robotron

Anleitung für den
Systemprogrammierer

# Steuerprogramm SCPX 1700

Arbeitsplatzcomputer A 7100 Betriebssystem SCP 1700

Anleitung für den Systemprogrammierer

# Steuerprogramm SCPX 1700

ACS


**VEB Robotron-Projekt Dresden**

C 1013-0000-1 M 3030

Die vorliegende Systemunterlagendokumentation, Anleitung für den Systemprogrammierer Steuerprogramm SCPX 1700 , entspricht dem Stand von 6/86.



Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Elektronik Dresden .

Herausgeber:

VEB Robotron-Projekt Dresden 8010 Dresden, Leningrader Str. 9



Kurzreferat

Das SCP 1700 ist ein Einzelnutzerbetriebssystem, das speziell für den Einsatz des AC A7100 im kommerziellen Bereich als Büro- und Personalcomputer entwickelt worden ist. In dieser Schrift sind die Eigenschaften des BS und ihre Nutzung durch die Anwenderschnittstellen (BDOS-Rufe und CCP-Kommandos) beschrieben.
Weiterhin wird der Aufbau des hardwareabhängigen Teiles, des BIOS, und seine Anwendung beschrieben.
Durch Veränderung der BIOS-Routinen läßt sich das SCP 1700 an anwenderspezifische Hardware anpassen. Methoden dazu und zur Neugenerierung eines SCP 1700 sind im letzten Teil beschrieben.

Druck und buchbinderische Verarbeitung:
TASTOMAT, Eggersdorf · Ag 706/205/87 — 4116

Inhaltsverzeichnis

## Bildverzeichnis



## Tabellenverzeichnis

## 1. Einleitung

Diese Schrift gibt dem Nutzer des SCP 1700 die Möglichkeit, alle Eigenschaften des Betriebssystems zu nutzen.
In den Abschnitten dieser Anleitung wird ein Überblick über die Arbeit des CCP, über die Nutzung der BDOS (Basic Disk Operating System)-Routinen und des BIOS (Basic Input/Output System) gegeben.
Weitere Abschnitte behandeln die Anpassung des SCP 1700 an andere Hardwarelösungen durch Veränderung der BIOS-Routinen und die Anfangsladeorganisation in Zusammenhang mit der SCP-System-Datei (SCP.SYS).
Zum Verständnis dieser Schrift ist die Kenntnis der folgenden Dokumentation erforderlich.

Anwendungsbeschreibung des SCP 1700
Anleitung für den Bediener des SCP 1700
Anleitung für den Programmierer des SCP 1700

# 2. Systemüberblick

## 2.1. Allgemeine Charakteristik

SCP 1700 ist ein Einzelnutzerbetriebssystem. Es ist konzipiert für Rechner des Typs A 7100 von Robotron.
SCP 1700 ist für die Bedienung eines Hauptspeichers mit einer Kapazität bis maximal 1 MByte geeignet.
SCP 1700 ist programm- und dateikompatibel zu allen existierenden Versionen von CP/M-86.
Die Dateistruktur des Betriebssystems erlaubt die Bedienung von maximal 16 Plattengeräteeinheiten (Floppy, Hard-Disk) mit bis zu 8 MByte Kapazität.
SCP 1700 ist in der Datei SCP.SYS auf der Systemplatte vorhanden. Diese Datei wird bei der Systeminitialisierung durch den Anfangslader in den Hauptspeicher geladen.
Dieser Anfangslader belegt die ersten Spuren der Systemplatte und wird beim System-Anfangsladevorgang vom Monitorprogramm in den Hauptspeicher geladen und gestartet. Die Datei SCP.SYS enthält die drei Programmoduln:

- Console Command Prozessor (CCP)
- Basic-Disk Operating System (BDOS)
- Basic I/O-System (BIOS)

CCP und BDOS erfordern einen Speicherplatzbedarf von etwa 10 K Bytes. Das Betriebssystem ist im Systemspeicher eines ACS-Rechners oberhalb der reservierten Interruptadressen und des Monitorprogrammes standardmässig ab der Adresse 1040 H angeordnet.
CCP, BDOS und BIOS bleiben ständig im Hauptspeicher. Sie werden nicht bei jedem System Reset neu geladen.
SCP 1700 lädt mit Hilfe des CCP Speicherabbilddateien (memory image files) von einer Platte und startet sie. Diese Dateien haben einen "Kopfsatz" und werden durch den Dateityp CMD gekennzeichnet. Sie stellen abarbeitungsfähige Programme dar.
Über einen Software-Interrupt können diese Programme mit dem Betriebssystem mittels eines BDOS-Rufes in Verbindung treten.
Diese Programme können durch Eingabe von CTRL C bzw. durch den BDOS-Ruf 0 beendet werden. Beides führt zu einem System Reset des Systems. Mit Hilfe von Basisseitenadressen wird unter SCP 1700 eine Unabhängigkeit von absoluten Adressen erreicht und damit die Nutzung eines 1 MByte-Adreßbereiches ermöglicht.
Folgende allgemeine Begriffe werden in dieser Schrift verwendet:

Tabelle 1: Begriffe

| Begriff | Bedeutung |
|---|---|
| Nibble | 4-Bit/Halbbyte |
| Byte | 8-Bit Wert |
| Word | 16 Bit Wert |
| Double Word | 32 Bit Wert |
| Paragraph | 16 zusammenhängende Bytes |
| Paragraph Boundary | Eine Adresse die ohne Rest durch 16 teilbar ist. |
| Segment | Ein Speicherbereich von 64 K zusammenhängender Bytes |
| Segmentregister | Eines der Register CS, DS, ES oder SS. |
| Offset | 16-Bit-Adresse relativ zu einem Segmentregister-inhalt. |
| Bereich | Eine zu einem Segmentregisterinhalt relative Programmeinheit. |
| Adress | Effektive Speicheradresse, die sich aus Zusammenfassung des Segmentregister-Wertes und des Offset-Wertes ergibt. |

Ein Bereich besteht aus einem Satz von Segmenten, die als eine Einheit in den Speicher geladen werden. Da ein Bereich aus mehr als 64 K Bytes bestehen kann, muß das Applikationsprogramm selbst die Segmentregister verändern, falls auf Code oder Daten über das jeweilige 64 K-Segment hinaus zugegriffen werden soll.
SCP 1700 unterstützt 8 Programmbereiche:
Code-, Daten-, Stack- und Extrabereiche sowie 4 Hilfsbereiche.
Nachdem ein Code-, Daten-, Stack- oder ein Extrabereich geladen ist, werden von SCP 1700 die Segmentregister (CS, DS, SS und ES) auf die Basisadressen der Bereiche gestellt.

## 2.2. Anwendungshinweise

Das Betriebssystem wird standardmäßig oberhalb der Interruptplätze ab Adresse 1040H geladen und die transienten verschieblichen Programme direkt oberhalb des Betriebssystems. Durch entsprechende Generierungen ist es möglich, das SCP 1700 und die transienten Programme in jeden beliebigen Teil des Hauptspeichers zu laden.
Programme, die unter der Steuerung des SCP 1700 laufen, müssen im Speicher nach bestimmten Grundsätzen aufgebaut sein.
Aus diesen Grundsätzen resultieren drei mögliche Speichermodelle, die durch SCP 1700 unterstützt werden:

- das 8080-Speichermodell
- das Small Speichermodell
- das Compact Speichermodell

Der Aufbau der Speichermodelle und ihre Anwendung werden im Abschnitt 3 beschrieben. Mit dem Systemprogramm GENCMD kann jeder Nutzer des SCP 1700 ein abarbeitungsfähiges Programm erzeugen. Mit Hilfe der GENCMD-Parameter läßt sich dafür jedes gewünschte Speichermodell erzeugen.
SCP 1700 selbst ist als 8080-Speichermodell aufgebaut. Das bedeutet, daß alle Segmentregister auf die Basisadresse von SCP 1700 eingestellt sind.
Für unterschiedliche Plattenlaufwerke existieren Plattendefinitionstabellen. In ihnen werden die wichtigsten Parameter für ihre Anwendung angegeben.
Eine Programmbeendigung kann unter SCP 1700 auf folgende zwei Arten geschehen:

- direkte Rückkehr zum CCP (Eingabe von CTRL C)
- BDOS-Ruf mit der Funktion 0

Der Ansprung des BDOS wird vom Programm über einen Softwareinterrupt (INT 224) erreicht.
Alle dem BDOS vom Applikationsprogramm übergebenen Adreßwerte sind Offsets zum Inhalt des DS (Data Segment)-Registers.

# 3. Eingabe und Ausführung der Kommandos unter SCP 1700

## 3.1. Allgemeines

Im folgenden wird die Arbeit des Console Command Processors (CCP) dargestellt.
Das Format und die Behandlung von Nutzerprogrammen wird zusammen mit den SCP 1700-Speichermodellen und den Speicherabbildformaten diskutiert.

## 3.2. Die Kommandos des CCP

Nach dem Anfangsladen des SCP 1700 wird eine Systemanfangsmeldung und die Eingabeanforderung auf dem Consol-Gerät ausgegeben. SCP 1700 erwartet anschließend die Eingabe einer Kommandozeile.
Es kann eines der residenten Kommandos DIR, ERA, REN, TYPE oder USER sowie auch der Name eines transienten Programms verwendet werden.
Transiente Programme haben den Dateityp CMD. Vom CCP können mehrere Programme in den Speicher geladen werden. Ein transientes Programm wie z.B. der Debugger (DDT86) kann weitere Programme laden und sie dann unter seiner Steuerung starten und abarbeiten. So kann z.B ein Hintergrund-Printspooler und anschließend das DDT86 geladen werden. Das DDT86 seinerseits kann nun ein zu testendes Programm laden, dem es zwischen den Unterbrechungspunkten die Steuerung überträgt.
SCP 1700 führt ein Verzeichnis über die Reihenfolge, in der die Programme geladen wurden. Mit Eingabe von CTRL-C wird die Abarbeitung des Programms abgebrochen, das der CCP zuletzt aktivierte. Ein CTRL-C auf DDT86-Kommando-Ebene führt zum Abbruch des DDT86 und seines Testprogramms. Im o.g. Beispiel bewirkt ein zweites CTRL-C auf der Ebene des CCP den Abbruch des Hintergrund-Printspoolers. Ein drittes CTRL-C bewirkt ein Systemrücksetzen.
Wird ein CTRL-C zu einem Zeitpunkt eingegeben, bei dem keine Programme im Speicher sind, so wird ein "Platten-Reset" ausgeführt.
Empfängt SCP 1700 von einem transienten Programm oder vom CCP eine Aufforderung zum Laden eines anderen transienten Programms, kontrolliert es zunächst den Speicherplatzbedarf des zu ladenden Programms. Kann der benötigte Speicherplatz bereitgestellt werden, wird dieser dem Programm zugeordnet und das Programm geladen. Erst ein geladenes SCP 1700-Programm kann weiteren Speicher vom BDOS als Pufferplatz anfordern. Mit der Programmbeendigung gibt das SCP 1700 den Programmspeicherplatz und den zusätzlich zugewiesenen Pufferplatz frei.

## 3.3. Ausführungsmodelle für transiente Programme

Die transienten Programme müssen im Hauptspeicher nach sogenannten Speichermodellen gestaltet sein.
SCP 1700 unterstützt drei Speichermodelltypen:

- das 8080-Speichermodell
- das Small-Speichermodell
- das Compact-Speichermodell

Welches Modell genutzt wird, ist aus dem CMD-Dateikopf des transienten Programms erkenntlich. Die Initialisierung der Segmentregister ist abhängig vom gewählten Speichermodell.
Das 8080-Speichermodell wird für Programme genutzt, bei denen der Codebereich nicht vom Datenbereich getrennt ist.
Es existiert nur ein gemeinsamer Teil, der den Code-, den Daten- und den Stackbereich enthält. Die Segmentregister werden mit der Startadresse der Region initialisiert, in der dieser eine Teil angelegt ist.
Die Segmentregister können jedoch während der Ausführung eines Applikationsprogramms so verwaltet werden, daß mehrere Segmente innerhalb des Codeteiles adressiert werden können.
Im Small-Speichermodell besteht ein Programm aus einem separaten Code- und einem separaten Datenteil. Es ist gut geeignet für Programme, bei denen der Code- und Datenteil sich leicht trennen lassen. Dabei kann sowohl der Code- als auch der Datenteil aus selbstständigen 64 KByte-Segmenten bestehen.
Im Compact-Speichermodell enthält jedes Programm zusätzlich einen Extra-, Stack- oder einen Hilfsteil.
Jeder Teil kann sich aus einem oder mehreren Segmenten zusammensetzen. Falls ein beliebiger Teil größer als ein Segment ist oder falls der Hilfsteil existiert, dann muß das Applikationsprogramm während der Abarbeitung seine Segmentregister selbst ordnungsgemäß verwalten, um alle Code- und Datenbereiche adressieren zu können.
Die 3 Modelle unterscheiden sich beim Laden eines transienten Programms vor allem in der Initialisierungsart der Segmentregister. Die Programmladefunktion des Betriebssystems bestimmt anhand der verwendeten Programmteile, welches Speichermodell ein transientes Programm benötigt.

## 3.4. Das 8080 Speichermodell

Das 8080-Speichermodell wird angewendet, wenn das Transientprogramm nur einen Codeteil beinhaltet. In diesem Fall werden die Register CS, DS, ES auf den Anfang des Codeteils initialisiert während die Register SS und SP auf den 96 Byte-Stackbereich im CCP gesetzt bleiben. Der Befehlszähler (Instrucktion Pointer Register, IP) wird auf 100H gesetzt. Damit ist die Belegung der Basisseite am Beginn des Codeteils gesichert.
Infolge des Programmladens ergibt sich das 8080-Modell.
Im Bild 1 liegen die niedrigen Adressen am oberen Diagrammende.

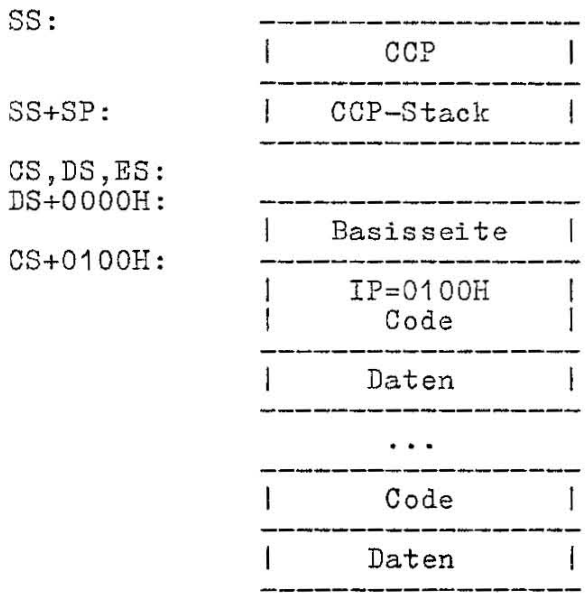



Bild 1: SCP 1700 8080 Speichermodell

Die gemischten Code- und Datenregionen lassen sich nicht voneinander unterscheiden.
Das folgende Beispiel zeigt wie ein Transientprogramm im 8080-Speichermodell codiert wird.

```
                cseg
                org     100h
                .
                .       (Code).
                .
      endCS     equ     ¤
                dseg
                org     offset    endCS
                .
                .       (Daten)
                .
                end
```

## 3.5. Das Small-Speichermodell

Das Small-Speichermodell wird angewendet, wenn das transiente Programm sowohl einen Code- als auch einen Datenteil hat. Es wird dem Codeteil die Anweisung CSEG und dem Datenbereich DSEG vorangestellt. Dabei ist der Anfang des Datensegmentes unabhängig vom Codesegment. Im Small-Speichermodell werden das Register CS auf den Beginn des Codeteils und die Register DS und ES auf den Beginn des Datenteils eingestellt. Die Register SS und SP zeigen weiterhin auf den CCP-Stack.

