Trackbytes enthalten.
Rechnerisch wird dazu die Tracknummer (0..80) mit 2 multipliziert und das Kopfbit (0 = oben, 1 = unten) addiert. Für die Ober- oder Unter- Seite einer Spur sind also zwei Trackbytes enthalten.
Die physikalische Tracknummer kann durch einfaches rechtsschieben mit z.B. SRL A errechnet werden. Das Carry Bit hat dann die Kopfnummer.

Bitte Beachten: WICHTIG! Auf die Spur, die man in REG08_0 angibt, muss vor Aufruf dieser OS Funktion ein SINI0/1 abgegeben werden, dieser wird dann durch diese OS Funktion beendet, und dannach geladen.

LADEN IN Erweiterungs-Speicher MAXIMAL 512 KB IBM, SYSTEM UND DATA

Kurzbeschreibung: Daten oder eine Datei wird in Erweiterungsspeicher geladen. Dieser kann komplett (512 KB) vollgeschrieben werden.

Label: L016 (interner FDC) bzw. L116 (externer FDC) ab &4000 in 1. E-RAM Block &C4
L015 (interner FDC) bzw. L115 (externer FDC) beliebige Adresse & E-RAM

ROM-Nummer: B

Startadresse: &D594 (L016), &D5A0 (L015) / &D5F3 (L116), &D5FF (L115)

Einsprungbedingungen: Bei allen Einsprüngen gilt:
HL = Adresse der TRK, SEK Tabelle
REG08_1 = Kopf (Bit 2), Laufwerk (Bits 1,0)
REG16_0 = Adresse der Spurlade OS Funktion des gewünschten Formats
- TRACK, SEKTOR Tabelle muss generiert sein.
- SINI0/1 muss abgeschickt sein
Bei den Einsprüngen L015 und L115 gilt außerdem:
DE = Adresse ab der geladen wird, kleiner &8000
AKT_RAM = &7FC4..&7FFF, 1. 16K Block Ziel, dieser Block muss bereits eingeblendet sein.

Aussprungbedingungen: ErweiterungsRAM ist beLADEN falls alles ok war.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, und die RAM-Variablen: AKT_RAM, REG08_0, FDC_RES, der FDC und die RAM-Konfiguration.

Der Speicherbereich von &8000 bis &8FFF ist unter Umständen verändert (Spurpuffer).

Beschreibung: Diese OS Funktion(n) laden von Diskette in das ErweiterungsRAM. Das ErweiterungRAM wird dabei aufsteigend, Block für Block beschrieben. Ist der Speicher komplett ausgebaut, dann können alle 512KB gefüllt werden. (Achtung auf eine Data Disk passen nur 178 KB Daten). Der Erweiterungsspeicher wird 16KB weise je von &4000 bis &7FFF beschrieben.
Es sind IBM, System und Data Format verwendbar. Alle acht Laufwerke sind selektierbar.
Verwendet man die Labels L016 bzw. L116, so wird immer ab dem Block &C4 (unterster Block) und ab Adresse &4000 geladen.
Verwendet man die Labels L015 bzw. L115, dann kann ab einer beliebigen Adresse, in einen beliebigen Block geladen werden. Die Adresse gibt man dann in DE an, sie muss unterhalb von &8000 liegen. Der erste Block wird in die RAM-Variable AKT_RAM geschrieben, und muss von Hand eingeblendet werden. Dies geschieht z.B. folgendermaßen:

```
LD  BC,&7FC6     ;&7F = GATE ARRAY, &C6 = 3. 16K Block (einer aus 32)

OUT (C),C        ;16K Block wird zwischen &4000 - &7FFF eingeblendet.

LD  (AKT_RAM),BC ;aktuelle RAM-Konfiguration ins RAM schreiben.
```

Die 16K Blöcke des ErweiterungsRAMs werden der Reihe nach über folgende Bytes eingeblendet: (logische Reihenfolge 1, 2, 3, ..., 31, 32)

&C4,&C5,&C6,&C7 , &CC,&CD,&CE,&CF , &D4,&D5,&D6,&D7 , &DC,&DD,&DE,&DF,

&E4,&E5,&E6,&E7 , &EC,&ED,&EE,&EF , &F4,&F5,&F6,&F7 , &FC,&FD,&FE,&FF.

Bitte Beachten: Ein SINI0/1 auf die erste Spur, von der geladen werden soll, muss abgeschickt worden sein. Er darf wie immer noch nicht beendet sein. Das LW muss bereit sein.

LADEN IN ERWEITERUNGSSPEICHER MAXIMAL 512KB VORTEX

Kurzbeschreibung: Daten oder eine Datei wird in den Erweiterungsspeicher geladen. Dieser kann komplett (512 KB) vollgeschrieben werden.

Label: L0V6 (interner FDC) bzw. L1V6 (externer FDC) ab &4000 in 1. E-RAM Block &C4
L0V5 (interner FDC) bzw. L1V5 (externer FDC) beliebige Adresse & E-RAM

ROM-Nummer: B

Startadresse: &D652 (L0V6), &D65E (L0V5) // &D6BE (L1V6), &D6CA (L1V5)

Einsprungbedingungen: Bei allen Einsprüngen gilt:
HL = Adresse der TRK, SEK Tabelle
REG08_0 = Spur, auf die ein SINI abgeschickt ist
REG08_1 = Kopf (Bit 2), Laufwerk (Bits 1,0)
- TRK, SEK Tabelle muss generiert sein.
- SINI0/1 muss abgeschickt sein.
Bei den Einsprüngen L0V5 und L1V5 gilt außerdem:
DE = Adresse ab der geladen wird, kleiner &8000
AKT_RAM = &7FC4..&7FFF, 1. 16K Block Ziel, dieser Block muss bereits eingeblendet sein.

Aussprungbedingungen: Erweiterungs-RAM ist beLADEN falls alles ok war.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, und die RAM-Variablen: AKT_RAM, REG08_0, REG08_1, FDC_RES, der FDC und die RAM-Konfiguration.

Der Speicherbereich von &8000 bis &8FFF ist unter Umständen verändert, er dient als Track, Sektor Puffer.

Beschreibung: Diese OS Funktion(n) laden von Diskette in das Erweiterungs-RAM. Das ErweiterungRAM wird dabei aufsteigend, Block für Block beschrieben. Ist der Speicher komplett ausgebaut, dann können alle 512KB gefüllt werden. Der Erweiterungsspeicher wird 16KB weise je von &4000 bis &7FFF beschrieben.

Alle acht Laufwerke sind selektierbar. Verwendet man die Labels L0V6 bzw. L1V6, so wird immer ab dem Block &C4 (unterster Block) und ab Adresse &4000 geladen.

Verwendet man die Labels L0V5 bzw. L1V5, dann kann ab einer beliebigen Adresse, in einen beliebigen Block geladen werden. Die Adresse gibt man dann in DE an, sie muss unterhalb von &8000 liegen. Der erste Block wird in die RAM-Variable AKT_RAM geschrieben, und muss von Hand eingeblendet werden. Dies geschieht z.B. folgendermaßen:

```
LD  BC,&7FC6     ;&7F = GATE ARRAY, &C6 = 3. 16K Block (einer aus 32)

OUT (C),C        ;16K Block wird zwischen &4000 - &7FFF eingeblendet.

LD  (AKT_RAM),BC ;aktuelle RAM-Konfiguration ins RAM schreiben.
```

Die 16K Blöcke des ErweiterungsRAMs werden der Reihe nach über folgende Bytes eingeblendet: (logische Reihenfolge 1, 2, 3, ..., 31, 32)

&C4,&C5,&C6,&C7 , &CC,&CD,&CE,&CF , &D4,&D5,&D6,&D7 , &DC,&DD,&DE,&DF,

&E4,&E5,&E6,&E7 , &EC,&ED,&EE,&EF , &F4,&F5,&F6,&F7 , &FC,&FD,&FE,&FF.

Bitte Beachten: Ein SINI0/1 auf die erste Spur, von der geladen werden soll, muss abgeschickt worden sein. Er darf wie immer noch nicht beendet sein. Das LW muss bereit sein.

INHALTSVERZEICHNISSES EINES LAUFWERKS LADEN

Kurzbeschreibung: Das Inhaltsverzeichnisses eines Laufwerks wird in den DIR-Puffer geladen.

Label: LESEDIR (interner FDC) // LES1DIR (externer FDC)

ROM-Nummer: B

Startadresse: &FDFA (LESEDIR) // &FDFD (LES1DIR)

Einsprungbedingungen: YL = Laufwerk 0..3
Die Systemvariable TURBO_X muß korrekte Werte enthalten.

Aussprungbedingungen: Das Inhaltsverzeichniss wurde unterhalb von TURBO_X eingelesen, TURBO_X wurde nach unten korregiert.

REG08_0 enthält das Laufwerk 0..7 (A,B,..,H) und A enthält die DIR-Länge in Pages (DIR Länge/256). Das Inhaltsverzeichniss ist noch unsortiert.

Manipuliert: AF, BC, DE, HL, AF', BC', HL', IX, YH und die RAM-Variablen REG08_0, REG16_0..7, FDC_RES, der FDC und die RAM-Konfiguration.

Beschreibung: Im FutureOS werden die Inhaltsverzeichnisse im RAM gepuffert, dies hat oft eine erhebliche Geschwindigkeitssteigerung zu Folge, es ist aber auch nötig die Disketten verantwortungsvoll zu wechseln.

Diese OS Funktionen dienen dazu ein Inhaltsverzeichnis von Diskette ins Erweiterungs-RAM, in den DIR-Puffer zu lesen. Es kann Data, System, IBM oder Vortex Format verwendet werden.

Im Register YL wird das einzulesende Laufwerk 0..3 angegeben, die Auswahl des FDC geschieht wie immer über das Label.

Die System-Variable TURBO_X gibt an wohin das Inhaltsverzeichnis gelesen werden soll. Es wird direkt darunter eingelesen, anschließend wird TURBO_X angepasst. Ist das Erweiteungs-RAM zuende, dann wird das DIR in die ersten 64K eingelesen.

Bitte Beachten: Das eingelesene Inhaltsverzeichnis wird nicht sortiert.

INHALTSVERZEICHNISSE ALLER MARKIERTEN LAUFWERKE LESEN

Kurzbeschreibung: Die Inhaltsverzeichnisse aller markierten Laufwerke werden gelesen und sortiert.

Label: LESDIR1

ROM-Nummer: B

Startadresse: &FDEB

Einsprungbedingungen: Die System-Variablen der Laufwerke TURBO_A...M und TURBO_X sollten korrekte Werte enthalten.

REG08_0 sollte &FF enthalten, ansonsten wird das DIR des entsprechenden Laufwerks als eingelesen und zu sortieren angesehen.

Aussprungbedingungen: Die Inhaltsverzeichnisse aller markierten Laufwerke wurden gelesen und sortiert. Die System-Variablen wurden angeglichen.

Manipuliert: AF, BC, DE, HL, AF', BC', HL', IX, IY und die RAM-Variablen REG08_0, REG16_0..7, TURBO_A..M, TURBO_X, FDC_RES, der FDC und die RAM-Konfiguration.

