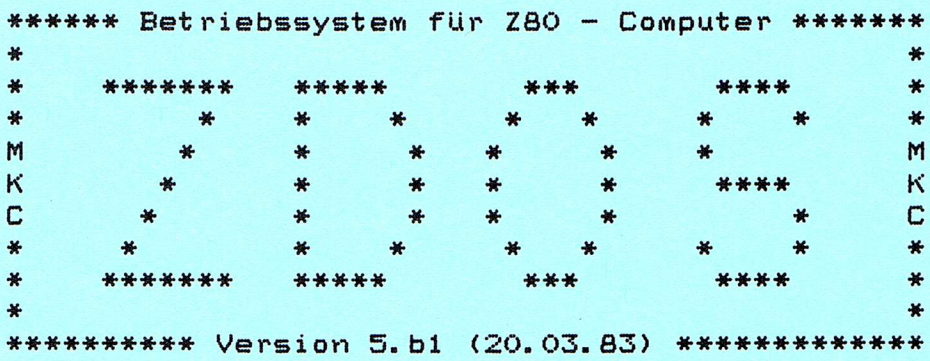

Inhaltsverzeichnis

HEAS ist der einzige Teil in **ZDOS**, der hardwareabhängig ist. In **HEAS** stehen alle Treiberroutinen für die Peripheriegeräte und die von **LEAS** benötigten Tabellen. Zusätzlich übernimmt **HEAS** die Zuordnung zwischen logischen und physikalischen Geräten.

HEAS besteht aus einzelnen Unterprogrammen, die alle über eine gemeinsame Sprungleiste erreicht werden können. Die **HEAS**-Routinen können von Programmen auf folgende Methode aufgerufen werden:

```
        LD    HL,(1)          ; Warmstart aus HEAS-Leiste
        LD    L,funktion      ; Nummer der Funktion * 3
        LD    DE,RETADR       ; diese Befehle ersetzen
        PUSH  DE              ; CALL  (HL)
        JP    (HL)
RETADR:
```

Hierbei wird aus der Nummer der gewünschten Funktion durch Multiplikation mit 3 die relative Adresse innerhalb der **HEAS**-Sprungleiste berechnet. Die Nummer erhält man durch Zählen der Einträge in der Sprungleiste. Als Beispiel hat die Funktion **CONIN** die Nummer 3 (vierter Eintrag, beginnend ab 0 zu zählen ergibt 3). Damit muß in L 3*3 = 9 geladen werden, um **CONIN** aufzurufen.

Achtung!

Nach **HEAS**-Aufrufen ist **nicht sichergestellt**, daß alle **LEAS**-Funktionen noch **einwandfrei** arbeiten. (Die **LEAS**-Funktionen, die auf dieselben **HEAS**-Routinen wie ein Anwenderprogramm zugreifen, können gestört werden; z.B. funktioniert die Tab-Auflösung der **LEAS**-Funktionen nach Benutzung von **HEAS-CONOUT** nicht mehr immer einwandfrei).

```
**********************************************************
**                                                      **
**    Die oben angeführte Methode ist die einzig        **
**    erlaubte Zugriffsversion auf HEAS-Routinen.       **
**    Jeder andere Zugriff verhindert die               **
**    Portabilität eines Programmes.                    **
**                                                      **
**********************************************************
```

Jetzt folgt eine Beschreibung der **HEAS**-Routinen

HEAS besteht aus der **Sprungleiste**, den in der Sprungleiste aufgerufenen **Routinen**, einem **Datenbereich** und den **Utility-Routinen.**

HEAS beginnt mit folgender Sprungleiste:

JP	BOOT	Einsprung nach Kaltstart
JP	WBOT	Warmstart
JP	CST	Konsolzeichen eingegeben ?
JP	CIN	Konsoleingabe
JP	COUT	Konsolausgabe
JP	LIST	Druckerausgabe
JP	PNCH	Lochstreifenausgabe
JP	RDR	Lochstreifeneingabe
JP	HOME	Kopf auf Spur 0
JP	SELDSK	Laufwerk selektieren
JP	SETTRK	Spur einstellen
JP	SETSEC	Sektor auswählen
JP	SETDMA	DMA-Puffer festlegen
JP	READ	Sektor lesen
JP	WRITE	Sektor schreiben
JP	LST	Drucker fertig ?
JP	SECTRA	Sektorverschränkung berechnen
JP	CBUF	Pufferadresse übergeben
JP	CONFIG	Konfiguration übergeben

Achtung

In **HEAS**-Routinen dürfen im wesentlichen nur die Register AF, BC, DE und HL benutzt werden. die Register IX und IY dürfen keinesfalls geändert werden, da sie in **LEAS** benötigt werden. Der zweite Registersatz wird von **ZDOS** generell nicht benutzt und steht daher dem Anwender uneingeschränkt zur Verfügung. Einzige Ausnahme sind einige Z80-CP/M-Programme, die wissen, daß CP/M nur die 8080-Register benutzt, und deshalb den zweiten Registersatz vor einem Systemaufruf nicht retten. Diese Programme sind aber sehr selten. Im normalen **HEAS** werden weder IX, IY noch der zweite Registersatz benutzt.

Nun folgt eine Beschreibung der in der **HEAS**-Sprungleiste aufgerufenen Rouinen in der Reihenfolge aus der Sprungleiste. **HEAS** hat keinen eigenen Stack; es benutzt den Stack des aufrufenden Programmes (also normalerweise den **LEAS**-Stack) mit. **HEAS** verändert im wesentlichen die Register **AF**, **BC**, **DE** und **HL**. Nicht alle **HEAS**-Routinen dürfen alle Register verändern. Dies ist bei den einzelnen Routinen angegeben.