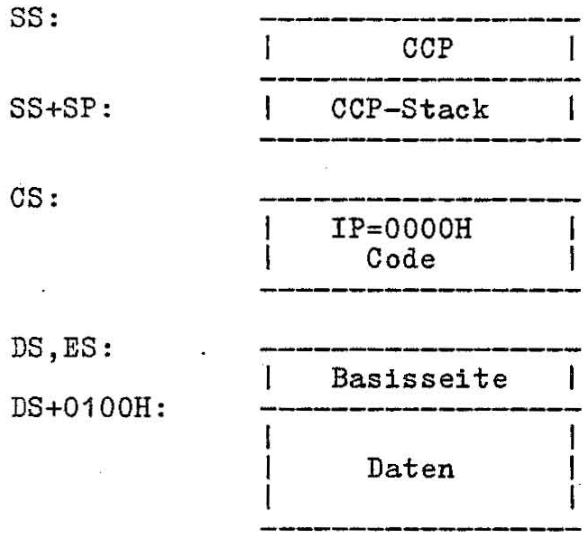



Bild 2: SCP 1700 Small Speichermodell

Der Programmcode beginnt bei CS + 0000H und der Datenbereich bei DS + 0100H. Die Basiswerte stehen ab DS + 0000H.
Das folgende Beispiel zeigt, wie ein Transientprogramm im Small-Speichermodell zu codieren ist.

```
cseg
 .
 .      (Code)
 .
dseg
org     100h
 .
 .      (Daten)
 .
end
```

## 3.6. Das Compact-Speichermodell

Das Compact-Speichermodell wird dann angewendet, wenn Code- und Datenteile zusammen mit einem oder mehreren Stack-, Extra- oder Hilfsteilen vorhanden sind. In diesem Fall werden die Register CS, DS, ES auf die Basisadressen der entsprechenden Bereiche gesetzt. Das Bild 3 zeigt die Anfangsbelegung der Segmentregister in einem Compact-Speichermodell.
Die Werte der Segmentregister können bei der Abarbeitung eines Programms geändert werden. Dazu sind die Anfangswerte, die der CCP in die Basisseite eintrug, zu laden.
Somit ist ein Zugriff zum gesamten Speicher möglich. Falls in einem Transientprogramm die Nutzung des Stackteils als Stackbereich vorgesehen ist, müssen die Register SS und SP als erstes verändert werden. Die Register SS und SP zeigen auch bei der Definition eines Stackbereichs beim Start eines Transientprogrammes in den CCP-Bereich. Obwohl es so scheint als würden die Register SS und SP den Stackbereich adressieren, gibt es gerade dort zwei Vorbehalte.
Erstens kann ein Transientprogramm den Stackbereich wie einen Datenbereich nutzen. In diesem Fall könnte eine "Far Call"-Anweisung des CCP bei der Übergabe der Steuerung an das

Transientprogramm Daten im Stackbereich überschreiben.
Zweitens würde das SS-Register logischerweise auf den Bereichsanfang und das SP-Register auf das Bereichsende zeigen. Falls der Stackbereich jedoch 64 KByte überschreitet, kann der Adressbereich vom Anfang bis Ende des Teiles nicht mehr mit einem 16-Bit-Offset im SP-Register erfasst werden.
Das folgende Beispiel verdeutlicht, wie ein Transientprogramm im Compactmodell zu programmieren ist.

```
        cseg
         .
         .      (Code)
         .
        dseg
        org     100 h
         .
         .      (Daten)
         .
        eseg
         .
         .      (weitere Daten)
         .
        sseg
         .
         .      (Stackbereich)
         .
        end
```

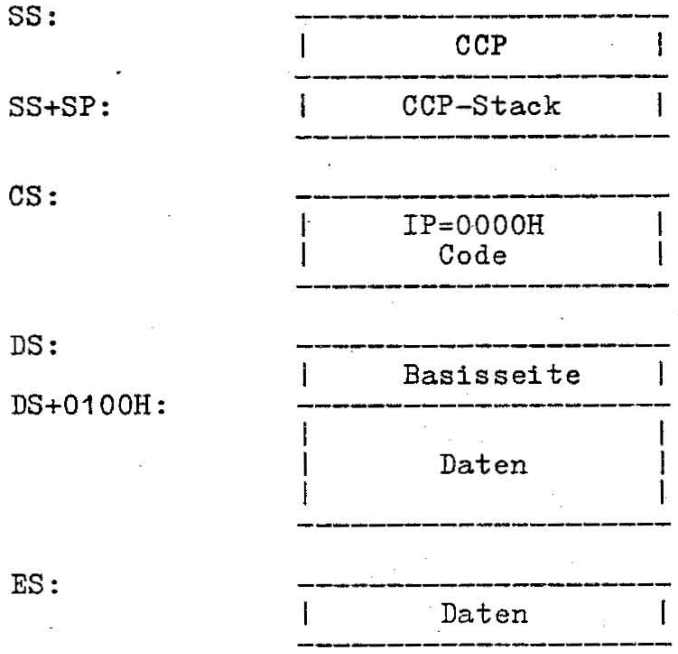



Bild 3: SCP 1700 Compact Speichermodell

## 3.7. Basisseiteninitialisierung

Die Basisseite enthält Standardwerte und Adressen, die vom CCP eingetragen und die vom Transientprogramm genutzt werden.
Die Basisseite belegt die Region zwischen dem Offset 0000H bis 00FFH relativ zum DS-Register.

| | | | |
|---|---|---|---|
| DS+0000 | LC0 | LC1 | LC2 |
| DS+0003 | BC0 | BC1 | M80 |
| DS+0006 | LD0 | LD1 | LD2 |
| DS+0009 | BD0 | BD1 | XXX |
| DS+000C | LE0 | LE1 | LE2 |
| DS+000F | BE0 | BE1 | XXX |
| DS+0012 | LS0 | LS1 | LS2 |
| DS+0015 | BS0 | BS1 | XXX |
| DS+0018 | LX0 | LX1 | LX2 |
| DS+001B | BX0 | BX1 | XXX |
| DS+001E | LX0 | LX1 | LX2 |
| DS+0021 | BX0 | BX1 | XXX |
| DS+0024 | LX0 | LX1 | LX2 |
| DS+0027 | BX0 | BX1 | XXX |
| DS+002A | LX0 | LX1 | LX2 |
| DS+002D | BX0 | BX1 | XXX |
| DS+0030 | ungenutzt | | |
| ... | | | |
| DS+005C | Standard-FCB | | |
| DS+0080 | Standardpuffer | | |
| DS+0100 | Beginn der Nutzerdaten | | |

Im obigen Bild ist jedem Byte ein Index 0, 1 und 2 zugeordnet, mit dem zwischen niederwertigen, mittleren und höherwertigen Byte unterschieden wird.
Durch XXX sind die ungenutzten Bytes gekennzeichnet.
LC ist die letzte Adresse des Codebereichs (24 Bits) wobei die 4 höchstwertigen Bits gleich Null sind.
Im 8080-Modell überschreiten das niederwertige und das mittlere Byte von LC (LC0 und LC1) niemals 0FFFFH und das höherwertige Byte (LC2) ist immer gleich Null. BC ist die Basis-Paragraphen-Adresse des Codebereichs (16-Bits). LD und BD geben die letzte Position und Paragraphen-Basis des Datenbereichs an. Die letzte Position liegt ein Byte niedriger als die Länge des Bereichs.
Das M80-Byte hat den Wert 1, wenn das 8080-Speichermodell angewendet wird. LE und BE beinhalten die Länge und die Paragraphenadresse des Extrabereichs und LS bzw. BS die Länge und die Basisadresse des Stackbereichs. Die mit LX und BX gekennzeichneten Bytes korrespondieren mit einem Satz von 4 wahlfreien, unabhängigen Bereichen, die Programme unter Anwendung des Compactspeichermodells nutzen. Ihre Anfangswerte werden aus dem Kopfsatz der Speicherabbilddatei entnommen.

## 3.8. Laden und Beenden eines transienten Programms

In einem Kommando können die Namen von zwei Dateien auftreten. Die formatierten FCB dieser Dateien legt der CCP relativ zum Register DS auf den Adressen 005CH und 006CH in der Basisseite ab.
Durch einen "Far CALL" überträgt der CCP die Steuerung an das Transientprogramm. Das Transientprogramm kann den 96 Byte-Stack des CCP nutzen und wahlweise mit Programmbeendigung durch eine "Far Return"-Anweisung direkt zum CCP zurückkehren. Ein Programmabbruch erfolgt mit Ausführung der BIOS-Funktion 0. Die BDOS-Funktion 0 kann ein Programm beenden ohne es im Speicher zu löschen oder den Speicherbelegungszustand zu ändern.
Der Bediener kann ein Programm durch Eingabe eines einzelnen CTRL-C abbrechen. Das hat den selben Effekt wie die programmtechnische Beendigung mit der BDOS-Funktion 0.

# 4. Erzeugung von CMD-Dateien

## 4.1. Allgemeines

CMD-Dateien sind Speicherabbilddateien von abarbeitungsfähigen SCP 1700-Programmen. Sie können mit Hilfe des Systemprogramms GENCMD erzeugt werden.
GENCMD verwendet als Eingabedateien solche im Hex-Format, die nach der Assemblierung durch den Assembler ASM86 gebildet werden.

## 4.2. Das Hex-Dateiformat

Eine Hex-Datei besteht aus einer festen Folge von ASCII-Sätzen folgenden Formats:

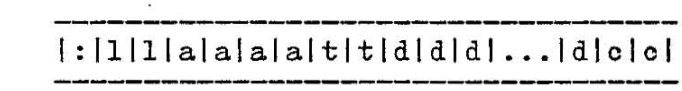



Bild 4: Hex-Dateiformat

Der Anfang jedes Satzes ist durch einen ASCII-Doppelpunkt gekennzeichnet. Jede weitere Einzelposition enthält ASCII-Hexadezimalzahlen zwischen 0 und F. Die einzelnen Felder der Sätze sind wie folgt definiert:

Tabelle 2: Hex-Datei-Felder

| Feld | Bedeutung |
|---|---|
| ll | Satzlänge 00 ... FF (0...255 dez) |
| aaaa | Ladeadresse, relativ zum aktuellen Segment |
| tt | Satztyp: |
| | 00 Datensatz, wird geladen ab Offset aaaa relativ zur aktuellen Basisparagraphenadresse |
| | 01 Dateiende, CC = FF |
| | 02 erweiterte Adresse, aaaa ist die Paragraphenadresse für die folgenden Datensätze |
| | 03 Die Anfangsadresse ist aaaa (IP wird entsprechend dem verwendeten Speichermodell gesetzt.) |
| | 81 wie 00, Daten gehören zum Code-Segment |
| | 82 " " , " " " Daten- " |
| | 83 " " , " " " Stack- " |
| | 84 " " , " " " Extra- " |
| | 85 Paragraphen Adresse für das absolute Code-S. |
| | 86 " " " " " Daten-S. |
| | 87 " " " " " Stack-S. |
| | 88 " " " " " Extra-S. |
| d | Datenbyte |
| cc | Prüfsumme aller Satzbytes |

Alle Zeichen eines Satzes, die vor dem Doppelpunkt stehen, werden ignoriert.

## 4.3. Die Arbeitsweise von GENCMD

GENCMD wird über CCP mit folgender Kommandozeile aufgerufen:

GENCMD Dateiname Parameterliste

wobei:
Dateiname: Name einer Datei im Hex-Format
Parameterliste: eine Folge von Schlüsselworten und Werten, die durch Kommas oder Leerzeichen getrennt wird.

Die möglichen Schlüsselworte sind:

8080 CODE DATA EXTRA STACK X1 X2 X3 X4

Das 8080-Schlüsselwort bewirkt die Bildung einer CMD-Datei in Form eines 8080-Speichermodells, bei der Codebereich und Datenbereich innerhalb eines Segmentes vermischt sind.
Die Form eines solchen Kommandos ist:

GENCMD Dateiname 8080

Die restlichen Schlüsselworte definieren spezielle Speicheranforderungen für jeden Segmentbereich. Sie sind nach dem Dateinamen oder der 8080-Option im Kommando anzugeben.
Den Schlüsselworten, die den jeweiligen Bereichen entsprechen, folgen Werte, die diesen zugeordnet sind. Jeder Wert ist eine Hexadezimalzahl, die eine Paragraphen-Adresse oder eine Segmentlänge in Paragrapheneinheiten darstellt. Die Werte sind in eckigen Klammern eingeschlossen und durch Kommata getrennt. Diesen Werten vorangestellt ist ein einzelner Buchstabe, der die Bedeutung jedes Wortes definiert:

Ahhhh Der Bereich wird auf die absolute Adresse hhhh geladen
Bhhhh Der Bereich beginnt auf hhhh in der Hex-Datei
Mhhhh Der Bereich erfordert eine Speichergröβe von mindestens hhhh x 16 Bytes
Xhhhh Der Bereich erfordert eine Speichergröβe von maximal hhhh x 16 Bytes

Die angegebenen Werte entsprechen denen der Basiswerte (siehe Abschnitt 3.7.).
Im allgemeinen müssen die o.g. Werte nicht angegeben werden, da diese Werte dem Hex-Dateiformat entnommen werden können. Bei folgenden Sonderfällen sind diese Angaben aber erforderlich:

- Das 8080-Schlüsselwort muβ angegeben werden, wenn der ASM86-Assembler zum Übersetzen von 8080-Parametern genutzt wurde und Code und Daten innerhalb eines Segmentes vermischt sind, unabhängig davon, ob im Quellprogramm CSEG- und DSEG-Anweisungen verwendet werden.

- Ein A-Wert (absolute Adresse) muβ für jeden Bereich angegeben werden, der auf einer absoluten Position im Speicher liegen muβ. Im Normalfall ist dieser Wert nicht definiert, da SCP 1700 nicht garantieren kann, daβ die geforderte Speicherregion verfügbar ist.

- Ein B-Wert muβ angegeben werden, wenn GENCMD Dateien verarbeitet werden, die nicht vom Assembler ASM86 erzeugt wurden. Diese Dateien enthalten im allgemeinen keine Informationen nach

der Code-, Daten-, Extra- oder Hilfsbereiche unterschieden werden können.

- Der M-Wert (Minimal-Speicherwert) wird nur angegeben, wenn die Hex-Format-Dateien nicht genau die Minimal-Speicheranforderungen des jeweiligen Bereichs definieren.
  Die Größe des Codebereichs ist im allgemeinen genau festgelegt und ergibt sich aus der Differenz der höchsten und der niedrigsten Byte-Adresse dieses Bereichs. Der Datenbereich kann jedoch am Ende nichtinitialisierten Speicher enthalten, der innerhalb des Hex-Formates der Datei nicht definiert ist. Die höchste Adresse innerhalb des Datenbereichs kann innerhalb des Quellprogramms durch die Anweisung "DB 0" als letzter Datenwert definiert werden.
  Der M-Wert kann dann genutzt werden, um zusätzlichen Speicherplatz am Ende des Bereichs zuzuweisen.
  Dasselbe trifft auf die Größen der Stack-, Extra- und Hilfsbereiche zu.