Beschreibung: Dieser Einsprung wird dazu verwendet die Inhaltsverzeichnisse aller markierten Laufwerke ins Erweiterungs-RAM, also den DIR-Puffer einzulesen. Die Inhaltsverzeichnisse werden wärend der Step-Wartezeit sortiert, so daß sie gleich verarbeitbar sind.

Ist eine markiertes Laufwerk nicht bereit, dann erfolgt kein Rücksprung, sondern eine Deaktivierung des LWs und Einsprung ins Desktop.

Es können Daten, System, IBM oder Vortex Disketten bearbeitet werden.

Bitte Beachten: Die Inhaltsverzeichnisse werden sortiert. Bei Lesefehlern erfolgt kein Rücksprung, sondern Fehlermeldung durch das Desktop.

Berechnung einer 67 Bytes TESTSUMME einer AMSDOS DATEI mit HEADER

Kurzbeschreibung: Die Prüfsumme des Header-Records einer AmsDos bzw. FutureOS Datei wird berechnet.

Label: TST_HED

ROM-Nummer: B

Startadresse: &D75B

Einsprungbedingungen: DE = Anfang des 128 Byte Header-Records.

Aussprungbedingungen: HL = Prüfsumme des Headers (eben errechnet)
DE = Zeiger auf das Low-Byte der (eventuellen) Prüfsumme des Header-Records der Datei.
B = &00

Manipuliert: AF, BC, DE, HL und XL

Beschreibung: Abgesehen von reinen ASCII Dateien hat jede unter AMSDOS verwendete Datei einen sogenannten Header - Record. Das sind die ersten 128 Bytes einer jeden Datei auf Diskette. Der Anwender bekommt diese 128 Bytes unter AMSDOS nicht zu Gesicht.

Auch FutureOS Dateien können einen solchen Header enthalten. Unter FutureOS ist dieser Header stark erweitert, aber dennoch kompatibel.

Über die ersten 67 Bytes (0-66) wird eine Prüfsumme gebildet, diese wird in den Bytes 67 und 68 des Headers gesichert.

Will man nun wissen ob eine Datei einen Headerrecord hat, dann kann man ja einfach die Prüfsumme berechnen lassen (mit dieser OS Funktion) und den errechneten Wert mit dem Wert der Bytes 67 und 68 vergleichen.

Sind beide Werte gleich, dann ist tatsächlich ein Header vorhanden.

Bitte Beachten: Da nur Daten gelesen und addiert werden, kein Problem.
Wie ein Header unter AMSDOS und unter FutureOS aussieht, kann im Handbuch nachgeschlagen werden.

WARTEN BIS LAUFWERK BEREIT (MAX. 5 SEKUNDEN)

Kurzbeschreibung: Testen ob bestimmtes Laufwerk bereit ist. Sonst warten bis es bereit ist. ABER: maximale Wartezeit.

Label: LWR0 (interner FDC) bzw. LWR1 (externer FDC) 5. Sekunden warten
LR0S (interner FDC) bzw. LR1S (externer FDC) freie Wartezeit

ROM-Nummer: B

Startadresse: &DC49 (LWR0), &DC59 (LWR1) // &DC4C (LR0S), &DC5C (LR1S)

Einsprungbedingungen: Bei allen Einsprüngen gilt:
D = zu testendes Laufwerk (0..3)
Bei den Einsprüngen LR0S und LR1S gilt außerdem:
HL = Anzahl der Versuche, die LW auf READY testen

Aussprungbedingungen:
Z-Flag = 0 (geleert) ==> Laufwerk IST bereit und A = Status 3.
Z-Flag = 1 (gesetzt) ==> Laufwerk NICHT bereit.

Manipuliert: AF, BC und HL

Beschreibung: Diese OS Funktion ermittelt den Laufwerksstatus. Damit man mit einem Laufwerk arbeiten kann muss dieses READY melden, d.h. das Laufwerk ist bereit Daten zu transferieren.

Bei Aufruf der OS Funktion gibt man das zu testende Laufwerk in D an (0..3). Jeh nachdem welchen FDC man benutzen will, verwendet man ein anderes Label zum Einsprung.

Beim Einsprung durch LWR0 (FDC 0) bzw. LWR1 (FDC 1) hat man eine maximale Wartezeit von 5. Sekunden. Ist das Laufwerk dannach immer noch nicht bereit, so bricht die OS Funktion mit gesetztem Zero Flag ab. Wenn beim Aussprung das Zero Flag gelöscht ist, dann ist das Laufwerk bereit um damit zu arbeiten.

ACHTUNG: Will man nicht fünf Sekunden warten, wenn das Laufwerk nicht bereit ist, dann kann man auch die Anzahl der Versuche direkt im Register HL angeben. Es gilt: Eine Anzahl von &8000 (32768 dezimal) Versuchen entspricht in etwa einer Wartezeit von 5 Sekunden. Man verwendet dann die Einsprünge LR0S und LR1S.

Bitte Beachten: Will man z.B. nicht bei jedem Diskettenzugriff eine obligatorische Hochlaufzeit abwarten, so kann man diese OS Funktion einsetzen, da sie sofort Zurückkehrt, sobald das Laufwerk erstmals READY meldet. Es wird dannach keine Wartezeit mehr vergäudet.

Aus diesem Grund kommt das FutureOS auch komplett ohne Hochlaufzeiten aus, da bei der Verwendung einer festen Zeitspanne immer etwas zu lange gewartet werden muss.

Warten bis LAUFWERK BEREIT (MAX. 5 SEK) - Abbruch bei SCHREIBSCHUTZ

Kurzbeschreibung: Testen ob bestimmtes Laufwerk bereit ist. Sonst max. Wartezeit warten. Bei Schreibschutz erfolgt Fehlermeldung.

Label: LWS0 (interner FDC) bzw. LWS1 (externer FDC) 5. Sekunden warten
LS0S (interner FDC) bzw. LS1S (externer FDC) freie Wartezeit

ROM-Nummer: B

Startadresse: &DC69 (LWS0), &DC7D (LWS1) / &DC6C (LS0S), &DC80 (LS1S)

Einsprungbedingungen: Bei allen Einsprüngen gilt:
D = zu testendes Laufwerk (0..3)
Bei den Einsprüngen LS0S und LS1S gilt außerdem:
HL = Anzahl d. Versuche, die LW auf READY testen

Aussprungbedingungen:
Z-Flag = 0 (geleert) ==> Laufwerk IST bereit und A = Status 3.
Z-Flag = 1 (gesetzt) ==> Laufwerk NICHT bereit.
Hat das Laufwerk Schreibschutz, dann wird eine Fehlermeldung ausgegeben, und der Stackpointer SP um zwei erhöht. Anschließend springt die OS Funktion ins Desktop (FORA). ==> Keine Rückkehr.

Manipuliert: AF, BC und HL

Beschreibung: Diese OS Funktion ermittelt den Laufwerksstatus. Damit man mit einem Laufwerk arbeiten kann muss dieses READY melden, d.h. das Laufwerk ist bereit Daten zu transferieren. Zusätzlich wird noch die Beschreibbarkeit der Diskette ermittelt.
Bei Aufruf der OS Funktion gibt man das zu testende Laufwerk in D an (0..3). Jeh nachdem welchen FDC man benutzen will, verwendet man ein anderes Label zum Einsprung.
Beim Einsprung durch LWS0 (FDC 0) bzw. LWS1 (FDC 1) hat man eine maximale Wartezeit von 5. Sekunden. Ist das Laufwerk dannach immer noch nicht bereit, so bricht die OS Funktion mit gesetztem Zero Flag ab. Wenn beim Aussprung das Zero Flag gelöscht ist, dann ist das Laufwerk bereit um damit zu arbeiten.
ACHTUNG: Will man nicht ganze fünf Sekunden warten, wenn das Laufwerk nicht bereit ist, dann kann man auch die Anzahl der Versuche direkt im Register HL angeben. Es gilt: Eine Anzahl von &8000 (32768 dezimal) Versuchen entspricht in etwa einer Wartezeit von 5 Sekunden. Es werden die Einsprünge LS0S bzw. LS1S verwendet.
Das Spezielle dieser OS Funktion ist es zusätzlich den Schreibschutz der Diskette zu testen. Ist die Diskette schreibgeschützt, dann wird der Stackpointer um zwei Bytes erhöht (POP HL), es wird eine Fehlermeldung "Disk hat Schreibschutz" in der Statuszeile ausgegeben, und es erfolgt der Einsprung ins Desktop mittels FORA.

Bitte Beachten: Will man z.B. nicht bei jedem Diskettenzugriff eine obligatorische Hochlaufzeit abwarten, so kann man diese OS Funktion einsetzen, da sie sofort zurückkehrt, sobald das Laufwerk erstmals READY meldet. Es wird dannach keine Wartezeit mehr vergäudet. Aus diesem Grund kommt das FutureOS auch komplett ohne Hochlaufzeiten aus, da bei der Verwendung einer festen Zeitspanne immer etwas zu lange gewartet werden muss. Wenn das Laufwerk schreibgeschützt ist erfolgt KEIN Rücksprung!

LAUFWERK RECALIBRIEREN (SPUR NULL SUCHEN)

Kurzbeschreibung: Ein bestimmtes Laufwerk wird recalibriert. Alle acht Laufwerke sind recalibrierbar.

Label: REA0 (interner FDC), REA1 (externer FDC)

ROM-Nummer: B

Startadresse: &C058 (REA0), &C05D (REA1)

Einsprungbedingungen: D = Laufwerk 0..3 (egal ob FDC0/1)
Das Laufwerk sollte 'Ready' sein.

Aussprungbedingungen: Das zuvcr in D angegebene Laufwerk wurde recalibriert. Der Kopf ist über Spur &00, siehe Register L!
A = Status Register 0 des FDC.
L = Spur, über der sich der Kopf nun befindet, solle &00 sein.

Manipuliert: AF, BC, L und AF'. Der Kopf einen LW wurde bewegt.

Beschreibung: Nach dem Einschalten des Computers 'weiss' der FDC nicht über welcher Spur sich die Köpfe der Laufwerke befinden. Deshalb sollte man mindestens nach jedem Einschalten des Computers die Laufwerke recalibrieren.

Recalibrieren heißt nichts anderes als dass der Kopf eines Laufwerkes über die Spur &00 gefahren wird. Jedes Laufwerk hat einen Spur Null Sensor, der dem FDC mitteilt, dass sich der Kopf auch wirklich über der Spur Null befindet.

Das FutureOS führt z.B. vor jedem Lesen eines Inhaltsverzeichnisses das Kommando Laufwerk recalibrieren aus. Ist ein LW physikalisch nicht vorhanden, so kann sich das OS aufhängen, da ein vorhandenes LW u.U. ein nicht vorhandenes LW vorgaukeln kann.

Bitte Beachten: Achtung! Es darf keinesfalls ein Laufwerk recalibriert werden, dass garnicht angeschlossen ist. Dies könnte verheerende Folgen haben. Vorher immer testen ob das Laufwerk da ist und ob das LW 'Ready' ist.

