ZSDOS 1.1

Ein Ersatz des CP/M 2.2 BDOS

Programmer's Manual

by

Harold F. Bower
Cameron W. Cotrill
Carson Wilson

Ein Red Nil Design *Dokument*

Inhaltsverzeichnis

1 Einleitung

ZSDOS ist ein mächtiger Ersatz für das Basic Disk Operating System (BDOS) von CP/M 2.2- oder ZRDOS 1.x-Systemen. Es verfügt über verschiedene neue und zahlreiche verbesserte Funktionen, wobei jedoch die Kompatibilität zu bestehenden CP/M 2.2 Programmen weitestgehend erhalten bleibt. Der Erhalt dieser Kompatibilität war auch das Hauptziel bei der Entwicklung von ZSDOS. In einigen Fällen erschienen uns die Möglichkeiten durch Einbindung von Elementen aus CP/M Plus (auch als CP/M 3 bekannt) und von ZRDOS jedoch wichtiger als der hundertprozentige Erhalt der Kompatibilität. Desweiteren wurden erweiterbare Datenstrukturen entworfen und in ZSDOS integriert, um die Basis für ein mächtigeres Betriebssystem zu schaffen.

Dieses Handbuch beschreibt ausführlich die Funktionen, Schnittstellen und Datenstrukturen von ZSDOS. Besonders die neuen Merkmale von ZSDOS stehen dabei im Vordergrund, aber auch die weniger bekannten oder häufig fehlverstandenen Eigenschaften von CP/M. Kurze Beispiele in Z80 Assemblersprache sollen den Gebrauch der ZSDOS Funktionsrufe verdeutlichen. In den Anhängen finden Sie Kurzübersichten der Funktionen von ZSDOS.

Dieses Programmer‘s Manual wurde nicht als vollständige Dokumentation des CP/M-Betriebssystems oder der Z80 Assemblersprache konzipiert. Es wird eine gewisse Kenntnis der Konventionen bei der Benutzung von CP/M-Funktionsrufen vorausgesetzt. Für umfassendere Informationen über CP/M und die Z80 Assemblersprache empfehlen wir Ihnen die in der Bibliographie des ***ZSDOS User‘s Guide*** aufgelisteten Werke. Sofern nicht gesondert auf ein spezielles DOS hingewiesen wird, gelten die über ZSDOS gemachten Aussagen in diesem Handbuch ebenso für ZDDOS.

1.1 Komponenten des Betriebssystems

Das Betriebssystem CP/M 2.2 (bzw. kompatible) besteht aus drei separaten Segmenten; dem Basic Input/Output System (BIOS), dem Basic Disk Operating System (BDOS) und dem Console Command Processor (CCP). Jedes dieser Segmente ist relativ unabhängig von den anderen und kann recht einfach ersetzt werden. Voraussetzung dafür ist allerdings die Einhaltung festgelegter Schnittstellenparameter.

Für jeden Computer muß das BIOS in irgendeiner Form existieren. Es erledigt alle hardwarenahen Aufgaben. Alle angeschlossenen Geräte werden ebenso vom BIOS gesteuert wie die interne Hardware. Aufgrund der sehr unterschiedlichen Hardware verschiedener Computertypen ist auch das BIOS sehr verschieden. Im Normalfall wurde das BIOS vom Computerhersteller geschrieben. Es gibt aber auch einige recht verbreitete BIOSe von Drittanbietern für spezielle Computer. ZSDOS wurde so gestaltet, daß es auf nahezu jedem Computer läuft, dessen BIOS zu CP/M 2.2 oder ZRDOS 1.x kompatibel ist.

Das dritte Systemsegment, der CCP, stellt die primäre Schnittstelle zum Benutzer des Computers dar. Diese Komponente ist wahrscheinlich die am häufigsten ersetzte. Vor allem die ZCPR-Familie erfreut sich größter Beliebtheit. Da Veränderungen am CCP für den Benutzer am deutlichsten zu spüren sind, halten ihn viele für den wichtigsten Teil des Betriebssystems. In diesem Handbuch wollen wir jedoch zeigen, daß die Eigenschaften des BDOS darüber entscheiden, wie flexibel und leistungsfähig der Computer letztendlich wird.

Vom BDOS werden alle logischen Ein- und Ausgaben des Systems kontrolliert. Es verwaltet alle Resourcen in Form von logischen Geräten, wie z. B. Konsole, Drucker und Diskettenlaufwerk. Durch das BDOS erhalten die reinen Hardwaretreiber des BIOS eine festgelegt Struktur, wodurch ein einheitliches Erscheinungsbild unterschiedlichster CP/M Computer gegenüber den Anwendungsprogrammen entsteht. Diese strikte Trennung hardwareunabhängiger Routinen von hardwareabhängigen war einer der bedeutendsten Fortschritte, die CP/M für die Entwicklung von Betriebssystemen für Mikrocomputer brachte.

ZSDOS ist ein vollständiger Ersatz des BDOS Segmentes und Gegenstand dieses Handbuches.

1.2 Speicherbelegung

Die Aufteilung des Arbeitsspeichers unter ZSDOS ist identisch zu CP/M 2.2- und ZRDOS 1.x-Systemen. Beginnend bei der absoluten Adresse 0 befinden sich die reservierten 256 Bytes (0 - 0FFH) der Systempage. Von Adresse 0100H bis zum unteren Ende des niedrigsten Systemsegments befindet sich der TPA-Bereich (Transient Program Area). Dort werden alle Anwendungsprogramme (z. B. Textverarbeitung, Datenbanken, Assembler usw.) ausgeführt. Die untere Adresse des niedrigsten Systemsegments ist durch ZSDOS nicht festgelegt.

1.2.1 Speicherbereich der Segmente

Als Systemsegmente werden alle Programm(teil)e bezeichnet, die nach einem Warmstart im System enthalten sind bzw. bleiben. Klassische Vertreter für derartige Segmente sind zunächst einmal BIOS und BDOS, die für Anwendungsprogramme unentbehrlich sind. Der CCP kann von Programmen überschrieben werden und wird jeweils bei der Rückkehr zum Betriebssystem nachgeladen.

Einige CP/M 2.2 BIOSe laden nach einem Warmstart sowohl den CCP als auch das BDOS nach. Diese Eigenart ist historisch bedingt und geht auf eine Aussage von Digital Research zurück, wonach Anwendungsprogramme im Bedarfsfall auch das BDOS überschreiben können. Die Praxiserfahrung zeigt aber, daß kein (uns bekanntes) Programm das BDOS überschreibt und die meisten modernen BIOSe das BDOS nicht nachladen. Für einige Funktionen von ZSDOS ("Nur-Lesen-Vektor erhalten" und "schnelles Wiedereinloggen")

muß das BDOS resident vorhanden sein, anderenfalls haben sie keine Wirkung. (Konfigurationsdaten werden überschrieben)

Außer den eben erwähnten Segmenten können noch weitere Systemsegmente in einem ZSDOS System vorhanden sein. Im Gegensatz zu den bereits genannten sind diese zusätzlichen Segmente nicht zum Betrieb eines ZSDOS Systems erforderlich. Es können jedoch zusätzliche Funktionen durch derartige Segmente zur Verfügung gestellt werden.

Beispiele für diese Segmente, die man auch als Resident System Extension (RSX) bezeichnet, sind: BackGrounder ii und DosDisk, ZCPR3 Systemkomponenten (wie ENV, RCP, IOP, NDR und FCP), Utilities wie DateStamper und nicht zuletzt die ZSDOS-Erweiterungen zur Unterstützung von Datumsstempeln für Dateien.

RSXe befinden sich normalerweise direkt unterhalb des CCP im Speicher. Alle anderen Systemerweiterungen sind im Normalfall oberhalb des BIOS angesiedelt. Zusammenfassend ist die Speicherbelegung in folgender Übersicht dargestellt:

Wie dieser Darstellung zu entnehmen ist, sind nur die Adressen der Systempage und der Beginn des TPA-Bereiches in einem ZSDOS System genau festgelegt. Alle anderen Systemadressen sind abhängig vom BIOS. Die Größen der Systempage, des ZSDOS und des CCP (außer in erweiterten Z-Systemen) sind genau definiert. Andere Systemsegmente können den eigenen Bedürfnissen angepaßt sein.

1.2.2 Systempage

Die Systempage (Adreßbereich von 0 - 0FFH) wird von ZSDOS für wichtige Systeminformationen genutzt. Für ZSDOS gelten dieselben Festlegungen für die Systempage wie unter CP/M 2.2. Die Systempage bildet die Schnittstelle zu ZSDOS, weshalb das Verstehen der Funktion jedes einzelnen Bereiches äußerst wichtig ist.

00H - 02H Sprung zur BIOS-Warmstartroutine (BIOS+03H)

Diese Adresse darf von keinem Programm verändert werden. Sie bietet die einzige Möglichkeit, die Basisadresse des ZSDOS Systemsegmentes mit Sicherheit festzustellen. Ein Beispiel für die "richtige" Benutzung des Warmboot Vektors folgt später in diesem Handbuch. Wenn Programme die BIOS Sprungvektoren verändern wollen, dann dürfen nur die Werte der BIOS Sprungtabelle angepaßt werden, nicht das Sprungziel der Adresse 0.

Einzig Alpha System‘s NZCOM verändert den BIOS Warmboot Vektor in akzeptabler Weise. Es wird ein BIOS einschließlich Sprungtabelle "imitiert", das sich weiter unten im Speicher befindet. Die Systemsegmente des ZCPR werden zwischen dem echten System-BIOS und dem "Imitat" von NZCOM geladen. Der BIOS Warmboot Vektor zeigt auf das "imitierte" BIOS. Dadurch laufen alle Programme, bis auf systemspezifische Utilities, weiterhin fehlerfrei. Wenn man ein BIOS schreibt oder überarbeitet, sollte man bei den Systemutilities die Funktionsweise von NZCOM beachten.

03H IOBYTE

Das IOBYTE enthält eine BIOS-abhängige Struktur. Es kann vom Programmierer des BIOS verwendet werden, um eine Umlenkung byteorientierter Ein-/Ausgaben zu ermöglichen. Im Byte selbst sind 4 Felder enthalten, die für die logischen Geräte Konsole (CON:), Leser (RDR:), Stanzer (PUN:) und Listgerät (LST:) stehen. Jedes logische Gerät kann einem von bis zu vier verschiedenen physikalischen Geräten zugeordnet werden.

Weil die Einbindung des IOBYTE optional und systemabhängig ist, schlagen Sie bitte in den Handbüchern Ihres Computers nach, um die genauen Festlegungen für Ihr System zu ermitteln.

04H aktuelles CCP Standardlaufwerk und -nutzerbereich

Im Byte dieser Speicherstelle legt der CCP die Werte für das aktuelle Standardlaufwerk und den aktuellen Nutzerbereich ab. Der Wert für das Laufwerk wird in den Bits 0 bis 3 gespeichert, beginnend bei 0 für das Laufwerk A. In den Bits 4 bis 7 wird der Nutzerbereich mit Modulo 16 gespeichert. Dabei ist zu beachten, daß nur auf die Nutzerbereiche 0 bis 15 direkt zugegriffen werden kann. Da im Directory für den Nutzerbereich fünf Bits zur Verfügung stehen, können Dateien bzw. Programme auch in den Nutzerbereichen 16 bis 31 gespeichert werden. Ab Version 3.3 des ZCPR ist das Einloggen in die Nutzerbereiche 16 bis 31 auf Kommandoebene optional möglich. Durch den zusätzlichen CCP Code bleibt (bis auf wenige Ausnahmefälle) die Nummer des hohen Nutzerbereiches erhalten.

05H - 07H Sprung zum BDOS

Ein Ruf auf die Adresse 5 wird verwendet, um eine Funktion von ZSDOS auszuführen. Der Wert auf Adresse 6 kann jedoch nicht als direkter Zeiger auf ZSDOS verwendet werden!

Wird von einem Programm die Größe des verfügbaren TPA-Bereiches benötigt, dann kann die auf den Adressen 6 und 7 gespeicherte Sprungadresse zur Berechnung verwendet werden. Das höherwertige Byte des Wertes auf Adresse 6 zeigt stets auf die letzte Page des TPA-Bereiches. Je nachdem, ob eine RSX geladen ist oder nicht, befindet sich auf der nächsten Page das ZSDOS Segment.

08H - 2FH frei für Systemerweiterungen

30H - 37H reserviert

38H - 3FH Restart-Vektor 38H

Im Normalfall wird diese Adresse von Debuggern für die Speicherung der Breakpoint-Routine benutzt. Während der Testphase wird vom Debugger somit nur ein Byte (Opcode: RST 38H, Wert: 0FFH) für den Eintrag des Breakpoints benötigt.

40H - 4FH frei für Systemerweiterungen

Auf einigen Computern werden Teile dieses Speicherbereiches für Systemfunktionen benutzt. So wird z. B. beim Ampro Little Board in den Bytes 40H bis 4DH der ZCPR3 Pfad abgelegt.

50H - 5BH frei für Programme

Verschiedene Modem/BBS Programme aus dem Public Domain Bereich speichern in diesem Bereich Parameter wie z. B. die Baudrate ab.

5CH - 6BH standardmäßiger Dateisteuerblock (FCB) 1

6CH - 7BH standardmäßiger Dateisteuerblock (FCB) 2

In den Dateisteuerblöcken werden durch den CCP die ersten beiden Parameter der Kommandozeile abgelegt. Bei Benutzung des ersten Dateisteuerblocks zum Öffnen einer Datei wird der Inhalt des zweiten Dateisteuerblocks überschrieben. Beim Öffnen einer Datei mit dem zweiten Dateisteuerblock wird ein Teil des DMA-Puffers überschrieben. Die 16 Bytes des zweiten Dateisteuerblocks sollten vor der Benutzung von 6CH auf einen anderen Speicherbereich kopiert und erst dort verwendet werden. Somit kann der Dateisteuerblock auf Adresse 5CH unverändert benutzt werden. Nach dem Öffnen der Datei umfaßt der vollständige Dateisteuerblock 36 Bytes (von 5CH bis 7FH).

80H - 0FFH standardmäßiger DMA Puffer / Kommandozeile

Dieser Speicherbereich wird für zwei Aufgaben verwendet. Wenn ein Programm durch den CCP gestartet wird, legt dieser hier alle Argumente der Kommandozeile zur weiteren Verwendung durch das Programm ab. Das erste Byte des Puffers (80H) enthält die Anzahl der gültigen Zeichen. Anschließend folgt der Rest der Kommandozeile selbst, beginnend mit dem Leerzeichen (auf 81H), welches die Argumente vom Kommando trennt.

Außerdem wird dieser Bereich für den standardmäßigen DMA Puffer benutzt. Solange die Adresse des DMA Puffers nicht über die BDOS Funktion 26 geändert wurde, erfolgen alle Datenübertragungen zwischen Speicher und Diskette über diese Speicherbereich. Die Kommandoparameter müssen vom Programm also ausgewertet werden, bevor der Puffer durch Diskettenoperationen überschrieben wird.

2 BDOS Funktionen

Das BDOS arbeitet nicht nur mit Diskettenlaufwerken zusammen, sondern auch mit anderen Ein- und Ausgabegeräten, wie z. B. Drucker, Konsole, Uhr, Modem usw. Disketteneingaben/-ausgaben werden mit Datenblöcken durchgeführt, während bei fast allen anderen Geräten nur einzelne Bytes übertragen werden. Darüberhinaus gibt es eine Reihe von BDOS Rufen zum Lesen oder Ändern von Datenstrukturen und zur Manipulation des Dateisystems. Wie man sieht, hat das BDOS etwas mehr zu tun, als nur das Disketten- bzw. Dateisystem zu verwalten.

2.1 Zeicheneingabe/-ausgabe

Bei Zeicheneingaben/-ausgaben wird jeweils nur ein Zeichen bzw. Byte übertragen. Diese Art der Übertragung findet normalerweise für das Terminal, den Drucker oder Geräte zur seriellen Kommunikation (Modem) Anwendung. Das BDOS unterstützt vier logische Geräte für die Zeicheneingabe/-ausgabe unvollständig:

- Konsole (Eingabe/Ausgabe)
- Zusatzeingabe [Leser]
- Zusatzausgabe [Stanzer]
- Listausgabe [Drucker]

Mit "unvollständig" meinen wir in diesem Zusammenhang, daß der Status (bereit/nicht bereit) nicht für alle logsischen Geräte verfügbar ist. Weil dies bei der Entwicklung der originalen CP/M Systemsegmente BDOS und BIOS übersehen wurde, gestaltet sich die Programmierung spezieller Anwendungen erheblich schwieriger. So lassen sich z. B. Kommunikationsprogramme nur mit großem Aufwand auf andere Computertypen portieren.

In ZSDOS stehen für die Zeicheneingabe/-ausgabe neun Funktionen zur Verfügung. Während die ersten sechs nur eine direkte Schnittstelle zu den entsprechenden BIOS Funktionen darstellen, führen die letzten drei rudimentäre Prozesse aus. Die neun Funktionen sind:

- ein Zeichen von der Konsole lesen
- ein Zeichen auf der Konsole ausgeben
- ein Zeichen von der Zusatzeingabe [Leser] lesen
- ein Zeichen auf die Zusatzausgabe [Stanzer] ausgeben
- ein Zeichen auf das Listgerät [Drucker] ausgeben
- Eingabestatus der Konsole abfragen
- direkte Konsoleneingabe/-ausgabe
- eine Zeichenkette auf der Konsole ausgeben
- Konsolenpuffer lesen (Zeichenkette von der Konsole holen, mit Editierfunktionen)

2.2 Diskettеneingabe/-ausgabe

Die Diskettenoperationen des BDOS speichern die Daten als logische Datenblöcke, die zu Gruppen zusammengefaßt eine Datei ergeben. Die Datei wird in einem von bis zu 32 Nutzerbereichen eines logischen Laufwerks gespeichert. In einem CP/M System können bis zu 16 logische Laufwerke vorhanden sein.

Von ZSDOS werden die Datenblöcke als 128 Bytes große logische Records verwaltet, wie sie vom BIOS zur Verfügung gestellt werden. Dabei nimmt ZSDOS die erforderliche Umrechnung zur Bestimmung der physikalischen Spur- und Sektornummern vor. Auf diese Weise bleibt die physikalische Sektorgröße, die im BIOS verarbeitet wird, den Anwendungsprogrammen verborgen. Egal, ob die physikalische Sektorgröße 128, 256, 512, 1024 oder 2048 Bytes beträgt, stellt das BDOS eine einheitliche Schnittstelle zur Verfügung. Folgende Dateioperationen bietet das BDOS:

- Datei suchen
- Datei umbenennen
- Datei löschen
- Datei erzeugen
- Datei öffnen
- zu einer bestimmten Stelle innerhalb einer Datei springen
- Record einer Datei lesen
- Record in eine Datei schreiben
- Datei schließen
- Größe einer Datei ermitteln

Einige dieser Funktionen können auf geschlossene Dateien angewendet werden. Dazu gehören: Suchen, Umbenennen, Löschen, Erzeugen, Öffnen und Dateigröße ermitteln. Alle anderen Funktionen können nur auf geöffnete Dateien angewendet werden.

2.3 Kontrolle und Status

Die letzte große Kategorie der BDOS Funktionen beinhaltet eine Reihe von Kommandos zur Kontrolle und Statusbeeinflussung des Systems. Dazu gehören zum einen Funktionen, die Adressen von Datenstrukturen liefern. Zum anderen aber auch zur Festlegung der Funktionen angeschlossener Ein-/Ausgabegeräte.

Durch die Erweiterungen in ZSDOS stehen in dieser Kategorie viele neue Kommandos (z. B. für Fehlermodus, Echtzeituhr und Datumsstempel) zur Verfügung. Um das System noch flexibler zu gestalten, können einige Funktionen durch die Verwendung neuer Kontrollstrukturen im laufenden System aktiviert oder deaktiviert werden.