BOOT Kaltstarteinsprung

Sprungleiste + 00H
Eingabe: A = Baudratencode
Ausgabe: C = 0 (Benutzer 0 und Betriebslaufwerk 0)
darf
ändern: AF, BC, DE, HL

BOOT ist die Routine, die vom Urlader **BOOT63** aufgerufen wird. **BOOT** initialisiert die Seite 0, setzt das Interrupt-System (falls erforderlich) und übergibt in Register C eine 0 für Laufwerk A. **BOOT** springt dann in den **KI**. (Adresse = Ladeadresse des **ZDOS**-Betriebssystems). Ferner hat **BOOT** die Aufgabe, alle I/O-Schnittstellen zu initialisieren; hierzu wird vom Urlader **BOOT63** in Register A der Wert, der in den CTC zu Baudratenerzeugung geschrieben wird, übergeben. **BOOT** trägt den vom Urlader erhaltenen Baudratencode in die Tabelle bei CINIT+10 und einen abgeleiteten Code bei CINIT+15 ein (für **CONFIG**).

Der Baudratencode ergibt sich aus folgender Aufstellung:

Baudraten-code	CINIT+15	Kommentar
13	80	9600 Baud, 4 MHz
26	81	4800 Baud, 4 MHz
52	82	2400 Baud, 4 MHz
104	83	1200 Baud, 4 MHz
8	00	9600 Baud, 2.5 MHz
16	01	4800 Baud, 2.5 MHz
32	02	2400 Baud, 2.5 Mhz
64	03	1200 Baud, 2.5 MHz
?	09	eine andere Baudrate, angenommen wird 2.5 MHz

WBOT Warmstartroutine

Sprungleiste + 03H

Eingabe: -
Ausgabe: C = Benutzernummer und Betriebslaufwerk (aus der Adresse 0004)
darf ändern: AF, BC, DE, HL

WBOT lädt das Betriebssystem (nur KI und **LEAS**) neu von der Diskette in Laufwerk A, setzt die Parameter der Seite 0 neu und übergibt beim Sprung in den **KI** in Register C die vor dem Warmstart als Betriebslaufwerk eingestellte Diskette (0 für A, 1 für B usw.)
Auf der Seite 0 sind zu setzen:

00H - 02H JP WBOT (**KI**-Ladeadresse + 1403H)
03H Grundeinstellung für das **IOBYTE**
05H - 07H JP LEAS (**KI**-Ladeadresse + 806H)

Ab der Version 5b1 (HEAS551) muß WBOT den LEAS-Datenbereich DATA löschen.

CST Konsolstatus

Sprungleiste + 06H

Eingabe: -
Ausgabe: A = FFH, falls Zeichen vorhanden
= 00H sonst
darf ändern: AF, BC

CST übergibt im Akku ein FFH, falls ein Zeichen von der Konsole eingegeben wurde, und ein 00H, falls kein Zeichen vorhanden ist.

CIN Konsoleingabe

Sprungleiste + 09H

Eingabe: -
Ausgabe: A = Zeichen
darf ändern: AF, BC

CIN liest das nächste Zeichen von der Konsole in den Akku, das höchste Bit muß gelöscht werden. **CIN** wartet, bis ein Zeichen eingegeben wird.

COUT Konsolausgabe

Sprungleiste + 0CH
Eingabe: C = Zeichen
Ausgabe: -
darf
ändern: AF, BC

COUT sendet das Zeichen aus Register C an die Konsole. Eventuell benötigte Warteschleifen nach CARRIAGE RETURN oder Schirm löschen sind hier zu implementieren.

LIST Druckerausgabe

Sprungleiste + 0FH
Eingabe: C = Zeichen
Ausgabe: -
darf
ändern: AF, BC

LIST sendet das Zeichen aus Register C an den logischen Drucker.

PNCH Lochstreifen stanzen

Sprungleiste + 12H
Eingabe: C = Zeichen
Ausgabe: -
darf
ändern: AF, BC

PNCH sendet das Zeichen aus Register C an den logischen Lochstreifenstanzer.

Diese Routine ist am Anfang nicht unbedingt erforderlich. Sie sollte erst eingefügt werden, wenn der Rest vollständig getestet ist.

RDR Lochstreifen lesen

Sprungleiste + 15H

Eingabe: -
Ausgabe: C = Zeichen
darf
ändern: AF, BC

RDR liest ein Zeichen vom Lochstreifenleser in den Akku, das höchste Bit wird gelöscht.

Diese Routine ist am Anfang nicht unbedingt erforderlich. Sie sollte erst eingefügt werden, wenn der Rest vollständig getestet ist.

HOME Laufwerk rücksetzen (Spur 0 suchen)

Sprungleiste + 18H

Eingabe: -
Ausgabe: -
darf
ändern: AF, BC, DE, HL

HOME positioniert den Kopf des selektierten Laufwerks auf Spur 00. Dies gilt solange, bis durch SETTRK eine andere Spur festgelegt wird.

SELDSK Laufwerk selektieren

Sprungleiste + 1BH

Eingabe: C = Laufwerksnummer
Ausgabe: HL = Adresse Disk-Header
= 0, falls Laufwerk nicht existent
darf
ändern: AF, BC, DE, HL

SELDSK selektiert ein Laufwerk. In Register C steht die Nummer des gewünschten Laufwerks (0 für A bis 7 für H). Bei Rückkehr wird im Register HL die Adresse des zugehörigen Disk-Headers bzw. eine 0, falls die Disk nicht existiert, übergeben. Dieses Laufwerk bleibt, bis durch einen erneuten Aufruf von SELDSK ein anderes Laufwerk bestimmt wird.

SETTRK — Spur für folgende Schreib- oder Leseoperationen festsetzen

Sprungleiste + 1EH
Eingabe: C = Spur (ab 0 gezählt)
Ausgabe: -
darf ändern: AF, BC, DE, HL

SETTRK positioniert den Kopf des selektierten Laufwerks auf die in Register C stehende Spur (gezählt wird ab Spur 0; die zweite Seite liegt hinter den Spuren der ersten Seite). **SETTRK** muß das SEEK-Kommando nicht unbedingt ausführen. Es reicht, wenn die Spurnummer gespeichert wird. Jede Ausführung kann der READ oder WRITE-Routine überlassen werden. Die hier festgelegte Spur gilt, bis durch SETTRK oder HOME eine andere Spur für ein Laufwerk bestimmt wird.