SPURWECHSELZEIT(KONFIG) EINES LW AKTIVIEREN

Kurzbeschreibung: Es wird die durch das OS vorgegebene Spurwechselzeit eines Laufwerks aktiviert.

Label: SWZ0 (interner FDC) bzw. SWZ1 (externer FDC)

ROM-Nummer: B

Startadresse: &C129 (SWZ0), &C121 (SWZ1)

Einsprungbedingungen: A = Laufwerk 0..3

Aussprungbedingungen: Spurwechselzeit (Step Rate) des entsprechenden LWs gesetzt.

Manipuliert: AF, BC, D und HL. Spurwechselzeit ist verändert.

Beschreibung: Im FutureOS Eprom B ist für jedes Laufwerk eine bestimmte Spurwechselzeit angegeben. Der User hat beim Kauf seiner FutureOS Kopie entsprechende Zeiten angegeben. Durch diese OS Funktion wird der FDC mit der Spurwechselzeit des angegebenen Laufwerks versorgt.

Diese OS Funktion sollte man immer dann aufrufen wenn man mit einem bestimmten LW arbeiten will, denn ein anderes LW könnte eine andere SWZ (Spurwechselzeit) haben.

Bitte Beachten: Unbedingt vor jedem Wechsel des LWs, diese OS Funktion aufrufen, da sonst der FDC u. U. mit einer falschen SWZ arbeitet.

7 BYTES DER RESULT/ERGEBNISSPHASE ABHOLEN

Kurzbeschreibung: Die sieben Bytes der Resultphase werden vom FDC abgeholt und ins RAM geschrieben.

Label: FDC0_RE (gilt für beide FDCs)

ROM-Nummer: B

Startadresse: &C1C4

Einsprungbedingungen: BC = Datenregister FDC0/1
FDC muß Resultphase signalisieren. Bytes müssen zum Abholen bereit sein.

Aussprungbedingungen: 7 Bytes ab FDC_RES ins RAM gelesen.

Manipuliert: AF, C, HL und 7 Bytes ab FDC_RES.

Beschreibung: Läßt man den FDC ein Kommando ausführen, daß auch eine Result- oder Ergebniss-Phase hat, dann kann man diese OS Funktion dazu benutzen, die Result-Bytes abzuholen und ins RAM zu schreiben.

Normalerweise macht dies das OS, wenn man aber den FDC direkt programmiert, dann kann diese OS Funktion hilfreich sein.

Die sieben Result-Bytes werden vom FDC gelesen und ab FDC_RES ins RAM geschrieben. Will man die Bytes an eine andere Adresse gelesen haben, so springt man in die OS Funktion 3 Bytes nach dem Einsprung ein, und gibt in HL die Zieladresse an.

Bitte Beachten: Befindet sich der FDC vor Aufruf dieser OS Funktion nicht in der Resultphase, so führt dies zu Fehlfunktionen, eventuell kehrt die OS Funktion dann nicht mehr zurück.

SPURWECHSELZEITEN ALLER LW'S AUS ROM HOLEN

Kurzbeschreibung: Die Standart-Spurwechselzeiten aller acht Laufwerke werden aus dem ROM ins RAM kopiert, und dadurch aktiv.

Label: SWZGEN

ROM-Nummer: B

Startadresse: &C01F

Einsprungbedingungen: -

Aussprungbedingungen: Standart-Spurwechselzeiten aus ROM B aktiviert.

Manipuliert: BC, DE, HL und SWZ.

Beschreibung: Im ROM B stehen ab RSWZ die Spurwechselzeiten aller acht Laufwerke, sie wurden ürsprünglich für jeden Anwender im KONFIG festgelegt.

Diese OS Funktion kopiert diese Standart-Spurwechselzeiten (SWZ) ins RAM, dort erst werden sie aktiv. Alle FutureOS Disk-OS Funktionen verwenden die SWZ aus dem RAM.

Der Aufruf dieser OS Funktion ist sinnvoll nachdem man die RAM-SWZ selbstständig verändert hat, und nun diese änderungen nicht mehr benötigt.

Bitte Beachten: Die alten RAM-SWZ werden überschrieben, die ROM SWZ sollten also sinnvoll sein.

DISKETTE FORMATIEREN (DATA, SYSTEM, IBM, FutureOS)

Kurzbeschreibung: Mit diesen OS Funktionen können Disketten im Data, System, FutureOS oder IBM Format auf FDC0 oder FDC1 formatiert werden. DS möglich.

Label: Formatieren von Spur &00 bis &28 / 40:
F0DAT (Data Format, interner FDC) F1DAT (Data Format, exter.FDC)
F0SAT (System Format, interner FDC) F1SAT (System Format, ext.FDC)
F0FAT (FutureOS Format, interner FDC) F1FAT (FutureOS For, ext.FDC)
F0IAT (IBM Format, interner FDC) F1IAT (IBM Format, extern.FDC)

Formatieren, mit freier Wahl der Start- und End-Spur:
F0DAU (Data Format, interner FDC) F1DAU (Data Format, exter.FDC)
F0SAU (System Format, interner FDC) F1SAU (System Format, ext.FDC)
F0FAU (FutureOS Format, interner FDC) F1FAU (FutureOS For, ext.FDC)
F0IAU (IBM Format, interner FDC) F1IAU (IBM Format, extern.FDC)

ROM-Nummer: B

Startadresse:

&C4B7 (F0DAT) //	&C518 (F1DAT) //	&C579 (F0SAT) //	&C5CD (F1SAT)
&C621 (F0FAT) //	&C675 (F1FAT) //	&C76B (F0IAT) //	&C7BF (F1IAT)
&C4BC (F0DAU) //	&C51D (F1DAU) //	&C57E (F0SAU) //	&C5D2 (F1SAU)
&C626 (F0FAU) //	&C67A (F1FAU) //	&C770 (F0IAU) //	&C7C4 (F1IAU)

Einsprungbedingungen: Für aller Einsprünge gilt:
D = Seite 0/1, Laufwerk 0..3
YH = &00 ==> normal Formatieren (z.B. auf 40 Spur LWs)
YH = &FF ==> Formatieren mit DoubleStep (auf 80 Spur LWs)

Für F0DAU, F1DAU, F0SAU, ... ,F1IAU, gilt zusätzlich:
YL = erste Spur (inclusive) ab der formatiert werden soll.
A = letzte Spur (inclusive) die formatiert werden soll.

Aussprungbedingungen: War LW bereit und beschreibbar, dann wurden die entsprechenden Spuren formatiert.

Manipuliert: AF, BC, DE, HL, AF', IX, IY und die akt. SWZ

Beschreibung: Diese OS Funktionen ermöglichen es Disketten zu formatieren. Es kann mit Data, System, FutureOS oder IBM Format formatiert werden. Entweder mit dem internen FDC0 (LWs A, B, C und D) oder mit dem extenen FDC1 (E,F,G,H). DoubleStep ist wahlweise möglich Man kann entweder die Standartspuren von 0 bis 40 formatieren, oder die Start- und End- Spur der Formatierung frei wählen.

Bitte Beachten: Das LW sollte auch beschreibbar sein. Die Daten auf formatierten Disketten sind unwiederbringlich verloren.

DISKETTE FORMATIEREN (VORTEX Format)

Kurzbeschreibung: Mit diesen OS Funktionen können Disketten im Vortex Format auf FDC0 oder FDC1 formatiert werden.

Label: Formatieren von Spur &00 bis &50 / 80:
F0VAT (Vortex Format, FDC) F1VAT (Vortex Format, FDC 1)

Formatieren, mit freier Wahl der Start- und End-Spur:
F0VAU (Vortex Format, FDC) F1VAU (Vortex Format, FDC)

ROM-Nummer: B

Startadresse:
&C6C9 (F0VAT) // &C71A (F1VAT) // // &C6CE (F0VAU) // &C71F (F1VAU)

Einsprungbedingungen: Für aller Einsprünge gilt:
D = Laufwerk 0..3

Für F0VAU, F1VAU gilt zusätzlich:
YL = erste Spur (inclusive) ab der formatiert werden soll.
A = letzte Spur (inclusive) die formatiert werden soll.

Aussprungbedingungen: War LW bereit und beschreibbar, dann wurden die entsprechenden Spuren formatiert.

Manipuliert: AF, BC, DE, HL, AF', IX, IY und die akt. SWZ

Beschreibung: Diese OS Funktionen ermöglichen es Disketten im Vortex Format zu formatieren. Entweder mit dem internen FDC0 (LWs A, B, C und D) oder mit dem extenen FDC1 (E, F, G und H).

Man kann entweder die Standartspuren von 0 bis 80 formatieren, oder die Start- und End-Spur der Formatierung frei wählen.

Bitte Beachten: Das LW sollte auch beschreibbar sein. Die Daten auf formatierten Disketten sind unwiederbringlich verloren.

SPEICHERN VON DATEN MAX. 64K (DATA, SYSTEM, IBM)

Kurzbeschreibung: Daten von maximal 64KB können auf eine Diskettem mit Data, System oder IBM Format gespeichert werden. Es erfolgt kein Eintrag ins Inhaltsverzeichnis.

Label: S0DS (interner FDC) // S1DS (externer FDC)

ROM-Nummer: B

Startadresse: &D771 (S0DS) // &D7CC (S1DS)

Einsprungbedingungen: Die Speichertabelle muß generiert sein, und der SEEK auf die ertste Quellspur muß abgeschickt worden sein.
HL = Anfang der Speichertabelle
DE = Startadresse der zu sichernden Daten
REG08_0 = erste Spur, ab der gesichert wird.
REG08_1 = Seite 0/1, Laufwerk 0..3
REG16_0 = Adresse der Spur-Speicher-OS Funktion (s.o.)

Aussprungbedingungen: Die Quellbytes wurden auf Disk auf die in der Tabelle angegebenen Spuren und Sektoren geschrieben.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen REG08_0, REG_PC(low), FDC_RES und der FDC.

Beschreibung: Mittls dieser OS Funktion ist es möglich beliebige Daten aus den gerade eingeblendeten 64K auf Diskette zu schreiben. Dabei können Data, System oder IBM Format verwendet werden.

Das gewünschte Format wird durch REG16_0 festgelegt, darin ist die Adresse der Spur-Speicher-OS Funktion enthalten. Diese ist je nach Format bzw. FDC unterschiedlich.

Siehe dazu auch "EINE SPUR LESEN ODER SCHREIBEN FDC 0/1"

Die OS Funktion benutzt eine Speicher-Tabelle. Der Aufbau dieser Tabelle entspricht der einer Lade-Tabelle (siehe dort) für Data, System oder IBM Format.

Nach erfolgreichem Speichern stehen die Speicher-Bytes ab der Startadresse auf den Spuren und Sektoren die durch die Tabelle angegeben wurden. Jedoch erfolgt kein Eintrag ins Inhaltsverzeichnis.

Vor dem Aufruf dieser OS Funktion ist der Kopf des Ziellaufwerks auf den Weg zur ersten Zielspur zu schicken, z.B.: mittels SINI0/1.

Bitte Beachten: Es erfolgt KEIN Eintrag in Inhaltsverzeichnis.