Grundsätzlich können die Funktionen dieser Kategorie folgendermaßen unterteilt werden:

- Rückgabe von Datenstrukturen
  (12 Kommandos)
- Festlegen von Kontrollelementen
  (7 Kommandos)
- Systemkontrolle
  (6 Kommandos)
- Schnittstelle zur Echtzeituhr
  (2 Kommandos)
- Zeit- und Datumsstempel für Dateien
  (2 Kommandos)

3 ZSDOS Datenstrukturen

3.1 Allgemein

Zum Datenaustausch zwischen ZSDOS und Anwendungsprogrammen werden verschiedene Datenstrukturen benutzt. Wenn das Programm Informationen ***an ZSDOS*** übergibt, so wird eine ZSDOS Funktion aufgerufen und ein Wert oder der Zeiger auf eine Datenstruktur im Register E übergeben, sofern es sich um ein Byte große Werte handelt. Für 16 Bit Werte wird das Registerpaar DE benutzt.

Die Übergabe mehrerer Werte an ZSDOS gestaltet sich nur wenig schwieriger. Im Normalfall sind dazu keine besonderen Vorkehrungen notwenig. Werden Zeiger auf bestimmte Datenstrukturen in wiederholten Rufen übergeben, dann sollte die Datenstruktur zwischen den Rufen nicht verschoben werden, da die Adresse der Struktur jedes Mal an ZSDOS übergeben wird. Ein Beispiel: Wird eine Datei mit dem Dateisteuerblock (FCB) auf einer bestimmten Adresse geöffnet, sollte der FCB nicht auf eine andere Adresse verschoben werden. Beachtet man dies nicht, kann das zu Problemen mit DateStamper, BGii oder anderen Programmen führen.

Werden Informationen ***von ZSDOS*** an das Programm zurückgegeben, stehen diese in Registern (byte- oder wortweise), im vorher definierten Pufferbereich, im aktuellen DMA-Puffer oder als Zeiger auf den aktuellen Datenbereich zur Verfügung. Soweit wie möglich werden gleiche Strukturen für die Übergabe der Informationen an und von ZSDOS benutzt. So entspricht z. B. der Dateisteuerblock (FCB) weitestgehend der Datenstruktur, mit der die Directory Informationen auf Diskette gespeichert werden. Sollen Eintragungen im Directory vorgenommen werden, so werden zuvor die Felder des FCB vom Anwendungsprogramm initialisiert, die den Directory Feldern entsprechen.

3.2 Logischer Record

Die grundlegendste Struktur, die von ZSDOS verwendet wird, ist der ***logische Record***. Er umfaßt 128 Bytes Daten/Informationen, die von Diskette gelesen oder darauf geschrieben werden. Bitte beachten Sie, daß der Ausdruck "logischer Record" nicht mit dem Begriff "Sektor" verwechselt werden darf. Im Normalfall beinhaltet ein Sektor der Diskette mehrere logische Records.

Mit den Diskettenfunktionen "Lesen" oder "ersten/nächsten Eintrag suchen" werden von ZSDOS die Informationen jeweils eines logischen Records in den aktuellen DMA Puffer gelesen. Analog dazu mit den Schreibfunktionen von dort auf Diskette geschrieben. Die Basisadresse des DMA Puffers wird über den Funktionsruf 26 festgelegt, wobei im Registerpaar DE die gewünschte Adresse übergeben wird. Mit Hilfe der Funktion 47 kann die aktuelle DMA Adresse abgefragt werden, die im Registerpaar HL zurückgegeben wird.

3.3 Dateisteuerblock (FCB)

Eine weitere grundlegende Struktur ist der ***Dateisteuerblock*** (nachfolgend ***FCB*** genannt), der zur Übergabe von Informationen *an ZSDOS* für die meisten dateibezogenen Funktionen dient. Der FCB ist ein 36 Bytes langer Datenbereich, der für die Dateimanipulation benötigte Informationen enthält. Der FCB ist wie folgt aufgebaut:

FCB + 00H	Laufwerksnummer (0 = Standardlaufwerk, 1...16 = A - P)
FCB + 01H	Dateiname in ASCII-Großbuchstaben [8 Bytes]
FCB + 09H	Dateityp in ASCII-Großbuchstaben [3 Bytes]
FCB + 0CH	Nummer des logischen Extents
FCB + 0DH	Nutzerbereich (bekannt als S1 Byte)
FCB + 0EH	Nummer des Datenmodules (S2 Byte)
FCB + 0FH	Recordzähler
FCB + 10H	16 Byte Belegungsvektor für diesen Extent
FCB + 20H	aktueller Record
FCB + 21H	Nummer für wahlfreien Record LSB
FCB + 22H	Nummer für wahlfreien Record ISB
FCB + 23H	Nummer für wahlfreien Record MSB

Die höchstwertigen Bits im ***Dateinamen*** und im ***Dateityp*** werden zur Speicherung der Attribute benutzt. Derartige Attribute kennzeichnen eine Datei als schreibgeschützt oder archiviert. Für weitere Informationen über die Verwendung von Dateiattributen schlagen Sie bitte bei Funktion 30 im Abschnitt 5.2.30 nach.

Die ***Nummer des logischen Extents*** wird von ZSDOS verwendet, um Dateien verarbeiten zu können, die größer als 128 logische Records (16 kB) sind. Für den ersten logischen Extent wird diese Nummer auf 0 gesetzt und jedes Mal für weitere 128 Records um eins erhöht.

Die ***Datenmodulnummer*** wird von ZSDOS verwendet, um Dateien verarbeiten zu können, die größer als 32 logische Extents (512 kB) sind. Sie ist anfangs ebenfalls auf 0 gesetzt und wird für jedes Mal für weitere 64 logische Extents um eins erhöht.

Normalerweise werden diese Felder des FCB von den Anwendungsprogrammen auf Null gesetzt, in bestimmten Fällen jedoch auf andere Werte. Die Nummer des logischen Extents kann Werte zwischen 0 und 31 annehmen. Für die Nummer des Datenmodules sind Werte zwischen 0 und 63 zulässig. Genauere Informationen können Sie dem Abschnitt 5.2.17 (Funktionsbeschreibung "ersten Eintrag suchen") dieses Handbuches entnehmen.

Im Byte des ***Recordzählers*** wird die Nummer des letzten verwendeten Records im aktuellen logischen Extent gespeichert. Der Belegungsvektor gibt die Anzahl der von der Datei belegten Blöcke wieder. Von keinem dieser Felder werden Daten an ZSDOS übergeben.

Im Byte ***aktueller Record*** wird die Nummer des nächsten Records im aktuellen Extent gespeichert, auf den von den Funktionen "sequentiell lesen" oder "sequentiell schreiben" zugegriffen wird. Im Normalfall wird dieses Byte beim Suchen oder Öffnen von Dateien vom Anwender auf Null gesetzt. Es sollte zwischen den sequentiellen Schreib- oder Lesezugriffen nicht verändert werden.

Das drei Byte große Feld mit der ***Nummer für den wahlfreien Record*** wird von den Funktionen "wahlfrei lesen" und "wahlfrei schreiben" benutzt. In den 24 Bits kann eine Zahl zwischen 0 und 262.143 stehen. Daraus resultiert die maximale Größe einer Datei von 262.144 Records.

Es gibt einen sehr wichtigen Unterschied zwischen ZSDOS und anderen DOS Segmenten für CP/M 2.2. Von ZSDOS werden neue Extents nur geöffnet oder erstellt, wenn sie benötigt werden und nicht wenn der letzte Record eines Extents gelesen oder beschrieben wird. Dies bedeutet, daß nach dem sequentiellen Lesen oder Schreiben des letzten Records eines Extents die Bytes des aktuellen Records und des Recordzählers auf 80H gesetzt sind. Einige ältere Anwendungsprogramme kommen mit dieser Funktionsweise des DOS nicht zurecht und laufen demzufolge nicht unter ZSDOS. Weil CP/M Plus mit den Extents genauso umgeht wie ZSDOS, laufen diese Programme auch nicht unter CP/M Plus.

Es ist äußerst wichtig, daß man sich darüber im Klaren ist, welche Bedeutung der FCB für das DOS bei Dateioperationen hat. Alle Statusinformationen werden dort gespeichert. Versucht ein Programm den FCB einer geöffneten Datei zu verändern, wird die Zerstörung des gesamten DOS Status riskiert! Dies trifft insbesondere dann zu, wenn DosDisk läuft und der FCB im MS-DOS Format vorliegt. Es wird daher ausdrücklich empfohlen, daß Anwendungen die Felder des FCB einer geöffneten Datei *unter keinen Umständen verändern* sollten. Einzige Ausnahmen bilden die Felder des aktuellen Record und der Nummer für den wahlfreien Record.

3.4 Directoryrecord

Der ***Directoryrecord*** ist eine Datenstruktur, die vom BDOS auf Diskette geschrieben wird. Darin sind Informationen über die aktuelle Belegung der Dateien enthalten. Jeder Directoryrecord ist 128 Bytes lang (ein logischer Record) und beinhaltet normalerweise vier Directoryeinträge. Für jede Datei auf der Diskette ist mindestens ein Directoryeintrag vorhanden. Ist eine Datei so groß, daß mehrere Directoryeinträge für sie benötigt werden, so bezeichnet man einen derartigen Eintrag als "physikalischen Extent". Jeder physikalische Extent hat eine eigene Nummer, so daß ZSDOS die Dateizugriffe korrekt durchführen kann.

Jeder ***Directoryeintrag*** ist 32 Bytes lang und ist ähnlich dem Dateisteuerblock aufgebaut. Der wesentlichste Unterschied besteht darin, daß im Directroyeintrag die Felder für den aktuellen Record und für den wahlfreien Record nicht vorhanden sind. Außerdem wird im ersten Byte des Directoryeintrags nicht der

Laufwerkscode sondern der Nutzerbereich der Datei gespeichert bzw. der Wert 0E5H, wenn die Datei gelöscht wurde.

Directoryeinträge sind folgendermaßen aufgebaut:

`DIR + 0`	Nutzerbereich der Datei (0..31)
`DIR + 1..8`	Dateiname in ASCII-Großbuchstaben [8 Bytes]
`DIR + 9..11`	Dateityp in ASCII-Großbuchstaben [3 Bytes]
`DIR + 12`	Nummer des logischen Extents
`DIR + 13`	S1 Systembyte (auf 0 gesetzt)
`DIR + 14`	Nummer des Datenmodules
`DIR + 15`	Recordzähler
`DIR + 16..31`	Belegungsvektor für diesen physikalischen Extent

Ebenso wie beim FCB werden auch hier die höchstwertigen Bits des Dateinamens bzw. -typs zur Speicherung der Attribute benutzt.

3.5 Diskettenbelegungsvektor

Der ***Belegungsvektor*** ist eine Struktur des BDOS, die im BIOS eingebettet ist. Für jedes logische Laufwerk im System ist ein Belegungsvektor vorhanden. Es handelt sich dabei um ein Bit-Abbild aller Blöcke der Diskette. Ist ein Bit auf 1 gesetzt, so bedeutet dies, daß der zugeordnete Block benutzt wird (belegt ist). Dementsprechend zeigt ein auf 0 zurückgesetztes Bit an, daß der Block verfügbar ist. Die Mindestgröße des Belegungsvektors in Bytes läßt sich für ein Laufwerk folgendermaßen berechnen: Anzahl der Blöcke /8 +1.

Für ein Anwendungsprogramm ist der Zugriff auf diese Struktur sehr ungewöhnlich (abgesehen von einigen Directoryprogrammen, die davon den verfügbaren Diskettenspeicherplatz ableiten). Anwendungsprogramme dürfen den Belegungsvektor niemals direkt verändern. Die Funktion 27 gibt die Adresse des Belegungsvektors für das aktuelle Standardlaufwerk im Registerpaar HL zurück. Wir empfehlen jedoch, daß auf den Vektor nicht zugegriffen werden sollte, da dies in zukünftigen Systemen unter Umständen nicht mehr möglich sein wird. Dies ist z. B. bereits bei CP/M Plus der Fall und wird eventuell auch in zukünftigen Versionen von ZSDOS so sein.

3.6 Diskettenparameterblock

Der ***Diskettenparameterblock*** (DPB) ist eine BIOS Struktur, die das Format eines logischen Laufwerks für ZSDOS definiert. Benötigt ein Programm Informationen aus dem DPB, so muß es direkt auf ihn zugreifen. Der DPB ist wie folgt aufgebaut:

`DPB + 0,1`	Anzahl der 128 Byte Records pro Spur
`DPB + 2`	Blockverschiebungsfaktor
`DPB + 3`	Blockmaske
`DPB + 4`	Extentmaske
`DPB + 5,6`	Anzahl der maximal verfügbaren Blöcke
`DPB + 7,8`	Anzahl der Directoryeinträge -1
`DPB + 9,10`	Bit-Abbild der Directory- und reservierten Blöcke
`DPB + 11,12`	Größe des Directoryprüfpuffers
`DPB + 13,14`	Anzahl der Systemspuren

Eine ausführliche Beschreibung der einzelnen Felder des DPB würde den Rahmen dieses Handbuches sprengen. Daher empfehlen wir Ihnen das sehr gute Buch "The Programmer‘s CP/M Handbook" von Andy Johnson-Laird (siehe *Bibliographie im ZSDOS User‘s Guide*). Von der Funktion 31 wird im Registerpaar HL die Adresse des DPB für das aktuelle Standardlaufwerk zurückgegeben.

3.7 Datumsangaben

Sind die Treiberroutinen zur Unterstützung von Datumsstempeln installiert, dann werden von ZSDOS zwei weitere Strukturen unterstützt. Die erste ist die ***Datumsangabe***, welche zum Austausch von Zeit- und Datumsinformationen zwischen Anwendungsprogrammen und ZSDOS verwendet wird. Die Datumsangabe basiert auf einer Reihe von gepackten BCD-Zahlen und ist folgendermaßen aufgebaut:

`TIME + 0`	letzte 2 Stellen des Jahres (von 78 bis 99 wird für das Jahrhundert 19 vorangestellt, sonst 20)	
`TIME + 1`	Monat	[1..12]
`TIME + 2`	Tag	[1..31]
`TIME + 3`	Stunde	[0..23]
`TIME + 4`	Minute	[0..59]
`TIME + 5`	Sekunde	[0..59]

Wer mit DateStamper vertraut ist, wird dieses Format sofort wiedererkennen. Die einzige Abweichung vom DateStamper Format ist das vorangestellte Jahrhundert. DateStamper nimmt für das Jahrhundert stets "19" an. Für weitere Informationen über das DateStamper Format sollten Sie im *DateStamper Handbuch* ab Seite A-19 nachschlagen.

Von ZSDOS wird bei Aufruf der Funktion 98 die aktuelle Zeit im Puffer zurückgegeben, auf dessen Anfangsadresse das Registerpaar DE zeigt. Durch Aufruf der Funktion 99 wird die Uhr auf die Werte im Puffer gesetzt, dessen Anfangsadresse im Registerpaar DE übergeben wird.

Auch die von DateStamper bekannte ***relative Uhr*** wird unterstützt. Dazu werden nur die Felder für Stunde und Minute benötigt, das Sekundenfeld wird auf Null gesetzt. Die relative Uhr ist nur ein einfacher binärer Zähler. Als niederwertiges Byte wird das Minutenfeld benutzt, während das Stundenfeld das

höherwertige Byte darstellt. Um Verwechslungen der relativen Uhr mit einer Echtzeituhr zu vermeiden, wird bei einer relativen Uhr das höchstwertige Bit (Bit 7) im Stundenfeld gesetzt.

3.8 Stempelformat

Das von ZSDOS benutzte ***Stempelformat für Dateien*** wurde ebenfalls von DateStamper abgeleitet. (Ehrlich Leute, wir wollten kein Plagiat von Bridger Mitchells Ideen produzieren. Wir haben jedes vorhandene Format genau untersucht, bevor wir uns für seine Methode entschieden haben, die für unsere Zwecke die besten Voraussetzungen bietet.) Unabhängig von der im System eingesetzten Stempelmethode - DateStamper oder P2DOS bzw. CP/M Plus Format - ist das intern verwendete Format stets gleich.

Das universelle Format besteht aus drei Feldern, die die Informationen über die Erstellung, den letzten Zugriff und die letzte Änderung enthalten. Dazu werden jeweils die ersten 5 Bytes der Datumsangabe verwendet. Wird ein Feld nicht oder nur teilweise unterstützt, dann werden die entsprechenden Bereiche im Feld beim Lesen auf 0 gesetzt und beim Schreiben ignoriert.

Alle Stempelinformationen werden im aktuellen DMA Puffer übergeben. Von der Funktion 102 werden die Informationen einer Datei zurückgegeben, dessen FCB Adresse im Registerpaar DE übergeben wurde. Mit der Funktion 103 werden die Stempelinformationen aus dem DMA Puffer auf eine Datei übertragen, deren FCB Adresse im Registerpaar DE übergeben wurde. Zur Zeit sind die Stempelinformationen nur 15 Bytes lang. Dies könnte sich jedoch in zukünftigen Versionen von ZSDOS ändern, so daß wir aus Kompatibilitätsgründen empfehlen, einen 128 Bytes großen Puffer im Anwendungsprogramm zu reservieren.

Stempelformat für Dateien (15 Bytes gepackte BCD-Zahlen):

`DMA + 0..4`	Feld für Erstellung	(erste 5 Bytes der Datumsangabe)
`DMA + 5..9`	Feld für Zugriff	(erste 5 Bytes der Datumsangabe)
`DMA + 10..14`	Feld für Änderung	(erste 5 Bytes der Datumsangabe)

4 ZSDOS Programmierkonventionen

4.1 Allgemein

ZSDOS ist kompatibel zu CP/M 2.2 und ZRDOS Anwendungen. Um jedoch eine solide Basis für zukünftige Erweiterungen von ZSDOS und kompatiblen BDOS Substituten zu schaffen, sollten die folgenden Praktiken während der Programmierung berücksichtigt werden.

Unter ZSDOS wird davon ausgegangen, daß die Systempage denselben Aufbau wie unter CP/M 2.2 oder ZRDOS hat - also ein Sprung zur Warmstartroutine des BIOS (Sprungziel: BIOS+3) auf Adresse 0000H und ein Sprung zum BDOS bzw. der untersten residenten Systemerweiterung (RSX) auf Adresse 0005H. Vom Sprungziel der Adresse 0005H kann keine korrekte Systemadresse abgeleitet werden, ausgenommen das Ende des TPA Bereiches. Die folgenden Quelltextzeilen sollen verdeutlichen, wie das obere Ende des TPA Bereiches mit Hilfe der Sprunganweisung auf Adresse 0005H berechnet werden kann:

```
GETSIZ: LD   HL,(0006H)  ; Endadresse des TPA holen
        DEC  HL          ; korrekte Adresse nun in HL
        ...
```

Oftmals wird diese Adresse von Anwendungsprogrammen zur Berechnung der Startadressen von BDOS und BIOS benutzt. Allerdings funktioniert diese Methode nicht, wenn residente Systemerweiterungen (RSXe) wie z. B. BGii, ZEX, DosDisk usw. installiert sind.