SETSEC — Sektor für folgende Schreib- oder Leseroutinen festsetzen

Sprungleiste + 21H
Eingabe: BC = Sektor
Ausgabe: -
darf ändern: AF, BC, DE, HL

SETSEC wählt den Sektor für nachfolgende Schreib- oder Leseoperationen aus. Register C enthält die Nummer des Sektors (gezählt wird ab Sektor 1). Die Sektornummer ist solange gültig, bis durch SETSEC eine neue festgelegt wird.

SETDMA — Pufferadresse für Schreib- oder Leseoperationen festlegen

Sprungleiste + 24H
Eingabe: BC = Schreib-/Lese-Adresse
Ausgabe: -
darf ändern: AF, BC, DE, HL

SETDMA legt den Speicherbereich für Schreib- oder Leseoperationen fest. Im Register BC wird die Anfangsadresse des DMA-Puffers übergeben. Seine Länge beträgt 128 Byte. Die Pufferadresse bleibt gleich, d.h. sie zählt nicht nach einer Operation um 128 Byte weiter. Die Pufferadresse kann nur durch einen SETDMA-Aufruf verändert werden.

READ Sektor lesen

Sprungleiste + 27H

Eingabe: -
Ausgabe: A = 0 falls erfolgreich
= 1 bei Fehler (ergibt: BAD SECTOR)
darf
ändern: AF, BC, DE, HL

READ liest den mit **SETSEC** gegebenen Sektor des selektierten Laufwerks von der eingestellten Spur in den DMA-Puffer. **READ** gibt im Akku eine 00H zurück, falls kein Fehler aufgetreten ist. **READ** sollte 10 Versuche zu lesen unternehmen, und erst wenn immer Fehler aufgetreten sind, eine 01H im Akku als Fehlermeldung übergeben.

WRITE Sektor schreiben

Sprungleiste + 2AH

Eingabe: C = Writetype (0, 1 oder 2)
Ausgabe: A = 0 falls erfolgreich
= 1 bei Fehler (ergibt: BAD SECTOR)
darf
ändern: AF, BC, DE, HL

WRITE beschreibt den vorher definierten Sektor mit den Daten aus dem DMA-Puffer. Auch hier sollten 10 Versuche durchgeführt werden, ehe im Akku eine 01H als Fehlermeldung zurückgegeben wird. Eine 00H im Akku bedeutet fehlerfreie Ausführung des **WRITE**-Befehls. Um mit anderen Sektorgrößen als 128 Byte effektiv arbeiten zu können, wird von LEAS beim Aufruf der **WRITE**-Routine der Writetyp mit übergeben. Es gilt folgender Code:

0 = normales Schreiben
1 = Schreiben ins Inhaltsverzeichnis
2 = Schreiben des ersten Sektors eines neuen Blocks.

Diese zusätzliche Information ermöglicht einen Zeitgewinn, da nicht mehr vor jedem Schreibzugriff in einen großen Sektor dieser gelesen werden muß (Typ 2 = neuer Block). Ferner kann das Inhaltsverzeichnis dadurch abgesichert sein, daß ein Schreibzugriff sofort durchgeführt wird, während normales Schreiben erst dann ausgeführt wird, wenn der Buffer benötigt wird. In einem HEAS mit einer von 128 Byte abweichenden Sektorgröße muß der Schreibtyp unter allen Umständen mit übergeben werden. Im Zweifelsfalle sollte ein Programm den Typ 1 übergeben. Dabei wird der Sektor vorher gelesen, dann geändert und sofort zurückgeschrieben. Dies ist zwar langsam, aber dafür sicher.

LST Druckerstatus

Sprungleiste + 2DH

Eingabe: -
Ausgabe: A = 0 falls Drucker nicht bereit
= FF falls Drucker bereit
darf
ändern: AF, BC

LST übergibt im Akku ein FFH, falls der Drucker bereit ist, ein Zeichen zu übernehmen; falls nicht, eine OOH.

SECTRA Sektorverschränkung berechnen

Sprungleiste + 30H

Eingabe: BC = logischer Sector (ab 0)
DE = Übersetzungstabelle für Sektorverschränkung
Ausgabe: HL = physikalischer (transformierter) Sektor (ab 1)
darf
ändern: AF, BC, DE, HL

SECTRA transformiert den logischen Sektor aus Register BC in den zugehörigen physikalischen Sektor und übergibt diesen im Register HL. Diese Sektorverschränkung erhöht die Geschwindigkeit bei aufeinanderfolgenden Diskzugriffen. Die Adresse der zu dem jeweiligen Laufwerk gehörenden Sektorverschränkungstabelle wird in Register DE von **LEAS** mit übergeben. (Achtung: bei nicht 128 Byte Sektorgröße ist **SECTRA** an dieser Stelle nicht empfehleswert.)
Ein Beispiel für die zur Transformation benutzte Tabelle, deren Adresse bei Aufruf von **SECTRA** in Register DE übergeben wird

```
XLT:    DEFB    1,7,13,19,25
        DEFB    5,11,17,23
        DEFB    3,9,15,21
        DEFB    2,8,14,20,26
        DEFB    6,12,18,24
        DEFB    4,10,16,22
```

Als Sektor wird an **SECTRA** 0 bis 25 übergeben. Durch Addition der Sektornummer zur Adresse XLT wird ein Zeiger auf die Sektornummer (jetzt physikalisch) gesetzt, der an das aufrufende Programm zurückgegeben werden soll. Durch Laden von Register L mit der Speicherzelle, auf die der soeben berechnete Zeiger hinweist, erhält man nach Löschen von Register H den physikalischen Sektor. Als Beispiel soll der Sektor 7 transformiert werden. Durch Addition von 7 zu XLT und anschließendes Laden des

Inhalts dieser soeben berechneten Position in der XLT-Tabelle erhält man eine 17. Dies ist der zu dem logischen Sektor 7 gehörende physikalische Sektor. **Logische Sektoren sind hier im Bereich von 0 bis 25; physikalische zwischen 1 und 26.**