SPEICHERN VON DATEN MAX. 64 KB (VORTEX Format)

Kurzbeschreibung: Daten von maximal 64 KB können auf eine Diskettem mit Vortex Format gespeichert werden. Es erfolgt kein Eintrag im Inhaltsverzeichnis.

Label: S0AV (interner FDC) // S1AV (externer FDC)

ROM-Nummer: B

Startadresse: &D826 (S0AV) // &D884 (S1AV)

Einsprungbedingungen: Die Speichertabelle muß generiert sein, und der SEEK auf die ertste Quellspur muß abgeschickt worden sein
HL = Anfang der Speichertabelle
DE = Startadresse der zu sichernden Daten
REG08_0 = erste Spur, ab der gesichert wird
REG08_1 = Seite 0/1, Laufwerk 0..3

Aussprungbedingungen: Die Quellbytes wurden auf Disk auf die in der Tabelle angegebenen Spuren und Sektoren geschrieben.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen REG08_0, REG08_1, REG_PC(low), FDC_RES und der FDC.

Beschreibung: Mittls dieser OS Funktion ist es möglich beliebige Daten aus den gerade eingeblendeten 64KB auf Diskette zu schreiben. Dabei wird Vortex verwendet.

Die OS Funktion benutzt eine Speicher-Tabelle. Der Aufbau dieser Tabelle entspricht der einer Lade-Tabelle (siehe dort) für Vortex Format.

Nach erfolgreichem Speichern stehen die Speicher-Bytes ab der Startadresse auf den Spuren und Sektoren die durch die Tabelle angegeben wurden. Jedoch erfolgt kein Eintrag ins Inhaltsverzeichnis.

Vor dem Aufruf dieser OS Funktion ist der Kopf des Ziellaufwerks auf den Weg zur ersten Zielspur zu schicken, z.B.: mittels SINI0/1.

Bitte Beachten: Es erfolgt KEIN Eintrag in Inhaltsverzeichnis.

SPEICHERN VON DATEN BIS 512KB (DATA, SYSTEM, IBM)

Kurzbeschreibung: Daten aus dem Erweiterungs-RAM werden auf eine Diskette im Data, System oder IBM Format gesichert.

Label:
S016 (interner FDC, von &4000 in ERAM &C4) // S015 (intern.FDC, freie Startadresse)
S116 (externer FDC, von &4000 in ERAM &C4) // S115 (extern.FDC, freie Startadresse)

ROM-Nummer: B

Startadresse:
&D8DD (S016) // &D8E9 (S015) // &D9AF (S116) // &D9BB (S115)

Einsprungbedingungen: für alle Einsprünge gilt:
Die Speicher-Tabelle muß generiert, und der LW-Kopf mittels SINI auf dem Weg zur richtigen Spur sein.
HL = Anfang der Speicher-Tabelle
YH = Spurlänge/256 also &10 für IBM bzw. &12 für Data, System
REG08_1 = Seite 0/1, Laufwerk 0..3 auf das gespeichert werden soll.
REG16_0 = Adresse der Spur-Speicher-OS Funktion (s.o.)
Die Labels S015 bzw. S115 benötigen noch folgende Daten:
DE = Startadresse der Daten &0000..&7E00 (normal &4000..&7E00)
AKT_RAM = erster 16K Exp-RAM Block ab dem gesichert werden soll.
Dieser Block (&C4, &C5, ..., &FF) muß bereits eingeblendet sein.

Aussprungbedingungen: Daten wurden auf Disk geschreiben.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen AKT_RAM, REG08_0, REG_PC(low), FDC_RES und der FDC, sowie der Sektor-Bereich &8000..&8FFF.

Beschreibung: Diese OS Funktionen ermöglichen es Daten des Erweiterungs-RAMs auf Diskette zu schreiben. Es können Daten, System oder IBM Format verwendet werden.
Benutzt man die Labels S016 bzw. S116, so kann man sich beim Einsprung einige Parameter sparen, und es wird automatisch ab der Adresse &4000 des ersten Exp.-RAM-Blocks &C4 gesichert.
Verwendet man die Labels S015 bzw. S115, dann sind Startadresse und Start-RAM-Block frei wählbar.
In jedem Fall wird das Exp.-RAM stückweise Block um Block gesichert.
In keinem Fall erfolgt ein Eintrag in Inhaltsverzeichnis.

Bitte Beachten: Es erfolgt KEIN Eintrag in Inhaltsverzeichnis.

SPEICHERN VON DATEN BIS 512 KB (VORTEX Format)

Kurzbeschreibung: Daten aus dem Erweiterungs-RAM werden auf eine Diskette im Vortex Format gesichert.

Label:
S0V6 (interner FDC, von &4000 in ERAM &C4) // S0V5 (intern.FDC, freie Startadresse)
S1V6 (externer FDC, von &4000 in ERAM &C4) // S1V5 (extern.FDC, freie Startadresse)

ROM-Nummer: B

Startadresse:
&DA81 (S0V6) // &DA8D (S0V5) // &DB61 (S1V6) // &DB6D (S1V5)

Einsprungbedingungen: für alle Einsprünge gilt:
Die Speicher-Tabelle muß generiert, und der LW-Kopf mittels SINI auf dem Weg zur richtigen Spur sein.
HL = Anfang der Speicher-Tabelle
REG08_0 = erste Spur auf die gesichert werden soll 0..79.
REG08_1 = Seite 0/1, Laufwerk 0..3 auf das gespeichert werden soll.
Die Labels S0V5 bzw. S1V5 benötigen noch folgende Daten:
DE = Startadresse der Daten &0000..&7E00 (normal &4000..&7E00)
AKT_RAM = erster 16K Exp-RAM Block ab dem gesichert werden soll.
Dieser Block (&C4, &C5, ..., &FF) muß bereits eingeblendet sein.

Aussprungbedingungen: Daten wurden auf Disk geschreiben.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, YL, und die RAM-Variablen AKT_RAM, REG08_0, REG08_1, REG_PC(low), FDC_RES und der FDC, sowie der Sektor-Bereich &8000..&8FFF.

Beschreibung: Diese OS Funktionen ermöglichen es die Daten des Erweiterungs-RAMs auf Diskette zu schreiben. Es wird Vortex das Format verwendet.

Benutzt man die Labels S0V6 bzw. S1V6, so kann man sich beim Einsprung einige Parameter sparen, und es wird automatisch ab der Adresse &4000 des ersten Exp.-RAM-Blocks &C4 gesichert.

Verwendet man die Labels S0V5 bzw. S1V5, dann sind Startadresse und Start-RAM-Block frei wählbar.

In jedem Fall wird das Exp.-RAM stückweise Block um Block gesichert.

In keinem Fall erfolgt ein Eintrag in Inhaltsverzeichnis.

Bitte Beachten: Es erfolgt KEIN Eintrag in Inhaltsverzeichnis.

INHALTSVERZEICHNIS EINENS LAUFWERKS SCHREIBEN

Kurzbeschreibung: Das Inhaltsverzeichnis eines Laufwerks wird auf Diskette geschrieben.

Label:
XSRIN0 (interner FDC, mit Wartezeit) / SRIN0 (int.FDC, OHNE Wartezeit)
XSRIN1 (externer FDC, mit Wartezeit) / SRIN1 (ext.FDC, OHNE Wartezeit)

ROM-Nummer: B

Startadresse:
&FDF7 (XSRIN0) // &FDF4 (SRIN0) // &FDF1 (XSRIN1) // &FDEE (SRIN1)

Einsprungbedingungen: D = LW 0..3
Die System-Variablen der Laufwerke müssen korrekte Werte enthalten.

Aussprungbedingungen: YL = LW 0..3

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL' und YL

Beschreibung: Diese OS Funktionen dienen dazu ein im RAM gepuffertes Inhaltsverzeichnis zurück auf Disk zu schreiben.

Da nach jedem Schreib-Vorgang nach dem Sektor noch die GAP#3 Bytes geschrieben werden müssen, muß eine entsprechende Wartezeit eingelegt werden.

XSRIN0 bzw. XSRIN1 wartet diese Zeit automatisch ab, erst dannach kehrt die OS Funktion zurück. Bei Verwendung von SRIN0 bzw. SRIN1 muß man selbstständig für die richtige Wartezeit sorgen, vorrausgesetzt man will das selbe Laufwerk sofort wieder benutzen. Dies geschieht z.B. durch das einlesen der nächsten Sektor ID durch HOLE_ID.

Für Formatinformationen usw. werden die System-Variablen benutzt.

Bitte Beachten: Benutzt man SRIN0 bzw. SRIN1 und will man anschließend das selbe Laufwerk benutzen, so MUß man unbedingt eine Wartezeit verstreichen lassen, da der FDC noch etwas Zeit braucht um die GAP#3 Bytes zu schreiben. Dies geschieht z.B. durch Einlesen der nächsten Sektor ID.

InHaltsVerzeichnisse aller MARKIERTEN und manipulierten LW'S SICHERN

Kurzbeschreibung: Die Inhaltsverzeichnisse aller markierten und manipulierten Laufwerke werden gesichert.

Label: SIDIR

ROM-Nummer:B

Startadresse: &FDE8

Einsprungbedingungen: System-Variablen aller LWs, HD, MD korrekt.

Aussprungbedingungen: DIRs aller markierten & manipulierten LWs wurden gesichet.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL' und YL

Beschreibung: Jedes Laufwerk, jede HD Partition und die Memory-Floppy haben ihre System-Variablen. Darin ist vermerkt ob ein Inhaltsverzeichnis eingelesen und verändert wurde.

Ist dies der Fall, so muß es irgendwann wieder auf Platte geschrieben werden.

Diese OS Funktion prüft die Variablen eines jeden Speichermediums (A-M). Ist ein Inhaltsverzeichnis eingelesen UND wurde es auch als verändert markiert, dann wird es auf Disk zurückgeschrieben.

Bitte Beachten: Es werden nur die Inhaltsverzeichnisse gesichert, die auch manipuliert wurden.

TABELLE PHYSIKALISCHER RAM-BLÖCKE GENERIEREN

Kurzbeschreibung: Erstellen einer Tabelle von freien 16 KB E-RAM-Blöcken des Erweiterungs-Speichers, physikalisch numeriert.

Label: GTPRB

ROM-Nummer: B

Startadresse: &FDE5

Einsprungbedingungen: HL = Ziel Tabelle wobei HL + 34 < &??00

Aussprungbedingungen: Tabelle wurde ab altem HL - Wert ins RAM geschrieben, beginnend mit &00.
A = &FD
BC = &7FFF
DE = &B9EF = XRAM_FF
HL = Ende der Tabelle im RAM

Manipuliert: AF, BC, DE und L

Beschreibung: Benötigt man einige freie Speicherblöcke (Kurzzeit-Speicher), so können diese durch diese OS Funktion ermittelt und in eine Tabelle zusammengefasst werden.

Es wird zuerst Block &C4, dann &C5, usw. und zuletzt &FF getestet. Die Tabelle ist folgendermaßen aufgebaut:

Sie beginnt mit einem Byte &00, d.h. sie kann von oben her (Block &FF) gelesen werden (HL zeigt nach Rücksprung auf Tabellenende).