Zur korrekten Berechnung der Systemadressen muß das Sprungziel der Adresse 0000H verwendet werden. Der Zeiger dieser Adresse weist stets auf BIOS+3 und sollte ***niemals*** durch irgendein Programm verändert werden. Wenn Programme die BIOS Eintrittspunkte abfangen müssen, z. B. für Warmstart, Konsolenstatus usw., sollte die Sprungtabelle des BIOS verändert werden und nicht das Sprungziel auf Adresse 0000H. Entsprechend dem folgenden Beispiel kann der ZSDOS Eintrittspunkt korrekt berechnet werden:

```
FINDZS: LD   HL,(0001H)  ; BIOS Warmstartadresse holen
        LD   DE,-0DFDH   ; Beginn ZSDOS liegt soweit unterhalb
        ADD  HL,DE       ; HL zeigt auf ZSDOS Eintrittspunkt
```

Die Basisadresse von ZSDOS liegt, ebenso wie bei CP/M 2.2, ZRDOS bis Version 1.9 und den meisten ungebankten Systemen, 0DFDH unterhalb des Sprungziels auf Adresse 0000H. Auch die Startadresse des CCP kann derart berechnet werden. Dazu muß nur der Wert 1603H vom Sprungziel abgezogen werden. Der Beginn der BIOS Sprungtabelle (Kaltstart) ergibt sich, wenn man 3 vom Sprungziel abzieht.

Kombiniert man nun die "korrekte" Berechnungmethode des BDOS Eintrittspunktes und das Sprungziel auf Adresse 0005H, kann man leicht feststellen, ob eine RSX installiert ist.

```
FINDZS: LD   HL,(0001H)  ; BIOS Warmstartadresse holen
        LD   DE,-0DFDH   ; Beginn ZSDOS liegt soweit unterhalb
        ADD  HL,DE       ; HL zeigt auf ZSDOS Eintrittspunkt
        EX   DE,HL
        LD   HL,(0006H)  ; BDOS Sprungvektor holen
        AND  A
        SBC  HL,DE       ; feststellen, ob Adressen identisch
        JR   Z,NORSX     ; ja - also keine RSX vorhanden
        ...
```

In ZCPR Systemen mit erweiterter Umgebung, stehen die BIOS, BDOS und CCP Adressen im Environment zur Verfügung. Wenn eine erweiterte Umgebung vorhanden ist, müssen alle Systemadressen dem Environment entnommen werden. Grund dafür sind unter Umständen "nicht normale" Größen von BDOS und CCP derartiger Systeme.

ZSDOS Funktionen werden aufgerufen, indem die Nummer der ZSDOS Funktion im Register C und Werte im Register E oder Registerpaar DE abgelegt werden. Anschließend wird ein Ruf (CALL) auf Adresse 0005H ausgeführt. Von ZSDOS werden Werte im Register A und/oder im Registerpaar HL zurückgegeben. Alle Registerinhalte mit Ausnahme von IX, IY und den alternativen Registern können verändert sein.

Bei der Entwicklung von BIOSen, BIOS Erweiterungen oder IOPs mit Wiedereintritt ist zu bedenken, daß alle Register, die nicht im originalen 8080 Registersatz enthalten sind, zwischen den Rufen unbedingt gesichert und wiederhergestellt werden müssen. Den Anwendungsprogrammen "gehören" die Register AF‘, BC‘, DE‘, HL‘, IX und IY - nicht der Systemsoftware! Entsprechend den Konventionen unterstehen hingegen die Register I und R dem BIOS. Bei neuen Prozessoren wie 64180 und Z280 gehören alle neuen Register (mit Ausnahme des Z280 SSP) dem BIOS, da sie hardwarespezifisch und in direktem Bezug zu Ein- und Ausgaben stehen. Der Z280 SSP sollte für das BDOS reserviert bleiben.

Viele Programmierer haben bei Dateioperationen Techniken verwendet, die mit weiterentwickelten Betriebssystemen zu Problemen führen. Davon ist z. B. das Suchen von Dateien betroffen, was bei aktiviertem Pfad unter bestimmten Umständen zu scheinbaren Fehlfunktionen mit ZSDOS führen kann. Der Test auf das Vorhandensein einer Datei sollte mit der dazu vorgesehenen BDOS Funktion 17 erfolgen und nicht auf der Auswertung des zurückgegebenen Codes der Öffnungsfunktion beruhen. Beim Öffnen sucht ZSDOS nämlich entlang des Pfades nach der angegebenen Datei, bei der Suche nach dem ersten Eintrag (Funktion 17) dagegen nicht.

Außerdem müssen Programmierer beachten, daß mit ZSDOS wesentlich größere Dateien als unter CP/M 2.2 oder ZRDOS erzeugt werden können. Während die letztgenannten Systeme lediglich Dateigrößen bis zu 65.536 logische

Records (8.192 kB) unterstützen, können Dateien unter ZSDOS bis zu 262.144 logische Records (32.768 kB) groß sein. Derartige Dateien können von CP/M 3.0 ebenfalls verarbeitet werden.

4.2 Wiedereintritt in ZSDOS Rufe

Wiedereintretende ZSDOS Funktionsrufe im Sinne der ZRDOS 1.x Festlegungen werden von ZSDOS vollständig unterstützt. Diese Eigenschaft wird im allgemeinen nur von ZCPR Input/Output Paketen (IOPs) genutzt.

Ein ***wiedereintretender ZSDOS Funktionsruf*** wird ausgelöst, wenn ZSDOS eine BIOS Funktion aufruft, die durch ein IOP abgefangen wird, welches wiederum eine ZSDOS Funktion aufruft. Ein Beispiel dafür wäre ein Druckerspooler (ein Programm, das die Druckerausgabe in eine Datei umleitet). ZSDOS würde die BIOS Funktion für die Listausgabe aufrufen. Das IOP fängt diesen Funktionsruf ab und ruft dann ZSDOS Funktionen auf, um die abgefangenen Daten in eine Datei zu schreiben.

Zwar ist die Möglichkeit des Wiedereintritts ein sehr mächtiges Werkzeug, aber es kann damit im System auch verherender Schaden bisher unbekannten Ausmaßes angerichtet werden! Alle wichtigen internen Statusinformationen von ZSDOS (einschließlich der Adresse des DMA Puffers) müssen vor jedem wiedereintretenden Funktionsruf gesichert und danach wiederhergestellt werden. Außerdem müssen einige BIOS Informationen ebenfalls gespeichert oder reinitialisiert werden. Eine Grundregel lautet: Es darf keine BDOS Diskettenfunktion unterbrochen werden.

Um ein Maximum an Kompatibilität zu bestehenden Programmen für ZRDOS Plus zu bieten, wurden die Datenbereiche von ZSDOS im unteren Teil des Systemsegmentes angesiedelt. Die verwendeten Adressen und die Methode des Wiedereintritts sind kompatibel zu bereits existierenden IOPs für ZRDOS Plus. Hier eine Gegenüberstellung der ZSDOS Parameter mit den Anforderungen von ZRDOS Plus:

	Beginn	***Länge***
ZSDOS	Basis+3	146 (92H) Bytes
ZRDOS Plus	Basis+5	147 (93H) Bytes

Weil der Datenbereich von ZSDOS kleiner ist, entsteht keinerlei Schaden, wenn die durch ZRDOS Plus erforderlichen 147 Bytes gesichert werden. Die unterschiedlichen Startadressen liegen darin begründet, daß sich in ZSDOS eine interne Fehlervektortabelle auf einem CP/M 2.2 kompatiblen Offset befindet. Auf den betroffenen Bytes liegt die Adresse der BAD SECTOR Fehlerroutine. Der Anwender sollte sich vergewissern, daß keine Routine beim ZSDOS Wiedereintritt mit ZRDOS Parametern den BAD SECTOR Fehlervektor verändert. Anderenfalls könnten merkwürdige Dinge geschehen.

Wie bereits gezeigt, kann die Basisadresse von ZSDOS mit Hilfe des BIOS Warmstartvektors auf den Adressen 0001 und 0002 leicht berechnet werden. Vom Vektor muß nur 0DFDH abgezogen werden, um die "Basis" zu erhalten.

Eine letzte Vorsichtsmaßnahme muß bei Nutzung der Möglichkeiten des Wiedereintritts ergriffen werden, für den Fall, daß Traps in den BIOS Sprungvektoren verwendet werden, um wiedereintretende ZSDOS Rufe einzuleiten: Der Anwender muß sicherstellen, daß alle Register einschließlich dem IX Register zwischen den wiedereintretenden Rufen gesichert werden!

Beispiel:

Zur Einbindung eines wiedereintretenden Funktionsrufes muß zuerst die Adresse des DMA Puffers vom DOS abgefragt und in Ihrem Programm gesichert werden. Anschließend muß der Datenbereich des DOS ebenfalls in Ihrem Programm gesichert werden. Ab diesem Zeitpunkt können alle Funktionsrufe ohne Rücksicht auf den früheren Zustand des DOS ausgeführt werden. Wurde Ihre Routine ausgeführt, sollten Sie das DOS wiederherstellen, indem die eingangs aufgeführten Schritte umgekehrt werden; zunächst wird der gesicherte Datenbereich auf seine ursprüngliche Position kopiert und dann die Adresse des DMA Puffers mittels Funktion 26 auf den alten Wert gesetzt. Die Adresse muß mit der DOS Funktion 26 wiederhergestellt werden, um die DMA Adresse des BIOS mit der im ZSDOS Datenbereich abzugleichen, falls sie durch Ihr Programm verändert wurde. Ein Beispiel für eine solche Codesequenz:

```
LD   C,47         ; aktuelle DMA Adresse holen
CALL 5            ; ...Ruf über den DOS Eintrittspunkt
LD   (DMASAV),HL  ; DMA Adresse lokal sichern
CALL FINDZS       ; Basisadresse von ZSDOS finden
                  ; ...(siehe Abschnitt 4.1)
LD   DE,9-6       ; Offset zum Beginn des Datenbereiches
ADD  HL,DE        ; ...vom Beginn des DOS
LD   DE,SAVAREA   ; Zeiger auf lokalen Sicherungsbereich
LD   BC,147       ; ...und den ganzen Bereich
LDIR              ; ...mittels Blockbefehl kopieren
...               ; hier steht Ihre Routine
CALL FINDZS       ; Basisadresse von ZSDOS erneut finden
LD   DE,9-6       ; Offset zum Beginn des Datenbereiches
ADD  HL,DE        ; ...vom Beginn des DOS
EX   DE,HL        ; in das Register für‘s Ziel
LD   HL,SAVAREA   ; Zeiger für Quelle auf
                  ; ...auf lokalen Sicherungsbereich
LD   BC,147       ; den ganzen Bereich
LDIR              ; ...mittels Blockbefehl kopieren
LD   DE,(DMASAV)  ; gesicherte DMA Adresse holen
LD   C,26         ; ...und mit Funktion 26
CALL 5            ; ...über DOS setzen (auch im BIOS)
...               ; weiter geht‘s...
```

4.3 ZSDOS Konfigurationsbereich

Der Konfigurationsbereich von ZSDOS befindet sich ab Adresse ZSDOS Basis+3. In diesem Bereich sind verschiedene Vektortabellen, die Adressen des Pfades und des Wheel Bytes sowie das Konfigurationsbyte der Flags

gespeichert. Für alle Versionen 1.x von ZSDOS und ZDDOS wird die gleiche Belegung beibehalten. In zukünftigen Veröffentlichungen können jedoch geänderte Offsets zum Einsatz kommen.

4.3.1 Fehlervektortabelle

Ab Adresse Basis+3 befindet sich die ***Fehlervektortabelle***. Dabei handelt es sich um eine CP/M 2.2 kompatible Struktur, die von einigen Programmen (z. B. Sektorenprüfprogramme) benutzt wird, um die BDOS Fehler abzufangen. Sie wurde nur wegen der Kompatibilität zu bestehenden Programmen in ZSDOS integriert. ZSDOS Anwendungen sollten die neuen BDOS Fehlermodi benutzen, um Fehler abzufangen.

4.3.2 Pfadadresse (nur ZSDOS)

Die Basisadresse des ***Suchpfades*** ist auf Adresse Basis+11 gespeichert. Beim Öffnen von Dateien verwendet ZSDOS den Suchpfad, wenn der Wert auf dieser Adresse ungleich Null und Bit 5 des Konfigurationsbytes auf 1 gesetzt ist.

Beachten Sie, daß bei Verwendung eines Kommandosuchpfades durch den CCP (wie bei ZCPR oder BGii) sooft entlang des DOS Pfades gesucht wird, wie Einträge im Kommandosuchpfad des CCP vorhanden sind. Dies verlangsamt selbstverständlich die Arbeit. Zukünftige Versionen dieser CCP Ersatzsysteme sollten überprüfen, ob ein ZSDOS Pfad vorhanden ist, um eine der folgenden Vorgehensweisen einzuleiten. Wird festgestellt, daß ein DOS Pfad vorhanden und aktiviert ist, dann wird dieser anstelle des CCP Suchpfades verwendet. Die zweite Möglichkeit wäre das Deaktivieren des DOS Pfades über Bit 5 des Flagbytes während der Suche entlang des CCP Kommandosuchpfades.

4.3.3 Adresse des Wheel Bytes

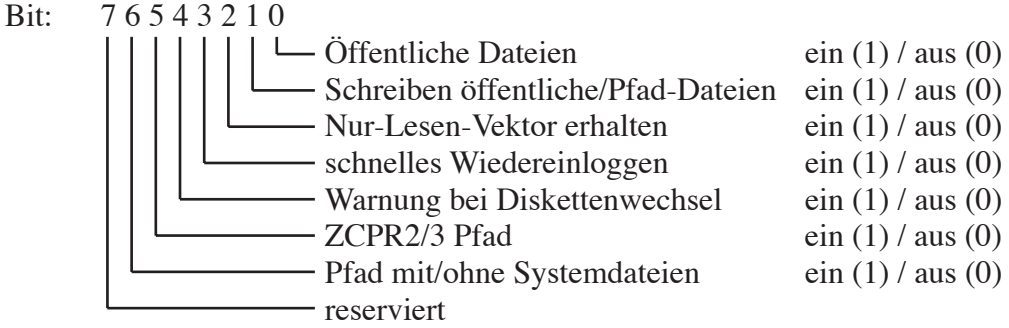
Auf Basis+13 wird die Adresse des ***Wheel Bytes*** gespeichert. Das Wheel Byte ist ein ZCPR Kontrollelement, mit dessen Hilfe die Sicherheitsfunktionen des Systems erweitert werden können. Ist das Wheel Byte aus (Wert gleich 0), dann schützt ZSDOS alle Dateien mit gesetztem Wheel-Schutz Attribut (f8) vor dem Überschreiben, Löschen und Umbenennen. Ist im BDOS die Adresse des Wheel Bytes auf 0 gesetzt (Zeiger auf den Sprung zum Warmstart), dann geht ZSDOS davon aus, daß der Anwender alle Nutzungsrechte besitzt und erlaubt ihm uneingeschränkten Zugriff auf die Dateien.

Beachten Sie bitte, daß das Wheel Byte ein Element von ZCPR3 ist und jede willkürlich festgelegte Adresse für den unabhängigen DOS Kontrollmechanismus festgelegt werden kann. Wenn Sie eine der ZCPR Versionen 3.x als Ersatz für den CCP benutzen, werden Sie sicherlich das gleiche Wheel Byte für beide Systemsegmente verwenden. Wir möchten hier nur darauf hinweisen, daß für spezielle Installationen unterschiedliche Speicherzellen möglich sind.

4.3.4 Konfigurationsbyte

Viele Eigenschaften von ZSDOS können während der Laufzeit durch Veränderung des ***Konfigurationsbytes*** kontrolliert werden. Das Byte befindet sich auf Adresse Basis+15. Von ZSDOS werden nicht alle Flagbits benutzt. Werden unbenutzte Bits gesetzt oder zurückgesetzt, hat dies keinen Einfluß auf ZSDOS.

Um Kompatibilität zu späteren Versionen von ZSDOS zu gewährleisten, sollten nur die Funktionen 100 (Hole Flags) und 101 (Setze Flags) benutzt werden, um auf die ZSDOS Flags zuzugreifen. Die Bedeutung der Flagbits im Konfigurationsbyte ist wie folgt definiert:



Bit 0 kontrolliert, ob ZSDOS Dateien mit gesetztem Attribut "öffentliche Datei" (f2) in anderen Nutzerbereichen der gleichen Diskette findet. Ist das Bit auf 1 gesetzt, dann werden derartige Dateien gefunden, wenn ein eindeutiger Dateiname angegeben wurde.

Bit 1 entscheidet darüber, ob in Dateien geschrieben werden darf, die über das Attribut öffentlich oder den Pfad (nur ZSDOS) gefunden wurden. Ist die Funktion eingeschaltet (Bit 1 gleich 1), kann in diese Dateien geschrieben werden. Anderenfalls (Bit 1 gleich 0) ist das Schreiben in Dateien, die mit Hilfe des Attributes öffentlich oder den Pfad gefunden wurden, nicht möglich.

Bit 2 legt fest, wann der Schreibschutzvektor in ZSDOS gelöscht wird. Ist das Bit gesetzt, dann wird der Vektor niemals gelöscht (solange ZSDOS nicht erneut von Diskette geladen wird). Bei zurückgesetztem Bit, wird das Schreibschutzbit für ein Laufwerk gelöscht, wenn es mit Funktion 13 oder 37 erneut eingeloggt wird.

Bit 3 veranlaßt ZSDOS, den Belegungsvektor einer "festen" Diskette (Festplatte oder RAM Disk) nicht neu zu erstellen, wenn das Laufwerk mit der Funktion 13 ausgeloggt wurde. Durch das Setzen dieses Bits wird die Arbeit mit Festplatten erheblich beschleunigt. Feste Disketten können jederzeit über die Funktion 37 wieder eingeloggt werden.

Bit 4 schaltet die Meldung von ZSDOS beim Wechsel von Disketten ein oder aus. Bei gesetztem Bit gibt ZSDOS nach jedem Wechsel einer Diskette eine Meldung auf dem Bildschirm aus. Diese Meldung wird selbstverständlich

unterdrückt, wenn der BDOS Fehlermodus dementsprechend eingestellt ist; unabhängig vom Zustand dieses Bits.

Bit 5 gibt die Benutzung des DOS Pfades beim Öffnen von Dateien frei, wenn es auf 1 gesetzt ist (nur ZSDOS). Wenn dieses Bit gesetzt und auf der Adresse für den Pfad ein Wert ungleich Null eingetragen ist, dann wird die angegebene Datei mit eindeutigem Namen von ZSDOS unter Verwendung des Pfades gefunden. Für ZDDOS hat dieses Bit keine Bedeutung.

Bit 6 legt den Pfadzugriff fest, wenn der Pfad aktiviert ist (nur ZSDOS). Wird das Bit 6 auf 1 gesetzt, dann werden alle Dateien in Verzeichnissen entlang des Pfades unabhängig vom Systemattribut gefunden (Pfad-Verzeichniszugriff). Schaltet man diese Funktion aus (Bit 6 gleich 0), dann werden in den Verzeichnissen entlang des Pfades nur Dateien gefunden, bei denen das Systemattribut gesetzt ist (Pfad-Dateizugriff). Für ZDDOS hat dieses Bit keine Bedeutung.

4.3.5 Datumsvektoren

In ZSDOS ist eine Vektortabelle integriert, die es Treibern für Datumsstempel erlaubt, sich selbst in ZSDOS einzubinden. Die Tabelle umfaßt 6 Einträge und beginnt bei Adresse Basis+16. Ein weiterer Dummy-Eintrag wird nur dazu benutzt, die Adresse der Ausschaltfunktion in ZSDOS aufzunehmen.

Die speziellen Treiber für Datumsstempel installieren sich selbst in ZSDOS, indem sie die Adressen der unterstützten Funktionen in die Tabelle eintragen. Zur Ausführung der Datumsstempelfunktionen ruft ZSDOS die benötigten Routinen auf. Zuvor wird jedoch der Directorypuffer aktualisiert und überprüft, ob die Diskette Schreib-/Lesestatus hat.

Weil ZDDOS bereits DateStamper™ beinhaltet, sind für dieses DOS nur die Adressen für den Uhrentreiber, für das Entfernen und für den Dummy-Eintrag notwendig.