- Der X-Wert wird angegeben, wenn zusätzlicher freier Speicher für E/A-Puffer oder Symboldateien benötigt wird.

Beispiele:

1. GENCMD X CODE[A40] DATA[M30,XFFF]

   Die Datei X.H86 wird in die Datei X.CMD transformiert. Ihr Codebereich beginnt auf der Paragraphenadresse 40H. Der Datenbereich erfordert ein Minimum von 300H Bytes, kann allerdings bis zu 0FFF0H Bytes nutzen.

2.GENCMD B:Y DATA[B30,M20] EXTRA[B50] STACK[M40] X1[M40]

   Die sich auf dem Plattenlaufwerk B befindliche Datei Y.HEX wird in die Datei Y.CMD transformiert. Der Codebereich beginnt bei 0000H, der Datenbereich beginnt auf der Adresse 300H und erfordert einen Speicherplatzbedarf von 200H Bytes. Der Extrabereich beginnt ab 500H während Stack- und Hilfsbereiche jeweils 400H Bytes erfordern, aber durch keine Adreßangabe definiert sind.

## 4.4. Das CMD-Dateiformat

Eine CMD-Datei, die durch GENCMD erzeugt wurde, besteht aus einem 128 Byte Kopfsatz, dem unmittelbar das Speicherabbild folgt. Der Aufbau dieses Kopfsatzes ist im Bild 5 dargestellt.

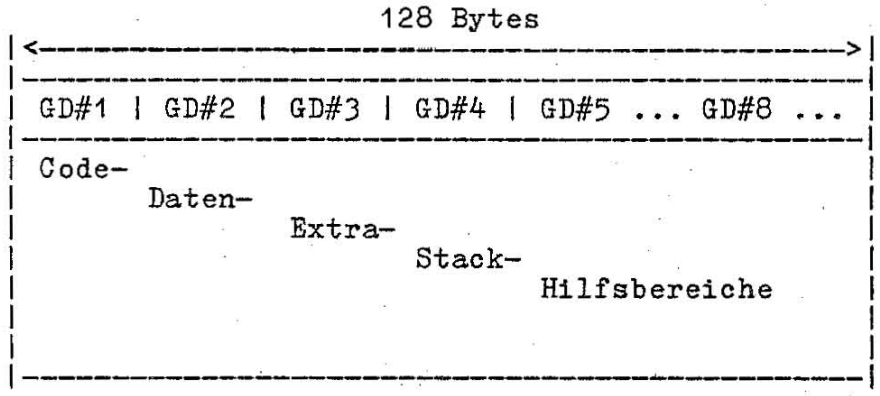



Bild 5: CMD-Datei-Kopfsatz

GD#1 ... GD#8 stellen Bereichs-Descriptoren dar, von denen jeder zu einer unabhängigen geladenen Speicherregion gehört.
Der Aufbau eines Bereichs-Descriptors ist im Bild 6 dargestellt:

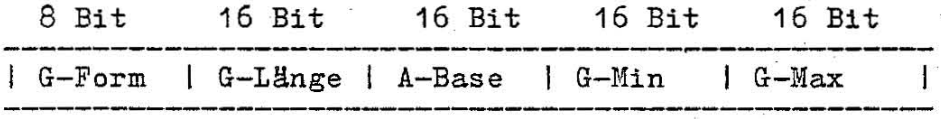



Bild 6: Bereichs-Descriptor

Dabei beschreibt G-Form das Bereichsformat oder hat den Wert 0, wenn keine weiteren Descriptoren folgen.
Der 8-Bit-Wert ist in zwei Felder unterteilt (siehe Bild 7).

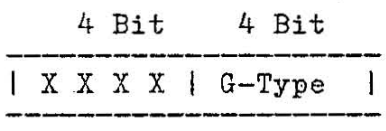



Bild 7: G-Form

Das Feld G-Type beschreibt den Bereichs-Descriptor-Typ und kann die in der Tabelle 3 genannten Werte annehmen:

Tabelle 3: Bereichs-Descriptor-Typen

| G-Type | Bereichs-Typ |
|---|---|
| 1 | Codebereich |
| 2 | Datenbereich |
| 3 | Extrabereich |
| 4 | Stackbereich |
| 5 | Hilfsbereich 1 |
| 6 | Hilfsbereich 2 |
| 7 | Hilfsbereich 3 |
| 8 | Hilfsbereich 4 |
| 9 | geteilter Codebereich |
| 10-14 | nicht verwendet |
| 15 | Escape-Code für zusätzliche Typen |

Alle weiteren Werte im Bereichs-Descriptor sind in Paragraphen-einheiten angegeben.
G-Länge enthält die Größe des Bereichs (z.B. G-Länge = 0080H, das entspricht einer Bereichsgröße von 800H bzw. 2048 (dez) Bytes).
A-Base bestimmt die Basis-Paragraphenadresse eines nicht verschieblichen Bereichs.
G-Min und G-Max enthalten die minimale bzw. maximale Größe des Speicherbereiches, der dem Bereich zugewiesen wurde.
Für den Ladevorgang des SCP 1700 ist die CMD-Datei durch ihren Kopfsatz vollständig beschrieben. Beim Ladevorgang wird bei Vorhandensein nur eines Codebereichs ein 8080-Speichermodell und bei Vorhandensein sowohl eines Code- als auch eines Datenbereichs ein Small-Speichermodell gebildet. In allen anderen Fällen wird ein Compakt-Speichermodell erzeugt.

# 5. Funktionen des BDOS

## 5.1. Allgemeines

Der Nutzer kann über Steuerprogrammrufe, sogenannte BDOS-Rufe, mit dem SCP 1700 in Verbindung treten. Über diese BDOS-Rufe können von den Applikationsprogrammen die Funktionen des BDOS genutzt werden. Diese Funktionen betreffen Ein-/Ausgabe-Anweisungen für die angeschlossenen Geräte, Organisationsanweisungen und Dateioperationen für die Plattengeräte. Im folgenden werden die einzelnen BDOS-Rufe ausführlich beschrieben.

## 5.2. BDOS-Parameter und Funktionscodes

Der Eintritt in das BDOS wird vom Applikationsprogramm mittels des Softwareinterrupts INT 224 erreicht. Die Parameterversorgung zur Ausführung der jeweiligen gewünschten BDOS-Funktion hat über die folgenden Register zu erfolgen.

CL : Funktionscode
DL : Byteparameter
DX : Wortparameter

Nach der Ausführung der Funktion werden die etwaigen Ausgabeparameter in den folgenden Registern dem Applikationsprogramm übergeben.

AL : Bytewerte
AX und BX : Wortwerte
ES und BX : Doppelwort (Offset in BX, Segmentadr. in ES)

Bei Eintritt in das BDOS werden vom BDOS alle Segmentregister, außer ES, gerettet und nach Ausführung der Funktion wieder rückgespeichert.

Tabelle 4: BDOS-Funktionen

| Nr. | Funktion |
|---|---|
| 0 | System rücksetzen |
| 1 | Eingabe eines Zeichens vom Gerät CONSOLE |
| 2 | Ausgabe eines Zeichens auf dem Gerät CONSOLE |
| 3 | Eingabe eines Zeichens vom Gerät AXI |
| 4 | Ausgabe eines Zeichens auf dem Gerät AXO |
| 5 | Ausgabe eines Zeichens auf dem Gerät LIST |
| 6 | Direkt-Ausgabe mit dem Gerät CONSOLE |
| 7 | Abfrage des I/O-Bytes |
| 8 | Setzen des I/O-Bytes |
| 9 | Ausgabe einer Zeichenkette auf dem Gerät CONSOLE |
| 10 | Eingabe einer Zeichenkette auf dem Gerät CONSOLE |
| 11 | Abfrage des Status des Gerätes CONSOLE |
| 12 | Abfrage der Versionsnummer des SCP 1700 |
| 13 | Rücksetzen des Dateisystems |
| 14 | Auswahl des Plattenlaufwerkes |
| 15 | Datei eröffnen |
| 16 | Datei schließen |
| 17 | Suchen nach dem ersten FCB einer Datei |
| 18 | Suchen nach dem jeweils nächsten FCB einer Datei |
| 19 | Datei löschen |
| 20 | Sequentielles Lesen eines logischen Satzes |
| 21 | Sequentielles Schreiben eines logischen Satzes |
| 22 | Einrichten einer Datei |
| 23 | Umbenennen einer Datei |
| 24 | Lesen des LOGIN-Vektors |
| 25 | Abfrage der aktuellen Plattenlaufwerknummer |
| 26 | Festlegen der aktuellen DMA-Adresse |
| 27 | Abfrage der Adresse des Zuteilungsvektors |
| 28 | Einstellen des Schreibschutzes für das aktuelle Plattenlaufwerk |
| 29 | Lesen des READ-ONLY-Vektors |
| 30 | Festlegen der Dateiattribute |
| 31 | Abfrage der Adresse der Tabelle der Plattenparameter |
| 32 | Einstellen/Abfragen des Nutzercodes |
| 33 | Direktes Lesen eines logischen Satzes |
| 34 | Direktes Schreiben eines logischen Satzes |
| 35 | Berechnen der Dateigröße |
| 36 | Bestimmen des nächsten direkt zu lesenden/ schreibenden Satzes |
| 37 | Rücksetzen des Plattenlaufwerkes |
| 40 | Direktes Schreiben leerer Sätze |
| 47 | Start eines anderen Programms |
| 50 | Direkter BIOS-Ruf |
| 51 | Einstellen der Segmentadresse des DMA-Bereichs |
| 52 | Abfrage der aktuellen DMA-Adresse |
| 53 | Ermittlung der Anfangsadresse einer Region definierter Länge |
| 54 | Test auf Verfügbarkeit einer definierten Region |
| 55 | Zuweisung von Speicherbereich für eine Region definierter Länge |
| 56 | Zuweisung von Speicherbereich für eine definierte Region |
| 57 | Freigabe von Speicherbereich |
| 58 | Löschen des gesamten zugewiesenen Speicherbereichs |
| 59 | Laden einer CMD-Datei |

Nach der Funktion werden die BDOS-Rufe in einfache BDOS-Rufe, BDOS-Dateioperationen und erweiterte Operationen für Speicherverwaltung und Programmladen unterschieden.

## 5.3. Einfache BDOS-Rufe

### 5.3.1. System rücksetzen

Funktion 0 SYSTEM RESET
Eingangsparameter:
CL: 00H
DL: Abbruchcode
Ausgangsparameter:
Das jeweilige Applikationsprogramm ist beendet.

Mit dem BDOS-Ruf "System rücksetzen" kann ein Applikationsprogramm beendet werden. Die Steuerung wird an das SCP 1700 auf CCP-Kommandoebene zurückgegeben. Der Abbruchcode im Register DL kann folgende Werte annehmen:

DL=00H Das den BDOS-Ruf enthaltende Programm wird beendet und die Steuerung an den CCP zurückgegeben.

DL=01H Das den BDOS-Ruf enthaltende Programm wird beendet, die Steuerung wird an den CCP zurückgegeben aber das Programm verbleibt weiterhin im Speicher, der Speicherzuweisungszustand bleibt unverändert.

### 5.3.2. Eingabe eines Zeichens vom Gerät CONSOLE

Funktion 1 CONSOLE INPUT
Eingangsparameter:
CL: 01H
Ausgangsparameter:
AL: eingegebenes ASCII-Zeichen

Es wird ein Zeichen von der logischen Terminaleinheit (CONSOLE) eingelesen und dem Programm im Register AL übergeben. Alle Zeichen auch CR, LF und BS (Backspace) werden auf der Terminaleinheit geechot. Tabulatoren werden auf 8 Zeichenstellen berechnet. Das rufende Programm befindet sich bis zur Übernahme des Zeichens im Wartezustand.

### 5.3.3. Ausgabe eines Zeichens auf das Gerät CONSOLE

Funktion 2 CONSOLE OUTPUT
Eingangsparameter:
CL: 02H
DL: auszugebendes ASCII-Zeichen
Ausgangsparameter:
Ausgabe des Zeichens

Das im Register DL eingeschriebene Zeichen wird auf dem logischen Terminalgerät CONSOLE ausgegeben. Tabulatoren werden auf 8 Zeichenstellen berechnet. Zusätzlich zur Ausgabe des Zeichens wird ein Test des Eingabestatus der Tastatur durchgeführt. Nach Eingabe eines CTRL-S wird die weitere Ausgabe von Zeichen solange blockiert, bis ein CR eingegeben wird.

### 5.3.4. Eingabe eines Zeichens vom logischen Gerät AXI

Funktion 3 READER INPUT
Eingangsparameter:
CL: 03H
Ausgangsparameter:
AL: eingegebenes ASCII-Zeichen

Es wird ein Zeichen vom logischen Gerät AXI eingelesen und dem Programm im Register AL übergeben. Das rufende Programm befindet sich bis zur Übernahme des Zeichens im Wartezustand.

### 5.3.5. Ausgabe eines Zeichens auf dem logischen Gerät AXO

Funktion 4 PUNCH OUTPUT
Eingangsparameter:
CL: 04H
DL: auszugebendes Zeichen
Ausgangsparameter:
Ausgabe des Zeichens

Das im Register DL übergebene Zeichen wird auf dem logischen Gerät AXO ausgegeben.

### 5.3.6. Ausgabe eines Zeichens auf dem Gerät LIST

Funktion 5 LIST OUTPUT
Eingangsparameter:
CL: 05H
DL: auszugebendes ASCII-Zeichen
Ausgangsparameter:
Ausgabe des Zeichens

Das im Register übergebene Zeichen wird auf dem logischen Druckgerät LIST ausgegeben.

### 5.3.7. Direktein- /ausgabe mit dem Gerät CONSOLE

Funktion 6 DIRECT CONSOLE I/O
Eingangsparameter:
CL: 06H
DL: 0FFH bei Eingabe Zeichen
0FEH bei Eingabe Status
ASCII-Zeichen bei Ausgabe
Ausgabeparameter:
AL: eingegebenes Zeichen/Status

Mit dem BDOS-Ruf der Funktion 6 wird eine Direktein- /ausgabe mit dem logischen Terminalgerät (CONSOLE) durchgeführt.
Je nach Angabe im Register DL wird ein Zeichen eingegeben oder ausgegeben bzw. der Gerätestatus abgefragt.
Bei der Eingabe werden die eingegebenen Zeichen nicht geechot und bei der Ausgabe die Steuerzeichen nicht überprüft und ausgewertet (z.B. CTRL-S, CTRL-P).
Wird durch die Angabe 0FEH im Register DL der Eingabestatus getestet, so wird im Register AL folgende Information dem Nutzer übergeben:

AL=00 Es ist kein Zeichen für die Eingabe bereit
AL=FF Ein Eingabezeichen ist bereit zur Übernahme

### 5.3.8. Abfrage des I/O-Bytes

Funktion 7 GET I/O-BYTE
Eingangsparameter:
CL: 07H
Ausgangsparameter:
AL: Wert des I/O-BYTES

Nach Ausführung der BDOS-Funktion wird dem Applikationsprogramm der aktuelle Wert des I/O-Bytes im Register AL übergeben. Das I/O-Byte enthält die aktuellen Zuweisungen für die logischen Geräte CONSOLE, AXI, AXO und LIST (siehe Abschnitt 6.3.)!