CBUF Zeichen aus Buffer holen

Sprungleiste + 33H

Eingabe: -
Ausgabe: HL = Adresse des Zeichenpuffers
A = nächstes Zeichen aus dem Buffer
darf ändern: AF, HL

CBUF übergibt in Register A das nächste Zeichen aus dem Prozedurbuffer und in HL die Adresse des Buffers.
Die Pufferlänge des Prozedurpuffers beträgt 128 Bytes. Ein Programm, das den Puffer lesen will, muß selbst dafür sorgen, daß der Inhalt des Puffers nach jedem Lesen um 1 Byte nach vorne geschoben wird. Eine 0 im Puffer entspricht dem Ende einer logischen Zeile (Eingabe am Terminal: LINE FEED); zwei Nullen signalisieren das Ende der Eingaben. Der Rest des Puffers ist nicht definiert. Der Puffer wird unter **ZDOS** bei Aufruf der Funktion **READ CONSOLE BUFFER** und Ausnutzung der **Multi-Line**-Fähigkeit von **ZDOS** gefüllt.

CONFIG Adresse des Initialisierungstabelle übergeben

Sprungleiste + 36H

Eingabe: -
Ausgabe: HL = Adresse der Initialisierungstabelle
darf ändern: HL

CONFIG übergibt in HL die Adresse der Initialisierungstabelle.
In der Initialisierungtabelle stehen die Werte, die bei der Initialisierung in SIO und CTC der CPU-Karte geschrieben worden sind. (Die Write-Register der SIO sind nicht lesbar!)

In **HEAS** sind alle Tabellen zur Beschreibung der verwendeten Hardware enthalten. **LEAS** ist voll hardwareunabhängig (Ausnahme: Z80-CPU).

Die erste Gruppe der Beschreibungstabellen bezieht sich auf die Disketten-Laufwerke. Hierin werden die Parameter des Disk-Layouts angeben.

1. Disk-Parameter-Basis-Tabellen

Diese Tabelle muß einmal für jedes Laufwerk vorhanden sein. Ferner müssen alle Disk-Parameter-Basis-Tabellen in aufsteigender Reihenfolge der Laufwerke fest aneinanderhängend angeordnet sein. In der Disk-Parameter-Basis-Tabelle sind folgende Einträge enthalten:

DPBT+ 0	Adresse des Vektors, der die Sektorverschränkung enthält.
+ 2	ohne Bedeutung
+ 4	ohne Bedeutung
+ 6	ohne Bedeutung
+ 8	Adresse eines 128 Byte großen Puffers, der von **LEAS** für das Directory benötigt wird. Dieser Wert ist in allen Disk-Parameter-Basis-Tabellen gleich.
+ A	Adresse des Disk-Parameter-Blocks. Im Disk-Parameter-Block stehen die Daten der jeweiligen Diskette. Gleiche Laufwerke benutzen denselben Disk-Parameter-Block
+ C	Adresse eines Datenbereichs für **LEAS**, in dem der Prüfvektor für das Directory von **LEAS** abgelegt wird. Jede DPBT muß einen eigenen Vektor zur Verfügung stellen.
+ E	Adresse des Belegungsvektors der jeweiligen Diskette. (Jede DPBT muß einen eigenen Belegungsvektor haben.)
+10	Beginn der nächsten Disk-Parameter-Basis-Tabelle, bzw. Ende der letzten DPBT.

Alle Werte sind 2 Byte lang; gespeichert sind sie in der Reihenfolge: lower und dann higher Byte.

2. Disk-Parameter-Block

Im Disk-Parameter-Block sind folgende Parameter des Laufwerks vorhanden:

DPB + 0 Sektoren / Spur (2 Byte lang)
Achtung: gemeint sind Sektoren zu 128 Byte!

+ 2 Block-Schiebe-Faktor (1 Byte Lang)
= 3 für 1K Blockgröße
= 4 für 2K Blockgröße
= 5 für 4K Blockgröße
= 6 für 8K Blockgröße
= 7 für 16K Blockgröße

+ 3 Block-Schiebe-Maske (1 Byte lang)
= 07H für 1K Blockgröße
= 0FH für 2K Blockgröße
= 1FH für 4K Blockgröße
= 3FH für 8K Blockgröße
= 7FH für 16K Blockgröße

+ 4 Extent-Maske (1 Byte lang)
= 0 für 1 Extent / Dir.-Eintrag
= 1 für 2 Extents / Dir.-Eintrag
= 3 für 4 Extents / Dir.-Eintrag
= 7 für 8 Extents / Dir.-Eintrag
= F für 16 Extents / Dir.-Eintrag

+ 5 Kapazität des Laufwerks in Blocks, gezählt wird ab 0! (Länge 2 Bytes)
Achtung, es gibt nur ganze Blocks!

+ 7 Anzahl der Einträge im Directory (2 Bytes lang)

+ 9 Bit-Maske für reservierte Directory-Blocks (2 Bytes lang)
Für jeden reservierten Block ist von vorne beginnend ein Bit auf 1 zu setzen. Beispiel:
2 reservierte Blocks ergeben folgendes:
(DPB+9) = 11000000B = 0C0H
(DPB+A) = 00000000B = 000H

+ B Länge des Prüfvektors; (2 Bytes lang) benötigt 1 Byte pro Sektor des Directories

+ D Anzahl der Systemspuren (2 Bytes lang); normal sind 2 Spuren

+ F Adresse des Sektor-Puffers für Spooling (für alle DPBs immer gleich). Die Länge des Sektorpuffers beträgt 128 Bytes. (2 Bytes lang)

+11 Adresse des File-Control-Blocks für Spooling (für alle DPBs immer gleich). Platzbedarf für den FCB: 33 Bytes.