Die Struktur der ***Datumsvektortabelle***:

`Basis + 16`	Vektor der Routine zum Lesen/Stellen der Echtzeituhr
`Basis + 18`	Vektor der Routine für den Stempel des letzten Zugriffs
`Basis + 20`	Vektor der Routine für den Stempel der Erstellung
`Basis + 22`	Vektor der Routine für den Stempel der letzten Änderung
`Basis + 24`	Vektor der Routine zum Stempel holen
`Basis + 26`	Vektor der Routine zum Stempel setzen
`Basis + 28`	Vektor der Dummy-Routine
`Basis + 30`	Adresse der Routine zum Entfernen des Datumsstempels

4.4 Routinen zur Unterstützung von Zeit- und Datumsstempeln

In Bezug auf die Routinen zur Unterstützung von Zeit- und Datumsstempeln unterscheiden sich ZSDOS und ZDDOS erheblich. In ZDDOS ist bereits DateStamper™ integriert, so daß nur noch ein Uhrentreiber benötigt wird. ZSDOS beinhaltet keine Stempelroutine. Zum Betrieb werden sowohl ein externer Uhrentreiber als auch eine externe Stempelroutine benötigt.

Mit ZSDOS sind verschiedene Formen von Datumsstempeln möglich. Zu den unterstützten Methoden gehören Plu*Perfect‘s DateStamper- und P2DOS- (kompatibel mit CP/M Plus) Datumsstempel. Solange keine entsprechende Routine zur Unterstützung von Datumsstempeln im ZSDOS System installiert ist, können die Datumsstempel nicht aktiviert werden. Zum Betrieb von ZSDOS ist es nicht notwendig, derartige Routinen zu installieren. Sie werden nur benötigt, wenn Sie die Funktionen 98, 99, 102 und 103 nutzen möchten.

In Abhängigkeit von der gewünschten Stempelmethode werden unterschiedliche Routinen benötigt. Durch die integrierte DateStamper™ Unterstützung erlaubt ZDDOS nur die Nutzung dieser Methode. Mit ZSDOS können DateStamper-, P2DOS- oder beide Stempelarten benutzt werden. Treiber für weitere Stempelmethoden können programmiert und problemlos in ZSDOS integriert werden. Dazu stellt ZSDOS lediglich definierte Verbindungspunkte zur Verfügung, während die eigentliche Routine außerhalb des BDOS Segmentes angesiedelt ist, typischerweise oberhalb des BIOS. Dahinter steht dieselbe Philosophie wie bei ZCPR – optionale Teile des Systems werden in reservierte Pufferbereiche oberhalb des BIOS verlagert.

ZSDOS erledigt für die Routinen den größten Teil der Kleinarbeit. Der angeforderte FCB wird in den Directorypuffer kopiert, die Diskette wird erforderlichenfalls auf Schreib-/Lesestatus geprüft, der DMA Puffer und der Offset des Directorypuffers werden entsprechend der Methode bereitgestellt.

Durch die enge Verbindung des DOS mit den Routinen für Datumsstempel beträgt der durchschnittliche Speicherbedarf für DateStamper mit einem Echtzeituhrtreiber unter ZSDOS ca. 3/4 kByte - viel weniger als bei früheren DateStampern. Mit ZDDOS und dem integrierten DateStamper wird der Speicherbedarf noch weiter reduziert, da nur noch der Uhrentreiber fehlt. Dieser ist im Normalfall unter 400 Bytes groß. Steht im System bereits ein Uhrentreiber zur Verfügung, wird unter ZDDOS gar kein zusätzlicher Speicherplatz benötigt.

Mit ZSDOS werden spezielle Treiber für DateStamper™, P2DOS (CP/M Plus kompatibel) und zur Unterstützung beider Formate geliefert. Die Treiber für beide Methoden lesen jeweils ein Format und schreiben beide. Sie sind besonders für Anwender interessant, die ein Höchstmaß an Kompatibiltät zwischen CP/M Plus und ZSDOS Systemen benötigen.

5 ZSDOS Funktionsrufe

5.1 Beschreibung der zurückgegebenen Werte

Die Funktionen von ZSDOS geben Werte zurück, um den Erfolg oder aufgetretene Fehler bei der Ausführung der Funktion anzuzeigen. Es werden fünf Kategorien dieser Codes unterschieden – Verzeichniscodes, Fehlercodes, Zeit-/Datumscodes, Schreib-/Lesecodes und erweiterte Fehlercodes. Folgende Übersicht zeigt die zurückgegebenen Werte der Codes jeder Kategorie:

Verzeichniscode:

A = 00H, 01H, 02H, 03H, wenn kein Fehler aufgetreten ist
A = 0FFH, bei einem Fehler

Fehlercode:

A = 00H, kein Fehler
A = 0FFH, Fehler aufgetreten

Zeit-/Datumscode:

A = 01H, wenn kein Fehler aufgetreten ist
A = 0FFH, Fehler aufgetreten

Schreib-/Lesecode:

A = 00H, wenn kein Fehler aufgetreten ist
A = 01H, Lesen - Dateiende
Schreiben - Directory voll
A = 02H, Diskette voll
A = 03H, Fehler während des Schließens beim wahlfreien Lesen/Schreiben
A = 04H, leerer Record beim wahlfreien Lesen
A = 05H, Directory voll beim wahlfreien Schreiben
A = 06H, Nummer des wahlfreien Records während des wahlfreien Lesens/Schreibens zu groß

erweiterte Fehlercodes im Fehlermodus:

A = 0FFH, weitere Fehlercodes in H

H = 01H, Diskettenein-/-ausgabefehler (defekter Sektor)
H = 02H, Diskette schreibgeschützt (nur lesen)
H = 03H, Datei schreibgeschützt
H = 04H, unzulässiges Laufwerk ausgewählt

Den folgenden Funktionsbeschreibungen kann man entnehmen, welche Werte von jeder Funktion zurückgegeben werden. Einzige Ausnahme bilden hierbei die erweiterten Fehlercodes, die von jeder Funktion zurückgegeben werden, die Diskettenzugriffe ausführt. Diese erweiterten Codes werden allerdings nur dann zurückgegeben, wenn einer der beiden Modi zum Zurückgeben des Fehlercodes über die Funktion 41 eingestellt wurde.

Funktion 0 - Boot	
Eingabe: keine	Ausgabe: keine

Dieser Ruf, der nur selten benutzt wird, sorgt für eine klare Abgrenzung zu Anwendungsprogrammen. Wird diese Funktion aufgerufen, so führt ZSDOS intern einen RST 0 Befehl aus. Bei einem ROM-basierten ZSDOS System wird dadurch das RAM Datensegment initialisiert und mit den Standardwerten geladen.

Von den meisten Programmierern wird für das Beenden ihres Programmes ein RET Befehl (wenn der CCP nicht überschrieben wurde) oder ein Sprung auf Adresse 0 benutzt. Die Funktion 0 ist eine Einbahnstraße - sie führt nicht zum rufenden Programm zurück.

Das Ergebnis dieses Rufes entspricht dem Warmstart des Systems - alle Laufwerke mit austauschbaren Medien werden zurückgesetzt, die DMA Adresse wird auf 80H gesetzt, der CCP wird nachgeladen (sofern er nicht durch eine RSX geschützt ist) und die Kontrolle an ihn übergeben. Zusätzlich wird durch Aufruf dieser Funktion der Fehlermodus von ZSDOS auf den Standard zurückgesetzt.

```
DONE:   LD   C,0
        CALL BDOS        ; Einbahnstraße - nicht mehr zurück
```

Funktion 1 - Konsoleneingabe	
Eingabe: keine	Ausgabe: A = Zeichen

Diese Funktion gibt das nächste Zeichen von der Konsole zurück. Steht kein Zeichen bei Aufruf dieser Funktion bereit, so wartet ZSDOS auf eine Eingabe, bevor zum rufenden Programm zurückgekehrt wird. Das zurückgegebene Zeichen wird auf der Konsole ausgegeben, wobei Steuerzeichen gefiltert werden.

Wagenrücklauf (0DH), Zeilenvorschub (0AH) und Backspace (08H) werden unverändert wiedergegeben. Alle Tabstopps (09H) werden in die entsprechende Anzahl Leerzeichen umgewandelt, um den Cursor auf die nächste durch 8 teilbare Spalte zu positionieren. Alle anderen Steuerzeichen werden nicht auf der Konsole wiedergegeben.

Control-S wird durch diese Funktion abgefangen und wie folgt behandelt: Wurde ein Control-S erkannt, dann werden alle Ausgaben auf die Konsole angehalten, bis irgendein anderes Zeichen (außer Control-C) eingegeben wird. Danach wird die Konsolenausgabe fortgesetzt. Wird nach der Eingabe von Control-S ein Control-C erkannt, dann setzt ZSDOS den Fehlermodus in den Standardmodus zurück und führt anschließend einen Warmstart aus.

```
CONIN:  LD   C,1
        CALL BDOS       ; nächstes Zeichen von der Konsole
        ...             ; Zeichen ist nun in Register A
```

Funktion 2 - Konsolenausgabe	
Eingabe: E = Zeichen	Ausgabe: keine (A = BIOS A Register)

Das im Register E enthaltene Zeichen wird mit dieser Funktion auf das aktuelle Konsolengerät ausgegeben. Ebenso wie bei Funktion 1 werden auch hier alle Tabstopps in Leerzeichen umgewandelt, so daß der Cursor auf der nächsten durch 8 teilbaren Spalte positioniert wird.

Die Funktion zur Konsoleneingabe kontrolliert diese Funktion bei Auftreten eines Control-S. Wurde ein Control-S eingegeben, dann wird die Ausgabe auf die Konsole solange unterbunden, bis irgendein anderes Zeichen (außer Control-C) eingegeben wird. Wird nach der Eingabe von Control-S ein Control-C erkannt, dann setzt ZSDOS den Fehlermodus in den Standardmodus zurück und führt anschließend einen Warmstart aus.

Anmerkung: Diese Funktion sollte nicht zur Ausgabe von Videosteuerzeichen verwendet werden, da bestimmte Steuerzeichen gefiltert werden. Für diese Zwecke sollte die Funktion 6 benutzt werden.

```
CONOUT: LD   E,A        ; angenommen, das Zeichen ist in A
        LD   C,2        ; Funktion Konsolenausgabe wählen
        CALL BDOS       ; Zeichen an Konsole senden
        ...
```

Funktion 3 - Zusatzeingabe [Leser]	
Eingabe: keine	Ausgabe: A = Zeichen

Mit dieser Funktion wird das nächste Zeichen vom aktuellen Zusatzeingabegerät geholt. Steht kein Zeichen bei Aufruf dieser Funktion bereit, so wartet ZSDOS auf eine Eingabe, bevor zum rufenden Programm zurückgekehrt wird. Ist kein Gerät für die Zusatzeingabe definiert, so hängt der zurückgegebene Wert von der Dummy-Routine des BIOS ab. Steuerzeichen werden durch diesen Funktionsruf nicht gefiltert.

In früheren BDOS Systemen wie CP/M 2.2 war als Gerät der Zusatzeingabe der "Leser" definiert. Diese Bezeichnung stammt noch aus der Zeit, als Lochstreifen ein verbreitetes Medium zur Datenspeicherung waren.

Die Eingabe vom Zusatzgerät könnte etwa so erfolgen:

```
AUXIN:  LD   C,3
        CALL BDOS        ; nächstes Zeichen der Zusatzeingabe
        ...              ; Zeichen ist nun im Register A
```

Funktion 4 - Zusatzausgabe [Stanzer]	
Eingabe: E = Zeichen	Ausgabe: keine (A = BIOS A Register)

Die Zusatzausgabefunktion sendet das Zeichen im Register E zum derzeit aktuellen Gerät der Zusatzausgabe. Bevor das Zeichen gesendet wird, wartet die Funktion auf die Bereitschaft des Gerätes.

In älteren BDOS Systemen wird die Zusatzausgabe oft als "Stanzerausgabe" bezeichnet, weil damit der Lochstreifenstanzer angesteuert wurde.

Ein Zeichen kann wie folgt an das Zusatzausgabegerät gesendet werden:

```
AUXOUT: LD   E,A          ; angenommen, das Zeichen ist in A
        LD   C,4          ; Zusatzausgabe wählen
        CALL BDOS         ; ...und Zeichen senden
        ...
```

Funktion 5 - Listausgabe	
Eingabe: E = Zeichen	Ausgabe: keine (A = BIOS A Register)

An das aktuelle Listgerät (LST:) wird das in Register E enthaltene Zeichen übertragen. Die BIOS Funktion zur Listausgabe wartet auf die Bereitschaft des Gerätes, bevor das Zeichen übertragen und zum BDOS zurückgekehrt wird. Durch das BDOS wird vor dem Senden des Bytes nicht die BIOS Routine zur Abfrage des Listausgabestatus aufgerufen.

```
LIST:   LD   E,A        ; angenommen, das Zeichen ist in A
        LD   C,5        ; Listausgabe wählen
        CALL BDOS       ; ...und Zeichen senden
        ...
```

Funktion 6 - direkte Konsoleneingabe/-ausgabe	
Eingabe: E = 0FFH (Eingabe) E = 0FEH (Eingabe) E = 0FDH (Eingabe) E = 0..0FCH (Ausgabe)	Ausgabe: A = eingeg. Zeichen (00 = kein) A = Konsolenstatus (00 = kein) A = eingegebenes Zeichen keine (A = BIOS A Register)

Dieser Funktionsruf (manchmal auch verkürzt DCIO genannt) wird benutzt, um die normale gefilterte Ein- und Ausgabe des BDOS zu umgehen und direkt über die BIOS Routinen mit der Konsole zu kommunizieren. Im Normalfall werden mit dieser Funktion Videosteuersequenzen an das Terminal übertragen.

In ZSDOS wurden einige Mängel des CP/M 2.2 BDOS beseitigt, wo Aufrufe der Funktion 6 mit der normalen BDOS Konsoleneingabe der Funktion 1 vermischt wurden. Dies betrifft den internen Zeichenpuffer des DOS, wenn Control-S Zeichen zum Starten und Anhalten der Bildschirmausgabe verwendet werden. Bei jedem Aufruf der Funktion 6 zur Zeicheneingabe oder Statusabfrage überprüft ZSDOS den Zeichenpuffer, um stets korrekte Ergebnisse beim Lesen der Konsole zu liefern. An dieser Stelle möchten wir Bridger Mitchell danken, der uns auf diese Eigenheit von CP/M hinwies, so daß wir sie beseitigen konnten.

Mit dem Wert im Register E wird beim Funktionsruf 6 in CP/M 2.2 und ZSDOS Systemen die auszuführende Konsolenfunktion bestimmt. Alle zurückgegebenen Werte werden im Register A zur Verfügung gestellt. Für das Register E sind folgende Werte definiert:

0FFH nächstes Zeichen von der Konsole holen oder 0, wenn kein Zeichen verfügbar ist

0FEH Status der Konsole abfragen; 0 zeigt an, daß kein Zeichen bereitsteht

0FDH* auf das nächste Zeichen von der Konsole warten und dieses zurückgeben

0..0FCH* Ausgabe des Zeichens im Register E auf der Konsole

* = neue oder geänderte Funktionen in ZSDOS

Die hinzugefügte Funktion (0FDH) entspricht in ihrer Funktionsweise der von CP/M Plus und stellt eine komfortablere Routine für "nächstes Zeichen holen" zur Verfügung.

```
OLDCODE: LD   E,0FEH      ; Konsolenstatus abfragen
         LD   C,6
         CALL BDOS
         AND  A           ; irgendwas angekommen?
         JR   Z,OLDCODE   ; ...nein, wiederholen
         LD   E,0FFH
         LD   C,6
         CALL BDOS        ; wenn endlich bereit, dann abholen
         ...              ; Zeichen jetzt in A

; neue Methode mit ZSDOS...

NEWCODE: LD   E,0FDH      ; hole nächstes Zeichen, sobald vorh.
         LD   C,6
         CALL BDOS
         ...              ; Zeichen jetzt in A
```

Funktion 7 - IOBYTE holen	
Eingabe: keine	Ausgabe: A = IOBYTE (Beginn System-page+03H)

Diese Funktion gibt im Register A den Wert des aktuellen IOBYTE zurück. Das IOBYTE ist eine optionale und BIOS-abhängige Struktur. Es kann zur Umlenkung byteorientierter Ein-/Ausgaben der logischen Geräte CON:, RDR:, PUN: und LST: benutzt werden. Bitte schlagen Sie in den Handbüchern Ihres Computers nach, um genaue Festlegungen für Ihr System zu ermitteln.

Anmerkung: Dieser Funktionsruf steht in zukünftigen ZSDOS Versionen eventuell nicht mehr zur Verfügung.

```
GETIOB: LD   C,7          ; hole IOBYTE
        CALL BDOS         ; ...wird in A zurückgegeben
        ...
```

Funktion 8 - IOBYTE setzen	
Eingabe: E = IOBYTE	Ausgabe: keine (A = IOBYTE)

Mit dieser Funktion wird der Wert des IOBYTE auf den im Register E übergebenen Wert gesetzt. Das IOBYTE ist eine optionale und BIOS-abhängige Struktur. Es kann zur Umlenkung byteorientierter Ein-/Ausgaben der logischen Geräte CON:, RDR:, PUN: und LST: benutzt werden. Bitte schlagen Sie in den Handbüchern Ihres Computers nach, um genaue Festlegungen für Ihr System zu ermitteln.

Anmerkung: Dieser Funktionsruf steht in zukünftigen ZSDOS Versionen eventuell nicht mehr zur Verfügung.

```
SETIOB: LD   E,A        ; angenommen, IOBYTE steht in A
        LD   C,8
        CALL BDOS       ; setze IOBYTE
        ...
```

Funktion 9 - Zeichenkette ausgeben	
Eingabe: DE = Adresse der Zeichenkette, die mit '$' endet	Ausgabe: keine (A = '$')

Eine aus ASCII Zeichen bestehende Zeichenkette, die mit einem Dollarzeichen '$' endet, wird mit dieser Funktion auf der Konsole ausgegeben. Alle Zeichen der Kette mit Ausnahme des Dollarzeichens werden an die Konsole übertragen. Alle Tabstopps werden in eine entsprechende Anzahl Leerzeichen umgewandelt, um den Cursor auf der nächsten durch 8 teilbaren Spalte zu positionieren.

Während der Ausgabe der Zeichenkette wird die Konsole auf die Eingabe von Control-S überprüft. Die Ausgabe wird in diesem Fall bis zur Eingabe eines weiteren Zeichens angehalten und danach fortgesetzt. Folgt auf Control-S allerdings Control-C, dann setzt ZSDOS den Fehlermodus auf den Standard zurück und führt anschließend einen Warmstart aus. Der Anwender kann auf diese Weise fehlerhafte Programme beenden, ohne einen Kaltstart auszuführen.

```
STROUT: LD   DE,STRING   ; was Sie anzeigen möchten
        LD   C,9         ; Zeichenkettenausgabe wählen
        CALL BDOS
        ...
STRING: DEFB 'Dies ist eine Zeichenkette.$'
        ...
```

Funktion 10 - Konsolenpuffer lesen	
Eingabe: DE = Adresse Eingabepuffer	Ausgabe: keine (A = 0DH)

Über diese Funktion bekommt man eine Zeichenkette vom momentanen Eingabegerät der Konsole. Um diese Funktion benutzen zu können, muß vom Anwendungsprogramm zunächst der Zeiger auf einen Eingabepuffer übergeben werden. Dieser Puffer ist wie folgt konfiguriert:

`BUFF + 0`	Größe des Puffers für die maximale Anzahl einzulesender Zeichen (höchstens 255)
`BUFF + 1`	tatsächliche Anzahl eingelesener Zeichen (wird bei Rückkehr vom BDOS gesetzt)
`BUFF + 2` bis max. Länge+2	Zeichen von der Konsole

Zur Bearbeitung der Eingabzeile stehen in der Funktion 10 einige Steuerzeichen zur Verfügung:

^H	löscht Zeichen links vom Cursor
^J	beendet die Eingabe
^M	beendet die Eingabe
^X	löscht gesamte Zeile
^U	wie ^X
^R	schreibt aktuelle Zeile neu (nur ZSDOS)
DEL	wie ^H

Alle Tabstopps werden von der Funktion 10 (wie bei den Funktionen 1, 2 und 9) in Leerzeichen umgewandelt, allerdings ***nur für die Bildschirmausgabe*** - im Puffer bleibt ^I erhalten. ZSDOS merkt sich die Cursorposition, wenn diese Funktion aufgerufen wurde, so daß Tabs ordnungsgemäß expandiert werden (solange nicht mit Funktion 6 in derselben Zeile etwas ausgegeben wurde). Die nicht druckbaren Control-Zeichen (alle außer Tab und Editiersteuerzeichen) werden für die Bildschirmausgabe in zwei Zeichen umgesetzt. Das erste Zeichen ist ein Caret '^', auf das das Control-Zeichen + 40H folgt. Zum Beispiel würde Control-Z als '^Z' angezeigt werden.