### 5.3.9. Setzen des I/O-Bytes

Funktion 8 SET I/O-BYTE
Eingangsparameter:
CL: 08H
DL: einzustellender I/O-Byte-Wert
Ausgangsparameter:
Das I/O-Byte ist neu eingestellt.

Diese BDOS-Funktion stellt den aktuellen Wert des I/O-Bytes ein. Der einzustellende Wert wird im Register DL dem BDOS übergeben. Nach Ausführung der Funktion sind die aktuellen Zuweisungen für die logischen Geräte CONSOLE, AXI, AXO und LIST modifiziert.

### 5.3.10. Ausgabe einer Zeichenkette auf dem Gerät CONSOLE

Funktion 9 PRINT STRING
Eingangsparameter:
CL: 09H
DX: Startadresse des Puffers in der die Zeichenkette steht.
Ausgangsparameter:
Ausgabe der spezifizierten Zeichenkette

Mit dieser Funktion wird eine im Register DX spezifierte Zeichenkette auf dem logischen Gerät CONSOLE ausgegeben. Die Ausgabe erfolgt solange, bis ein "¤" in der Zeichenkette gefunden wird. Tabulatoren werden auf 8 Zeichenstellen berechnet und als Leerzeichen ausgegeben. Vor Ausgabe eines Zeichens wird die Tastatur auf Eingabezeichen überprüft. Nach Eingabe eines CTRL-S wird die Ausgabe bis zur Eingabe eines CR gestoppt. Nach Eingabe eines CTRL-P wird die Ausgabe auf dem logischen LIST-Gerät geechot.

### 5.3.11. Eingabe einer Zeichenkette vom Gerät CONSOLE

Funktion 10 READ CONSOLE BUFFER
Eingangsparameter:
CL: 0AH
DX : Anfangsadresse des Eingabepuffers im AP
Ausgangsparameter:
Die eingegebenen Zeichen befinden sich im angegebenen Puffer.

Es wird eine Zeichenkette vom logischen Gerät CONSOLE in den im Register DX spezifizierten Puffer eingelesen. Dieser Eingangspuffer ist wie folgt aufzubauen:

| DX: | +0 | +1 | +2 | +3 | +4 | +5 | +6 | | +n |
|---|---|---|---|---|---|---|---|---|---|
| | mx | nc | c1 | c2 | c3 | c4 | c5 | | cn |

wobei:

mx: maximale Zeichenzahl die in den Puffer eingetragen werden kann (maximaler Wert 255).
nc: laufender Bytezähler der eingelesenen Bytes.
c1...cn: eingelesene Bytes

Die Eingabe vom logischen Gerät CONSOLE wird beendet, wenn entweder der Eingabepuffer gefüllt ist (mx=nc), oder die Steuerzeichen CR (CTRL-M) bzw. LF (CTRL-J) in der Eingabezeichenfolge erkannt werden.
Der Wert mx muß vor dem Ruf der BDOS-Funktion gesetzt sein und kann im Bereich von 1 bis 255 liegen. Der Wert nc wird dem Nutzer nach Rückkehr ins rufende Programm übergeben und kann im Bereich von 0 bis mx liegen.
Ein Endezeichen (CR oder LF) wird nicht in den Puffer übertragen und ist auch nicht im Zähler nc enthalten.
Im Fall nc < mx folgen dem letzten eingegebenen Zeichen im Puffer undefinierte Werte. Während der Eingabe werden vom BDOS die in Tabelle 5 aufgeführten Steuerzeichen ausgewertet und behandelt.

Tabelle 5: Steuerzeichen, die von der BDOS Funktion 10 behandelt werden

| Steuerzeichen | Funktion |
|---|---|
| Rubout/DEL | löscht und echot das zuletzt eingegebene Zeichen |
| CONTROL-C | führt zu einem System-Reset des SCP 1700, wenn es am Beginn der Zeile eingegeben wird |
| CONTROL-E | veranlaßt die weitere Eingabe auf der nächsten physischen Zeile |
| CONTROL-H | Rückpositionierung um eine Zeichenposition |
| CONTROL-J | beendet die Eingabezeile (LF) |
| CONTROL-M | beendet die Eingabezeile (CR) |
| CONTROL-R | veranlaßt ein nochmaliges Schreiben der aktuellen Zeile |
| CONTROL-U | löschen der aktuellen Zeile |
| CONTROL-X | Rückpositionierung an den Beginn der aktuellen Zeile |

Bestimmte Steuerzeichen, die den Cursor des aktuellen logischen Gerätes CONSOLE beim Echoen der eingegebenen Zeile an die äußerste linke Position zurückbewegen (z. B.: CTRL-X), führen den Cursor nur bis zu der Position, wo die Systemmeldung (promt) endet.

### 5.3.12. Abfrage des Eingabestatus des Gerätes CONSOLE

Funktion 11 GET CONSOLE STATUS
Eingangsparameter:
CL: 0BH
Ausgangsparameter:
AL: Eingabestatus des Gerätes CONSOLE
AL: 01H Zeichen verfügbar
AL: 00H kein Zeichen verfügbar

Mit Hilfe der BDOS-Funktion 11 kann festgestellt werden, ob im logischen Gerät CONSOLE ein eingegebenes Zeichen zur Übernahme bereitsteht. Nach Ausführung des BDOS-Rufes steht im Register AL der Eingabestatus.

### 5.3.13. Abfrage der Betriebssystemsversionsnummer

Funktion 12 RETURN VERSION NUMBER
Eingangsparameter:
CL: 0CH
Ausgangsparameter:
BX: Versionsnummer

Ueber diesen BDOS-Ruf kann die Versionsnummer des SCP 1700 abgefragt werden.

## 5.4. BDOS-Dateioperationen

### 5.4.1. Allgemeines

Die Funktionen 13 bis 52 beziehen sich auf Plattendateioperationen des SCP 1700. Bei vielen dieser Operationen wird in DX die Adresse des Dateisteuerblocks (FCB) übergeben. Der FCB ist für Dateioperationen mit sequentiellem Zugriff 33 Bytes und für solche mit Direktzugriff 36 Bytes lang.
Der Standard-FCB steht normalerweise ab Offset 005CH relativ zum DS-Register.

Tabelle 6: Standard FCB

| Offset | Länge in Bytes | Name | Inhalt |
|---|---|---|---|
| 0 | 1 | dr | Laufwerkcode (0-16)<br>0 = Standardlaufwerk<br>1 = Laufwerk A<br>2 = Laufwerk B<br>.<br>.<br>.<br>16= Laufwerk P |
| 1 | 8 | f1...f8 | Dateiname linksbündig in ASCII-Zeichen, wird mit 20H (Leerzeichen) aufgefüllt |
| 9 | 3 | t1,t2,t3 | Dateityp, in ASCII-Zeichen.<br>Die höchstwertigen Bits (Bit 7) der Bytes t1 und t2 haben folgende Bedeutung:<br>Bit 7 von t1=1: READ/ONLY-Datei<br>Bit 7 von t2=1: SYS-Datei |
| 12 | 1 | ex | Ein FCB beschreibt einen Dateibereich, dessen Größe von der Blockmaske BLM (siehe Abschnitt 7) abhängt(bei 5,25"-Disketten beträgt sie 32k Byte). Das Byte ex enthält die aktuelle Bereichsnummer, d.h. die Angabe um den wievielten Bereich einer Datei es sich handelt. ex nimmt Werte von 0...31 an, wird normalerweise vom Nutzer auf Null gesetzt. |
| 13 | 2 | s1,s2 | s1,s2 sind reserviert für interne Systembenutzung; s2 wird durch die BDOS-Funktionen OPEN, MAKE und SEARCH auf Null gesetzt. |
| 15 | 1 | rc | Satzzähler innerhalb eines Dateibereiches (ex), nimmt Werte von 0 bis 128 an.<br>rc enthält die Satzanzahl, die durch diesen FCB belegt wird. |
| 16 | 16 | d0...dn | reserviert für interne Systembenutzung; (wird von SCP 1700 gefüllt). |

Tabelle 6: Standard FCB (Fortsetzung)

| Offset | Länge in Bytes | Name | Inhalt |
|---|---|---|---|
| 32 | 1 | cr | Satzzähler, enthält die Nummer des aktuellen Satzes einer sequentiellen Dateioperation (lesen oder schreiben), wird normalerweise vom Nutzer auf Null gesetzt. |
| 33 | 3 | r0,r1,r2 | enthält bei Direktzugriffsoperationen die aktuelle Satznummer im Bereich von 0...65535 wobei in r0 und r1 ein 16 Bit-Wert steht und in r2 der Überlauf<br>r0 = niederwertiges Byte<br>r1 = höherwertiges Byte |

SCP 1700 führt Verzeichnisoperationen in einem reservierten Speicherbereich aus, so daß der Schreibpufferinhalt nicht beeinflußt wird. Ausgenommen davon sind die Funktionen SEARCH und SEARCH NEXT, hier wird der FCB auf der angegebenen DMA-Adresse abgespeichert. Treten während der BDOS-Dateiverarbeitung Fehler auf, so wird folgende Fehlermitteilung auf dem aktuellen logischen Gerät CONSOLE ausgegeben:

BDOS ERR ON x: error

wobei:

x = Name des Laufwerks
error = Fehlercode FC
FC = BAD SECTOR
SELECT
R/O

Bedeutung der Fehlermitteilungen:

- FC = BAD SECTOR Bei der Ausführung des BIOS-Rufes Sektor lesen bzw. Sektor schreiben trat ein Hardwarefehler auf. Das Nutzerprogramm wird nach der Ausschrift solange blockiert, bis eine der folgenden Eingaben durch den Rechner erfolgt ist:

  CTRL-C - Abbruch des Programms
  CR - Ignorieren des Fehlers und Weiterbearbeitung des Programms

- FC = SELECT Das angewählte Plattenlaufwerk wird vom SCP 1700 nicht unterstützt. Das Nutzerprogramm wird abgebrochen und die Steuerung an das CCP übergeben.

- FC = R/O (Schreibschutz) Es wurde ein Schreibversuch auf einem schreibgeschützten Plattenlaufwerk ausgeführt. Nach der Ausschrift wird das Nutzerprogramm blockiert und nach Eingabe eines beliebigen Zeichens auf der Tastatur abgebrochen. Die

Steuerung wird an das CCP übergeben.

### 5.4.2. Beschreibung der einzelnen Funktionen

#### 5.4.2.1. Rücksetzen des Dateisystems

Funktion 13 RESET DISK SYSTEM
Eingangsparameter:
CL: 0DH
Ausgangsparameter:
Das Dateisystem ist in einen definierten Anfangszustand gesetzt.

Diese Funktion wird benutzt, um das Dateisystem vom Programm her in einen definierten Anfangszustand zu setzen. Alle Laufwerke haben den READ/ONLY-Status, Laufwerk A ist ausgewählt. Diese Funktion kann z. B. durch ein Anwenderprogramm genutzt werden, welches Plattenwechsel während der Abarbeitung fordert. Anstelle dieser Funktion kann auch die Funktion RESET DRIVE (37) verwendet werden.

#### 5.4.2.2. Auswahl des Plattenlaufwerkes

Funktion 14 SELECT DISK
Eingangsparameter:
CL: 0EH
DL: n n=1...15(für Laufwerke A,B...,P)
Ausgangsparameter:
Das spezifizierte Laufwerk ist als Standardlaufwerk ausgewählt.

Das in DL angegebene Laufwerk wird als Standardlaufwerk für Dateioperationen ausgewählt. DL ist 0, 1,..., 15 für die Laufwerke A, B,..., P. Zusätzlich wird das Laufwerk dem System bekannt gemacht (logged-in), wenn es rückgesetzt war, d. h. bis zum nächsten System-Reset, Rücksetzen des Dateisystems bzw. bis zum nächsten Laufwerkrücksetzen bleibt dieses Laufwerk ausgewählt, das Verzeichnis dieses Laufwerkes ist aktiv. FCB mit dem Laufwerkcode Null (dr=0) beziehen sich automatisch auf dieses ausgewählte Standardlaufwerk. Die Laufwerkcodes 1 bis 16 ignorieren den Standard und beziehen sich direkt auf die Laufwerke A bis P.

#### 5.4.2.3. Datei eröffnen

Funktion 15 OPEN FILE
Eingangsparameter:
CL: 0FH
DX: FCB-Adresse
Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=FF Datei nicht gefunden
RC=0...3 Datei gefunden

Mit dieser Funktion wird ein FCB einer Datei aktiviert, die im Verzeichnis einer Platte für den aktiven aktuellen Nutzercode existiert. BDOS durchsucht das Verzeichnis des im FCB angegebenen Plattenlaufwerks (Byte dr) entsprechend der im DX angegebenen FCB-Adresse (Bytes dr bis ex des FCB), wobei ein Fragezeichen (hex 3F) an einer beliebigen Stelle im FCB als Übereinstimmung

mit dem Zeichen an der gleichen Stelle in der Verzeichniseintragung gewertet wird. Das Byte "ex" ist vorher auf Null zu setzen. Wird eine Verzeichniseintragung gefunden, so werden die zugehörigen Informationen in die Bytes d0 bis dn des FCB übertragen.

### 5.4.2.4. Datei schließen

```
Funktion 16   CLOSE FILE
        Eingangsparameter:
                C1: 10H
                DX: FCB-Adresse
        Ausgangsparameter:
                AL: RC        (Rückkehrcode)
                    RC=FF     Datei nicht gefunden
                    RC=0...3  Datei erfolgreich geschlossen
```

Diese Funktion ist die Umkehrung der OPEN-Funktion. Vorausgesetzt, der durch DX adressierte FCB wurde vorher durch eine OPEN- oder MAKE-Funktion (Funktion 15 und 22) aktiviert, überträgt die CLOSE-Funktion den derzeitigen Stand des FCB in das zugehörige Plattenverzeichnis. Der FCB-Vergleichsprozeß entspricht dem beim OPEN, ebenso haben die Rückkehrcodes 0 bis 3 (erfolgreich) und FF (Fehler) die gleiche Bedeutung. Wurden nach der Dateieröffnung nur Leseoperationen ausgeführt, muß die Datei nicht abgeschlossen werden; sind Schreiboperationen ausgeführt worden, muß sie abgeschlossen werden.

### 5.4.2.5. Suchen nach dem ersten FCB einer Datei

```
Funktion 17   SEARCH FOR FIRST
        Eingangsparameter:
                CL; 11H
                DX: FCB-Adresse
        Ausgangsparameter:
                AL: RC        (Rückkehrparameter)
                    RC=FF     Datei nicht gefunden
                    RC=0...3  Datei vorhanden
```

Diese Funktion durchsucht das Plattenverzeichnis nach dem ersten FCB einer Datei, deren FCB-Adresse in DX steht. Wird die Datei nicht gefunden, steht in AL der Wert FF, sonst 0, 1, 2 oder 3. Ist die Datei vorhanden, d.h. der FCB wurde auf der durchsuchten Platte gefunden, so wird der FCB in den Puffer ab der gültigen DMA-Adresse eingeschrieben, wobei seine Anfangsadresse sich aus:
Inhalt von AL multipliziert mit 32 ergibt

Der FCB kann ab dieser Adresse aus dem Puffer gelesen werden.
Ein Fragezeichen an einer beliebigen Stelle innerhalb "f1" bis "ex" führt zur Übereinstimmung (Gleichheit) an der gleichen Position der Verzeichniseintragung. Wenn im Byte "dr" ein Fragezeichen angegeben ist, so wird das Verzeichnis der Diskette auf dem Standardlaufwerk durchsucht, unabhängig vom Nutzercode. Ist in "dr" kein Fragezeichen enthalten, wird das Byte "s2" automatisch auf Null gesetzt.