Beispiel: 8" Diskette, 26 Sektoren, 1 K Blocks, 242 Blocks Kapazität, 64 Directory-Einträge, dafür zwei Blocks reserviert, damit 16 Sektoren für das Directory und 2 Systemspuren.

```
; DISK PARAMETER BLOCK ( FOR ALL DISKS )
DPB:        DEFW    26              ; SECTORS / TRACK
            DEFB    3,7,0
            DEFW    242,63
            DEFB    0C0H,0
            DEFW    10H,2
            DEFW    SBUF            ; SECTOR BUFFER
            DEFW    FCB             ; FILE CONTROL BLOCK
```

Im Folgenden werden die benötigten Puffer beschrieben:

1. Der **Prüfvektor CSV** für die Directories hat 1 Byte für jeden Sektor des Directories als Länge. Damit ergibt sich in diesem Bespiel eine Länge von 16 Bytes.

```
CSV0:       DEFW    0,0,0,0,0,0,0,0
CSV1:       DEFW    0,0,0,0,0,0,0,0
```

2. Der **Belegungsvektor ALV** muß beim Kaltstart des Systems gelöscht sein. Seine Länge berechnet sich folgendermaßen:
Für jeden Block Kapazität der Diskette muß ein Bit zur Verfügung gestellt werden. Das so erhaltene Ergebnis ist auf ganze Bytes aufzurunden. Für das Beispiel folgt:
242 Blocks ergeben 242 Bits Länge
Das ergibt 242/8 Byte = 30,25 Bytes
Durch Aufrunden ergeben sich 31 Bytes pro Diskette

```
ALV0:       DEFW  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
ALV1:       DEFW  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

3. Der Prozedurbuffer hat die Länge 128 Byte. Direkt davor müssen zwei Bytes stehen (**READ CONSOLE BUFFER**). Das erste dieser Bytes hat den Wert 07EH, der des zweiten ist ohne Belang. Der Prozedurbuffer bietet die Möglichkeit, beim Kaltstart des Systems automatisch ein Kommando ausführen zu lassen. Im folgenden Beispiel ist das der Aufruf der Prozedur **INIT**.

```
            DEFB    07EH,7
BUF:        DEFB    'DO INIT',0,0,0,0,0,0,0,0,0
            DEFS    70H
```

4. Für SPOOLING sind ein **FCB** und ein 128 Byte langer Sektorpuffer SBUF erforderlich.

```
FCB:        DEFB    0           ; DISK / EMPTY - FLAG
            DEFW    0,0,0,0,0,0,0,0
            DEFW    0,0,0,0,0,0,0,0

SBUF        DEFS    80H         ; SPOOL BUFFER
```

4. HEAS-Puffer

5. Zur richtigen Funktion des Programmes CONFIG ist die Initialisierungtabelle CINIT im HEAS enthalten. In ihr stehen:
1, CTCA-Mode, CTCA-Time-Constant, 1, SIOA-Write-Register 3, SIOA-Write-Register 4, SIOA-Write-Register 5, der Code für die Baudrate des Ports A,
2, CTCB-Mode, CTCB-Time-Constant, 2, SIOB-Write-Register 3, SIOB-Write-Register 4, SIOB-Write-Register 5 und der Code für die Baudrate des Ports B. Beispiel:

```
CINIT:      DEFB    01,5,8,01,0C1H,44H,6AH,03
            DEFB    02,4DH,64,02,0C1H,44H,6AH,09
```

Der Baudratencode ist:

0 für 9600 Baud
1 für 4800 Baud
2 für 2400 Baud
3 für 1200 Baud
4 für 600 Baud
5 für 300 Baud
6 für 150 Baud
7 für 110 Baud
8 für 75 Baud
9 für eine unbekannte Baudrate

Das höchste Bit des Baudratencodes unterscheidet zwischen 2,5 und 4 MHz-Systemen. Bit 7 = 1 heißt 2,5 und Bit 7 = 0 heißt 4 MHz.

Achtung

Bei Umgehung der Baudratensuchroutine des **BOOT641** wird in **HEAS** 2,5 MHz als Systemtakt angenommen; dies ist bei 4 MHz-Systemen zu ändern. Im Normalfall wird vom **BOOT641** eine Information über die Terminalbaudrate und den Systemtakt an **HEAS** übergeben. (der Wert CTCB-Time-Constant).

6. Ferner enthält HEAS noch einen 128 Byte großen Directorybuffer DIRB, der von LEAS bei Directory-Zugriffen benutzt wird.

```
DIRB:       DEFS    80H     ; DIRECTORY-BUFFER
```

7. ab Version 5b1 ist der LAES-Datenbereich nach HEAS ausgelagert. Benötigt werden ca. 180 Byte direkt anschließend an die Sprungleiste, d.h. ab Adresse HEAS+03AH. Es ist darauf zu achten, daß die Routine CONFIG noch in der selben Page beginnt. (Falls nicht wird WRONG-SYSTEM gemeldet!)

Beschreibung von HEAS55

HEAS55 ist der hardwareabhängige Teil von ZDOS 5.b. HEAS55 ist über Steuervariable einfach an unterschiedliche Hardware adaptierbar. HEAS55 benutzt folgende Diskettenformate:

5,25" MFM = Double Density A oder H
512 Byte / Sektor
10 Sektoren / Spur
40 Spuren / Seite
einseitige Aufzeichnung mit invertieren Daten

MFM = Double Density B, I, J
512 Byte / Sektor
10 Sektoren / Spur
40 Spuren / Seite
beidseitige Aufzeichnung mit invertieren Daten

MFM = Double Density C
512 Byte / Sektor
10 Sektoren / Spur
80 Spuren / Seite
einseitige Aufzeichnung mit invertieren Daten

MFM = Double Density D oder K
512 Byte / Sektor
10 Sektoren / Spur
80 Spuren / Seite
beidseitige Aufzeichnung mit invertieren Daten

8" MFM = Double Density E
512 Byte / Sektor
15 Sektoren / Spur
77 Spuren / Seite
einseitige Aufzeichnung mit invertieren Daten

MFM = Double Density F
512 Byte / Sektor
15 Sektoren / Spur
77 Spuren / Seite
beidseitige Aufzeichnung mit invertieren Daten

FM = Single Density G
128 Byte / Sektor
26 Sektoren / Spur
77 Spuren / Seite
einseitige Aufzeichnung mit invertieren Daten
Standard CP/M - Format zum Austausch

Die fett geschriebenen Kennbuchstaben sind die bei FORMAT und SYSGEN zu wählenden Formate. In HEAS ist dies so nicht möglich. Hier muß das gewünschte Format über die Steuervariablen gewählt werden.

Erläuterung der Steuervariablen:

Es kann immer nur eine der beiden Zeilen aktiv sein, die andere ist durch das vorgestellte Semikolon auszublenden. Zum Ändern ist also das Semikolon zu versetzen.

EPC wählt zwisccchen der EPC-Version und der Version für die Floppycontrollerkarten FDC5 bzw. FDC8/5 aus:

```
;=============================================================
;       FDC OR EPC WANTED                                    I
;=============================================================
EPC     EQU     0                   ; EPC
;EPC    EQU     1                   ; FDC5 OR FC8/5
;=============================================================
```

DMAMOD wählt zwischen DMA und NON-DMA-Betrieb des Floppy-Controllers; 8" Disketten müssen immer im DMA-Betrieb vereinbart werden, der EPC kann nur im NON-DMA-Betrieb arbeiten:

```
;=============================================================
;       DEFINE MODE (DMA OR NON-DMA)                         I
;=============================================================
DMAMOD  EQU     0                   ; DMA MODE WANTED
;DMAMOD EQU     1                   ; NON-DMA-MODE WANTED
;=============================================================
```

Es folgt jetzt die Entscheidung zwischen 5,25" und 8" Disketten:

```
;=============================================================
;       5,25" OR 8" DISKS WANTED ?                           I
;=============================================================
MINI    EQU     0       ; 5,25" DISKS
;MINI   EQU     1       ; 8" DISKS
;=============================================================
```

Und die Anzahl der normalen Disketten:

```
;=============================================================
;       DEFINE NUMBER OF DD-DISKS                            I
;=============================================================
NDISKS  DEFL    2       ; NUMBER OF DISKS (MAX: 4)
;=============================================================
```

-- Achtung --

Hier zählen nur die normalen Disketten, d.h. z.B. die RAM-Disks zählen hier genauso nicht wie die Disk B im MIXED-System, die 40 Spur Disketten liest.

bei 5,25" Disketten die Entscheidung, ob 80 oder 40 Spur Laufwerke verwendet werden:

```
;=============================================================
;       DEFINE 5,25" DD-DISKS                                 I
;=============================================================
;       40 OR 80 TRACKS / SIDE                                I
;=============================================================
;MAXT   EQU     40      ; 40 TRACKS / SIDE
MAXT    EQU     80      ; 80 TRACKS / SIDE
;=============================================================
```

bei 80 Spur Laufwerken (5,25") die Abfrage, ob Laufwerk B 40 Spur Disketten lesen und schreiben soll:

```
;=============================================================
;       MIXED SYSTEM (A reads 80 track, B 40 track  I
;=============================================================
MIX     EQU     0       ; MIXED WANTED
;MIX    EQU     1       ; NORMAL SYSTEM
;=============================================================
```

Bei 8" Laufwerken kann mit SD das zweite 8" Laufwerk für Standard-CP/M-Disketten vereinbart werden. Dann liest und schreibt Laufwerk "A" nach wie vor ZDOS-Disketten, während Laufwerk "B" Standard-CP/M-Disketten lesen und schreiben kann (zum Austausch von Daten und Programmen)

-- S/D benötigt das File SD.MAC auf Disk "B" --

```
;=============================================================
;       DEFINE 8" DD-DISKS                                    I
;=============================================================
;       SINGLE DENSITY DISK WANTED ?                          I
;       INCLUDES FILE:  B:SD.MAC                              I
;=============================================================
;SD     EQU     0               ; SINGLE DENSITY
SD      EQU     1               ; DOUBLE DENSITY
;=============================================================
```

Die als Option lieferbare MBYTE-Version wird mit MBYTE eingeschaltet. Änderungen für die RAM-Floppy erfolgen in dem getrennten File MBYTE.MAC.

-- MBYTE benötigt das File MBYTE.MAC auf Disk "B" --

```
;=============================================================
;       OPTIONAL MBYTE  ( RAM - FLOPPY )                      I
;       INCLUDES FILE:  B:MBYTE.MAC                           I
;=============================================================
;MBYTE  EQU     0               ; MBYTE-OPTION
MBYTE   EQU     1               ; NORMAL
;=============================================================
```

Das als Option lieferbare gemischte 5,25"/8" System wird mit dem WINCH-Schalter eingeschaltet. Dies ist gedacht für eine spätere Erweiterung durch ein Winchster-Laufwerk, wird aber vorläufig nur für das 58-System benutzt.

-- WINCH benötigt die Datei WINCH.MAC auf Disk "B" --

```
;=========================================================
;        OPTIONAL WINCHESTER - DISK (UNIT A)             I
;        INCLUDES FILE:  B:WINCH.MAC                     I
;=========================================================
;WINCH  EQU     0                ; WINCHESTER OPTION
WINCH   EQU     1                ; NORMAL
;=========================================================
```

Ferner interessant sind die Portadressen der I/O-Chips. Sie sind, soweit von **ZDOS** benutzt, angegeben:

```
;=========================================================
;         ***** I/O-PORTS *****
;=========================================================
CONB    EQU     0E3H    ; CONSOLE B STATUS
CONA    EQU     0E1H    ; CONSOLE A STATUS
                        IFF     EPC
FDC     EQU     0F2H    ; EPC ONLY
TIMER   EQU     0E9H    ; MOTO-TIMER FOR EPC
RBOOT   EQU     0ECH    ; SWITCH OFF EPROM
                        ELSE
FDC     EQU     0       ; FLOPPY CONTROLL
                        ENDIF
FDD     EQU     FDC+1   ; FLOPPY DATA PORT
FTC     EQU     FDC+2   ; FLOPPY TERMINATE
DMA     EQU     FDC+3   ; FLOPPY-DMA-PORT
CTC     EQU     0E8H    ; CLOCK FOR CONA
PIOA    EQU     0E4H    ; PRINTER CONTROL
PIOB    EQU     0E6H    ; PRINTER DATA
```

Jetzt folgen die Wartezeiten für die Peripheriegeräte, nach denen von **HEAS55** eine Fehlermeldung ausgegeben wird.