Von der Funktion wird Control-P erkannt und das Druckerflag entsprechend umgeschaltet. Control-S (zum Anhalten der Konsolenausgabe) wird von dieser Funktion nicht erkannt.

Die Funktion 10 wird in folgenden Fällen beendet:

1. Enter (Carriage Return = Wagenrücklauf) oder Zeilenvorschub wurde gedrückt.

2. Der Eingabepuffer ist voll.

3. Ist das erste eingegebene Zeichen in der Zeile ein Control-C, dann wird das Programm abgebrochen und ein Warmstart des Systems ausgeführt. In diesem Fall bleibt das Control-C im Puffer erhalten, so daß der ZCPR Kommandoprozessor nicht durcheinander gerät.

Bei der Rückkehr zum aufrufenden Programm wird in BUFF + 1 die Anzahl der gelesenen Zeichen eingetragen. Die Zeichenkette selbst beginnt ab BUFF + 2. Beachten Sie dabei, daß Enter oder Zeilenvorschub zwar gelesen werden, aber nicht im Puffer erscheinen.

Ein Beispiel für die Benutzung der Funktion 10:

```
BUFFRD: LD   DE,BUFF     ; Zeiger auf den Textpuffer
        LD   C,10        ; Funktion Konsolenpuffer lesen
        CALL BDOS
        ...
                         ; Struktur insgesamt 128 Zeichen
BUFF:   DEFB 126         ; max. 126 Zeichen werden gelesen
        DEFB 0           ; tatsächliche Anzahl hier von ZSDOS
        DEFS 126         ; eigentlicher Pufferbereich
        ...
```

Funktion 11 - Konsolenstatus holen	
Eingabe: keine	Ausgabe: A = 0, kein Zeichen A = 1, Zeichen vorhanden

Diese Funktion wird benutzt, um das Konsolengerät abzufragen, ob ein Zeichen eingegeben wurde. Von ZSDOS wird im Register A der Wert 0 zurückgegeben, wenn kein Zeichen bereitsteht bzw. der Wert 1 im anderen Fall.

```
CONST:  LD   C,11        ; Konsolenstatus prüfen
        CALL BDOS        ; gibt Status in A zurück
        AND  A           ; etwas vorhanden?
        JR   Z,NOCHAR    ; ...Sprung, wenn kein Zeichen vorh.
        ...
```

Funktion 12 - CP/M Versionsnummer holen	
Eingabe: keine	Ausgabe: HL = 22H (CP/M-kompatibel)

Von dieser Funktion wird im Registerpaar HL der Wert 22H zurückgegeben, um die Kompatibilität zu CP/M 2.2 anzuzeigen. Um die Version von ZSDOS abzufragen, muß die Funktion 48 benutzt werden.

Im Falle von ZDDOS bzw. ZSDOS mit installiertem ZDS DateStamper gibt die Funktion 12 im Registerpaar DE die Adresse der DateStamper Uhrenroutine zurück, wenn beim Aufruf im Register E der Wert ‘D‘ (044H) übergeben wurde. Dann wird im Register H das ASCII-Zeichen ‘D‘ zurückgegeben. Durch diese Funktionsweise wird sichergestellt, daß für Plu*Perfect‘s DateStamper geschriebene Software auch unter ZSDOS läuft. Speziell an ZSDOS angepaßte Programme sollten jedoch die Funktionsrufe von ZSDOS für die Zugriffe auf Uhr und Datumsstempel anstelle der alten DateStamper Methode benutzen.

```
GETCPV: LD   C,12
        CALL BDOS        ; CP/M Versionsnummer holen
        ...
```

Funktion 13 - Diskettensystem zurücksetzen	
Eingabe: keine	Ausgabe: A = 0, keine Datei namens $*.* A = 0FFH, Datei namens $*.* vorhanden

Durch die Funktion 13 werden alle Laufwerke ausgeloggt und die Adresse des DMA Puffers auf 80H zurückgesetzt. Laufwerk A wird als Standardlaufwerk festgelegt. Der eingestellte Nutzerbereich wird nicht verändert. Vor dem Aufruf dieser Funktion sind alle Dateien zu schließen, in die Daten geschrieben wurden.

Eine undokumentierte Eigenschaft von CP/M 2.2 benutzt der CCP, um Submitdateien abzuarbeiten. Jedes Mal, wenn unter CP/M ein Laufwerk zurückgesetzt oder angewählt wird, enthält das Register A den Wert 0FFH, sofern eine Datei namens $*.* auf dem Laufwerk im aktuellen Nutzerbereich vorhanden ist. Dieser zurückgegebene Wert wird vom CCP benutzt. Er zeigt an, wo sich die Datei $$$.SUB befinden könnte. Der Kommandoprozessor bezieht dann die nächste Eingabezeile nicht vom Benutzer, sondern aus dieser Datei.

Etliche DOS Systeme, die das Einloggen von Festplatten überspringen, haben Schwierigkeiten, dieses Flag an den CCP korrekt zu übergeben. ZSDOS prüft bei jedem Einloggen sowie beim Erstellen und Löschen von Dateien das Vorhandensein einer Datei namens $*.*. Das Flag wird gesetzt, wenn eine Datei namens $*.* erzeugt oder beim Einloggen (in irgendeinen Nutzerbereich) entdeckt wird. Das Flag wird zurückgesetzt, wenn die Datei $*.* erfolgreich gelöscht wurde. Durch diese Verfahrensweise spiegelt das Submit Flag das Vorhandensein einer $*.* Datei im System stets korrekt wider - selbst wenn schnelles Wiedereinloggen aktiviert ist.

Da diese Methode nicht ganz CP/M 2.2 entspricht, kann der CCP nur solange richtig funktionieren, wie nicht mehrere Dateien namens $*.* im System existieren (eigentlich sehr unwahrscheinlich!).

Im Gegensatz zur Funktion 13, die alle Laufwerke ausloggt, arbeitet die Funktion 37 differenzierter. Damit werden nur ausgewählte Laufwerke zurückgesetzt. Programme sollten daher möglichst die Funktion 37 anstelle 13 benutzen. Weil gewechselte Disketten automatisch eingeloggt werden, ist es nur selten erforderlich, Laufwerke zurückzusetzen oder zwischen festen und wechselbaren Disketten zu unterscheiden, wenn man unter ZSDOS arbeitet.

```
RESSYS: LD   C,13
        CALL BDOS        ; Laufw. zurücksetzen, A: einloggen
        AND  A           ; Submitdatei vorhanden?
        JR   NZ,DOSUB    ; ...ja, also Datei öffnen
        ...
```

Funktion 14 - Laufwerk auswählen	
Eingabe: E = Nummer des Laufwerks (0 = A, 1 = B..)	Ausgabe: A = 0, keine Datei namens $*.* A = 0FFH, Datei namens $*.* vorhanden

Diese Funktion wird benutzt, um ein Standardlaufwerk auszuwählen. Auf die Diskette im Standardlaufwerk wird immer dann zugegriffen, wenn im FCB bei Dateizugriffen kein Laufwerk festgelegt ist. War das gewählte Laufwerk bisher noch nicht eingeloggt, dann geschieht dies ebenfalls durch die Funktion 14.

Ist einer der erweiterten Fehlermodi von ZSDOS aktiv, so zeigt der Wert 0FFH im Register nicht unbedingt einen Fehler an. Vielmehr muß der Inhalt des Registers H geprüft werden. Ist der Wert in H gleich Null, dann ist kein Fehler aufgetreten, sondern ZSDOS zeigt an, daß eine Submitdatei auf dem Laufwerk vorhanden sein könnte. Wenn der Wert im Register H nicht Null ist, dann gab es ein Problem bei der Auswahl des Laufwerkes (im allgemeinen heißt das: Laufwerk ist nicht vorhanden!).

Ist kein erweiterter Fehlermodus aktiv und wurde das angegebene Laufwerk nicht gefunden, so bricht ZSDOS das Anwendungsprogramm nach der Ausgabe einer entsprechenden Fehlermeldung ab.

Es wäre noch zu beachten, daß CP/M 2.2 und alle anderen kompatiblen BDOS Substitute (mit Ausnahme von ZRDOS 1.9) einen Fehler in dieser Routine haben. Wurde ein nicht vorhandenes Laufwerk ausgewählt, so geht das BDOS trotzdem davon aus, daß das unzulässige Laufwerk das Standardlaufwerk ist. In normalen CP/M 2.2 Systemen tritt dieser Fehler nicht in Erscheinung, da das BIOS den CCP nachlädt und der erste ausgeführte BDOS Ruf eine gesonderte Auswahl darstellt. NZCOM benutzt zum Nachladen des CCP das BDOS, wodurch der Fehler in Erscheinung tritt.

```
; bei diesem Quelltext wird davon ausgegangen, daß ein
; erweiterter Fehlermodus AKTIV ist (BDOS Fehler zurückgeben)

SELDK:  LD   E,A        ; angenommen, A enthält Laufwerksnr.
        LD   C,14
        CALL BDOS       ; Laufwerk auswählen
        AND  A
        RET  Z          ; zurück, wenn kein Fehler auftrat
        LD   A,H
        AND  A          ; Steht OFFH für Submitdatei?
        RET  Z          ; handelte sich um Submitdatei, Lw. OK
        ...             ; ansonsten Laufwerk unzulässig
```

Funktion 15 - Datei öffnen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Verzeichniscode

Verglichen mit anderen Z80 BDOS Systemen bietet ZSDOS eine stark erweiterte Funktion zum Öffnen von Dateien. Wie bereits erwähnt, kann ZSDOS das Attribut für öffentliche Dateien und den Pfad zum Dateiöffnen verwenden (sowohl für das Lesen als auch für das Schreiben). Wurde eine Datei erfolgreich geöffnet, so enthält die Speicherzelle FCB + 13 die Nummer des Nutzerbereichs, OR-verknüpft mit 80H:

```
FOPEN:  LD   A,(USER)
        LD   E,A          ; Nutzerbereich der Datei
        LD   C,32         ; Funktionsruf Nutzerbereich setzen
        CALL BDOS
        LD   DE,FCB
        LD   C,15
        CALL BDOS         ; Datei öffnen
        INC  A
        JP   Z,ERROR      ; Fehler beim Öffnen der Datei
        ...               ; FCB + 13 = Nutzerbereich
                          ; ...OR-verknüpft mit 80H
```

Im Gegensatz zu einigen anderen ZSDOS Funktionen wird von dieser nicht die Nummer des Nutzerbereichs auf FCB + 13 akzeptiert. Ursprünglich wurde von Digital Research das S1 Byte auf FCB + 13 als "für das System reserviert" deklariert. Allerdings wurde nicht festgelegt, ob das Byte für das Öffnen einer Datei auf einen bestimmten Wert gesetzt werden muß.

Weil viele Programme FCBs wiederbenutzen, kann dieses Feld eine gültige Nummer enthalten (und oftmals ist es auch so), allerdings für die ehemalige Datei! Nach vielen Experimenten haben wir entschieden, daß es der sicherste Weg sei, wenn ZSDOS beim Dateiöffnen den Wert auf FCB + 13 ignoriert, wenn der Fehlermodus des BDOS gleich Null ist. Dies ist die einzige zuverlässige Methode, die Nutzerbereichsnummer im FCB zu unterstützen und trotzdem abwärtskompatibel zu bleiben.

Programme, die für die Arbeit unter ZSDOS geschrieben werden, sollten alle Bytes von FCB + 0 bis FCB + 0EH initialisieren. Zusätzlich zu den Einträgen von Laufwerk und Dateiname muß der Wert des S1 Bytes auf Null oder die Nummer des Nutzerbereichs OR-verknüpft mit 80H gesetzt werden. Außerdem müssen die Bytes des aktuellen Extents (FCB + 12), des Datenmodules (FCB + 14) und des aktuellen Records (FCB + 32) auf Null gesetzt werden, es sei denn, die Datei soll nicht an ihrem Anfang geöffnet werden. Es ist möglich, eine Datei mit irgendeinem Extent des ersten Datenmodules (erste 512 kBytes) zu öffnen. Für das Datenmodul kann jedoch kein Wert ungleich Null angegeben werden.

Möchte man in einer Anwendung das S1 Byte zur Bestimmung des Nutzerbereiches beim Öffnen einer Datei (oder einer anderen dateibezogenen Funktion) benutzen, dann muß der BDOS Fehlermodus verwendet werden, um anzuzeigen, daß die Anwendung die Gegebenheiten von ZSDOS kennt. Erst dann kann die Nummer des Nutzerbereiches in FCB + 13 übergeben werden. Wurde der Fehlermodus auf einen Wert ungleich Null eingestellt, dann benutzt ZSDOS das Feld FCB + 13 (siehe Funktion 45).

Nachdem eine Datei mit der Funktion 15 erfolgreich geöffnet wurde, ist das S1 Byte entweder auf den Nutzerbereich OR-verknüpft mit 80H gesetzt oder unverändert, wenn der Fehlermodus gleich Null war. Die Datenmodulnummer wird stets auf Null gesetzt. Der Dateiname, der Recordzähler (FCB + 15) und der Belegungsvektor (FCB + 16..31) werden vom Directoryeintrag der entsprechenden Datei kopiert. Die Felder von Extent, aktuellem Record und wahlfreiem Record bleiben erhalten.

Für einige Anwendungen ist es von Bedeutung, ob die Datei über Pfad- oder öffentlichen Zugriff geöffnet wurde. ZSDOS liefert auch diese Information. Nach dem Öffnen wird das Attributbit f7 gesetzt, wenn der Pfad oder das Attribut öffentliche Datei verwendet wurden. Zur Verzweigung bei Pfad- oder öffentlichem Zugriff könnte der Code etwa so aussehen:

```
LD   C,15
CALL BDOS        ; Datei öffnen
INC  A
JP   Z,ERROR     ; ...Sprung bei aufgetretenem Fehler
LD   HL,FCB+7    ; Zeiger auf das f7 Bit
BIT  7,(HL)      ; Test auf Pfad-/öffentlichen Zugriff
JR   NZ,ISPS     ; ...Sprung, wenn benutzt
...
```

Eine schlechte Angewohnheit beim Programmieren ist es, den FCB einer bereits geöffneten Datei zu verlagern oder denselben FCB mehrfach für das Öffnen von weiteren Dateien zu benutzen, solange eine Datei vor dem Öffnen der nächsten nicht geschlossen wurde. Mangelhafte Programmierpraktiken können Probleme mit verschiedenen populären Betriebssystemerweiterungen wie BGii verursachen.

Funktion 16 - Datei schließen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Verzeichniscode

Durch diese Funktion werden alle internen Puffer auf Diskette übertragen und das Directory aktualisiert, wenn in eine Datei geschrieben wurde. Zu einem guten Programmierstil gehört aber auch das Schließen von Dateien mit der Funktion 16, wenn diese nur zum Lesen geöffnet waren. Nur so kann volle Kompatibilität zu zukünftigen Versionen von ZSDOS und anderen Betriebssystemerweiterungen garantiert werden.

```
FCLOSE: LD   DE,FCB      ; Zeiger auf die zu schließende Datei
        LD   C,16
        CALL BDOS        , Datei schließen
        INC  A           ; alles OK?
        JR   Z,ERROR     ; ...Sprung bei aufgetretenem Fehler
        ...
```

Funktion 17 - ersten Eintrag suchen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Verzeichniscode

Von dieser Funktion wird das erste Auftreten eines übereinstimmenden Directoryeintrages zurückgegeben. Die Übereinstimmung basiert sowohl auf den ersten 13 Bytes des FCB (Laufwerk, Name und Nummer des Extents) als auch auf der Nummer des Nutzerbereichs (entweder per Default oder über den Inhalt von FCB + 13) und der Nummer des Datenmoduls (FCB + 14), wenn dort ein ‘?‘ eingetragen wird. Die Suche wird stets am Anfang des Directorys begonnen. In den ersten 13 Bytes des FCB (FCB + 0..12) und in der Datenmodulnummer sind Jokerzeichen erlaubt.

Fragezeichen werden von den Funktionen "ersten Eintrag suchen" und "nächsten Eintrag suchen" auf zwei verschiedene Arten verwendet. Zum einen kann ein ‘?‘ in den Bytes FCB + 1 bis FCB + 14 eingesetzt werden, um eine Übereinstimmung mit jedem beliebigen Zeichen für diese Position zu erreichen. Ist z. B. im ersten Byte des Dateinamens (FCB + 1) ein Fragezeichen eingetragen, so werden alle Dateien ohne Berücksichtigung des ersten Zeichens gefunden (nur die anderen Zeichen des Dateinamens sind ausschlaggebend). Setzt man in den Bytes für Extent und Datenmodul Fragezeichen ein, so werden dementsprechend alle physikalischen Extents der Datei oder Dateien gefunden. Diese beiden Bytes werden z. B. von Programmen auf ‘?‘ gesetzt, die die Dateigröße durch Summierung aller Extents berechnen.

Für die zweite Art zur Verwendung von Fragezeichen im FCB wird das Byte für das Laufwerk (FCB + 0) mit einem ‘?‘ belegt. In diesem Fall werden aber nicht alle Laufwerke ausgewählt (wie man vielleicht meinen möchte), sondern alle Directoryeinträge des aktuellen Laufwerkes (auch bereits gelöschte Einträge).

Nach einem Aufruf der Funktion 17 wird der übereinstimmende Directoryrecord in den aktuellen DMA Puffer kopiert. Wird der im Register A zurückgegebene Verzeichniscode um 5 Stellen nach links verschoben und zur Basisadresse des DMA Puffers addiert, zeigt er auf das erste Byte des übereinstimmenden Directoryeintrages.

```
SEARCF: LD   DE,DMAADDR  ; DMA Adresse auf eigenen Puffer
        LD   C,26
        CALL BDOS
        LD   DE,FCB      ; diese Übereinstimmung suchen
        LD   C,17
        CALL BDOS        ; nach Übereinstimmungen suchen
        CP   0FFH        , hat‘s geklappt?
        JR   Z,NOMAT     ; ...Sprung bei keiner Übereinstimmung
        ADD  A,A
        ADD  A,A
        ADD  A,A
        ADD  A,A
        ADD  A,A         ; Index mit 32 multiplizieren
        LD   L,A
        LD   H,0         ; Wortwert daraus machen
        LD   DE,DMAADDR  ; Zeiger auf Beginn des Puffers
        ADD  HL,DE       ; HL zeigt auf übereinstimm. Eintrag
        ...
```

Funktion 18 - nächsten Eintrag suchen	
Eingabe: keine	Ausgabe: A = Verzeichniscode

Nach einer erfolgreichen Suche des ersten Eintrags (Funktion 17) wird diese Funktion dazu benutzt, weitere Übereinstimmungen für den angegebenen FCB zu suchen (je eine pro Aufruf). Für eine fehlerfreie Arbeitsweise dieser Funktion müssen zwei Bedingungen erfüllt sein: 1.) Die Funktion "ersten Eintrag suchen" muß für die erste Übereinstimmung ausgeführt worden sein. 2.) Zwischen den beiden Aufrufen zur Suche nach Übereinstimmungen dürfen keine anderen Diskettenoperationen des BDOS ausgeführt werden.