Nach diesem Byte &00 folgen, in aufsteigender Folge, die physikalischen RAM-Nummern der Exp.-RAMs.

Bitte Beachten: Die Tabelle beginnt mit einem &00 Byte, hat aber keine Endmarkierung.

Beim Aufruf muß das Low-Byte von HL kleiner als &DE sein, da bei der Generierung der Tabelle nur L erhöht wird, nicht aber H.

HD20 - LESEN EINER 8,5 KB SPUR

Kurzbeschreibung: Eine 8,5KB Spur wird von Festplatte (Dobbertin HD20) eingelesen.

Label: HDLSPUR oder HDLSP (SCHNELLER).

ROM-Nummer: B

Startadresse: &DCC6 (HDLSPUR), &DCCC (HDLSP)

Einsprungbedingungen:
HDLSPUR:
B = Kopfnummer von 0 bis 3
DE = Cylindernummer von 0 bis 613
HL = Zieladresse für 8.5 KB (= &2200) Daten

HDLSP:
D = CC000000 = 2 Bit MSB Cylinder
E = LSB Cylinder, untere 8 Bit der Cylindernummer von 0 bis 613
HL = Zieladresse für 8.5 KB Daten
XL = Kopfnummer von 0 bis 3

Aussprungbedingungen:
A = &00 ==> Erfolg, Daten gelesen, alles o.k. HL=HL+&2200
A = &02 ==> ein Sektor wurde nicht gefunden.
A ungleich &00, dann Fehler aufgetreten. HL ist undefiniert.
Bei Erfolg: HL wurde um &2200 erhöht, HL zeigt auf Byte hinter gelesenem Datenblock.
Es wurden 8.5 KB von der HD gelesen, wenn alles gut ging (A=0 bei Aussprung).

Manipuliert: AF, BC, D, HL und IX

Beschreibung: Diese OS Funktion liest eine ganze Spur (8.5 KB) von der Dobbertin HD ein. Man kann entweder bei HDLSPUR oder bei HDLSP einspringen. Der Unterschied liegt lediglich darin, wie die Daten beim Aufruf in den versch. Registern übergeben werden. Siehe unter Punkt Einsprungbedingungen. HDLSPUR formt lediglich die Daten in den Registern für HDLSP um, und macht dann dort weiter. Die OS Funktion liest die Sektoren einer Spur genau in der Reihenfolge, in der sie auf der Platte erreichbar sind. - Schneller gehts nicht -
Will man eine große Datei einlesen, dann kommt es öfter vor, daß man komplette Spuren von HD lesen muß. Würde man die Sektoren einzeln lesen, dann würde dies um ein vielfaches länger dauern.
Benutzt man den direkten Einsprung in HDLSP, dann muss man die Cylindernummer 0..613 (10 Bit) auf zwei Register aufteilen. Die unteren 8 Bit der Cylindernummer werden in Register E übergeben. Die oberen 2 Bit werden in den oberen 2 Bits von Register D übergeben, die unteren 6 Bits von Register D müssen auf Null gesetzt werden.

Bitte Beachten: Die HD muß bereit sein. Ist die HD nicht bereit, oder gar nicht aktiv, dann kann das zu unvorhersehbaren Fehlern führen. Beim Aufruf von HDLSP müssen die unteren 6 Bits im Register D auf Null gesetzt werden, sonst wird die Spur nicht komplett gelesen.

HD20 - SCHREIBEN EINER 8,5 KB SPUR

Kurzbeschreibung: Eine 8,5KB Spur wird auf Festplatte (Dobbertin HD20) geschrieben.

Label: HDSSPUR oder HDSSP (SCHNELLER).

ROM-Nummer: B

Startadresse: &DF09 (HDSSPUR), &DF0F (HDSSP)

Einsprungbedingungen:
HDSSPUR:
B = Kopfnummer von 0 bis 3
DE = Cylindernummer von 0 bis 613
HL = Quelladresse für 8.5 KB (= &2200) Daten

HDSSP:
D = CC000000 = 2 Bit MSB Cylinder
E = LSB Cylinder, untere 8 Bit der Cylindernummer von 0 bis 613
HL = Quelladresse für 8.5 KB Daten
XL = Kopfnummer von 0 bis 3

Aussprungbedingungen:
A = &00 ==> Erfolg, Daten geschrieben, alles o.k. HL=HL+&2200
A = &02 ==> ein Sektor wurde nicht gefunden.
A ungleich &00, dann Fehler aufgetreten. HL ist undefiniert.
Bei Erfolg: HL wurde um &2200 erhöht, HL zeigt auf Byte hinter geschriebenem Datenblock.
Es wurden 8.5KB auf die HD geschrieben, wenn alles gut ging (A=0 bei Aussprung).

Manipuliert: AF, BC, D, HL und IX

Beschreibung: Diese OS Funktion schreibt eine ganze Spur (8.5 KB) auf die Dobbertin HD. Man kann bei HDSSPUR oder bei HDSSP einspringen. Der Unterschied liegt lediglich in der Datenübergabe in den Registern.
Siehe unter Punkt Einsprungbedingungen. HDLSPUR formt lediglich die Daten in den Registern für HDLSP um, und macht dann dort weiter. Die OS Funktion schreibt die Sektoren einer Spur genau in der Reihenfolge, in der sie auf der Platte erreichbar sind. - Schneller gehts nicht - Will man eine große Datei schreiben, dann kommt es öfter vor, daß man komplette Spuren auf HD schreiben muß. Würde man die Sektoren einzeln schreiben, dann würde dies um ein vielfaches länger dauern.
Benutzt man den direkten Einsprung in HDSSP, dann muss man die Cylindernummer 0 .. 613 (10 Bit) auf zwei Register aufteilen. Die unteren 8 Bits der Cylindernummer werden in Register E übergeben. Die oberen 2 Bits werden in den oberen 2 Bits von Register D übergeben, die unteren 6 Bits von Register D müssen auf Null gesetzt werden.

Bitte Beachten: Die HD muß bereit sein. Ist die HD nicht bereit, oder gar nicht aktiv, dann kann das zu unvorhersehbaren Fehlern führen. Beim Aufruf von HDSSP müssen die unteren 6 Bits im Register D auf Null gesetzt werden, sonst wird die Spur nicht komplett beschrieben!

HD20 - LESEN EINES 512 BYTES SEKTORS

Kurzbeschreibung: Ein 512 Bytes großer Sektor wird von der Dobbertin HD20 gelesen.

Label: HDLESE oder HDLES (schneller)

ROM-Nummer: B

Startadresse: &DCF8 (HDLESE), &DCFF (HDLES)

Einsprungbedingungen:
HDLESE:
B = Kopfnummer von 0 bis 3
C = Sektornummer von 0 bis 16 // &00 bis &10
DE = Cylindernummer von 0 bis 613
HL = Zieladresse für 0.5 KB Daten

HDLES:
D = CCSSSSSS = 2 Bit MSB Cylinder, die unteren 6 Bit enthalten die Sektornummer von 0 – 16
E = LSB Cylinder, untere 8 Bit der Cylindernummer von 0 bis 613
HL = Zieladresse für 0.5 KB Daten
XL = Kopfnummer von 0 bis 3

Aussprungbedingungen:
A = &00 ==> Erfolg, Daten gelesen, HL = HL + &0200, Zero-Flag gesetzt
A = &02 ==> Sektor wurde nicht gefunden
A ungleich &00, dann Fehler aufgetreten
Bei Erfolg: HL wurde um 0,5 KB (= &0200) erhöht, HL zeigt auf Byte hinter gelesenem Datenblock.
Es wurden 0.5 KB von HD gelesen, wenn A=0 bei Aussprung

Manipuliert: AF, BC, und HL, bei HDLESE außerdem: D, XL

Beschreibung: Mit dieser OS Funktion kann man einen Sektor von 512 Bytes von der Dobbertin HD einlesen. Man kann entweder bei HDLESE oder bei HDLES einspringen. Der Unterschied liegt nur darin, wie die Daten beim Aufruf in den versch. Registern übergeben werden. Siehe unter Punkt Einsprungbedingungen. HDLESE formt lediglich die Daten in den Registern für HDLES um, und macht dort weiter. Will man eine ganze Spur lesen, so sollte man lieber auf die OS Funktionen HDLSPUR bzw. HDLSP zurückgreifen.

Benutzt man den direkten Einsprung in HDLES, dann muss man die Cylindernummer 0 ... 613 (10 Bits) auf zwei Register aufteilen. Die unteren 8 Bit der Cylindernummer werden in Register E übergeben. Die oberen 2 Bit werden in den oberen 2 Bit von Register D übergeben, aber Vorsicht, denn Register D hat in den unteren 6 Bits noch die Sektornummer (von 0 bis 16).

Bitte Beachten: Die HD sollte vor Ausführung des Kommandos auf Bereitschaft getestet werden, sonst könnte eine Fehlfunktion auftreten.

HD20 - SCHREIBEN EINES 512 BYTES SEKTORS

Kurzbeschreibung: Ein 512 Bytes großer Sektor wird auf die Dobbertin HD20 geschrieben.

Label: HDSCHR oder HDSCH (SCHNELLER).

ROM-Nummer: B

Startadresse: &DF36 (HDSCHR), &DF3D (HDSCH)

Einsprungbedingungen:
HDSCHR:
B = Kopfnummer von 0 bis 3
C = Sektornummer von 0 bis 16 // &00 bis &10
DE = Cylindernummer von 0 bis 613
HL = Quelladresse für 0.5 KB Daten

HDSCH:
D = CCSSSSSS = 2 Bit MSB Cylinder, die unteren 6 Bit enthalten die Sektornummer von 0 – 16
E = LSB Cylinder, untere 8 Bit der Cylindernummer von 0 bis 613
HL = Quelladresse für 0.5 KB Daten
XL = Kopfnummer von 0 bis 3

Aussprungbedingungen:
A = &00 ==> Erfolg, Daten geschrieben, HL = HL + &0200, Zero-Flag gesetzt
A = &02 ==> Sektor wurde nicht gefunden
A ungleich &00, dann Fehler aufgetreten
Bei Erfolg: HL wurde um &0200 erhöht, HL zeigt auf Byte hinter geschriebenem Datenblock.
Es wurden 0.5 KB auf HD geschrieben, wenn A=0 bei Aussprung.

Manipuliert: AF, BC, und HL, bei HDSCHR außerdem: D, XL

Beschreibung: Mit dieser OS Funktion kann man einen Sektor von 512 Bytes auf die Dobbertin HD20 schreiben. Man kann entweder bei HDSCHR oder bei HDSCH einspringen. Der Unterschied liegt nur darin, wie die Daten beim Aufruf in den versch. Registern übergeben werden. Siehe unter Punkt Einsprungbedingungen. HDSCHR formt lediglich die Daten in den Registern für HDSCH um, und macht dort weiter. Will man eine ganze Spur schreiben, so sollte man lieber auf die OS Funktionen HDSSPUR bzw. HDSSP zurückgreifen.