### 5.4.2.6. Suchen nach dem jeweils nächsten FCB einer Datei

Funktion 18 SEARCH FOR NEXT

Eingangsparameter:
CL: 12H

Ausgangsparameter:
AC: RC (Rückkehrcode)
RC=FF keine Datei gefunden
RC=0...3 Datei gefunden

Diese Funktion wird nach SEARCH FOR FIRST angewendet und durchsucht das Verzeichnis nach dem jeweils nächsten FCB einer Datei. Diese Funktion läuft genau so ab wie SEARCH FOR FIRST, nur, daß das Durchsuchen des Verzeichnisses nach dem letzten gefundenen FCB beginnt. Wenn kein weiterer FCB gefunden wird, so wird im AL-Register der Wert FF zurückgeliefert. In der Anweisungsfolge muß SEARCH FOR NEXT entweder SEARCH FOR FIRST oder einem anderen SEARCH FOR NEXT folgen. Es muß aber beachtet werden, daß zwischen zwei dieser Rufe keine plattenbezogene BDOS-Rufe liegen dürfen.

### 5.4.2.7. Datei löschen

Funktion 19 DELETE FILE

Eingangsparameter:
CL: 13H
DX: FCB-Adresse

Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=FF Datei nicht vorhanden
RC=00 erfolgreiche Ausführung

Die angegebene Datei (FCB-Adresse in DX, Dateiname im FCB) wird aus dem Verzeichnis der Platte des angegebenen Laufwerkes gelöscht. Dateiname und Dateityp können Fragezeichen enthalten; das Laufwerk ist direkt anzugeben. Bei erfolgreicher Ausführung ist der Rückkehrcode Null. Ist die Datei nicht vorhanden, wird im Register AL der Wert FF zurückgeliefert.

### 5.4.2.8. Sequentielles Lesen eines logischen Satzes

Funktion 20 READ SEQUENTIAL

Eingangsparameter:
CL: 14H
DX: FCB-Adresse

Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=00 erfolgreiche Ausführung
RC=01 kein weiterer Satz vorhanden (normalerweise Dateiende); lesen eines nicht geschriebenen Satzes; Zugriff zu einem nicht vorhandenen Bereich (durch WRITE RANDOM entstanden)

Von der Datei wird ein 128Byte-Satz gelesen und ab der aktuellen DMA-Adresse im Speicher abgespeichert. Der Satz wird im FCB adressiert durch die Bytes "cr" (Satznummer) und "ex" (Bereichsnummer). Anschließend wird die Satznummer erhöht. Falls "cr" größer als 128 wird, wird automatisch der nächste Bereich eröffnet (Inhalt von "ex" um 1 erhöhen) und die Satznummer auf Null

gesetzt (Byte "cr").
Will der Nutzer das Lesen mit dem ersten Satz der Datei beginnen, so muß er nach OPEN die Satznummer selbst auf Null setzen.
Vor Ausführung der READ SEQUENTIAL-Funktion muß der FCB mittels OPEN oder MAKE aktiviert worden sein.

### 5.4.2.9. Sequentielles Schreiben eines logischen Satzes

Funktion 21 WRITE SEQUENTIAL
Eingangsparameter:
CL: 15H
DX: FCB-Adresse
Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=00 erfolgreiche Ausführung
RC=01 kein verfügbarer Platz im Verzeichnis (Es soll ein neuer Bereich eröffnet werden, aber im Verzeichnis ist kein Platz für eine neue FCB-Eintragung
RC=02 kein verfügbarer Platz auf der Platte (alle Datenblöcke belegt)

Analog zu READ SEQUENTIAL (Funktion 20) wird ab der aktuellen DMA-Adresse ein 128Byte-Satz auf die Platte geschrieben.

### 5.4.2.10. Einrichten einer Datei

Funktion 22 MAKE FILE
Eingangsparameter:
CL: 16H
DX: FCB-Adresse
Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=0...3 erfolgreiche Ausführung
RC=FFH kein verfügbarer Platz im Verzeichnis

Diese Funktion ähnelt der OPEN-Funktion. Die Datei darf aber auf der Platte des ausgewählten Laufwerkes noch nicht vorhanden sein BDOS erstellt die Datei, trägt sie in das Verzeichnis ein und initialisiert Speicherinhalt zu einer leeren Datei. Der Anwender muß selbst absichern, daß noch keine Datei mit dem gleichen Namen existiert (notfalls vorher mit DELETE löschen). Im Anschluß an die MAKE-Funktion ist die Datei eröffnet und der FCB aktiviert (OPEN ist nicht notwendig).

### 5.4.2.11. Datei umbenennen

Funktion 23 RENAME FILE
Eingangsparameter:
CL: 17H
DX: FCB-Adresse
Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=FF Datei nicht vorhanden
RC=00 erfolgreiche Ausführung

Bei den Verzeichniseintragungen der angegebenen Datei wird der Dateiname ausgetauscht. Der mit dem Register DX adressierte FCB hat hier einen anderen Aufbau:

```
Byte 0      = Laufwerksangabe
     1-15   = alte Dateibezeichnung
     16     = unbenutzt
     17-33  = neue Dateibezeichnung
```

Der Anwender muß dafür sorgen daß SCP 1700 gerechte Dateibezeichnungen gewählt werden.

### 5.4.2.12. Lesen des LOGIN-Vektors

```
Funktion 24   RETURN LOGIN VECTOR
        Eingangsparameter:
                CL: 18H
        Ausgangsparameter:
                BX: LOGIN-Vektor
```

Der LOGIN-Vektor ist ein 16-Bit-Wort, in dem alle verfügbaren Platten-Laufwerke gekennzeichnet sind, wobei Bit 0 das Laufwerk A und Bit 15 das Laufwerk P repräsentiert. Im LOGIN-Vektor, der im Register BX nach Ausführung der Funktion steht, sind alle die Bit auf "1" gesetzt, deren entsprechende Laufwerke on-line sind. Alle anderen Bits sind Null (Laufwerke sind nicht on-line). Der LOGIN-Vektor wird gesetzt durch die Ausführung der BDOS-Funktion 14 (SELECT DISK) oder durch die Ausführung einer anderen BDOS-Funktion, die die SELECT DISK-Funktion einschließt.

### 5.4.2.13. Abfrage der aktuellen Laufwerknummer

```
Funktion 25   RETURN CURRENT DISK
        Eingangsparameter:
                CL: 19H
        Ausgangsparameter:
                AL= aktuelle Laufwerknummer
```

Im Register AL steht nach Ausführung dieser Funktion die Nummer des aktuellen Standardlaufwerks; 0 für A, 1 für B usw. bis 15 für Laufwerk P.

### 5.4.2.14. Festlegen der aktuellen DMA-Adresse

```
Funktion 26   SET DMA ADRESS
        Eingangsparameter:
                CL: 1AH
                DX: DMA-Adresse
        Ausgangsparameter:
                die aktuelle DMA-Adresse ist eingestellt.
```

Es wird die Offset-Adresse des aktuellen DMA-Bereichs für Lese- und Schreiboperationen spezifiziert. Das ist die Adresse, ab der der 128 Byte lange Satz vor einem WRITE bzw. nach einem READ steht. Zur Einstellung der kompletten DMA-Adresse ist neben SET DMA ADDRESS die Funktion SET DMA SEGMENT BASE (Funktion 51) erforderlich. Diese DMA-Adresse bleibt bis zur nächsten Ausführung von SET DMA ADDRESS bestehen.

### 5.4.2.15. Abfrage der Adresse des Zuteilungsvektors

Funktion 27 GET ADDR (ALLOC)

Eingangsparameter:
CL: 1BH

Ausgangsparameter:
BX: Offset-Adresse des Zuteilungsvektors
ES: Segment-Adresse

Nach Ausführung der Funktion 27 wird dem Anwenderprogramm über die Register BX (Offset) und ES (Segment) die Adresse des Zuteilungsvektors mitgeteilt. Der Zuteilungsvektor ist Bestandteil des BIOS und besteht aus einer Anzahl von 16-Bit-Worten in denen jedes Bit einen Datenblock der spezifizierten Platte repräsentiert. Gesetzte Bits bedeuten, daß der zugeordnete Datenblock belegt ist. Der Inhalt des Vektors kann falsch sein, wenn die Platte sich im Zustand READ/ONLY befindet.

### 5.4.2.16. Einschalten Schreibschutz

Funktion 28 WRITE PROTECT DISK

Eingangsparameter:
CL: 1CH

Ausgangsparameter
das aktuelle Disk-Laufwerk ist schreibgeschützt

Das aktuell ausgewählte Plattenlaufwerk wird schreibgeschützt. Schreibversuche auf diese Platte führen zur Fehlerauschrift

BDOS ERR ON d: R/O

Der Schreibschutz kann wieder aufgehoben werden durch System-Reset, Rücksetzen des Dateisystems oder Rücksetzen des Laufwerkes.

### 5.4.2.17. Lesen des READ/ONLY-Vektors

Funktion 29 GET READ/ONLY VECTOR

Eingangsparameter:
CL: 1DH

Ausgangsparameter:
BX: READ/ONLY-Vektor

Ähnlich der Funktion RETURN LOGIN VECTOR wird in BX ein 16-Bit-Vektor bereitgestellt, dessen Bits 0 bis 15 die Plattenlaufwerke A bis P repräsentieren. Ein gesetztes Bit bedeutet Schreibschutz für dieses Laufwerk. Die Bits werden gesetzt durch die Funktion WRITE PROTECT DISK oder automatisch beim Plattenwechsel.

### 5.4.2.18. Festlegen der Dateiattribute

Funktion 30 SET FILE ATTRIBUTES

Eingangsparameter:
CL: 1EH
DX: FCB-Adresse

Ausgangsparameter:
AL: RC (Rückkehrcode)
RC=00 erfolgreiche Ausführung
RC=FF Datei nicht vorhanden

Diese Funktion erlaubt verschiedene Dateikennzeichen (Attribute) zu setzen bzw. zu löschen. Im FCB, dessen Adresse im Register DX

angegeben ist, müssen die gewünschten Dateiattribute in den Vorzeichenbits der Bytes t1 und t2 (siehe auch Abschnitt 5.4.1 und Tabelle 6 Standard FCB) verschlüsselt angegeben werden.

Folgende Dateiattribute sind möglich:

Vorzeichenbit von t1:
Bei gesetztem Bit wird die Datei zur READ/ONLY-Datei erklärt, d.h. jede Schreiboperation, die diese Daten betrifft, führt zu einer Fehlerausschrift.

Vorzeichenbit von t2:
Ist das Bit gesetzt, wird die Datei zu einer sogenannten SYS-Datei erklärt. Bei der Verzeichnisausgabe mit dem Kommando DIR erscheint diese Datei nicht in der Verzeichnisliste. Dateiverzeichnisse fÜR SYS-Dateien können mit dem Kommando DIRS ausgegeben werden.

### 5.4.2.19. Abfrage der Adresse der Plattenparameter

Funktion 31 GET ADDR (DISK PARMS)
Eingangsparameter:
CL: 1FH
Ausgangsparameter:
BX: Adresse des Plattenparameterblocks (DPB)
ES: Basisadresse des Segments

Nach Ausführung der Funktion wird dem Nutzerprogramm in den Registern BX und ES die Adresse des BIOS residenten DPB des aktuell ausgewählten Laufwerkes zur Verfügung gestellt. Diese Information kann verwendet werden für:
- Anzeige bzw. Berechnung des freien Platzes
- Ändern der aktuellen Werte bei Plattenwechsel
  (Aufbau DPB siehe Abschnitt 7.3.)

### 5.4.2.20. Einstellen/Abfragen des Nutzercodes

Funktion 32 SET/GET USER CODE
Eingangsparameter:
CL: 20H
DL: FF (bei GET)
DL: Anwendercode (bei SET)
Ausgangsparameter:
AL: aktueller Anwendercode (bei GET)
AL: unbestimmt (bei SET)

Mit Hilfe dieser Funktion ist es möglich den aktuellen Nutzercode abzufragen oder zu ändern. Enthält das Register DL FF, so übergibt die Funktion den Nutzercode im AL-Register. Der Nutzercode liegt im Bereich zwischen 0 und 15. Ist der Inhalt von DL ungleich FF, so wird er als Nutzercode interpretiert (Modulo 16) und als aktueller Code eingetragen.

## 5.4.2.21. Direktes Lesen eines logischen Satzes

```
Funktion 33   READ RANDOM
        Eingangsparameter:
                CL: 21H
                DX: FCB-Adresse
        Ausgangsparameter:
                AL: RC    (Rückkehrcode)
                    RC=00 erfolgreiche Ausführung
                    RC=01 Lesen nichtgeschriebener Daten
                          Der Satz der angegebenen Satznummer
                          wurde vorher nicht geschrieben.
                    RC=02 nicht benutzt
                    RC=03 aktueller Bereich kann nicht abge-
                          schlossen werden
                          Dieser Fehler kann auftreten, wenn der
                          FCB zerstört wurde, oder nicht eröffnet
                          ist.
                    RC=04 Lesen aus einem nicht beschriebenen
                          Bereich
                          Die Satznummer zielt auf einen Bereich,
                          in den noch kein Satz geschrieben wurde
                          (analog zu RC=01)
                    RC=05 nicht benutzt
                    RC=06 unzulässige Satznummer
                          Das Byte r2 des FCB ist ungleich Null.
```

Die READ RANDOM-Funktion ist ähnlich der sequentiellen Dateileseoperation. Es wird aber immer nur der Satz gelesen, dessen Satznummer als 3 Byte-Wert in den Bytes r0...r2 des FCB steht. SCP 1700 nutzt das Byte r2 nicht (außer bei der Funktion 35). Das Byte r2 muß daher immer mit Null eingestellt sein, d.h. die Satznummer kann Werte von 0 bis 65535 annehmen. Der so adressierte Satz wird gelesen und ab der aktuellen DMA-Adresse gespeichert. Erforderlich ist, daß der Bereich Null (Byte ex) zuerst eröffnet wurde, unabhängig davon, ob in diesen Bereich bereits Sätze geschrieben wurden. Nach Ausführung der Leseoperation wird im Register AL ein Rückkehrcode übergeben.
Diese Funktion erhöht nicht automatisch die Satznummer!
Es ist aber möglich nach READ RANDOM sequentielle Ein- oder Ausgaben folgen zu lassen. Dabei ist zu beachten, daß mit der zuletzt verwendeten Satznummer weitergearbeitet wird, da die Funktion READ RANDOM die Bytes cr und ex entspechend der absoluten Satznummer (r1, r2) füllt.