Bis auf zwei Ausnahmen entsprechen die Ruf- und Rückkehrsequenzen denen der Funktion 17. Die Funktion 18 benötigt keinen Zeiger auf den FCB, da er von der Suche nach dem ersten Eintrag DOS-intern gespeichert wurde und wiederverwendet wird. Desweiteren ist das Register C mit dem Wert 18 anstelle 17 zu laden, um die Funktion "nächsten Eintrag suchen" auszuwählen.

Funktion 19 - Datei löschen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Fehlercode

Diese Funktion löscht alle Dateien von der Diskette, deren Directoryeintrag dem übergebenen FCB entspricht. Voraussetzung dafür ist allerdings, daß Diskette und Datei(en) Schreib-/Lesestatus haben und der Anwender alle Wheel Nutzungsrechte besitzt, sofern das Attribut Wheel Schutz der Datei gesetzt ist. Die Funktion gestattet die Verwendung von Jokerzeichen.

Ebenso wie CP/M kennzeichnet auch ZSDOS gelöschte Dateien dadurch, daß das Byte des Nutzerbereichs der Dateien (DIR + 0) auf 0E5H gesetzt wird und die Bits der Dateien im Belegungsvektor gelöscht werden. Durch dieses Verfahren werden die Directoryeinträge für ZSDOS freigegeben, jedoch werden weder die Daten gelöscht, noch wird der Directorybelegungsvektor (DIR + 16) verändert. Solange keine Schreiboperationen ausgeführt werden, ist es daher häufig möglich, den Löschvorgang rückgängig zu machen ("unerase"). Dazu müssen nur alle physikalischen Extents der Datei gesucht werden und deren 0E5H in eine zulässige Nutzerbereichsnummer geändert werden. Anschließend wird die Funkton 37 aufgerufen, um eine Überarbeitung des Belegungsvektors zu veranlassen.

```
FKILL:  LD   DE,FCB      ; was gelöscht werden soll
        LD   C,19
        CALL BDOS        ; Löschen ausführen
        INC  A           ; alles OK?
        JR   Z,ERROR     ; ...Sprung bei Problemen
        ...
```

Funktion 20 - sequentiell lesen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Schreib-/Lesecode

Nachdem eine Datei mit der Funktion 15 geöffnet wurde, kann diese Funktion verwendet werden, um den nächsten 128 Bytes langen Record in den aktuellen DMA Puffer zu übertragen. Im FCB werden die Einträge für den aktuellen Record, den aktuellen Extent und die aktuelle Nummer des Datenmodules überarbeitet, um die nächste Position für den sequentiellen Zugriff bereitzustellen. Durch wiederholten Aufruf der Funktion 20 kann der nächste Record ohne Eingriff des Anwendungsprogrammes gelesen werden.

```
READS:  LD   DE,FCB      ; Datei muß bereits geöffnet sein
        LD   C,20
        CALL BDOS        ; nächsten Record in DMA Puffer lesen
        AND  A           ; Fehler aufgetreten?
        JR   NZ,ERROR    ; ...Sprung, wenn es Probleme gab
        ...
```

Funktion 21 - sequentiell schreiben	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Schreib-/Lesecode

Diese Funktion bildet das Gegenstück zur Funktion 20 - sie schreibt den Inhalt des aktuellen DMA Puffers auf Diskette in die angegebene Datei, die zuvor mit der Funktion 11 geöffnet wurde. Im FCB werden die Einträge für den aktuellen Record, den aktuellen Extent und die aktuelle Nummer des Datenmodules überarbeitet, um die nächste Position für den sequentiellen Zugriff bereitzustellen. Neue Belegungsblöcke und neue Extents werden geöffnet bzw. erzeugt, wenn dies notwendig ist.

Bei jedem Schreibzugriff auf eine Datei über den Funktionsruf 21 wird das Attributbit "Archiv" (t3) des ersten Extents gelöscht, wenn die Datei über Funktion 16 geschlossen wird. Über das Archiv Attribut ist ein gezieltes Backup mit Programmen wie COPY, ZFILER, PPIP, DATSWEEP sowie mit allen anderen Programmen, die das Archiv Attribut unterstützen, möglich.

```
WRITS:  LD   DE,FCB      ; Datei muß bereits geöffnet sein
        LD   C,21
        CALL BDOS        ; DMA Pufferinhalt auf Diskette schr.
        AND  A           ; Fehler aufgetreten?
        JR   NZ,ERROR    ; ...Sprung, wenn es Probleme gab
        ...
```

Weil das BDOS den BIOS Belegungsvektor eines Datenblockes vor dem sequentiellen oder wahlfreien Schreiben setzt, kann es passieren, daß der FCB bei auftretendem Fehler "Diskette voll" (zurückgegebener Code 02) einen unzulässigen Zustand einnimmt. Daher muß stets die Funktion 16 zum Schließen der Datei verwendet werden, welche aufgetretene Fehler anzeigt. Erst dann können weitere Operationen des BDOS (z. B. Löschen der fehlerhaften Datei) ausgeführt werden. Zum Abgleichen des BIOS Belegungsvektors wird der aktualisierte FCB benutzt, wodurch die entsprechenden Bits des Vektors nach dem Löschen der fehlerhaften Datei zurückgesetzt werden.

Funktion 22 - Datei erstellen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Verzeichniscode

Mit dieser Funktion wird eine neue Datei mit dem im FCB angegebenen Namen auf Diskette erzeugt. Durch den Aufruf wird kein Datenspeicherplatz auf der Diskette belegt, jedoch im Directory der Platz für den ersten Extent der Datei reserviert. Durch die Verwendung der Funktion 22 wird die Datei gleichzeitig zum Lesen und Schreiben geöffnet, so daß die Funktion 15 nicht mehr gesondert aufgerufen werden muß.

WARNUNG: Von der Funktion "Datei erstellen" wird vor dem Erzeugen nicht überprüft, ob auf der Diskette bereits eine Datei gleichen Namens vorhanden ist. Der Programmierer muß also dafür sorgen, daß ein bereits vorhandener Dateiname nicht ein weiteres Mal für die Dateierstellung benutzt wird.

Beispiel:

```
FMAKE:  LD   DE,FCB      ; diese Datei soll erstellt werden
        LD   C,17        ; zuerst gegen Duplikate absichern
        CALL BDOS
        INC  A
        JR   Z,FMAKE1    ; ...noch nicht vorhanden - erzeugen
        LD   DE,DUPWRN   ; Anwender auf Vorhandensein hinweisen
        LD   C,9
        CALL BDOS
        LD   C,1
        CALL BDOS        ; was möchte der Anwender tun?
        AND  5FH         ; in Großbuchstaben umwandeln
        CP   'Y'
        LD   C,0
        CALL NZ,BDOS     ; ...wenn NEIN, dann Abbruch
        LD   DE,FCB
        LD   C,19
        CALL BDOS        ; ansonsten Duplikat löschen

FMAKE1: LD   DE,FCB
        LD   C,22
        CALL BDOS        ; jetzt wird die Datei erstellt
        INC  A           ; Fehler aufgetreten?
        JR   Z,ERROR     ; ...Sprung, wenn es Probleme gab
        ...
DUPWRN: DEFB 'File exists! Erase it (Y/N)? $'
        ...
```

Funktion 23 - Datei umbennen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Fehlercode

Mit dieser Funktion wird die Datei umbenannt, die mit den ersten 13 Bytes des FCB (einschließlich Laufwerk und optionale Nutzerbereichsangabe) bezeichnet wird. Der neue Dateiname wird ab Byte 17 des FCB übergeben. Im Gegensatz zu CP/M 2.2 und ZRDOS erlaubt ZSDOS die Angabe von Jokerzeichen bei diesem Funktionsruf. Steht an irgendeiner Stelle der originalen Dateibezeichnung ein Fragezeichen '?', so wird dieses Zeichen im Directory durch die Umbenennen Funktion nicht verändert. Alle Attribute mit Ausnahme von "öffentliche Datei" bleiben erhalten. Die umbenannten Dateien erhalten aus Sicherheitsgründen alle das Attribut "privat", um Konflikten aufgrund von mehreren öffentlichen Dateien gleichen Namens auf einer Diskette vorzubeugen.

Ebenso wie bei der Funktion "Datei erstellen" liegt die Verantwortung beim Programmierer, die Erzeugung von Duplikaten im Directory mit dieser Funktion zu vermeiden.

Format des FCB beim Umbenennen:

Offset:	0	1	9	12	13	14	15	16	17		25	28	29	30	31
Daten:	dr	oldfn	typ	0	us	0	0	0	newfn		typ	0	0	0	0

(verwendete Abkürzungen:
dr - drive = Laufwerk
oldfn - old filename = alter Dateiname
typ - filetype = Dateityp
us - user = Nutzerbereich
newfn - new filename = neuer Dateiname)

```
RENAME:                 ; Vorauss.: Name noch nicht vergeben
        LD   DE,RENFCB  ; entsprechend formatierter FCB
        LD   C,23
        CALL BDOS       ; Datei umbennen
        INC  A          ; irgenwelche Probleme?
        JR   Z,ERROR    ; ...Sprung bei aufgetretenem Fehler
        ...
```

Funktion 24 - Login-Vektor holen	
Eingabe: keine	Ausgabe: HL = Login-Vektor

Von dieser Funktion wird im Registerpaar HL ein Bit-Abbild der momentan eingeloggten Laufwerke zurückgegeben. Die Belegung ist wie folgt definiert:

Register:	H								L							
Bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Laufwerk:	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

```
GETLGV: LD   C,24        ; Login-Vektor holen
        CALL BDOS        ; Vektor jetzt in HL
        ...
```

Funktion 25 - aktuelles Standardlaufwerk holen	
Eingabe: keine	Ausgabe: A = aktuelles Laufwerk

Mit dieser Funktion kann das momentane Standardlaufwerk festgestellt werden (das Laufwerk, welches ohne gesonderte Angabe benutzt wird). Im Register A wird ein Wert zwischen 0 und 15 zurückgegeben, der das entsprechende Laufwerk von A bis P bezeichnet. So steht z. B. der Wert 0 für A und 2 für Laufwerk C.

```
SHOWDD: LD   C,25        ; hole Standardlaufwerk
        CALL BDOS
        ADD  A,41H       ; Umwandlung zu ASCII
        LD   C,2         ; auf Konsole ausgeben
        LD   E,A         ; für ZSDOS in E laden
        CALL BDOS        ; als ASCII auf CON: ausgeben
        ...
```

Funktion 26 - Adresse DMA Puffer setzen	
Eingabe: DE = DMA Adresse	Ausgabe: keine (A = 00H)

Diese Funktion legt die Basisadresse des 128 großen Puffers, der für Diskettenschreib- und -lesefunktionen sowie für die Übertragung der Datumsstempel benutzt wird, auf die im Registerpaar DE übergebene Adresse fest. Standardmäßig beginnt der DMA Puffer auf Adresse 80H, wenn ein Programm gestartet wird. Die aktuelle Adresse kann mit Hilfe der Funktion 47 festgestellt werden.

Beachten Sie, daß der Ausdruck DMA etwas unzutreffend ist. Die Abkürzung DMA steht für "Direct Memory Access", womit eigentlich ein Peripherie-Chip bezeichnet wird. In Abhängigkeit von der Hardware und dem BIOS des jeweiligen Systems wird für die Übertragung der Diskettendaten ein derartiger Schaltkreis benutzt oder auch nicht.

```
STDMA:  LD   DE,DMAADR   ; Datenübertragungen finden dort statt
        LD   C,26
        CALL BDOS
        ...
```

Funktion 27 - Adresse des Belegungsvektors holen	
Eingabe: keine	Ausgabe: HL = Adresse Belegungsvektor

Im Registerpaar HL gibt diese Funktion einen Zeiger auf den Belegungsvektor des aktuellen Standardlaufwerkes zurück.

```
GETALV: LD   E,0        ; Belegungsvektor von Laufwerk A holen
        LD   C,14
        CALL BDOS       ; A als Standardlaufwerk festlegen
        LD   C,27       ; jetzt Belegungsvektor holen
        CALL BDOS
        ...
```

Funktion 28 - Schreibschutz für Diskette setzen	
Eingabe: DE = Schreibschutzvektor	Ausgabe: keine (A = 00H)

Diese Funktion dient in erster Linie dazu, die Disketten in den ausgewählten Laufwerken vor Schreib-, Umbenennungs-, Löschversuchen usw. zu schützen. Im Registerpaar DE wird ein Bit-Abbild zur Bestimmung der Laufwerke übergeben.

Die Möglichkeit, Laufwerke zu einem späteren Zeitpunkt auf Lesen/Schreiben zurückzusetzen, hängt vom Zustand des Flagbits "Nur-Lesen-Vektor erhalten" im Konfigurationsbyte ab (Bit 2 der Speicherzelle auf Adresse ZSDOSBASIS + 15). Wenn das Bit gesetzt ist, wird der Schreibschutzvektor niemals gelöscht. Ist das Bit gelöscht, dann wird das jeweilige Laufwerk durch einen Aufruf der Funktionen 13 oder 37 in den Status Lesen/Schreiben versetzt.

Jedem Bit im Registerpaar DE ist ein Laufwerk zugeordnet:

Register:	D								E							
Bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Laufwerk:	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

```
SETDRO: LD   C,25        ; hole Standardlaufwerk
        CALL BDOS
        LD   HL,1        ; Maske in HL initialisieren
        AND  A           ; Test für Laufwerk A
        JR   Z,SETDR2    ; ...Sprung, wenn A Standardlaufw. ist
        LD   B,A         ; ansonsten Wert = Verschiebungszähler
SETDR1: ADD  HL,HL       ; 16 Bits nach links verschieben
        DJNZ SETDR1      ; ...wiederholen, bis fertig
SETDR2: EX   DE,HL       ; Vektor in DE
        LD   C,28        ; setze Standardlaufwerk auf R/O
        CALL BDOS
        ...
```

Funktion 29 - Schreibschutzvektor holen	
Eingabe: keine	Ausgabe: HL = Schreibschutzvektor

Von dieser Funktion wird ein Abbild der Laufwerke zurückgegeben, die mit dem Funktionsruf 28 schreibgeschützt wurden. Im Gegensatz zu CP/M werden Laufwerke nicht automatisch schreibgeschützt, wenn ein Diskettenwechsel erkannt wird. Das Format des Schreibschutzvektors, der im Registerpaar HL zurückgegeben wird, ist wie folgt definiert:

Register:	H								L							
Bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Laufwerk:	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

```
CHKRO:  LD   C,25        ; hole Standardlaufwerk
        CALL BDOS
        LD   HL,1        ; Maske in HL initialisieren
        AND  A           ; Test für Laufwerk A
        JR   Z,CHKRO2    ; ...Sprung, wenn A Standardlaufw. ist
        LD   B,A         ; ansonsten Wert = Verschiebungszähler
CHKRO1: ADD  HL,HL       ; 16 Bit nach links verschieben
        DJNZ CHKRO1      ; ...wiederholen, bis fertig
CHKRO2: PUSH HL          ; Maske sichern
        LD   C,29        ; hole Schreibschutzvektor
        CALL BDOS
        POP  DE          ; Maske wiederherstellen
        LD   A,E
        AND  L
        LD   L,A         ; Test auf Übereinstimmung durch
                         ; AND-Verknüpfung der LSBs
        LD   A,D
        AND  H
        LD   H,A         ; ...und MSBs
        OR   L           ; Kontrolle auf Übereinstimmung
        JR   NZ,ISWP     ; wenn ungleich Null, dann
                         ; Standardlaufwerk schreibgeschützt
        ...
```

```
; Routine zum Rücksetzen des Schreibschutzes für alle Laufwerke

RST2RW: LD   C,100       ; in den Flags nachsehen,
        CALL BDOS        ; ...ob "Nur-Lesen-Vektor erhalten"
                         ; aktiv ist
        BIT  2,L
        JR   Z,RESET     ; deaktiviert, Rücksetzen funktioniert
        RES  2,L         ; ansonsten erst deaktivieren
        EX   DE,HL
        LD   C,101
        CALL BDOS        ; Flags setzen
RESET:  LD   C,29
        CALL BDOS        ; überprüfen, ob ein Laufwerk
                         ; schreibgeschützt ist
        LD   A,H
        OR   L
        RET  Z           ; nein, also Rücksetzen weglassen
        EX   DE,HL       ; Schreibschutzvektor in DE
        LD   C,37        ; mehrere Laufwerke zurücksetzen
        CALL BDOS        ; ...um den Schreibschutz aufzuheben
        ...
```

Funktion 30 - Dateiattribute setzen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Fehlercode

Dateiattribute werden von ZSDOS benutzt, um den Status von Dateien zwischen den BDOS Rufen zu kontrollieren. Die Attribute sind in den höchstwertigen Bits (Bit 7) der Dateinamen (8 Bytes Name, 3 Bytes Typ) von FCB und Directoryeintrag enthalten. Folgende Bedeutungen sind definiert:

FCB + 1	f1 (für Anwender verfügbar; Attribut "nicht laden" bei Plu*Perfect)
FCB + 2	öffentliche Datei
FCB + 3	kein Zugriffstempel
FCB + 4	f4 (für Anwender verfügbar)
FCB + 5	reserviert für interne Benutzung durch ZSDOS
FCB + 6	reserviert für interne Benutzung durch ZSDOS
FCB + 7	reserviert für interne Benutzung durch ZSDOS
FCB + 8	Wheel Schutz
FCB + 9	Nur-Lesen (Schreibschutz)
FCB + 10	Systemdatei
FCB + 11	archiviert

Diese Funktion gestattet dem Programmierer das Setzen oder Löschen von Attributen durch Setzen oder Rücksetzen der entsprechenden Bits im FCB, dessen Adresse in DE übergeben wird. Im FCB dürfen Jokerzeichen enthalten sein.

Der Anwender darf die Attributbits, die für die interne Benutzung reserviert sind, ***nicht verändern***. Bereits unter CP/M und ZRDOS waren diese Bits reserviert, so daß dies keine neue Einschränkung darstellt.

Wird das Attribut-Bit für *öffentliche Dateien* auf 1 gesetzt, dann kann diese Datei von jedem Nutzerbereich der gleichen Diskette gefunden werden, wenn eine eindeutige Dateiangabe zur Suche benutzt wird. Es liegt in der Verantwortung des Programmierers dafür zu sorgen, daß auf einer Diskette keine zweite Datei existiert, die mit einer öffentlichen Datei namensgleich ist.

Das Attribut *Wheel-Schutz* verhindert, daß die Datei überschrieben, gelöscht, umbenannt oder eines ihre Attribute verändert wird, solange das Wheel Byte nicht an ist. In einer Systemumgebung ohne ZCPR hat dieses Attribut im allgemeinen keine Wirkung. ZSDOS geht dann davon aus, daß der Anwender alle Nutzungsrechte besitzt.

Mit dem Attribut *Nur-Lesen* wird uneingeschränkter Schutz vor dem Überschreiben, Löschen oder Umbenennen einer Datei erreicht.