Benutzt man den direkten Einsprung in HDSCH, dann muss man die Cylindernummer 0 ... 613 (10 Bits) auf zwei Register aufteilen. Die unteren 8 Bit der Cylindernummer werden in Register E übergeben. Die oberen 2 Bit werden in den oberen 2 Bit von Register D übergeben, aber Vorsicht, denn Register D hat in den unteren 6 Bits noch die Sektornummer (von 0 bis 16).

Bitte Beachten: Die HD sollte vor Ausführung des Kommandos auf Bereitschaft getestet werden, sonst könnte eine Fehlfunktion auftreten.

HD20 - LESE IHV/DIR EINER PARTITION NACH &4000

Kurzbeschreibung: Das Inhaltsverzeichnis einer HD20 Partition wird nach &4000 ins RAM eingelesen.

Label: L_DIR_HD

ROM-Nummer: B

Startadresse: &E156

Einsprungbedingungen: A = Partition 0..3

Aussprungbedingungen:
A = &00 ==> DIR korrekt gelesen. DIR steht ab &4000 im RAM.
A = &02 ==> Sektor nicht gefunden. Das DIR hat wohl einen Fehler!
A > &00 ==> anderer Fehler aufgetreten

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL' und IX.
Außerdem wird 1KB als Puffer benötigt. Der Bereich von &B400 bis &B7FF wird also verändert.

Beschreibung: Die Dobbertin HD20 ist in vier Partitionen aufgeteilt. Jede dieser Partitionen (im OS als I, J, K und L bezeichnet) hat ein eigenes IHV (= Inhaltsverzeichnis = DIR). Jedes IHV hat eine Länge von 16 KB.

Diese OS Funktion läd nun das IHV einer Partition (Akku: 0, 1, 2, 3) an die Adresse &4000 im RAM. Der Bereich von &B800 bis &BBFF wird dabei als Puffer benutzt und überschrieben.

Blendet man vor Aufruf der OS Funktion einen 16 KB langen E-RAM Block im Bereich von &4000 bis &7FFF ein, so wird das IHV dorthin geladen.

Bitte Beachten: Der Puffer-Bereich von &B800 bis &BBFF wird verändert.

HD20 - SCHREIBE IHV EINER PARTITION AB &4000

Kurzbeschreibung: Das Inhaltsverzeichnis einer Festplattenpartition der HD20 wird auf Platte geschrieben.

Label: S_DIR_HD

ROM-Nummer: B

Startadresse: &E1E5

Einsprungbedingungen: A = Partition 0..3 (entspricht I..L)
Das IHV muss ab &4000 im eingeblendeten RAM stehen.

Aussprungbedingungen:
A = &00 ==> Erfolg, IHV korrekt gesichert
A = &02 ==> Sektor wurde nicht gefunden.
A ungleich &00, dann Fehler aufgetreten.
IHV wurde auf Platte gesichert, wenn A=0 bei Aussprung.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX.
Außerdem dient der Bereich von &B400 bis &B7FF als Puffer.

Beschreibung: Diese OS Funktion dient dazu, das Inhaltverzeichnis / IHV / DIR einer Partition der Dobbertin HD20 zu sichern.

Beim Einsprung muss der Akku mit der Nummer der Partition geladen werden. Gültige Werte sind 0, 1, 2 oder 3, was den Partitionen I, J, K oder L entspricht. Außerdem muß das IHV von &4000 bis &7FFF (16KB) im RAM stehen. Und dieses RAM muß natürlich auch eingeblendet sein.

Nach Aufruf der OS Funktion gibt das A Register über den Erfolg der Operation Aufschluß. Ein &00 Byte bedeutet wie so oft, daß alles gut gegangen ist. Das IHV wurde auf Platte geschrieben.

Da die schnellen Spur-Sicherungs-OS Funktionen benutzt werden, dient der Bereich von &B400 bis &B7FF als Pufferbereich.

Bitte Beachten: Hat der Akku beim Aussprung einen anderen Wert als &00, dann ist ein Fehler aufgetreten.

HD20 - BLOCK IN CYLINDER, KOPF UND SEKTOR UMRECHNEN

Kurzbeschreibung: Eine Blocknummer der Dobbertin HD20 wird in ihren Cylinder, Kopf und Sektor umgerechnet.

Label: HDB2C

ROM-Nummer: B

Startadresse: &E676

Einsprungbedingungen:
HL = Blocknummer > 3 !!!
REG08_1 = Partition 0..3 (entspricht I, J, K und L)

Aussprungbedingungen:
C = Sektor &00 - &10 (0 - 17)
DE = Cylinder * 4 + Kopfnummer (untere 2 Bits 1,0)
HL = &0264 (= &0099 * 4)

Manipuliert: AF, BC, DE und HL

Beschreibung: Im Inhaltsverzeichnis der Dobbertin HD werden 16 Bit Blocknummern verwendet. Diese OS Funktion dient dazu die logische 16 Bit Blocknummer in die physikalischen Werte von Cylinder, Kopfnummer und Sektor umzurechnen.

Beim Aufruf der OS Funktion wird die Blocknummer in HL übergeben. Die OS Funktion liefert in DE die Cylindernummer mit 4 multipliziert. Die Kopfnummer steht in den beiden unteren Bits von DE. Das Register C erhält die Sektornummer zwischen &00 und &10 (0 - 17 dez.).

Wichtig: In der RAM-Variablen REG08_1 muß die Nummer der Partition von 0 bis 3 (entspricht I, J, K und L) angegeben werden.

Bitte Beachten: Die Blocknummer muß größer als 3 sein. Der erste freie Block ist Block &0004. Denn zwischen Block 3 und 4 sind 25.5KB als "Systemspur" reserviert. Diese 25.5KB entsprechen drei Spuren zu 8.5KB

HD20 - CYLINDER-KOPF-SEKTOR-TABELLE GENERIEREN

Kurzbeschreibung: Aus den Blocknummern einer Datei wird eine Lade-bzw. Speicher-Tabelle generiert. BLOCK ==> CYLINDER-KOPF-SEKTOR.

Label: HD_TAB

ROM-Nummer: B

Startadresse: &E4F1

Einsprungbedingungen:
DE = 1. Eintrag im 32er DIR (&4000,&XX20,&XX40,&XX60 .. &7FE0)
HL = Zieladresse, ab der Cyl-K-Sek-Tabelle generiert werden soll
REG08_1 = Partition 0..3 (=I..L)

Aussprungbedingungen: Tabelle wurde ab HL generiert.

Manipuliert: AF, BC, DE, HL, BC', DE', HL' und IX

Beschreibung: Wenn man unter FutureOS etwas von Festplatte läd, oder darauf speichert, dann wird mit Lade-bzw.Speicher-Tabellen gearbeitet.
Durch diese Technik läßt sich ein erheblicher Geschwindigkeitsvorteil erzielen. Bevor mit einer Datei gearbeitet wird, werden alle ihre Blocknummern in Cylinder, Kopf und Sektoren umgewandelt. Diese Daten stehen dann in einer Tablle zur Verfügung.

Aufbau der Lese/Schreib-Tabelle für die Dobbertin HD20:
1.) 16 Bit:
- Cylindernummer * 4 + Kopf(0..3) oder ...
- &FFFF als Marke für das Tabellenende.

2.) 8 Bit:
- Sektoranzahl (1..16) oder ...
- &FF um anzuzeigen daß die gesammte Spur (17 Sektoren) bearbeitet wird.

3.) X * 8 Bit: (wenn Sektorzahl-Byte <> &FF)
- Sektornummern, in aufsteigender Reihenfolge, maximal 16 * 8 Bit.

4.) nun folgt die nächste "Cylinder*4+Kopf" Nummer (siehe 1.)) oder &FFFF um das Ende der Tabelle anzuzeigen.

Beim Aufruf von HD_TAB zeigt DE auf den ersten Eintrag/Extent der Datei. Diese Adresse muß durch 32 teilbar sein. Es sind also folgende Adressen möglich:
&XX00, &XX20, &XX40, &XX60, &XX80, &XXA0, &XXC0 oder &XXE0.
Die Zieladresse ist völlig frei wählbar, man sollte jedoch beachten, daß die generierte Tabelle bei großen Dateien auch eine beachtliche Länge erreichen kann.
Außerdem muß die RAM-Variable REG08_1 mit der Nummer der Partition 0-3 geladen werden.
Nach Aufruf der OS Funktion steht die Tabelle im RAM.

Bitte Beachten: Die Quelladresse sollte auf einen korrekten Eintrag zeigen. Die Bytes oberhalb der Zieladresse werden überschrieben.

HD20 - LADEN EINER DATEI (maximal 40 KB) VON HD20

Kurzbeschreibung: Eine Datei von maximal 40(44)KB wird von der HD20 ins RAM gelesen.

Label: LHD4

ROM-Nummer: B

Startadresse: &E254

Einsprungbedingungen:
HL = Adresse der Cylinder/Kopf/Sektor Tabelle der Datei
REG16_1 = Zieladresse im RAM

Aussprungbedingungen:
A = &00 ==> Erfolg, Datei korrekt gelesen
A = &02 ==> Sektor wurde nicht gefunden
A ungleich &00, dann Fehler aufgetreten
Datei wurde ins RAM geladen, wenn A=&00

Manipuliert: AF,BC,DE,HL,IX,IY und die Variablen REG16_0, REG16_1.

Beschreibung: Mit LHD4 ist es möglich eine Datei von maximal 40 KB in das gerade eingeblendete RAM zu lesen. Wenn man bereit ist, den Bereich der Datei-Tagging-Bytes zu opfern, dann kann man auch bis zu 44 KB laden (&0000 bis &AFFF). Beim Aufruf der OS Funktion muß HL mit der Startadresse einer zuvor generierten Cylinder/Sektor Tabelle geladen sein. Die Adresse, an die die Datei geladen werden soll, wird in REG16_1 übergeben. Wurde die Datei ordnungsgemäß geladen, dann enthält der Akku beim Aussprung ein &00 Byte.

Bitte Beachten: Die Datei darf den Bereich ab &A000 (bzw. B000) nicht überschreiben. Die HD muß bei Aufruf der OS Funktion natürlich bereit sein.

HD20 - LADEN EINER DATEI (maximal 512 KB) VON HD20 INS E-RAM

Kurzbeschreibung: Eine Datei von maximal 512 KB wird von der HD20 ins Erweiterungs-RAM geladen.

Label: LHD6

ROM-Nummer: B

Startadresse: &E355

Einsprungbedingungen:
HL = Adresse der Cylinder/Kopf/Sektor Tabelle der Datei
REG16_1 = Zieladresse im RAM
AKT_RAM = Ziel-RAM-Block ab dem geladen werden soll

Aussprungbedingungen:
A = &00 ==> Erfolg, Datei korrekt gelesen
A = &02 ==> Sektor wurde nicht gefunden
A ungleich &00, dann Fehler aufgetreten
Datei wurde ins RAM geladen, wenn A=&00

Manipuliert: AF, BC, DE, HL, IX, IY, die Variablen REG16_0, REG16_1, AKT_RAM und der RAM-Status. Außerdem der RAM-Bereich von &8000 bis &9FFF (8 KB Spurpuffer).