## 5.4.2.22. Direktes Schreiben eines logischen Satzes

```
Funktion 34   WRITE RANDOM
        Eingabeparameter:
                CL: 22H
                DX: FCB-Adresse
        Ausgabeparameter:
                AL: RC    (Rückkehrcode)
                    RC=00 erfolgreiche Ausführung
                    RC=01 nicht benutzt
                    RC=02 kein Platz
                          Auf der Platte des aktuellen Laufwerks
                          ist kein Block mehr verfügbar
                    RC=03 aktueller Bereich kann nicht abge-
                          schlossen werden. Die Satznummer in r1,
```

r0 bezieht sich auf einen anderen Bereich. Der vorhergehende genutzte kann aber nicht abgeschlossen werden, da ev. der FCB zerstört wurde oder die Datei gar nicht eröffnet ist.

RC=04 nicht benutzt

RC=05 Dateiverzeichnis voll
Wird ein neuer Bereich benötigt, so erfordert das eine neue Verzeichniseintragung für diese Datei, aber es ist kein Platz mehr vorhanden.

RC=06 unzulässige Satznummer
Das Byte r2 des FCB ist ungleich Null.

Diese Funktion entspricht der Funktion READ RANDOM, nur das hier die Daten von der aktuellen DMA-Adresse zur Platte übertragen werden. Ist der Bereich auf der Platte oder der Datenblock noch nicht eingerichtet (Funktion 22) wird das Einrichten von der WRITE RANDOM-Funktion übernommen. Die Satznummer (Bytes r1 und r0) wird nicht erhöht. Jedoch werden die Bytes cr und ex entsprechend der Satznummer geändert, so daß nach WRITE RANDOM mit sequentiellen E/A-Funktionen weitergearbeitet werden kann.
Wie bei READ RANDOM muß gleichfalls der Bereich Null (ex = 0) als erstes eröffnet werden, unabhängig davon, ob in diesem Bereich geschrieben wird. Ist die gesamte Datei noch leer, so muß mit der MAKE-Funktion Plattenplatz zugewiesen und in das Dateiverzeichnis die Datei eingetragen werden.

### 5.4.2.23. Berechnen der Dateigröße

Funktion 35 COMPUTE FILE SIZE

Eingangsparameter:
CL: 23H
DX: FCB-Adresse

Ausgabeparameter:
Satznummer in den Bytes r0 bis r2 des FCB

Das Verzeichnis der sich auf dem im FCB angegebenen Laufwerk befindlichen Platte wird nach der im FCB angegebenen Datei durchsucht. Die Funktion liefert in den Bytes r1, r0 des FCB die Satznummer des ersten freien Satzes auf der Platte. Es ist zu beachten, daß bei Platten die Direktzugriffsdateien enthalten, Lücken auf der Platte vorhanden sein können. Die gelieferte Satznummer bezieht sich immer auf den ersten freien Satz nach allen Dateien (Lücken innerhalb einer Direktzugriffsdatei gelten als zugehörig zu dieser Datei und somit als belegt).
Das Byte r2 zeigt dabei die EOV-Bedingung an. Ist r2 ungleich Null, dann bedeutet das, daß alle Blöcke der Platte belegt sind, d.h. auf die Platte wurden 65536 Sätze geschrieben bzw. Dateien zugewiesen. Die gelieferte Satznummer kann in nachfolgenden direkten E/A-Operationen verwendet werden.

### 5.4.2.24. Bestimmen des nächsten direkt zu lesenden/schreibenden Satzes

Funktion 36 SET RANDOM RECORD

Eingangsparameter:
CL: 24H
DX: FCB-Adresse

Ausgangsparameter:
Satznummer in den Bytes r0 bis r2 des FCB

Nach sequentiellen Lese- oder Schreiboperationen kann mit Hilfe dieser Funktion die absolute Satznummer des nächsten sequentiellen Satzes bestimmt werden. Diese sind aus den Werten der FCB-Bytes cr und ex berechnet und in die Bytes r1, r0 geschrieben. Nachfolgende Direktzugriffsoperationen können bezugnehmend auf diese Satznummer ausgeführt werden. Damit ist es möglich, eine Datei sequentiell zu verarbeiten und ab einem definierten Zeitpunkt direkt zuzugreifen.

### 5.4.2.25. Rücksetzen des Plattenlaufwerkes

Funktion 37 RESET DRIVE
Eingangsparameter:
CL: 25H
DX: Laufwerksvektor
Ausgangsparameter:
AL: 00

Im Laufwerksvektor repräsentiert jedes Bit ein Laufwerk (Bit 0 = Laufwerk A, Bit 15 = Laufwerk P). Gesetztes Bit bedeutet: das entsprechende Laufwerk ist zurückzusetzen (nicht logged-in, READ/ONLY-Zustand).

### 5.4.2.26. Direktes Schreiben leerer Sätze

Funktion 40 WRITE RANDOM WITH ZERO FILL
Eingangsparameter:
CL: 28H
DX: FCB-Adresse
Ausgangsparameter:
AL: RC (Rückkehrcode)
RC siehe Abschnitt 5.4.2.22.

Diese Funktion entspricht dem direkten Schreiben. Es wird aber nur ein leerer Satz geschrieben. Damit ist der Satz initialisiert und gehört zur Datei. Er gilt aber als nicht geschrieben.

### 5.4.2.27. Start eines anderen Programmes

Funktion 47 CHAIN TO PROGRAM
Eingangsparameter:
CL: 2FH
DMA Puffer: Kommandozeile
Ausgangsparameter:
Das Programm ist gestartet

Mit der Funktion 47 kann von einem Programm aus ein anderes gestartet werden. Das entsprechende Startkommando mit etwaigen Parametern ist als Kommandozeile im Standard-DMA-Puffer des rufenden Programms einzutragen. Die Kommandozeile ist mit einem Nullbyte abzuschließen.
Nach dem Start des Programms ist die übergebene Kommandozeile im Standard-DMA-Puffer des neuen Programms eingetragen. Das alte Programm wird nicht fortgesetzt.

### 5.4.2.28. Direkter BIOS-Ruf

Funktion 50 DIRECT BIOS CALL
Eingangsparameter:
CL: 32H
DX: Adresse des BIOS-Parameter-Blocks

Mit der Funktion 50 können die BIOS-Routinen des SCP (siehe Abschnitt 6.) von einem Anwenderprogramm direkt aufgerufen werden. Der übergebene Parameterblock besitzt folgenden Aufbau und ist 5 Byte lang:

Byte 1: BIOS-Funktionsnummer
Bytes 2/3: 16-Bit-Wort, das in das Register CX vor dem Start des BIOS eingetragen wird
Bytes 4/5: 16-Bit-Wort, das in das Register DX eingetragen wird, bevor BIOS aufgerufen wird

### 5.4.2.29. Einstellen der Segmentadresse des DMA-Bereichs

Funktion 51 SET DMA BASE
Eingangsparameter:
CL: 33H
DX: DMA-Basisadresse
Ausgangsparameter:
Die DMA-Basisadresse ist eingestellt.

Diese Funktion setzt das Basisregister für den DMA-Verkehr. Die Basisadresse wird als 16-Bit Paragraphenadresse im DX-Register übergeben. Diese Basisadresse zusammen mit der DMA-Offset-Adresse ergibt dann die Adresse eines 128 Byte langen Puffers für Plattenein- und -ausgaben. Ist keine Adressangabe im DX-Register enthalten, gilt als Standardadresse die des Nutzerdatensegmentes (DS-Wert). Die DMA-Offset-Adresse ist in diesem Fall 80H (siehe Abschnitt 3.)

### 5.4.2.30. Übernahme der aktuellen DMA-Adresse

Funktion 52 GET DMA BASE
Eingangsparameter:
CL: 34H
Ausgangsparameter:
BX: relative DMA-Adresse
ES: DMA-Basisadresse

Die Funktion liefert die aktuellen Werte der DMA-Basis und der relativen DMA-Adresse in den angegebenen Registern BX und ES.

## 5.5. BDOS Speicherverwaltung und Ladeoperationen

### 5.5.1. Allgemeines

Der Speicher wird unter SCP 1700 auf zwei verschiedene Arten verwaltet.
Im ersten Fall wird durch eine statische Zuordnungstabelle, die im BIOS festgelegt ist, der physische Speicher definiert.
Auf diese Weise ist es möglich mit SCP 1700 in einer Speicherstruktur zu arbeiten, die sich aus bis zu 8 nicht zusammenhängenden RAM- und ROM-Bereichen und reservierten, fehlenden bzw. fehlerhaften Speicherregionen zusammensetzt.
In einem einfachen System mit zusammenhängendem Speicherplatz wird von der Zuordnungstabelle nur eine einzelne Region verwaltet, die gewöhnlich hinter dem BIOS-Bereich beginnt und bis zum Ende des vorhandenen Speichers reicht.
Neben der beschriebenen statischen Zuordnung wird von SCP 1700 eine dynamische Speicherverwaltung ausgeführt.
Diese dynamische Speicherverwaltung basiert auf der statischen und läßt eine dynamische Zuordnung von wiederum 8 Regionen zu.
Speicherplatz kann entweder durch eine Programmladeoperation oder durch BDOS-Rufe zur Speicherverwaltung angefordert werden.
Zum Laden von Programmen gibt es 2 Möglichkeiten:

- Eingabe des CCP-Kommandos
- Verwendung der BDOS-Funktion 59 (PROGRAMM LOAD).

Mit dem CCP-Kommando können mehrere Programme geladen werden, wenn jedes Programm nach dem Start ein Systemrücksetzen (BDOS-Funktion 0 mit DL=01H) ausführt und im Speicher verbleibt.
Programme eines solchen Mehrprogrammsystems erhalten nur durch direktes Starten von einem anderen Programm aus die Steuerung.
Im Normalfall gibt es immer nur ein Transientprogramm zu jeder beliebigen Zeit im Speicher. Falls trotzdem mehrere Programme gleichzeitig im Speicher stehen, so können sie durch die Eingabe von CONTROL-C gelöscht werden, Sie werden in der umgekehrten Reihenfolge in der sie geladen wurden, gelöscht, unabhängig davon welches Programm gerade die CONTROL-C-Eingabe ausgelöst hat.

#### Beispiele

Ein Programm, das durch ein CCP-Kommando geladen wurde, kann selbst weitere Programme laden und Datenbereiche zuteilen.
Angenommen es sind 4 Speicherregionen in folgender Art zugewiesen:
Ein Programm wurde auf CCP-Ebene durch ein Bedienerkommando geladen. Der CMD-Dateikopf wurde eingelesen, das gesammte Speicherabbild des Programms und seiner Daten in die Region A geladen und die Abarbeitung gestartet. Dieses Programm nutzt wiederum die BDOS-Programmladefunktion 59, um ein anderes Programm in die Region B zu laden und diesem dann die Steuerung zu übergeben.
Das Programm in der Region B belegt zusätzlich die Region C und D.
Die Aufgliederung des Speichers zeigt das Bild 8

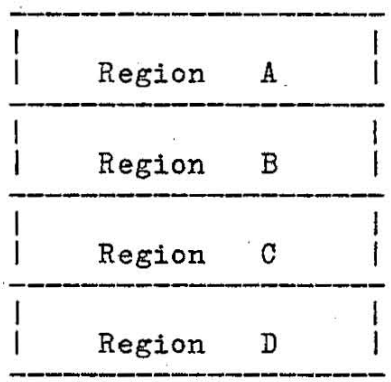



Bild 8: Beispiel für eine Speicherzuordnung

Es gibt eine hierarchische Zuordnung dieser Regionen. Das Programm mit A kontrolliert den ganzen Speicher von A bis D. Das Programm in B kontrolliert die Regionen B bis D. Das Programm in A kann, falls gewünscht, die Regionen B bis D freigeben und ein anderes Programm laden (Arbeitsweise von DDT86).
Bevor ein anderes Programm geladen wird, sorgt der BDOS-Ruf 57 (FREE MEMORY) für die Freigabe des Speichers, den das aktuelle Programm belegt. Weiterhin kann das Programm in B, falls gefordert, die Region C und D freigeben. Alle 4 Regionen A bis D werden freigegeben, wenn das Programm in A oder B ein Systemreset (BDOS-Funktion 0 mit DL = 00H) ausführt.
Ein transientes Programm kann einen Teil der Region freigeben und somit die Bearbeitung weiterer Speicherplatzanforderungen ermöglichen. Der freigegebene Speicherplatz muß aber am Anfang oder am Ende einer Region liegen.
Im nächsten Beispiel erhält ein Programm in der Region C nach seiner ersten Speicheranforderung 600H Paragraphen ab der Paragraphen-Adresse 100H. So ergibt sich folgendes Bild:

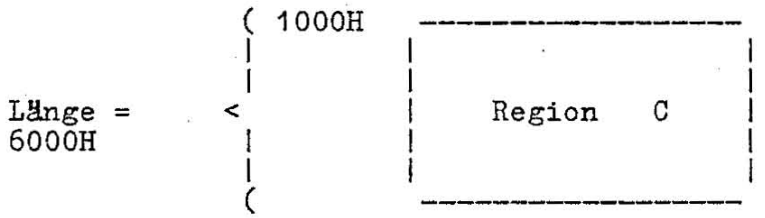



Bild 9: Beispiel einer Speicherregion

Außerdem sei die Region D belegt. Ohne Einfluß auf die Region D können dann 200H Paragraphen in der Region C freigegeben werden, wenn diese 200H Paragraphen ab der Paragraphenbasis 700H beginnen. Bild 10 spiegelt zu diesem Beispiel die Speicherzuordnung wieder.

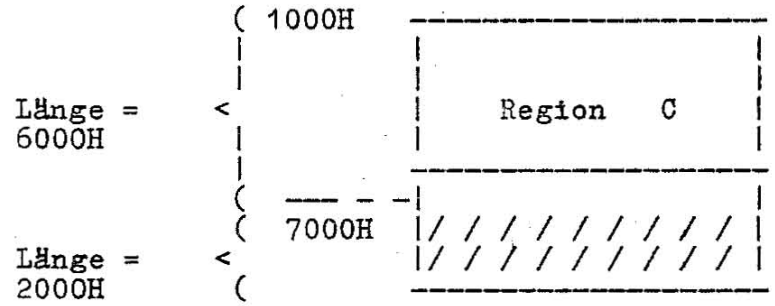



Bild 10: Beispiel einer Speicherregion

Die Region ab der Paragraphenadresse 700H ist nun für die nächsten Speicheranforderungen verfügbar. Es ist aber zu beachten, daß eine Speicheranforderung zum Fehler führt, wenn bereits schon 8 Speicherregionen bestehen.
In der Regel vergibt das System nur eine Region, wenn alle Programmeinheiten in eine zusammenhängende Region passen.
Die in den folgenden Abschnitten beschriebenen BDOS-Funktionen zur Speicherverwaltung beziehen sich auf einen Speichersteuerblock (MCB), der im jeweils aufrufenden Programm definiert sein muß.
Er hat folgende Form:

| | 16-Bit | 16-Bit | 8-Bit |
|---|---|---|---|
| MCB | M-Base | M-Length | M-Ext |

M-Base und M-Length sind entweder Eingangs- oder Ausgangsparameter, die in 16-Byte-Paragraphen-Einheiten ausgedrückt werden, und der Funktion beim Aufruf übergeben werden bzw. dem Nutzer nach Ausführung der Funktion zur Verfügung gestellt werden.
Im M-Ext wird ein Ergebniswert nach Ausführung der Funktion geliefert.
Eine fehlerhafte Abarbeitung wird im Register AL mit OFFH quittiert.