Das *System-Attribut* hat zwei Funktionen. Zum einen werden Systemdateien von den meisten Directoryprogrammen nicht angzeigt und bleiben dem Anwender dadurch "verborgen". Zum anderen werden derartige Dateien beim Öffnen unter ZSDOS entlang des Pfades gefunden, wenn der Pfad-Dateizugriff aktiv ist.

Das *Archiv-Attribut* zeigt an, ob eine Datei verändert wurde. Dieses Bit muß von einem Anwendungsprogramm gesetzt werden (typische Vertreter sind Programme für Sicherheitskopien). Wird in die Datei geschrieben, dann löscht ZSDOS das Archiv-Attribut des ersten Extents.

```
SETATT: LD   DE,FCB      ; FCB enthält zu setzende Attribute
        LD   C,30        ; ...Setzen oder Rücksetzen
        CALL BDOS        ; Dateiattribute setzen
        ...
```

Funktion 31 - Adresse des DPB holen	
Eingabe: keine	Ausgabe: HL = Adresse des DPB

Durch diese Funktion wird ein Zeiger auf den Diskettenparameterblock (DPB) des aktuellen Standardlaufwerkes im Registerpaar HL übergeben. Aus dem DPB können Informationen wie z. B. die Kapazität des Laufwerkes, die Anzahl der Directoryeinträge, die Anzahl der Spuren usw. gewonnen werden.

```
GETDPB: LD   E,0          ; DPB von Laufwerk A
        LD   C,14
        CALL BDOS         ; Standardlaufwerk ist jetzt A
        LD   C,31         ; hole DPB
        CALL BDOS         ; Zeiger auf DPB von A: in HL
        ...
```

Funktion 32 - Nutzerbereich holen/setzen	
Eingabe: E = 0FFH (holen) E = 0 - 31 (setzen)	Ausgabe: A = Nutzerbereich A = 0

Diese Funktion holt oder setzt den standardmäßigen Nutzerbereich für Dateioperationen. Gegenüber CP/M und ZRDOS gestattet ZSDOS das Überschreiben des standardmäßigen Nutzerbereichs durch eine gesonderte Angabe im FCB. Ansonsten wird für alle Dateioperationen der standardmäßige Nutzerbereich verwendet.

```
GETUSR: LD   C,32
        LD   E,0FFH      ; hole aktuellen Nutzerbereich
        CALL BDOS
        ...


SETUSR: LD   E,A         ; angenommen, Nutzerbereich in A
        LD   C,32
        CALL BDOS        ; setze neuen Nutzerbereich
        ...
```

Funktion 33 - wahlfrei lesen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Schreib-/Lesecode

Mit dieser Funktion ist der wahlfreie Lesezugriff auf die im FCB angegebene Datei möglich. Vor dem Aufruf der Funktion muß die Nummer für den wahlfreien Record (FCB + 33..FCB + 35, LSB..MSB) auf die Zahl des gewünschten 128 Bytes langen Records gesetzt werden. Zuvor muß allerdings die Datei mit dem Extent 0 über die Funktion 15 geöffnet worden sein.

Während die Funktion zum seriellen Lesen den Wert für den aktuellen Record (FCB + 32) nach jedem Zugriff erhöht, bleibt dieser von der Funktion 33 unverändert. Vom Anwendungsprogramm muß vor jedem Aufruf die Nummer für den wahlfreien Record gesetzt werden.

```
RDRAN:  ...             ; FCB + 33...35 bereits gesetzt
        LD   DE,FCB     ; Datei ist bereits geöffnet worden
        LD   C,33
        CALL BDOS       ; liest Record in den DMA Puffer
        AND  A          ; Fehler aufgetreten?
        JR   NZ,ERROR   ; ...Sprung, wenn es Probleme gab
        ...
```

Funktion 34 - wahlfrei schreiben	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Schreib-/Lesecode

Diese Funktion gestattet den wahlfreien Schreibzugriff auf die im FCB angegebene Datei. Ebenso wie bei Funktion 33 muß die Nummer für den wahlfreien Record (FCB + 33..FCB + 35, LSB..MSB) auf die Zahl des gewünschten 128 Bytes großen Records gesetzt werden. Zuvor muß die Datei bereits mit der Funktion 15 geöffnet oder mit der Funktion 22 erstellt worden sein. Es ist zu beachten, daß beim zuvor ausgeführten Öffnen mit der Funktion 15 die Nummer des Extents 0 sein muß (Basisextent).

Auch von dieser Funktion wird der Wert für den aktuellen Record (FCB + 32) nicht beeinflußt, so daß vom Anwendungsprogramm die Nummer für den wahlfreien Record vor jedem Funktionsaufruf gesetzt werden muß.

Von dieser Funktion werden Extents je nach Bedarf geöffnet, geschlossen und erstellt. Außerdem bewirkt die Funktion, daß das Archiv-Bit des ersten Extents beim Schließen der Datei über Funktion 16 zurückgesetzt wird. Dadurch wird angezeigt, daß die Datei verändert wurde.

Anmerkungen:

1.) Es ist unter ZSDOS möglich, Dateien zu erstellen, die für die Arbeit unter CP/M 2.2 oder ZRDOS zu groß sind. Während von CP/M 2.2 und ZRDOS Dateien nur bis zu 65.536 Records (8.192 kB) verarbeitet werden können, unterstützt ZSDOS Dateien bis zu einer Größe von 262.144 Records (32.768 kB). Derart große Dateien können problemlos unter CP/M 3.0 benutzt werden.

2.) Dateien, die im wahlfreien Zugriff erstellt wurden und "Löcher" enthalten, werden von den meisten Kopierprogrammen nicht fehlerfrei übertragen, da diese sequentielle Lese- und Schreiboperationen ausführen. Zu diesen Programmen gehören z. B. COPY, PIP und PPIP.

```
WRRAN:  ...             ; FCB + 33...35 bereits gesetzt
        LD   DE,FCB     ; Datei ist bereits geöffnet worden
        LD   C,34
        CALL BDOS       ; schreibt Record von DMA auf Diskette
        AND  A          ; Fehler aufgetreten?
        JR   NZ,ERROR   ; ...Sprung, wenn es Probleme gab
        ...
```

Bei Auftreten eines Fehlers "Diskette voll" gilt ebenso wie bei Funktion 21: Vor Ausführung anderer Operationen sollte die Datei geschlossen werden (siehe auch Abschnitt 5.2.21).

Funktion 35 - Dateigröße berechnen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Fehlercode FCB+33..35 = letzter Record+1

Diese Funktion berechnet die "virtuelle" Größe einer Datei. Die zurückgegebene Größe ergibt sich durch das Setzen der Nummer für den wahlfreien Record im FCB auf die des letzten gefundenen Records plus eins. Verwechseln Sie diesen Wert nicht mit der realen Größe der Datei auf der Diskette – wahlfrei erstellte Dateien können "verlorene" Extents enthalten, die zwar keinen Platz auf der Diskette belegen, aber in die Berechnung dieser Funktion einbezogen werden.

Oftmals wird dieser Funktionsruf benutzt, um für das Anfügen weiterer Records die Nummer des wahlfreien Records auf einen Wert nach dem Ende der Datei zu setzen.

```
GETSIZ: LD   DE,FCB      ; Größe dieser Datei holen
        LD   C,35
        CALL BDOS        ; Größe in FCB + 33...35 lesen
        AND  A           ; Fehler aufgetreten?
        JR   NZ,ERROR    ; ...Sprung, wenn es Probleme gab
        ...
```

Funktion 36 - wahlfreien Record setzen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = 00H

Von dieser Funktion wird die Nummer des wahlfreien Records im FCB auf die aktuelle Position in der Datei gesetzt, auf die lesend und/oder schreibend sequentiell zugegriffen wird. Diese Information kann für eine spätere Verwendung gespeichert werden, um schnell auf die gewünschte Stelle in der Datei zuzugreifen.

Eine mögliche Anwendung wäre z. B. die Erstellung eines Indexes während dem sequentiellen Schreiben einer Datei. Durch Verwendung des Indexes kann mit wahlfreien Zugriffen schnell auf die indizierten Positionen zugegriffen werden. Bitte beachten Sie, daß sequentielle Lese- und Schreibfunktionen die Nummer des wahlfreien Records nicht verändern.

```
SETRAN: LD   DE,FCB      ; Datei ist bereits geöffnet
        LD   C,36        ; ...und wurde sequentiell zugegriffen
        CALL BDOS        ; setze FCB + 33..FCB + 35 auf
        ...              ; ...aktuellen Record
```

Funktion 37 - mehrere Laufwerke zurücksetzen	
Eingabe: DE = Maske	Ausgabe: A = 0 oder A = 0FFH, wenn Datei namens $*.* auf Standardlaufw. vorh.

Unter CP/M 2.2 war diese fehlerhaft programmierte Funktion kaum zum Ausloggen mehrerer Laufwerke zu gebrauchen. Die meisten Programmierer gingen davon aus, daß die Funktion 37 unter CP/M ein zurückgesetztes Laufwerk wieder einloggen würde - das war aber nicht der Fall. Die meisten BDOS Substitute übernahmen diesen Fehler, in ZSDOS ist er jedoch behoben. Wird das Standardlaufwerk zurückgesetzt, dann wird es anschließend wieder eingeloggt und der Belegungsvektor neu aufgebaut - selbst wenn es sich dabei um eine Festplatte handelt. Zuvor sollte aber sichergestellt sein, daß alle Dateien auf der Diskette geschlossen sind, bevor das Laufwerk mit der Funktion 37 zurückgesetzt wird.

Anmerkung: Enthält ein Programm eine Stelle zum Diskettenwechsel, sollte die Funktion erst ***danach*** ausgeführt werden.

Die Funktion 37 bietet die einzige verfügbare Methode zum erneuten Aufbau der Belegungsvektoren von festen Disketten, wenn "schnelles Wiedereinloggen fester Disketten" aktiviert ist. Desweiteren muß diese Funktion stets ausgeführt werden, wenn direkte BIOS Rufe von einen Anwendungsprogramm benutzt wurden, die Directories oder Belegungsvektoren von festen Disketten verändern.

Im Registerpaar DE ist jedem Bit ein Laufwerk wie folgt zugeordnet:

Register:	D								E							
Bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Laufwerk:	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Hier ein Beispiel zum Wiedereinloggen aller festen Disketten im System:

```
LD   C,39        ; hole Vektor fester Disketten
CALL BDOS
EX   DE,HL       ; Vektor in DE übertragen
LD   C,37
CALL BDOS        ; feste Disketten wiedereinloggen
...
```

Funktion 39 - Vektor fester Disketten holen	
Eingabe: keine	Ausgabe: HL = Vektor fester Disketten

Von der Funktion 39 wird ein Bit-Abbild der Laufwerke im System wiedergegeben, die als feste Disketten eingeloggt sind, wenn "schnelles Wiedereinloggen fester Disketten" eingeschaltet ist. Ein Beispiel für den Gebrauch dieses Funktionsrufes ist dem Listing für die Funktion 37 dieses Abschnittes zu entnehmen.

Das Bit-Abbild, welches von ZSDOS im Registerpaar HL zurückgegeben wird, ist wie folgt definiert:

Register:	H								L							
Bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Laufwerk:	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Funktion 40 - wahlfrei schreiben, auffüllen mit Nullen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Schreib-/Lesecode

Die Arbeitsweise dieser Funktion ist der Funktion 34 sehr ähnlich, jedoch werden alle Records eines neu belegten Blockes durch Füllen mit 00H initialisiert. Werden Records mittels Funktion 34 in Blöcken erstellt, die noch nie beschrieben wurden, können undefinierte Daten in den Blöcken enthalten sein.

```
WRRANZ: ...             ; FCB + 33...35 bereits gesetzt
        LD   DE,FCB     ; Datei ist bereits geöffnet worden
        LD   C,40
        CALL BDOS       ; schreibt Record von DMA auf Diskette
                        ; ...und füllt den Rest mit Nullen
        AND  A          ; Fehler aufgetreten?
        JR   NZ,ERROR   ; ...Sprung, wenn es Probleme gab
        ...
```

Funktion 45 - BDOS Fehlermodus setzen	
Eingabe: E = 0FFH Fehlercode zurückgeben E = 0FEH Code zurückgeben und Fehler anzeigen E = 080H ZSDOS Standard-Fehlermodus setzen	Ausgabe: A = 00H A = 00H A = 00H

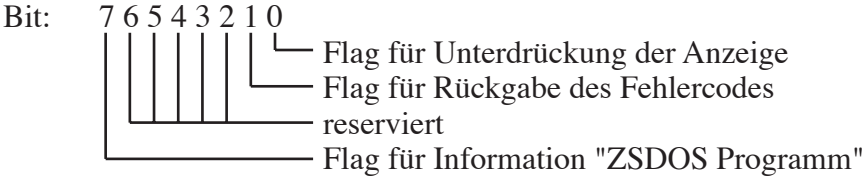
Mit der Funktion 45 haben Anwendungsprogramme die Möglichkeit, die Fehlerbehandlung von ZSDOS zu beeinflussen. Alle vom BDOS erkannten Fehler, einschließlich Auswahl- und Schreibschutzfehler, können an das Anwendungsprogramm übergeben und eine zusätzliche Bildschirmmeldung von ZSDOS angezeigt oder unterdrückt werden.

Um den Fehlermodus festzulegen, wird im Register E ein entsprechender Wert beim Aufruf der Funktion 45 übergeben. Das Byte 0FEH bewirkt die Ausgabe der Fehlermeldung durch ZSDOS auf dem Bildschirm. Anschließend wird die Kontrolle wieder an das Programm übergeben. Das Register A enthält dann den Wert 0FFH und der erweiterte Fehlercode steht im Register H zur Verfügung.

Enthält das Register E beim Aufruf der Funktion 45 stattdessen den Wert 0FFH, dann werden zwar in den Register A und H die eben genannten Werte übergeben, jedoch erscheint keine Fehlermeldung von ZSDOS auf dem Bildschirm. Wird bei der Einstellung des Fehlermodus im Register E der Wert 0 übergeben, dann stellt ZSDOS den standardmäßigen CP/M Fehlermodus ein. Diese Werte der Eingabeparameter entsprechen denen der Funktion 45 unter CP/M Plus.

Der an die Funktion 45 übergebene Wert erfüllt noch einen weiteren Zweck - er informiert das DOS darüber, daß ein ZSDOS Programm läuft. Speziell für ZSDOS geschriebene Programme setzen das S1 Byte im FCB auf den Nutzerbereich OR-verknüpft mit 80H und das S2 Byte auf den Wert 0, wenn Dateioperationen wie Öffnen, Umbenennen, Löschen usw. durchgeführt werden. Ist der Fehlermodus auf einen Wert ungleich Null gesetzt worden, dann geht ZSDOS davon aus, daß das Anwendungsprogramm diese Bytes ordnungsgemäß setzt.

Um das DOS unter Beibehaltung des standardmäßigen Fehlermodus darüber zu informieren, daß ein ZSDOS Programm läuft, wird das Bit 7 im Register E bei der Übergabe als Flag benutzt. Die Bits für den Fehlermodus sind momentan wie folgt festgelegt:



Hat ein Anwendungsprogramm den Fehlermodus geändert, dann muß er vor der Rückkehr zur Betriebssystemebene wieder auf Null gesetzt werden. Nur so kann eine volle Kompatibilität zu Programmen gewährleistet werden, die nicht speziell für ZSDOS geschrieben wurden. Beim Abbruch eines Programmes vom BDOS wird der Fehlermodus vor dem Booten auf 0 gesetzt. Auch der Aufruf der Funktion 0 setzt den Fehlermodus auf den CP/M Standard zurück.

Bei der Rückgabe von erweiterten Fehlercodes in den entsprechenden Modi zeigt der Wert 0FFH im Register A das Auftreten eines Fehlers an. Der jeweilige Fehlercode wird im Register H zurückgegeben. Die Codes haben folgende Bedeutung:

Wert in H	***Bedeutung***
0	kein erweiterter Fehlercode
1	Diskettenein-/-ausgabefehler (defekter Sektor)
2	Diskette schreibgeschützt
3	Datei schreibgeschützt
4	unzulässige Laufwerksangabe

Derzeitig sind nur 4 erweiterte Fehlercodes definiert. Es ist jedoch möglich, daß in zukünftigen Versionen weitere hinzukommen. Aus diesem Grund sollte bei der Programmierung nicht davon ausgegangen werden, daß nur bestimmte Fehler auftreten können, sondern alle definierten Fehlercodes getestet werden.

Beispiele für das Setzen des Fehlermodus:

```
SET$RET$ERR:
        LD   E,0FFH      ; Fehlercode zurück, keine Anzeige
        LD   C,45
        CALL BDOS
        ...


SET$QRET$ERR:
        LD   E,0FEH      ; Fehlercode zurück, Bildschirmanzeige
        LD   C,45
        CALL BDOS
        ...


SET$DEF$ERR:
        LD   E,0         ; Standard-Fehlermodus einstellen
        LD   C,45
        CALL BDOS
        ...
```

Beispiele für die Fehlerbehandlung:

```
FOPEN:   LD   DE,FCB
         LD   C,15
         CALL BDOS        ; Datei öffnen
         INC  A
         JR   NZ,OKOPEN   ; kein Fehler, Datei ist geöffnet
         LD   A,H         ; erweiterten Fehlercode laden
         AND  A           ; Test auf erweiterten Fehlercode
         JR   Z,NOFILE    ; kein erweiterter Fehler
                          ; ...Datei wurde nicht gefunden
         CP   1
         JR   Z,BADSEC    ; Fehler "defekter Sektor"
;
; beim Öffnen können keine Schreibschutzfehler auftreten
;
         CP   4
         JR   Z,SELERR    ; Fehler "unzulässige Laufwerksangabe"
;
; Diese erweiterten Fehlercodes sind bisher definiert.
; In zukünftigen Versionen sind es vielleicht noch mehr, also
; sollte man diese Möglichkeit einplanen!
;
         JR   UNKERR      ; ...ansonsten Sprung zur Routine
                          ; "unbekannter Fehler"
         ...


FWRITE:  LD   DE,FCB
         LD   C,21
         CALL BDOS        ; Sektor in Datei schreiben
         AND  A
         JR   Z,OKWR      ; kein Fehler, Datei ist geöffnet
         CP   OFFH        ; Ist es ein erweiterter Fehler?
         JR   NZ,NRMERR   ; Nein, normaler Fehler
                          ; ...wie "Disk voll", "Dir voll" usw.
         LD   A,H         ; ansonsten erweit. Fehlercode laden
         CP   1
         JR   Z,BADSEC    ; ...Sprung Routine "defekter Sektor"
         CP   2
         JR   Z,DISKWP    ; ...Sprung Routine "Disk R/O"
         CP   3
         JR   Z,FILEWP    ; ...Sprung Routine "Datei R/O"
         CP   4
         JR   Z,SELERR    ; ...Sprung Routine "unzulässiges Lw"
;
; Diese erweiterten Fehlercodes sind bisher definiert.
; In zukünftigen Versionen sind es vielleicht noch mehr, also
; sollte man diese Möglichkeit einplanen!
;
         JR   UNKERR      ; ...ansonsten Sprung zur Routine
                          ; "unbekannter Fehler"
         ...
```

Funktion 47 - Adresse des DMA Puffers holen	
Eingabe: keine	Ausgabe: HL = Zeiger auf DMA Puffer

Diese Funktion gibt im Registerpaar HL die Basisadrese des DMA Puffers zurück. Der DMA Puffer wird für alle Übertragungen von und zur Diskette sowie von den Datumsstempelfunktionen benutzt.

Hauptsächlich IOPs, die in ZSDOS wiedereintreten, benutzen diese Funktion. Die Adresse des DMA Puffers wird vom IOP geholt, um sie nach dem wiedereintretenden Ruf auf den alten Wert zu setzen.