```
PTIME   EQU     16      ; TIME-OUT (CENTRONICS)
STIME   EQU     16      ; TIME-OUT (V24)
```

An externen Adressen ist die Systemadresse = Anfangsadresse des KI = Ladeadresse anzugeben. Dies ist die Adresse, die auch bei GENCOM für alle Systemteile anzugeben ist.

```
;
;         ***** EXTERNALS *****
;
KI      EQU     0E000H
```

In HEAS55 ist die IOBYTE-Funktion implementiert. Hier sind die "Geräte" UC1, PTR, PTP, UP2 und UL1 noch frei. Zum Einfügen eigener Routinen sind die Zeilen der ASSIGN - TABLE:

```
;
;       **** ASSIGN - TABLES
;
TCIN:   DEFW    CAIN,CBIN,RDR,CBIN        ; TTY,CRT,BAT,UC1
TCOUT:  DEFW    CAOUT,CBOUT,LIST,CBOUT    ; TTY,CRT,BAT,UC1
TCST:   DEFW    CASTI,CBSTI,BATST,CBSTI   ; TTY,CRT,BAT,UC1
TRDR:   DEFW    CAIN,DUMY,CBIN,CBIN8      ; TTY,PTR,UR1,UR2
TPNCH:  DEFW    CAOUT,NO,CBOUT,NO         ; TTY,PTP,UP1,UP2
TLIST:  DEFW    CAOUT,CBOUT,PIO8,NO       ; TTY,CRT,LPT,UL1
TLST:   DEFW    CASTO,CBSTO,PIOST,BATST   ; TTY,CRT,LPT,UL1
```

zu ändern und die entsprechenden eigenen Routinen einzufügen.

Hierbei bedeutet:	TCIN	CONSOLE INPUT
	TCOUT	CONSOLE OUTPUT
	TCST	CONSOLE STATUS
	TRDR	READER INPUT
	TPNCH	PUNCH OUTPUT
	TLIST	LIST OUTPUT
	TLST	LIST STATUS
und	CAIN	SIO A IN (7 Bit)
	CASTI	SIO A Receiver Status
	CAOUT	SIO A OUT (8 Bit)
	CASTO	SIO A Transmitter Status
	CBIN8	SIO B IN (8 Bit)
	PIO8	Centonics Out (8 bit)
	PIOST	Centronics Status

Beschreibung des Urladers

Der Urlader **BOOT65** wird in einem EPROM 2716 geliefert. Er gehört zum Lieferumfang von **ZDOS**. Das EPROM ist z.B. als IC 3 auf der CPU-Karte HKM-Z-105x einzusetzen. In einem Standardsystem ist der Urlader direkt lauffähig. Für abweichende Hardware sind im **BOOT641** Patch-Areas vorgesehen, um den Urlader adaptieren zu können.

Für den Einplatinencomputer und die MKC CPU II existieren spezielle Urlader.(Die Beschreibung stimmt überein; einige Adressen und Routinen sind verändert.)

BOOT65 bestimmt zuerst die Baudrate des an SIO Port B angeschlossenen Terminals. (kein Handshake!) Der Reihe nach wird auf 9600, 1200, 4800 und 2400 Baud verglichen. Hierzu ist am Terminal maximal 8 mal die Leertaste zu betätigen. Anschließend erscheint diese oder eine ähnliche Meldung:

BOOTSTRAPLOADER VERS. 6.5
COPYRIGHT (C) 1981, 82 BY H.K.M.
???? BAUD,

mit ???? = gefundene Baudrate des Terminals.
Jetzt folgt ein einfacher nicht zerstörender Speichertest. Ausgegeben wird der verfügbare RAM-Speicher in 16K-Seiten. Falls ein Speicherfehler gefunden wurde, wird die Seite, in der der Fehler auftrat, ausgegeben. Geprüft wird der Speicher ab Adresse 2000H oder 8000H (EPC-I). 48K MEMORY FAILED bedeutet, daß zwischen 32 und 48K ein Fehler gefunden wurde.
Jetzt wird die Betriebsart des Floppy-Controllers festgestellt. (Vorrang hat der DMA-Mode)
Meldung: DMA-MODE oder NON-DMA-MODE
Anschließend wird **ZDOS** von den ersten Spuren der Diskette in Laufwerk A an das obere Ende des lückenlosen Speichers geladen und anhand der **HEAS**-Sprungleiste eine evtl. notwendige Verschiebung berechnet und ausgeführt.
BOOT65 kann nur ein gültiges **ZDOS** laden, bei anderem Inhalt erfolgt die Meldung **NO SYSTEM. BOOT641** wartet auf die Diskette, d.h. es wird **kein** DISK NOT READY gemeldet.

Beschreibung der Patch-Area im **BOOT65** (vgl. Listing im Anhang)

Prinzipiell sind am Anfang des **BOOT65** ab Adresse 0020H 9 mal 3 Bytes leer (Inhalt OFFH). Hier können Sprünge zu eigenen Routinen des Users einprogrammiert werden. Für diese Routinen steht kein Stack zus Verfügung. Die Rückkehradresse wird deshalb in Register IX übergeben.

0020H	USER0	wird vor Ausführung von BOOT65 angesprungen. z.B. Speicher einschalten
0023H	USER1	Initialisierung der User-Konsole, umgeht die Baudratenbestimmung
0026H	USER2	User-Send-Message-Routine
0029H	USER3	Festlegung der höchsten verfügbaren Adresse im Speicher, umgeht den Speichertest
002CH	USER4	DMA oder NON-DMA-Mode festlegen
002FH	USER5	Disk-Timing festlegen (nur für NON-DMA-Mode)
0032H	USER6	Disk-Timing festlegen (nur für DMA-Mode)
0035H	USER7	wird am Ende von **BOOT65** vor dem Sprung in **HEAS** ausgeführt