## 5.5.2. BDOS-Operationen für die Speicherverwaltung

### 5.5.2.1. Ermittlung der Anfangsadresse einer definierten Region

Funktion 53 GET MAX MEM
Eingangsparameter:
CL: 35H
DX: Offset zum MCB
Ausgangsparameter:
AL: RC (Rückkehrcode)

Beim Aufruf der Funktion ist in M-Length die gewünschte Länge der Region einzutragen. Nach Ausführung der Funktion ist in M-Base die Segmentadresse der gewünschten Region und in M-Length die maximal zur Verfügung stehende Länge der Region (Längen und Adreßangaben in Paragrapheneinheiten) eingetragen.

### 5.5.2.2. Test auf Verfügbarkeit einer definierten Region

```
Funktion 54   GET ABS MAX
        Eingangsparameter:
                CL: 36H
                DX: Offset zum MCB
        Ausgangsparameter:
                AL: RC      (Rückkehrcode)
                    RC=FFH Region nicht verfügbar
                    RC=00H Region verfügbar
```

Bei Aufruf der Funktion ist in M-Length und in M-Base die gewünschte Region zu definieren. Nach Ausführung der Funktion kann die Verfügbarkeit an Hand des Rückkehrcodes im Register AL ausgetestet werden.

### 5.5.2.3. Zuweisung von Speicherbereich für eine Region definierter Länge

```
Funktion 55   ALLOC MEM
        Eingangsparameter:
                CL: 37H
                DX: Offset zum MCB
        Ausgangsparameter:
                AL: RC      (Rückkehrcode)
                    RC=00H Funktion erfolgreich ausgeführt
                    RC=FFH kein Speicherplatz zugewiesen
```

Die Speicherzuweisungsfunktion weist Speicherbereich entsprechend des MCB zu. Der MCB wird durch DX adressiert. Die Größe der Speicheranforderung wird durch M-Length festgelegt. Die Funktion 55 trägt in den M-Base die Basisparameteradresse der zugewiesenen Region ein.

### 5.5.2.4. Zuweisung von Absolutspeicherbereich

```
Funktion 56   ALLOC ABS MEM
        Eingangsparameter:
                CL: 38H
                DX: Offset zum MCB
        Ausgangsparameter:
                AL: RC      (Rückkehrcode)
                    RC=00H erfolgreiche Ausführung
                    RC=FFH kein Speicherplatz zugewiesen.
```

Diese Absolutspeicher-Zuweisungsfunktion ordnet Speicherbereich entsprechend dem MCB zu. Der MCB wird durch DX adressiert. Der Speicherbereich ergibt sich aus M-Length und der absoluten Basisadresse von M-Base. Die Werte sind vom Nutzer im MCB vor dem Aufruf der Funktion einzustellen.

### 5.5.2.5. Freigabe von Speicherbereich

```
Funktion 57   FREE MEM
        Eingangsparameter:
                CL: 39H
                DX: Offset zum MCB
        Ausgangsparameter:
                keine
```

Die Funktion 57 wird genutzt um den Speicherbereich freizugeben, der einem Programm zugeordnet ist. Der Wert von M-Ext steuert die Ausführung der Funktion. Falls M-Ext = 0FFH ist, werden alle Speicherbereiche, die das rufende Programm belegt, gelöscht. Anderenfalls, bei M-Ext = 00H, wird der Speicherbereich mit der Länge M-Length ab der Adresse M-Base freigegeben.
Wie oben beschrieben, kann entweder die ganze zugewiesene Region oder ein Teil am Ende der Region freigegeben werden. Ein mittlerer Teil einer Region kann unter SCP 1700 nicht freigegeben werden.

### 5.5.2.6. Löschen des gesamten Speichers

Funktion 58 FREE ALL MEM
Eingangsparameter:
CL: 3AH
Ausgangsparameter:
keine

Die Funktion 58 wird genutzt, um den gesamten verfügbaren Speicher bei SCP 1700 freizugeben. (Normalerweise wendet diese Funktion nur der CCP zur Initialisierung an).

### 5.5.2.7. Laden einer CMD-Datei

Funktion 59 PROGRAM LOAD
Eingangsparameter:
CL: 3BH
DX: Offset zum FCB
Ausgangsparameter:
AX: Rückkehrcode/Basisseitenadresse
BX: Basisseitenadresse

Die Funktion 59 lädt eine CMD-Datei. DX adressiert einen ordnungsgemäß geöffneten FCB, der die betreffende CMD-Datei beschreibt.
Nach Ausführung der Funktion enthält das Register AX den Inhalt 0FFFFH, wenn die Datei nicht geladen werden konnte.
Andernfalls beinhalten die Register AX und BX die Paragraphenadresse der Basisseite, die zur geladenen Datei gehört . Die Basisadresse und die Länge von jedem Segment sind in der Basisseite abgelegt. Es ist zu beachten, daß beim Laden auf CCP-Ebene die DMA-Basisadresse so initialisiert wird, daß sie auf die Basisseite der geladenen Datei zeigt und der DMA-Offset den Wert 0080H erhält. Die Funktion 59 an sich richtet keine Standard-DMA-Adresse ein. Es ist Aufgabe des Programms, das die Funktion 59 aufruft, die Funktion 51 zum Setzen der DMA-Basis und die Funktion 26 zum Setzen der DMA-Offsets aufzurufen, ehe der geladenen Datei die Steuerung übergeben wird.

# 6. Organisation des Basic I/O-Systems (BIOS)

## 6.1. Struktur des BIOS

BIOS liegt im obersten Teil des Betriebssystems SCP 1700. Die Lage im SCP 1700 ist im folgenden Bild dargestellt.

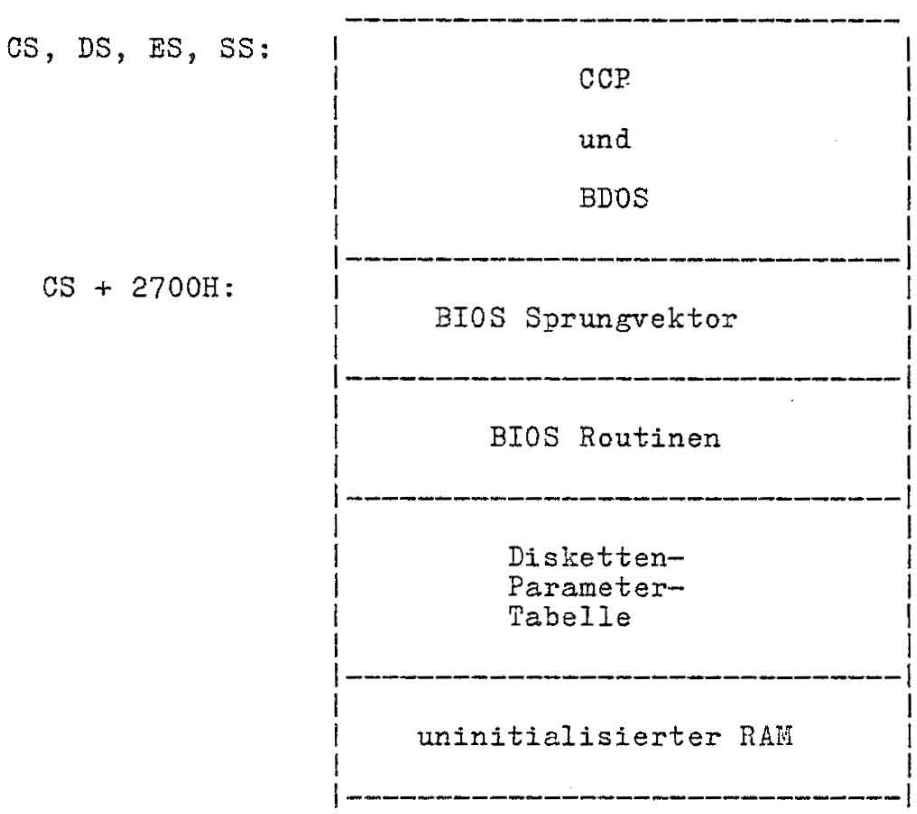



Bild 11: Generelle Struktur von SCP 1700

CCP und BDOS werden mit dem Liefersatz des Betriebssystem SCP 1700 als HEX-Datei SCP.H86 geliefert. Dazu wird als ASM-Datei der Qellcode eines BIOS mitgeliefert, das alle wesentlichen Hardwareteile des A 7100 bedient. Dieses BIOS wird im folgenden beschrieben. Um SCP 1700 auf nichtstandardgemäßer Hardware implementieren zu können, muß vom Nutzer nach dem mitgelieferten Vorbild ein der jeweiligen Hardware angepaßtes BIOS erstellt werden. Das erzeugte BIOS ist als HEX-Datei an das Ende von SCP.H86 anzufügen. Dann wird mit Hilfe des Systemprogramms GENCMD die Datei SCP.SYS erzeugt, die später beim BOOT-Vorgang über ein Ladeprogramm geladen werden kann.
Das Ladeprogramm enthält selbst eine vereinfachte BIOS-Variante, das sogenannte LDBIOS (Loader BIOS).

## 6.2. Der BIOS-Sprungvektor

Der Eintritt in das BIOS erfolgt über einen "Sprungvektor", der ab Offset 2700H von der Betriebssystembasis (siehe Bild 11) liegt.
Der Sprungvektor besteht aus einer Folge von 22 3-Byte-Sprungbefehlen, die die Programmsteuerung an die BIOS-Eintrittspunkte übertragen. Obwohl einige unwesentliche BIOS-Subroutinen einen einfachen Return (RET)-Befehl enthalten können, muß das entsprechende Sprungvektor-Element in der in Tabelle 7 beschriebenen Weise vorhanden sein.
Die Parameter für die einzelnen BIOS-Subroutinen werden in den Registern CX und DX übergeben, wenn das die Subroutine verlangt. CX erhält den ersten Parameter, DX wird für das zweite Argument benutzt. Rückkehrcodes werden entsprechend ihrem Typ in folgenden Registern übergeben:

Byte-Werte in AL
Wort-Werte in BX

Die spezifischen Parameter und Rückkehrcodes sind in der Beschreibung der einzelnen Subroutinen enthalten. Der BIOS-Sprungvektor hat folgende Form:

Tabelle 7: BIOS-Sprungvektor

| Kommando | BIOS-Ruf | Beschreibung |
|---|---|---|
| JMP INIT | 0 | Initialisieren des SCP 1700 |
| JMP WBOOT | 1 | System-Reset |
| JMP CONST | 2 | Prüfen Status vom Gerät CONSOLE |
| JMP CONIN | 3 | Lesen eines Zeichens vom Gerät CONSOLE |
| JMP CONOUT | 4 | Schreiben eines Zeichens auf dem Gerät CONSOLE |
| JMP LISTOUT | 5 | Schreiben eines Zeichens auf dem Gerät LIST |
| JMP PUNCH | 6 | Schreiben eines Zeichens auf dem Gerät AXO |
| JMP READER | 7 | Lesen eines Zeichens vom Gerät AXI |
| JMP HOME | 8 | Schreib-/Lesekopf auf Spur 00 positionieren |
| JMP SELDSK | 9 | Laufwerk auswählen |
| JMP SETTRK | 10 | Setzen Spurnummer |
| JMP SETSEC | 11 | Setzen Sektornummer |
| JMP SETDMA | 12 | Setzen DMA-Offsetadresse |
| JMP READ | 13 | ausgewählten Sektor lesen |
| JMP WRITE | 14 | ausgewählten Sektor schreiben |
| JMP LISTST | 15 | Prüfen Status LIST-Gerät |
| JMP SECTRAN | 16 | Übersetzen Sektornr. (log./phys.) |
| JMP SETDMAB | 17 | Setzen DMA-Segmentadresse |
| JMP GETSEGT | 18 | Lesen Offset MEM DESC Tabelle |
| JMP GETIOBF | 19 | Lesen I/O-Byte |
| JMP SETIOBF | 20 | Setzen I/O-Byte |
| JMP KGSIN | 21 | ABS/KGS-Eingabe |

Die BIOS-Routinen, die über den BIOS-Sprungvektor aufgerufen werden, können in drei Gruppen aufgeteilt werden:
- System-(Re)-Initialisierungs-Subroutinen
- einfache Zeichen-E/A-Subroutinen
- Platten-E/A-Subroutinen.

## 6.3. Einfache periphere Geräte

Die Ausführung aller einfachen Zeichen-E/A-Operationen ist im ASCII-Format mit Groß- und Kleinbuchstaben vorgesehen.
Der End-of-File-Zustand für ein Eingabegerät wird durch das ASCII-Zeichen CTRL/Z (1AH) spezifiziert. SCP 1700 betrachtet periphere Geräte als "logische" Geräte. Die Zuweisung zu physischen Geräten erfolgt innerhalb des BIOS. Die Gerätecharakteristika werden in der Tabelle 8 definiert:

Tabelle 8: Charakteristik logischer Geräte für SCP 1700

| logischer Gerätename | Charakteristik |
|---|---|
| CONSOLE | Das logische Gerät CONSOLE besteht aus dem Bildschirmgerät und der Tastatur und wird zur Kommunikation des Bedieners mit dem Rechnersystem verwendet. Im BIOS wird das logische Gerät CONSOLE von den Routinen CONIN, CONOUT und CONST angesprochen. |
| LIST | Das logische Gerät LIST ist im Standardfall ein Drucker. Es dient zur Ausgabe von Listings und als Hardcopygerät. |
| AXO | Das logische Gerät AXO ist als Hilfsausgabegerät deklariert. Im Standardfall ist das Bildschirmgerät zugewiesen. |
| AXI | Das logische Gerät AXI ist als Hilfseingabegerät deklariert. Im Standardfall ist die Tastatur zugewiesen. |

Im BIOS des Liefersatzes ist eine Veränderung der Anordnung logischer Geräte zu physischen Geräten mit Hilfe der IOBYTE-Funktion möglich. Die IOBYTE-Funktion erzeugt eine Zuordnung von logischen zu physischen Geräten, die während der Arbeit von SCP 1700 geändert werden kann (siehe STAT-Kommando).
Die Definition der IOBYTE-Funktion ist wie folgt festgelegt:

- ein einzelner Speicherplatz, genannt IOBYTE, wird im BIOS belegt
- dieses Byte definiert die Zuordnung logischer zu physischen Geräten, wie es zum entsprechenden Zeitpunkt existiert.

Das IOBYTE ist in vier 2-Bit-Felder aufgeteilt:

| IOBYTE | LIST | | AXO | | AXI | | CONSOLE | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Der Wert jedes Feldes kann zwischen 0 und 3 liegen und definiert die zugewiesene Quelle bzw. das Ziel für jedes logische Gerät.