Es ist zu beachten, daß nur die DMA Adresse zurückgegeben wird, wie sie dem *DOS* bekannt ist. Die DMA Adresse im *BIOS* kann davon abweichen, wenn ein Anwendungsprogramm diese durch direkte BIOS Rufe verändert hat. Um auch zu späteren Betriebssystemen kompatibel zu bleiben, sollte der Gebrauch von BIOS Routinen minimiert oder gänzlich vermieden werden.

```
GETDMA: LD   C,47        ; hole Adresse des DMA Puffers
        CALL BDOS        ; wird in HL zurückgegeben
        ...
```

Funktion 48 - Version des erweiterten DOS holen	
Eingabe: keine	Ausgabe: H = DOS Kennzeichen (‘S‘ für ZSDOS) (‘D‘ für ZDDOS) A, L = Versionsnummer

Mit dieser Funktion kann man herausfinden, ob ZSDOS in einem System läuft und wenn ja, welche Version. Es ist unbedingt erforderlich das Vorhandensein von ZSDOS festzustellen, bevor eine der neuen bzw. erweiterten ZSDOS Funktionen benutzt wird. Um sicherzugehen, daß ZSDOS vorhanden ist, ruft man zunächst die Funktion 12 auf, von der 22H im Register A zurückgegeben werden muß. Anschließend wird die Funktion 48 aufgerufen.

Die Funktion zur Abfrage der Kennzeichnung von erweiterten Betriebssystemen wurde in Zusammenarbeit mit Joe Wright, Bridger Mitchell und den Autoren von ZSDOS entwickelt. Der Aufruf ist identisch zur ZRDOS Funktion "Versionsnummer holen". Anhand des im Register H zurückgegebenen Kennzeichens können die DOS Substitute unterschieden werden. Ein erweitertes DOS erkennt man daran, daß im Gegensatz zu CP/M in den Registern A und L die Versionsnummer zurückgegeben wird. Im normalen CP/M BDOS ist diese Funktion nicht enthalten und es wird nur der Wert Null zurückgegeben. Zur Zeit sind folgende Kennzeichen für das DOS vergeben:

Kennzeichen	***DOS***
00H	ZRDOS
‘D‘	ZDDOS
‘S‘	ZSDOS

Die Zuweisung neuer Kennzeichen sollte mit einer der oben genannten Personen abgestimmt werden, um keine Mißverständnisse in Bezug auf diese Funktion aufkommen zu lassen. Von den 256 verschiedenen Kennzeichen stehen noch 253 zur Verfügung!

```
LD   C,12        ; hole CP/M Version
CALL BDOS
CP   22H         ; zu Version 2.2 kompatibel?
JR   NZ,NOTZS    ; ...nein, kann also nicht ZSDOS sein
LD   C,48
CALL BDOS        ; hole Version des erweiterten DOS
LD   A,H         ; Versionskennzeichen laden
CP   ‘S‘         ; Ist es ZSDOS?
JR   NZ,NOTZS    ; ...Sprung, wenn nicht ZSDOS
...
```

Funktion 98 - Uhrzeit holen	
Eingabe: DE = Zieladr. Zeitinformationen	Ausgabe: A = Zeit-/Datumscode

Diese Funktion ermöglicht Anwendungsprogrammen, die Systemuhr zu lesen. Im Registerpaar DE wird eine Zieladresse eines Pufferbereiches übergeben, in den die Zeitinformationen von ZSDOS geschrieben werden.

Für diese Funktion werden Treiberroutinen benötigt. Ist kein Treiber installiert oder kann die Uhr nicht gelesen werden, dann wird im Register A der Wert 0FFH zurückgegeben und der Puffer bleibt unverändert. Enthält das Register A bei der Rückkehr den Wert 1, stehen im Puffer die Zeit- und Datumsinformationen der Systemuhr bereit.

```
GETTIM: LD   DE,TIMEAD   ; Anfangsadresse des Puffers
        LD   C,98
        CALL BDOS        ; Zeitinformationen auf Zieladresse
        INC  A           ; Fehler aufgetreten?
        JR   Z,ERROR     ; ...Sprung, wenn Fkt. nicht vorhanden
        ...

TIMEAD: DEFB 0,0,0,0,0,0 ; Puffer mit Nullen initialisiert
        ...
```

Funktion 99 - Uhrzeit setzen	
Eingabe: DE = Zeiger Zeitinformationen	Ausgabe: A = Zeit-/Datumscode

Mit dieser Funktion kann ein Anwendungsprogramm die Systemuhr stellen. Die Anfangsadresse des Puffers mit den Zeitinformationen wird im Registerpaar DE übergeben.

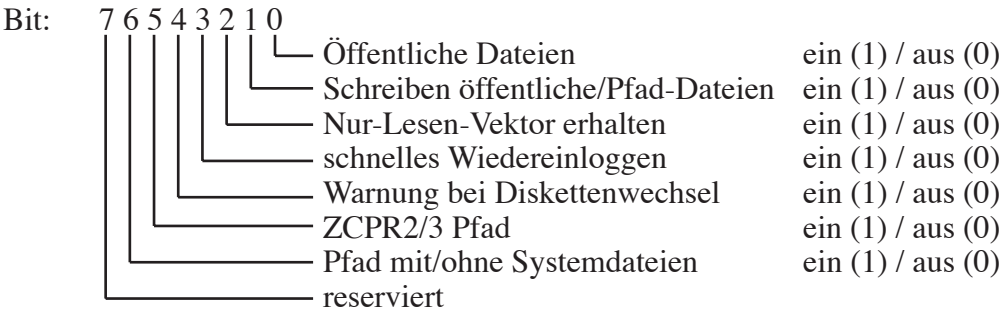
Für diese Funktion werden Treiberroutinen benötigt. Ist kein Treiber installiert, wird im Register A der Wert 0FFH zurückgegeben. Enthält das Register A bei der Rückkehr den Wert 1, wurde die Uhr gestellt.

```
SETTIM: LD   DE,TIMEAD   ; Anfangsadresse des Puffers
        LD   C,99
        CALL BDOS        ; Uhr stellen
        INC  A           ; Fehler aufgetreten?
        JR   Z,ERROR     ; ...Sprung, bei Fehler oder
                         ; ...wenn Funktion nicht vorhanden
        ...
```

Funktion 100 - Konfigurationsflags holen	
Eingabe: keine	Ausgabe: HL = Flags

Von dieser Funktion werden im Registerpaar HL die aktuellen ZSDOS Konfigurationsflags zurückgegeben. In der Version 1.1 von ZSDOS ist nur das Register L von Bedeutung. Aus Kompatibilitätsgründen zu späteren Versionen von ZSDOS, die eventuell mehr Statusbits benutzen, wird im Register H der Wert 0 zurückgegeben.

Register H = 0, Register L enthält:

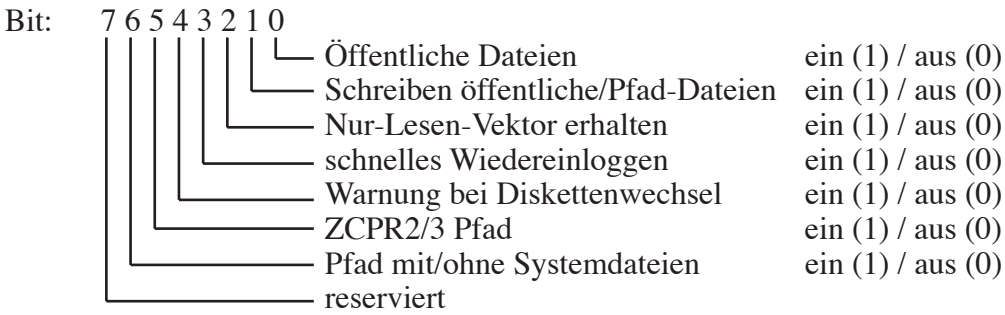


Bitte schlagen Sie im Abschnitt 4.3.4 für eine detaillierte Beschreibung der Funktionen der einzelnen Bits nach.

Funktion 101 - Konfigurationsflags setzen	
Eingabe: DE = Flags	Ausgabe: keine

Mit diesem Funktionsruf werden die ZSDOS Konfigurationsflags auf die im Registerpaar DE übergebenen Werte gesetzt. In der Version 1.1 von ZSDOS ist nur der Wert im Register E von Bedeutung. Das Register D sollte mit dem Wert 0 geladen werden, um zu späteren Versionen kompatibel zu bleiben.

Register D = 0, Register E enthält:



Bitte schlagen Sie im Abschnitt 4.3.4 für eine detaillierte Beschreibung der Funktionen der einzelnen Bits nach.

Funktion 102 - Datumsstempel holen	
Eingabe: DE = Adresse des FCB	Ausgabe: A = Zeit-/Datumscode Datumsstempel im DMA Puffer

Diese Funktion gibt die Datumsstempel der Datei zurück, deren Name im durch DE adressierten FCB steht. Die Extent- und Modulnummern (FCB + 12...FCB + 14) müssen vor dem Aufruf der Funktion auf Null gesetzt sein. Außerdem muß der richtige Nutzerbereich eingestellt sein. Dazu wird entweder FCB + 13 mit der Nummer des Nutzbereichs OR-verknüpft mit 80H geladen oder zuvor die Funktion 32 aufgerufen. Die gewünschten Stempelinformationen stehen nach dem Aufruf der Funktion 102 in den ersten 15 Bytes des DMA Puffers zur Verfügung. Zukünftige Versionen von ZSDOS benutzen eventuell ein erweitertes Stempelformat, so daß wir die Bereitstellung eines 128 Bytes großen Puffers für die Datumsstempel empfehlen.

Um diese Funktionen ausführen zu können, müssen entsprechende Treiberroutinen installiert sein. Stehen keine Treiber zur Verfügung oder können die Stempel nicht gelesen werden, dann wird im Register A der Wert 0FFH zurückgegeben. In diesem Fall ist der Inhalt des DMA Puffers undefiniert. Wird im Register A der Wert 1 zurückgegeben, dann stehen im DMA Puffer gültige Stempelinformationen zur Verfügung.

Funktion 103 - Datumsstempel setzen	
Eingabe: DE = Adresse des FCB Datumsstempel im DMA Puffer	Ausgabe: A = Zeit-/Datumscode

Diese Funktion schreibt die Stempelinformationen im DMA Puffer auf Diskette. Das Registerpaar DE zeigt beim Aufruf der Funktion auf den FCB mit dem Namen der zu stempelnden Datei. Ebenso wie beim Funktionsruf 102 müssen auch hier die Extent- und Modulnummern initialisiert und der Nutzerbereich festgelegt worden sein, bevor die Funktion aufgerufen wird.

Um diese Funktionen ausführen zu können, müssen entsprechende Treiberroutinen installiert sein. Stehen keine Treiber zur Verfügung oder können die Stempel nicht geschrieben werden, dann enthält das Register A bei der Rückkehr den Wert 0FFH. Wird im Register A der Wert 1 zurückgegeben, dann wurden die Stempelinformationen erfolgreich auf Diskette geschrieben. Beachten Sie bitte, daß ZSDOS nicht die normale Fehlerbehandlung aufruft, wenn die Diskette schreibgeschützt ist. In diesem Fall wird im Register A der Wert 0FFH zurückgegeben und kein Datumsstempel auf Diskette geschrieben.

Beispiel zur Benutzung der Funktionen 102 und 103:

```
COPYDS: LD   DE,DSBUF    ; Puffer für Stempelinformationen
        LD   C,26        ; setze Adresse des DMA Puffers
        CALL BDOS
        LD   DE,SRCFCB   ; Quell-FCB
                         ; (Nutzerbereich bereits gesetzt)
        LD   C,102
        CALL BDOS        ; hole Stempel der Quelldatei
        LD   DE,DSTFCB   ; Ziel-FCB
                         ; (Nutzerbereich bereits gesetzt)
        LD   C,103
        CALL BDOS        ; Stempel auf Zieldatei übertragen
        ...
```

Schnellübersicht der Funktionen von ZSDOS

Nr.	Funktionsname	Eingabeparameter	zurückgegebene Werte
0	Boot	keine	keine
1	Konsoleneingabe	keine	A=Zeichen
2	Konsolenausgabe	E=Zeichen	keine (A=BIOS A)
3	Zusatzeingabe	keine	A=Zeichen
4	Zusatzausgabe	E=Zeichen	keine (A=BIOS A)
5	Listausgabe	E=Zeichen	keine (A=BIOS A)
6	direkte Konsolen-eingabe/-ausgabe	E=0FFH (ein) E=0FEH (ein) E=0FDH (ein) E=0..0FCH (aus)	A=eingeg. Zeichen A=Konsolenstatus A=eingeg. Zeichen keine (A=BIOS A)
7	IOBYTE holen	keine	A=IOBYTE
8	IOBYTE setzen	E=IOBYTE	keine (A=IOBYTE)
9	Zeichenkette ausgeben	DE=Adr. Zeichenk.	keine (A='$')
10	Konsolenpuffer lesen	DE=Adr. Puffer	keine (A=0DH)
11	Konsolenstatus holen	keine	A=00H - kein Zeichen A=01H - Zeichen vorh.
12	CP/M Versionsnummer holen	keine	HL=22H
13	Diskettensystem rücksetzen	keine	A=00H keine $*.* Datei A=0FFH $*.* vorhanden
14	Laufwerk auswählen	E=Nr. Laufwerk	A=00H keine $*.* Datei A=0FFH $*.* vorhanden
15	Datei öffnen	DE=Adr. FCB	A=Verzeichniscode
16	Datei schließen	DE=Adr. FCB	A=Verzeichniscode
17	ersten Eintrag suchen	DE=Adr. FCB	A=Verzeichniscode
18	nächsten Eintrag suchen	keine	A=Verzeichniscode
19	Datei löschen	DE=Adr. FCB	A=Fehlercode
20	sequentiell lesen	DE=Adr. FCB	A=Schreib-/Lesecode
21	sequentiell schreiben	DE=Adr. FCB	A=Schreib-/Lesecode
22	Datei erstellen	DE=Adr. FCB	A=Verzeichniscode
23	Datei umbennen	DE=Adr. FCB	A=Fehlercode
24	Login-Vektor holen	keine	HL=Login-Vektor
25	akt. Standardlaufwerk holen	keine	A=akt. Standardlaufwerk
26	Adresse DMA Puffer setzen	DE=DMA Adr.	keine (A=00H)
27	Adr. Belegungsvektor holen	keine	HL=Belegungvektor
28	Diskettenschreibschutz setzen	DE=R/O-Vektor	keine (A=00H)
29	Schreibschutzvektor holen	keine	HL=R/O-Vektor
30	Dateiattribute setzen	DE=Adr. FCB	A=Fehlercode
31	Adresse des DPB holen	keine	HL=Adresse DPB
32	Nutzerbereich holen/setzen	E=0FFH (holen) E=Nutzerbereich (setzen)	A=Nutzerbereich A=00H

33	wahlfrei lesen	DE=Adr. FCB	A=Schreib-/Lesecode
34	wahlfrei schreiben	DE=Adr. FCB	A=Schreib-/Lesecode
35	Dateigröße berechnen	DE=Adr. FCB	A=Fehlercode FCB+33..35=ltzt. Rec.+1
36	wahlfreien Record setzen	DE=Adr. FCB	A=00H
37	mehrere Laufwerke zurücksetzen	DE=Maske	A=00H A=0FFH $*.* vorhanden
38	*nicht enthalten*		
39	Vektor fester Disk holen	keine	HL=Vektor fixed Disks
40	wahlfrei schreiben, auffüllen mit Nullen	DE=Adr. FCB	A=Schreib-/Lesecode
41-44	*nicht enthalten*		
45	BDOS Fehlermodus setzen	E=0FFH Code E=0FEH Code+Msg. E=80H ZSDOS-Mode E=00H CP/M-Mode	A=00H A=00H A=00H A=00H
46	*nicht enthalten*		
47	Adr. DMA Puffer holen	keine	HL=Zeiger auf DMA
48	Version des erweiterten DOS holen	keine	H=DOS-Typ 'S'=ZSDOS 'D'=ZDDOS A, L=Version (BCD)
49-97	*nicht enthalten*		

Die Funktionen 98 und 99 sind nur verfügbar, wenn ein Uhrentreibermodul installiert ist.

98	Uhrzeit holen	DE=Zieladr. Zeit	A=Zeit-/Datumscode
99	Uhrzeit setzen	DE=Zeiger Zeit	A=Zeit-/Datumscode
100	Konfigurationsflags holen	keine	HL=Flags
101	Konfigurationsflags setzen	DE=Flags	keine

Die Funktionen 102 und 103 sind unter ZSDOS nur verfügbar, wenn ein Modul für Datumsstempel installiert ist.

102	Datumsstempel holen	DE=Adr. FCB	A=Zeit-/Datumscode Stempel im DMA
103	Datumsstempel setzen	DE=Adr. FCB Stempel im DMA	A=Zeit-/Daumscode

Übersicht der BDOS Fehlercodes

Verzeichniscode:

A = 00H, 01H, 02H, 03H, wenn kein Fehler aufgetreten ist

A = 0FFH, bei einem Fehler

Fehlercode:

A = 00H, kein Fehler

A = 0FFH, Fehler aufgetreten

Zeit-/Datumscode:

A = 01H, wenn kein Fehler aufgetreten ist

A = 0FFH, Fehler aufgetreten

Schreib-/Lesecode:

A = 00H, wenn kein Fehler aufgetreten ist

A = 01H, Lesen - Dateiende

Schreiben - Directory voll

A = 02H, Diskette voll

A = 03H, Fehler während des Schließens beim wahlfreien Lesen/Schreiben

A = 04H, leerer Record beim wahlfreien Lesen

A = 05H, Directory voll beim wahlfreien Schreiben

A = 06H, Record während des wahlfreien Lesens/Schreibens zu groß

erweiterte Fehlercodes im Fehlermodus:

A = 0FFH, weitere Fehlercodes in H

H = 01H, Diskettenein-/-ausgabefehler (defekter Sektor)

H = 02H, Diskette schreibgeschützt (nur lesen)

H = 03H, Datei schreibgeschützt

H = 04H, unzulässiges Laufwerk ausgewählt

Kurzübersicht der BIOS Funktionen

Nr.	Funktionsname	Eingabeparameter	zurückgegebene Werte
0	BOOT	keine	keine
1	WBOOT	keine	keine
2	CONST	keine	A=0FFH, bereit A=00H, nicht bereit
3	CONIN	keine	A=Zeichen von CON:
4	CONOUT	C=Zeichen an CON:	keine
5	LIST	C=Zeichen an LST:	keine
6	PUNCH	C=Zeichen an PUN:	keine
7	READER	keine	A=Zeichen von RDR:
8	HOME	keine	keine
9	SELDSK	C=Laufwerk (0..15) E=Init Select Flag	HL=Adresse DPH HL=0 unzulässiges Laufw.
10	SETTRK	BC=Spurnummer	keine
11	SETSEC	BC=Sektornummer	keine
12	SETDMA	BC=Adr. DMA Puffer	keine
13	READ	keine	A=00H fehlerfrei A=01H Fehler aufgetreten
14	WRITE	C=00H Daten schreiben C=01H Directory schr. C=02H neue Daten schreiben	A=00H fehlerfrei A=01H Fehler aufgetreten
15	LISTST	keine	A=00H bereit A=0FFH nicht bereit
16	SECTRN	BC=log. Sektornummer	HL=phys. Sektornummer DE=Adr. Übersetzungstab.

Anmerkung: Das BIOS darf das IX-Register nicht verändern!

Harold F. Bower
7914 Redglobe Court
Severn, MD 21144
Ladera Z-Node
213/670-9465

Cameron W. Cotrill
2935 Manhatten Ave.
La Crescenta, CA 91214
Ladera Z-Node
213/670-9465

Carson Wilson
1359 W. Greenleaf
Chicago, IL 60626
Antelope Freeway Z-Node
312/764-5162

Notizen