0038H.	USER8	User-Konsoleingaberoutine

Die Adresse, ab der Benutzer-Routinen in das **BOOT63**-EPROM einprogrammiert werden dürfen, ist dem Listing im Anhang zu entnehmen. (Sie liegt ca. bei 6FEH)

Achtung

Bei Umgehung der Baudratenbestimmung fehlt dem System die Kenntnis des Systemtaktes. Dies ist in **HEAS** zu beachten. Das normale **HEAS** nimmt jetzt 2,5 MHz Systemtakt an. Bei 4 MHz Takt ist dies in **HEAS** entsprechend zu ändern. Eine andere Möglichkeit besteht darin, den Urlader die Zeitkonstante, die in den CTC geschrieben werden müsste, übergeben zu lassen (z.B. ODH für 9600 Baud bei 4 MHz Systemtakt).

Adressen im System (Hardware)

********** I/O-Map **********

Adresse	Baustein	Funktion	
00	uPD 765	Floppy-Status-Port	
01	uPD 765	Floppy-Daten-Port	
02	uPD 765	Floppy-Terminate	
03	Z80 DMA	Floppy-DMA-Port	
E0	Z80 SIO	Data SIO A	TTY
E1	Z80 SIO	Command SIO A	
E2	Z80 SIO	Data SIO B	CRT
E3	Z80 SIO	Command SIO B	
E4	Z80 PIO	PIO Data A	LPT
E5	Z80 PIO	PIO Command A	
E6	Z80 PIO	PIO Data B	
E7	Z80 PIO	PIO Command B	
E8	Z80 CTC	Kanal 0	Baudrate für SIO A
E9	Z80 CTC	Kanal 1	
EA	Z80 CTC	Kanal 2	Baudrate für SIO B
EB	Z80 CTC	Kanal 3	

Die MKC CPU-II und die MKC EPC-I Rechnerplatinen haben folgende abweichende Adressbelegung:

MKC EPC-I:

F0	uPD 765	Floppy-Status-Port
F2	uPD 765	Floppy-Status-Port
F1	uPD 765	Floppy-Daten-Port
F3	uPD 765	Floppy-Daten-Port
F4..F7	uPD 765	Floppy-Terminate
EC..EF		Boot-Reset

MKC CPU-II:

EC..EF	Paging-Registerfile

Alle anderen I/O-Adressen sind nicht belegt.

I/O-Initialisierung

SIO-Initialisierung

Beide Kanäle der SIO sind initialisiert für:
asynchrone Übrtragung
8 Bit, 1 Stop-Bit, kein Parity, kein Handshake,
* 16 Clock-Mode, DTR aus, RTS aktiv
Damit folgt:
WR1 = 0
WR2 = 0
WR3 = 0C1H
WR4 = 044H
WR5 = 06AH
WR6 nicht geladen
WR7 nicht geladen

CTC-Initialisierung

Die Kanäle 0 und 2 des CTC werden im Counter-Mode betrieben. (Mode-Control-Byte = 04DH), die Zeitkonstanten sind für Kanal 0 (9600 Baud) entweder 8 oder 13 bei 2,5 oder 4 MHz; und für Kanal 2 entsprechend der Terminal-Baudrate eingestellt.

PIO-Initialisierung

Die PIO ist initialisiert als CENTRONICS-Druckerschnittstelle. Kanal A ist das Status/Control-Port, Kanal B das Datenausgabeport. Folgende Belegung der PIO ist auf allen H.K.M. und MKC Karten einheitlich vorgesehen:

PIO Port A (Steuer-Port)	Bit 0	= Selected (Input)
	Bit 1	= Busy (Input)
	Bit 2	= Paper end (Input)
	Bit 3	= Error not (Input)
	Bit 4	= Strobe not (Output)
	Bit 5	= Select not (Output)
	Bit 6	= Autolf not (Output)
	Bit 7	= Init not (Output)
PIO Port B	Bit 0 bis 7	= Daten (Output)

Initialisierung:

Port B	0FH	Output Mode 0
Port A	CFH	Control Mode 3 mit
	0FH	Bit 4..7 = out
		Bit 0..3 = in

********** Memory-Map **********

0000 ... 0002	JP Warmstart
0003	**IOBYTE**
0004	Betriebslaufwerk des **KI**
0005 ... 0008	JP **LEAS** (**LEAS**-Service-Call)
005C ... 007F	File-Control-Block
0080 ... 00FF	**DMA**-Puffer
0100 ... E7FF	Programmbereich
D300 ... DFFF	**PSP**, falls geladen
E000 ... E7FF	**KI**
E800 ... F3FF	**LEAS**
E806	**LEAS**-Einsprungstelle
F400 ... FEFF	**HEAS**
F400	**HEAS**-Sprungleiste
FF00 ... FFFF	Vector-Interrupt-Page
FF00 ... FF7F	reserviert für **HEAS**
FF80 ... FFFF	frei für Anwenderprogramme

Die Adressen, die hier ab D300 angegeben sind, sind abhängig von der Ladeadresse des Betriebssystems. Sie verschieben sich dementsprechend, wenn das Betriebssystem für eine tiefere Adresse generiert wird.

Achtung

Die Lage der Vector-Interrupt-Page ist nicht absolut festgelegt. Sie befindet sich immer am oberen Ende des verfügbaren Schreib/Lesespeichers. Damit kann ihre Lage nur durch Auslesen des I-registers der Z80-CPU bestimmt werden. Die Vector-Interrupt-Page beginnt immer an einer Seitengrenze. Eine ZDOS-DMA-Version verlangt einen Vector-Interrupt, d.h. kein Programm darf den Interrupt "disablen". Ferner darf das I-Register nicht verändert werden. Die ZDOS-NON-DMA-Versionen schalten den Interrupt bei Floppyzugriffen aus, versetzen ihn aber anschließend wieder in den Ursprungszustand.