Beschreibung: Mit LHD6 ist es möglich eine Datei von maximal 512 KB in das Erweiterungs-RAM zu laden. Beim Aufruf der OS Funktion muß HL mit der Startadresse einer zuvor generierten Cylinder/Sektor Tabelle geladen sein. Die Adresse, an die die Datei geladen werden soll, wird in REG16_1 übergeben. Die physikalische Nummer des RAM-Blocks, ab dem geladen werden soll, wird in AKT_RAM übergeben. Dabei kommen wie immer nur &C4, &C5, &C6, &C7, &CC, &CD, &CE, &CF, &D4 ... &FF in Frage.

Wurde die Datei ordnungsgemäß geladen, dann enthält der Akku beim Aussprung ein &00 Byte.

Bitte Beachten: Das RAM, in das man läd, sollte auch existieren. Die HD muß bei Aufruf der OS Funktion natürlich bereit sein.

HD20 - SICHERN EINER DATEI (maximal 64 KB) AUF HD20

Kurzbeschreibung: Eine Datei von maximal 64 KB wird vom RAM auf die HD20 gesichert.

Label: SHD4

ROM-Nummer: B

Startadresse: &E3CB

Einsprungbedingungen:
HL = Adresse der Cylinder/Kopf/Sektor Tabelle der Datei
REG16_1 = Quelladresse im RAM

Aussprungbedingungen:
A = &00 ==> Erfolg, Datei korrekt gesichert
A = &02 ==> Sektor wurde nicht gefunden
A ungleich &00, dann Fehler aufgetreten
Datei wurde auf HD gesichert, wenn A = &00

Manipuliert: AF, BC, DE, HL, IX und die Variablen REG16_0, REG16_1.

Beschreibung: Mit SHD4 ist es möglich eine Datei von maximal 64KB auf die HD20 (von Dobbertin) zu speichern. Gespeichert wird dabei aus der gerade aktiven Speicherkonfiguration. Beim Aufruf der OS Funktion muß HL mit der Startadresse einer zuvor generierten Cylinder / Kopf / Sektor Tabelle geladen sein. Die Adresse, ab der die Datei gespeichert werden soll, wird in REG16_1 übergeben.

Wurde die Datei ordnungsgemäß gesichert, dann enthält der Akku beim Aussprung ein &00 Byte.

Bitte Beachten: Die HD20 muß bei Aufruf der OS Funktion natürlich bereit sein.

HD20 - SICHERN EINER DATEI (maximal 512 KB) VOM E-RAM AUF HD20

Kurzbeschreibung: Eine Datei von maximal 512 KB wird vom E-RAM auf die HD20 gesichert.

Label: SHD6

ROM-Nummer: B

Startadresse: &E423

Einsprungbedingungen:
HL = Adresse der Cylinder/Kopf/Sektor Tabelle der Datei
REG16_1 = Quelladresse im RAM
AKT_RAM = Quell-RAM-Block ab dem gespeichert werden soll

Aussprungbedingungen:
A = &00 ==> Erfolg, Datei korrekt gesichert
A = &02 ==> Sektor wurde nicht gefunden
A ungleich &00, dann Fehler aufgetreten
Datei wurde gesichert, wenn A=&00

Manipuliert: AF, BC, DE, HL, BC', DE', HL', IX, die Variablen REG16_0, REG16_1, AKT_RAM und der RAM-Status. Außerdem der RAM-Bereich von &8000 bis &9FFF (8 KB Spurpuffer).

Beschreibung: Mit SHD6 ist es möglich eine Datei von maximal 512KB vom Erweiterungs-RAM auf HD zu sichern. Beim Aufruf der OS Funktion muß HL mit der Startadresse einer zuvor generierten Cylinder/Sektor Tabelle geladen sein. Die Adresse, ab der die Datei gespeichert werden soll, wird in REG16_1 übergeben. Die physikalische Nummer des RAM-Blocks, ab dem gespeichert werden soll, wird in AKT_RAM übergeben. Dabei kommen wie immer nur &C4, &C5, &C6, &C7, &CC, &CD, &CE, &CF, &D4 ... &FF in Frage.

Wurde die Datei ordnungsgemäß gesichert, dann enthält der Akku beim Aussprung ein &00 Byte.

Bitte Beachten: Die HD muß bei Aufruf der OS Funktion bereit sein.

SELEKTIERE NÄCHSTEN 16 KB ERWEITERUNGS-RAM BLOCK (IN AKT_RAM)

Kurzbeschreibung: Selektieren nächsten 16 KB Block des Erweiterungs-RAMs. RAM Konfiguration in Variable AKT_RAM schreiben. Bis 8 MB RAM.

Label: NXX_ERM

ROM-Nummer: B

Startadresse: &E72B

Einsprungbedingungen: System RAM Variable AKT_RAM hat korrekten Wert.

Aussprungbedingungen: Variable AKT_RAM enthält RAM Auswahl des nächst höheren 16 KB Erweiterungs-RAM Blocks. Der Block ist zwischen &4000 und &7FFF eingeblendet.

Manipuliert: F, BC und RAM Variable AKT_RAM.

Beschreibung: Die OS Funktion NXX_ERM dient der Verwaltung von bis zu 8 MB Erweiterungs-RAM, heutige Speichererweiterungen unterstützen jedoch nur bis zu 4 MB. Die OS Funktion liest die aktuelle RAM Konfiguration aus der System-RAM-Variable AKT_RAM, selektiert dann den nächst höheren 16 KB Block des Erweiterungs-RAM, blendet den neuen 16 KB RAM Block zwischen &4000 und &7FFF ein und vermerkt die neue RAM-Konfiguration schließlich in der System-RAM-Variable AKT_RAM.
Die 16 KB Blöcke des 4 MB Erweiterungs-RAMs sind wie hier beschrieben in aufsteigender Reihenfolgen anzusprechen, es ist die physikalische RAM Konfiguration angegeben:

```
&7FC4, &7FC5, &7FC6, &7FC7, &7FCC, &7FCD, &7FCE, &7FCF,
&7FD4, &7FD5, &7FD6, &7FD7, &7FDC, &7FDD, &7FDE, &7FDF,
&7FE4, &7FE5, &7FE6, &7FE7, &7FEC, &7FED, &7FEE, &7FEF,
&7FF4, &7FF5, &7FF6, &7FF7, &7FFC, &7FFD, &7FFE, &7FFF,

&7EC4, &7EC5, &7EC6, &7EC7, &7ECC, &7ECD, &7ECE, &7ECF,
&7ED4, &7ED5, &7ED6, &7ED7, &7EDC, &7EDD, &7EDE, &7EDF,
&7EE4, &7EE5, &7EE6, &7EE7, &7EEC, &7EED, &7EEE, &7EEF,
&7EF4, &7EF5, &7EF6, &7EF7, &7EFC, &7EFD, &7EFE, &7EFF,

&7DC4, &7DC5, &7DC6, &7DC7, &7DCC, &7DCD, &7DCE, &7DCF,
&7DD4, &7DD5, &7DD6, &7DD7, &7DDC, &7DDD, &7DDE, &7DDF,
&7DE4, &7DE5, &7DE6, &7DE7, &7DEC, &7DED, &7DEE, &7DEF,
&7DF4, &7DF5, &7DF6, &7DF7, &7DFC, &7DFD, &7DFE, &7DFF,

&7CC4, &7CC5, &7CC6, &7CC7, &7CCC, &7CCD, &7CCE, &7CCF,
&7CD4, &7CD5, &7CD6, &7CD7, &7CDC, &7CDD, &7CDE, &7CDF,
&7CE4, &7CE5, &7CE6, &7CE7, &7CEC, &7CED, &7CEE, &7CEF,
&7CF4, &7CF5, &7CF6, &7CF7, &7CFC, &7CFD, &7CFE, &7CFF,

... &7BXX ... &7AXX ... &79XX ...

&78C4, &78C5, &78C6, &78C7, &78CC, &78CD, &78CE, &78CF,
&78D4, &78D5, &78D6, &78D7, &78DC, &78DD, &78DE, &78DF,
&78E4, &78E5, &78E6, &78E7, &78EC, &78ED, &78EE, &78EF,
&78F4, &78F5, &78F6, &78F7, &78FC, &78FD, &78FE, &78FF.
```

Bitte Beachten: Man sollte den Berich oberhalb der 4 MB Grenze nur nutzen, wenn das entsprechende RAM auch angeschlossen ist.

SELEKTIERE NÄCHSTEN 16 KB ERWEITERUNGS-RAM BLOCK (REGISTER BC)

Kurzbeschreibung: Selektieren nächsten 16 KB Block des Erweiterungs-RAMs. RAM Konfiguration in Register BC. Bis 8 MB E-RAM nutzbar.

Label: NXT_ERM

ROM-Nummer: B

Startadresse: &FE81

Einsprungbedingungen: Register BC enthält aktuelle RAM Konfiguration.

Aussprungbedingungen: Register BC enthält RAM Auswahl des nächsten 16 KB Erweiterungs-RAM Blocks (zwischen &4000 und &7FFF eingeblendet).

Manipuliert: F und BC.

Beschreibung: Die OS Funktion NXT_ERM dient der Verwaltung von bis zu 8 MB Erweiterungs-RAM, heutige Speichererweiterungen unterstützen jedoch nur bis zu 4 MB. Der OS Funktion wird die aktuelle RAM Konfiguration in Register BC übergeben. NXT_ERM selektiert dann den nächst höheren 16 KB Block des Erweiterungs-RAM, blendet den neuen 16 KB RAM Block zwischen &4000 und &7FFF ein und vermerkt die neue RAM-Konfiguration schließlich in BC.
Die 16 KB Blöcke des 4 MB Erweiterungs-RAMs sind wie hier beschrieben in aufsteigender Reihenfolgen anzusprechen, es ist die physikalische RAM Konfiguration angegeben:

```
&7FC4, &7FC5, &7FC6, &7FC7, &7FCC, &7FCD, &7FCE, &7FCF,
&7FD4, &7FD5, &7FD6, &7FD7, &7FDC, &7FDD, &7FDE, &7FDF,
&7FE4, &7FE5, &7FE6, &7FE7, &7FEC, &7FED, &7FEE, &7FEF,
&7FF4, &7FF5, &7FF6, &7FF7, &7FFC, &7FFD, &7FFE, &7FFF,

&7EC4, &7EC5, &7EC6, &7EC7, &7ECC, &7ECD, &7ECE, &7ECF,
&7ED4, &7ED5, &7ED6, &7ED7, &7EDC, &7EDD, &7EDE, &7EDF,
&7EE4, &7EE5, &7EE6, &7EE7, &7EEC, &7EED, &7EEE, &7EEF,
&7EF4, &7EF5, &7EF6, &7EF7, &7EFC, &7EFD, &7EFE, &7EFF,

&7DC4, &7DC5, &7DC6, &7DC7, &7DCC, &7DCD, &7DCE, &7DCF,
&7DD4, &7DD5, &7DD6, &7DD7, &7DDC, &7DDD, &7DDE, &7DDF,
&7DE4, &7DE5, &7DE6, &7DE7, &7DEC, &7DED, &7DEE, &7DEF,
&7DF4, &7DF5, &7DF6, &7DF7, &7DFC, &7DFD, &7DFE, &7DFF,

&7CC4, &7CC5, &7CC6, &7CC7, &7CCC, &7CCD, &7CCE, &7CCF,
&7CD4, &7CD5, &7CD6, &7CD7, &7CDC, &7CDD, &7CDE, &7CDF,
&7CE4, &7CE5, &7CE6, &7CE7, &7CEC, &7CED, &7CEE, &7CEF,
&7CF4, &7CF5, &7CF6, &7CF7, &7CFC, &7CFD, &7CFE, &7CFF,

... &7BXX ... &7AXX ... &79XX ...

&78C4, &78C5, &78C6, &78C7, &78CC, &78CD, &78CE, &78CF,
&78D4, &78D5, &78D6, &78D7, &78DC, &78DD, &78DE, &78DF,
&78E4, &78E5, &78E6, &78E7, &78EC, &78ED, &78EE, &78EF,
&78F4, &78F5, &78F6, &78F7, &78FC, &78FD, &78FE, &78FF.
```

Bitte Beachten: Man sollte den Berich oberhalb der 4 MB Grenze nur nutzen, wenn das entsprechende RAM auch angeschlossen ist.

Selektiere vorherigen 16 KB Erweiterungs-RAM BLOCK (in AKT_RAM)

Kurzbeschreibung: Selektiert vorherigen 16 KB Block des Erweiterungs-RAMs. RAM Konfiguration in Variable AKT_RAM schreiben. Bis 8 MB RAM.

Label: LXX_ERM

ROM-Nummer: B

Startadresse: &E745

Einsprungbedingungen: System RAM Variable AKT_RAM hat korrekten Wert.

Aussprungbedingungen: Das Sign-Flag informiert über den Erfolg.
Sign-Flag = M, Fehler, RAM-Untergrenze wurde erreicht.
Sign-Flag = P, alles bestens, neuer RAM Block selektiert. Dann:
Variable AKT_RAM enthält RAM Konfiguration des neuen Erweiterungs-RAM Blocks (16 KB, zwischen &4000 und &7FFF eingeblendet).

Manipuliert: F, BC und RAM Variable AKT_RAM.

Beschreibung: Die OS Funktion LXX_ERM dient der Verwaltung von bis zu 8 MB Erweiterungs-RAM, heutige RAM-Erweiterungen unterstützen nur bis 4 MB.
Die OS Funktion liest die aktuelle RAM Konfiguration aus der System-RAM-Variable AKT_RAM, selektiert dann den vorhergehenden 16 KB Block des Erweiterungs-RAM, blendet den neuen 16 KB RAM Block zwischen &4000 und &7FFF ein und vermerkt die neue RAM-Konfiguration in AKT_RAM.
Die 16 KB Blöcke des 4 MB Erweiterungs-RAMs sind wie hier beschrieben in aufsteigender Reihenfolgen anzusprechen, es ist die physikalische RAM Konfiguration angegeben:

```
&7FC4, &7FC5, &7FC6, &7FC7, &7FCC, &7FCD, &7FCE, &7FCF,
&7FD4, &7FD5, &7FD6, &7FD7, &7FDC, &7FDD, &7FDE, &7FDF,
&7FE4, &7FE5, &7FE6, &7FE7, &7FEC, &7FED, &7FEE, &7FEF,
&7FF4, &7FF5, &7FF6, &7FF7, &7FFC, &7FFD, &7FFE, &7FFF,

&7EC4, &7EC5, &7EC6, &7EC7, &7ECC, &7ECD, &7ECE, &7ECF,
&7ED4, &7ED5, &7ED6, &7ED7, &7EDC, &7EDD, &7EDE, &7EDF,
&7EE4, &7EE5, &7EE6, &7EE7, &7EEC, &7EED, &7EEE, &7EEF,
&7EF4, &7EF5, &7EF6, &7EF7, &7EFC, &7EFD, &7EFE, &7EFF,

&7DC4, &7DC5, &7DC6, &7DC7, &7DCC, &7DCD, &7DCE, &7DCF,
&7DD4, &7DD5, &7DD6, &7DD7, &7DDC, &7DDD, &7DDE, &7DDF,
&7DE4, &7DE5, &7DE6, &7DE7, &7DEC, &7DED, &7DEE, &7DEF,
&7DF4, &7DF5, &7DF6, &7DF7, &7DFC, &7DFD, &7DFE, &7DFF,

&7CC4, &7CC5, &7CC6, &7CC7, &7CCC, &7CCD, &7CCE, &7CCF,
&7CD4, &7CD5, &7CD6, &7CD7, &7CDC, &7CDD, &7CDE, &7CDF,
&7CE4, &7CE5, &7CE6, &7CE7, &7CEC, &7CED, &7CEE, &7CEF,
&7CF4, &7CF5, &7CF6, &7CF7, &7CFC, &7CFD, &7CFE, &7CFF,

... &7BXX ... &7AXX ... &79XX ...

&78C4, &78C5, &78C6, &78C7, &78CC, &78CD, &78CE, &78CF,
&78D4, &78D5, &78D6, &78D7, &78DC, &78DD, &78DE, &78DF,
&78E4, &78E5, &78E6, &78E7, &78EC, &78ED, &78EE, &78EF,
&78F4, &78F5, &78F6, &78F7, &78FC, &78FD, &78FE, &78FF.
```

Bitte Beachten: Man sollte den Berich oberhalb der 4 MB Grenze nur nutzen, wenn das entsprechende RAM auch angeschlossen ist.

Selektiere VORHERIGEN 16 KB ERWEITERUNGS-RAM BLOCK (Register BC)

Kurzbeschreibung: Selektiert vorherigen 16 KB Block des Erweiterungs-RAMs. RAM Konfiguration in Register BC. Unterstützt bis zu 8 MB E-RAM.

Label: LST_ERM

ROM-Nummer: B

Startadresse: &FEA3

Einsprungbedingungen: Register BC enthält RAM Konfiguration.

Aussprungbedingungen: Das Sign-Flag informiert über den Erfolg.
Sign-Flag = M, Fehler, RAM-Untergrenze wurde erreicht.
Sign-Flag = P, alles bestens, neuer RAM Block selektiert. Dann:
Register BC enthält RAM Konfiguration des neuen 16 KB E-RAMs.

Manipuliert: F und BC.

Beschreibung: Die OS Funktion LST_ERM dient der Verwaltung von bis zu 8 MB Erweiterungs-RAM. Die OS Funktion liest die aktuelle RAM Konfiguration aus Register BC, selektiert dann den vorhergehenden 16 KB Block des Erweiterungs-RAM, blendet den neuen 16 KB RAM Block zwischen &4000 und &7FFF ein und vermerkt die neue RAM-Konfiguration in Register BC.
Die 16 KB Blöcke des 4 MB Erweiterungs-RAMs sind wie hier beschrieben in aufsteigender Reihenfolgen anzusprechen, es ist die physikalische RAM Konfiguration angegeben:

```
&7FC4, &7FC5, &7FC6, &7FC7, &7FCC, &7FCD, &7FCE, &7FCF,
&7FD4, &7FD5, &7FD6, &7FD7, &7FDC, &7FDD, &7FDE, &7FDF,
&7FE4, &7FE5, &7FE6, &7FE7, &7FEC, &7FED, &7FEE, &7FEF,
&7FF4, &7FF5, &7FF6, &7FF7, &7FFC, &7FFD, &7FFE, &7FFF,

&7EC4, &7EC5, &7EC6, &7EC7, &7ECC, &7ECD, &7ECE, &7ECF,
&7ED4, &7ED5, &7ED6, &7ED7, &7EDC, &7EDD, &7EDE, &7EDF,
&7EE4, &7EE5, &7EE6, &7EE7, &7EEC, &7EED, &7EEE, &7EEF,
&7EF4, &7EF5, &7EF6, &7EF7, &7EFC, &7EFD, &7EFE, &7EFF,

&7DC4, &7DC5, &7DC6, &7DC7, &7DCC, &7DCD, &7DCE, &7DCF,
&7DD4, &7DD5, &7DD6, &7DD7, &7DDC, &7DDD, &7DDE, &7DDF,
&7DE4, &7DE5, &7DE6, &7DE7, &7DEC, &7DED, &7DEE, &7DEF,
&7DF4, &7DF5, &7DF6, &7DF7, &7DFC, &7DFD, &7DFE, &7DFF,

&7CC4, &7CC5, &7CC6, &7CC7, &7CCC, &7CCD, &7CCE, &7CCF,
&7CD4, &7CD5, &7CD6, &7CD7, &7CDC, &7CDD, &7CDE, &7CDF,
&7CE4, &7CE5, &7CE6, &7CE7, &7CEC, &7CED, &7CEE, &7CEF,
&7CF4, &7CF5, &7CF6, &7CF7, &7CFC, &7CFD, &7CFE, &7CFF,

... &7BXX ... &7AXX ... &79XX ...

&78C4, &78C5, &78C6, &78C7, &78CC, &78CD, &78CE, &78CF,
&78D4, &78D5, &78D6, &78D7, &78DC, &78DD, &78DE, &78DF,
&78E4, &78E5, &78E6, &78E7, &78EC, &78ED, &78EE, &78EF,
&78F4, &78F5, &78F6, &78F7, &78FC, &78FD, &78FE, &78FF.
```

Bitte Beachten: Man sollte den Berich oberhalb der 4 MB Grenze nur nutzen, wenn das entsprechende RAM auch angeschlossen ist.

LESE 128 BYTES VON EINEM I/O PORT

Kurzbeschreibung: Diese OS Funktion liest Blöcke von 128 Bytes von einem 16 Bit I/O Port.

Label: F_INP

ROM-Nummer: B

Startadresse: &DD84

Einsprungbedingungen:
A = Anzahl der 128 Byte Blöcke, die gelesen werden sollen
BC = 16 Bit Portadresse von der gelesen werden soll
HL = Ziel-Adresse der zu lesenden Daten

Aussprungbedingungen: Daten wurden gelesen

Manipuliert: AF, HL und der Speicherbereich ab HL

Beschreibung: Diese OS Funktion erlaubt es 1-256 Blöcke von jeweils 128 Bytes von einer 16 bit I/O Adresse ins RAM zu lesen.

Bitte Beachten: Es können maximal 32 KB am Stück gelesen werden.

SENDE 128 BYTES AN EINEN I/O PORT

Kurzbeschreibung: Diese OS Funktion schickt Blöcke von 128 Bytes an einen 16 Bit I/O Port.

Label: F_OUT

ROM-Nummer: B

Startadresse: &DFC2

Einsprungbedingungen:
A = Anzahl der 128 Byte Blöcke die an die I/O Adresse geschickt werden sollen
BC = 16 Bit Port Adresse, an die Daten gesendet werden sollen
HL = Quell-Adresse an der die Daten beginnen

Aussprungbedingungen: Daten wurden gesendet

Manipuliert: AF, HL

Beschreibung: Diese OS Funktion erlaubt es 1-256 Blöcke von 128 Bytes an eine 16 Bit I/O Portadresse zu senden.

Bitte Beachten: Maximal 32 KB können auf einmal gesendet werden