Die Standardeinstellung des IOBYTE ist folgende:

| LIST | | AXO | | AXI | | CONSOLE | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Die verschiedenen möglichen Werte der Zuweisungen sind in Tabelle 9 angegeben:

Tabelle 9: IOBYTE-Felddefinition

| | Wert | phys. Name | Bedeutung |
|---|---|---|---|
| CON | 0 | TTY: | Ausgabe: Bildschirmeinheit über ABS<br>Eingabe: Tastatur über 8251A der ZVE |
| | 1 | CRT: | Ausgabe: Bildschirmeinheit über ASP (IFSS)<br>Eingabe: Tastatur über ASP (IFSS) |
| | 2 | BAT: | Ausgabe: Logisches LIST-Gerät<br>Eingabe: Logisches Gerät AXI, geeignet für Stapelverarbeitung |
| | 3 | UC1: | Ausgabe: Ausgabegerät über ASP (V24)<br>Eingabe: Eingabegerät über ASP (V24) |
| AXI | 0 | TTY: | Tastatur über 8251A der ZVE |
| | 1 | PTR: | Eingabegerät über ASP (V24) |
| | 2 | UR1: | Eingabegerät über ASP (IFSS) |
| | 3 | UR2: | frei |
| AXO | 0 | TTY: | Bildschirmeinheit über ABS |
| | 1 | PTP: | Ausgabegerät über ASP (IFSP) |
| | 2 | UP1: | Ausgabegerät über ASP (IFSS) |
| | 3 | UP2: | Ausgabegerät über ASP (V24) |
| LST | 0 | TTY: | Bildschirmeinheit über ABS |
| | 1 | CRT: | Ausgabegerät über ASP (IFSS) |
| | 2 | LST: | Drucker über Centronix-Interface |
| | 3 | UL1: | Ausgabegerät über ASP (IFSP) |

Die o.a. Angaben zum IOBYTE betreffen die IOBYTE-Implementierung im BIOS des Liefersatzes. Der Nutzer kann durch Änderung des Liefer-BIOS eine eigene Zuweisung von logischen zu physischen

Geräten über das IOBYTE verwirklichen.

## 6.4. Eintrittspunkte der BIOS-Subroutinen

Die BIOS-Routinen werden vom BDOS aus zur Ausführung der gewünschten Funktionen angesprungen. Im folgenden werden die Funktionen beschrieben, die unter jedem BIOS-Eintrittspunkt erfüllt werden.
Dabei ist zum Beispiel zu beachten, daß Platten-E/A-Anforderungen immer über eine Folge von verschiedenen Subroutinenaufrufen realisiert werden. Diese Subroutinen wählen das Laufwerk aus, setzen die Spur- und Sektoradresse und liefern der E/A-Operation Offset- und Segmentadresse für den direkten Speicherzugriff (DMA).
Erst nachdem diese Parameter übergeben sind, erfolgt der Ruf für READ oder WRITE und die geforderte E/A-Operation wird ausgeführt. Dabei können einem einzelnen SELDSK-Ruf zur Auswahl eines Laufwerks eine ganze Reihe von READ- oder WRITE-Anweisungen folgen, ehe ein Wechsel der o.g. Parameter folgt. READ- und WRITE-Anweisungen führen normalerweise eine Reihe Wiederholungen aus, bevor sie eine Fehlermeldung an das BDOS zurückgeben.

Tabelle 10: BIOS-Rufe

| Subroutine | Beschreibung |
|---|---|
| INIT | Diese Subroutine wird durch den SCP 1700-Lader unmittelbar nach dem Laden von SCP.SYS in den Speicher aufgerufen. Das Programm ist verantwortlich für die Hardwareinitialisierung, die noch nicht vom BOOT-LADER vorgenommen worden ist, das Setzen von Anfangswerten für das BIOS (einschließlich IOBYTE), das Ausgeben einer Startmeldung und die Initialisierung des Interruptvektors auf die BDOS Offset- und Basisadresse. Nach Beendigung der Routine erfolgt ein Sprung zum CCP-Offset (0H). Alle Basisregister enthalten durch die Initialisierung zu diesem Zeitpunkt die Basisadresse des Betriebssystems. |
| WBOOT | Diese Subroutine wird nach einem Ruf BDOS #0 ausgeführt. An dieser Stelle werden einige Neuinitialisierungen von Hard- und Software vorgenommen. Nach Abschluß dieser Routine wird dierekt zum CCP-Reset (06H) gesprungen. |
| CONST | Es wird der Status des gerade zugewiesenen CONSOLE-Gerätes geprüft. Sofern ein ASCII-Zeichen zum Lesen bereit ist, erhält das Register AL als Rückkehrcode den Wert OFFH, anderenfalls 00H. |
| CONIN | Liest das nächste ASCII-Zeichen vom Gerät CONSOLE in Register AL ein. Ist vom Gerät CONSOLE kein Zeichen bereit, wartet die Routine auf ein Zeichen und gibt erst dann die Steuerung an BDOS zurück. |
| CONOUT | Übergibt ein ASCII-Zeichen in Register CL an das Ausgaberegister des Gerätes CONSOLE. |
| LISTOUT | Übergibt im Register CL ein ASCII-Zeichen an das aktuelle LIST-Gerät. |
| PUNCH | Übergibt im Register CL ein BYTE an das aktuelle AXO-Gerät. |
| READER | Liest das nächste BYTE vom aktuellen AXI-Gerät in das Register AL. Ein END-OF-FILE wird mit dem Zeichen CTRL/Z (1AH) erkannt. |

Tabelle 10 BIOS-Rufe (Fortsetzung)

| Subroutine | Beschreibung |
|---|---|
| HOME | Setzt den Schreib/Lesekopf des angewählten Laufwerks auf die Spur 00 zurück. Dasselbe kann auch mit der Funktion SETTRK (mit Parameter 0) erreicht werden. |
| SELDSK | Wählt das im Register CL angegebene Laufwerk für spätere Operationen aus:<br>- 0 für A<br>- 1 für B<br>.<br>.<br>.<br>- 15 für P<br>Bei der Rückkehr aus dieser Routine enthält das Register BX die Basisadresse des Platten-Parameter-Kopfes (DPH siehe Abschnitt 7.2.) des angewählten Laufwerks. Sollte ein nichtexistierendes Laufwerk angesprochen werden, so liefert SELDSK als Fehlermeldung im Register BX den Wert 0000H. Anderenfalls kehrt SELDSK immer mit der aktuellen Kopfadresse zurück. Es ist vorteilhaft, eine Aktualisierung der Plattenauswahl bis zum Abschluß einer I/O-Operation (READ, WRITE) zurückzustellen und SELDSK nicht in einer Folge von Plattenoperationen anzusprechen. Mit Aufruf von SELDSK ist es möglich festzustellen, ob das angesprochene Laufwerk zum ersten Male ausgewählt wird. Im Register DL ist Bit 0 (das niederwertigste Bit) gleich Null, wenn das Laufwerk zum ersten Mal angesprochen wurde. |
| SETTRK | Beim Aufruf der SETTRK-Routine enthält das Register CX die Spuradresse für einen folgenden Plattenzugriff auf dem angewählten Laufwerk. Das Register CX kann bei Standarddisketten Werte zwischen 0 und 76, bei Minidisketten zwischen 0 und 79, enthalten und von 0 bis 65535 bei nichtstandardgerechten Subsystemen. |

Tabelle 10 BIOS-Rufe (Fortsetzung)

| Subroutine | Beschreibung |
|---|---|
| SETSEC | Beim Aufruf der SETSEC-Routine enthält das Register CX die physische Sektornummer für einen folgenden Plattenzugriff auf dem angewählten Laufwerk (siehe auch SECTRAN). |
| SETDMA | Register CX enthält bei Aufruf der SETDMA-Routine den DMA-Offset für eine folgende READ- oder WRITE-Operation. Wenn SETDMA z. B. mit CX=80H gerufen wird, dann lesen alle folgenden READ-Operationen ihre Daten in den Bereich von 80H bis 0FFH ab der aktuellen DMA-Basissegmentadresse, und alle WRITE-Operationen holen ihre Daten von dieser Adresse, bis ein neuer SETDMA- oder SETDMAB-Ruf erfolgt. |
| READ | Nachdem das Laufwerk ausgewählt wurde, Spur- und Sektornummer gesetzt wurden, liest diese Routine, aufbauend auf diese Parameter, einen Sektor, und übergibt im Register AL folgende Fehlermeldungen:<br>0 kein Fehler<br>1 unkorrigierbarer Fehler<br>Mit Rückkehrcode Null nimmt SCP 1700 an, daß die Operation erfolgreich abgeschlossen wurde. Wenn ein Fehler gemeldet wird, so bringt BIOS die Meldung<br>READ/WRITE ERROR<br>Das BIOS geht danach in einen HALT-Zustand und wartet auf eine Bedienereingabe. Der Bediener hat die Möglichkeit, durch Eingabe folgender Zeichen die Bearbeitung fortzusetzen:<br>C - Abbruch der Operation<br>R - Wiederholen der Operation<br>I - Ignorieren des Fehlers |
| WRITE | Schreibt die Daten vom selektierten DMA-Puffer zum ausgewählten Laufwerk, der angewiesenen Spur und dem angewiesenen Sektor. Fehlermeldungen werden wie im READ-Kommando im Register AL übergeben. Diese besitzen die gleiche Bedeutung wie oben beschrieben. Die Bedienerreaktionen sind die gleichen wie bei READ.<br>Für die WRITE-Routine sind folgende CL-Werte erforderlich:<br>CL = 0 normales Schreiben<br>CL = 1 Directory schreiben<br>CL = 2 erste Schreiboperation auf neuen Datenblock |

Tabelle 10 BIOS-Rufe (Fortsetzung)

| Subroutine | Beschreibung |
|---|---|
| LISTST | Liefert den Status des LIST-Gerätes. Der Wert 00 steht in Register AL, wenn das Gerät nicht zur Übernahme eines Zeichens bereit ist, 0FFH wenn es bereit ist. |
| SECTRAN | Mit dieser Routine wird eine logische in eine physische Sektorangabe umgewandelt. Bei Aufruf erhält SECTRAN die logische Sektornummer im Register CX. Die logische Sektornummer kann im Bereich von 0 bis Anzahl der Sektoren -1 liegen. Zusätzlich wird im Register DX der Offset einer Umkodierungstabelle benötigt. Die resultierende physische Sektornummer wird in Register BX übergeben. Ist DX gleich 0000H, so erfolgt keine Umrechnung und CX wird einfach nach BX geliefert. Zu beachten ist, daß SECTRAN auch aufgerufen wird, wenn im Disketten-Parameter-Kopf keine Umrechnung angewiesen wird. |
| SETDMAB | Register CX enthält beim Aufruf die Segmentbasisadresse für eine folgende DMA-Operation (READ oder WRITE). BIOS benutzt für eine READ- oder WRITE-Operation einen 128 Byte langen Puffer, der durch die DMA-Basisadresse und den DMA-Offset adressiert wird. |
| GETIOBF | Liefert in Register AL den aktuellen Wert des IOBYTE. |
| SETIOBF | Der im Register CL bei Aufruf der SETIOB-Routine übergebene Wert wird in das IOBYTE geladen. |
| KGSIN | Liefert im Register AL das am KGS bzw. ABS anliegende Zeichen (z.B. zum Lesen der Cursorposition verwendbar). Liegt kein Zeichen an, wartet die Routine auf ein Zeichen und gibt erst dann die Steuerung an BDOS zurück. |

Tabelle 10 BIOS-Rufe (Fortsetzung)

| Subroutine | Beschreibung |
|---|---|
| GETSEGT | Liefert in Register BX die Adresse der Speicherzuordnungstabelle (MRT). Der Inhalt des Registers BX ist der Offset der Tabelle relativ vom Beginn des Betriebssystems. Die Tabelle gibt Beginn und Größe des Speicherbereiches an, der von transienten Programmen genutzt werden kann. Die Speicherzuordnungstabelle hat folgende Form:<br><br>8-Bit<br>MRT: R-CNT<br>0: R-BASE \| R-LENGTH<br>1: R-BASE \| R-LENGTH<br>. . .<br>N: R-BASE \| R-LENGTH<br>16-Bit \| 16-Bit<br><br>R-CNT ist die Anzahl der Speicherzuordnungsbeschreibungen (im obigen Beispiel N+1). R-BASE gibt die Paragraphenbasis und R-LENGTH die Länge jedes physisch zusammenhängenden Speicherbereiches an. In der Tabelle sind nicht die Speicherbereiche der Interruptvektoren und des SCP 1700-Betriebssystems enthalten, da diese Bereiche nicht zu dem Speicherbereich gehören, der durch transiente Programme belegt werden darf. Besteht der Speicher aus einem zusammenhängenden Bereich, so ist R-CNT gleich 1 und N gleich Null mit nur einer Speicherzuordnungsbeschreibung für den gesamten Speicher. |

# 7. BIOS Plattendefinitionstabellen

## 7.1. Allgemeines

SCP 1700 ist ein tabellengesteuertes Betriebssystem mit einem getrennten feldkonfigurierbarem Basis-E/A-System (BIOS) Durch den Austausch bestimmter Subroutinen im BIOS, die im vorhergehenden Abschnitt aufgeführt sind, kann SCP 1700 für jedes RAM-orientierte Mikroprozessorsystem arbeitsfähig gemacht werden. Die Verwaltung der Speicherplätze auf den Platten erfolgt unter SCP 1700 dynamisch, d.h. daß zum Zeitpunkt der Neueröffnung einer Datei deren mögliche Länge noch nicht angegeben werden braucht. Speicherplatz wird in einer Datei in bestimmten Portionen zugeordnet, deren Größe im Plattenparameterblock (siehe Tabelle 12) mit der Blockmaske (BLM) definiert wird. Die Mittel, die SCP 1700 für diese Plattenverwaltung benötigt, sind

- der FCB (siehe Tabelle 6)
- das Plattenbelegungsverzeichnis
- das Dateiverzeichnis

Die 1. nicht reservierte Spur jeder Platte (Diskette) enthält das Dateiverzeichnis des Datenträgers. Es enthält für jede Datei die jeweils ersten 32 Bytes ihres FCB. Bei jedem Neuaufruf einer Platte liest das BDOS dieses Dateiverzeichnis und berechnet aus den Einträgen der Bytes 16 bis 31 jedes FCB die Belegung der Platte. Diese Belegung faßt das BDOS zusammen und legt diese innerhalb des BDOS in dem Plattenbelegungsverzeichnis (Allocation-Bit-Map) zusammen. Dies Verzeichnis spiegelt die Belegung der Diskette wieder. Seine Größe ist abhängig von der Kapazität der Platte. Seine Adresse steht im Plattenparameterkopf (DPH) (siehe Tabelle 11) in den Bytes 14 und 15. Jedes Bit dieses Verzeichnisses repräsentiert einen Block auf der Platte, dessen Größe die BLM festlegt (für 8"-Disketten im Standardformat Blockgröße=1k Byte, für 5,25"-Disketten Blockgröße =2k Byte) Der Ablauf einer Dateieröffnung durch das BDOS läßt sich wie folgt beschreiben:

1. Das BDOS durchsucht das Plattenbelegungsverzeichnis bis es eine Null findet.

2. Es ersetzt die Null durch eine 1, kennzeichnet damit den Block als belegt, und trägt die Blocknummer (1 bis 256) in den o.g. Bereich des FCB (Byte 16 bis 31) ein.

3. Vor jeder Schreiboperation berechnet das BDOS aus der letzten Blocknummer und der nächsten Satznummer (Byte RC des FCB) die Spur und den logischen Sektor wohin der nächste Satz geschrieben wird.

4. Wenn alle logischen Sektoren (128 Byte) eines Blockes beschrieben sind, sucht das BDOS die nächste Null im Plattenbelegungsverzeichnis und macht weiter wie unter 2.

Da das BDOS das Plattenbelegungsverzeichnis stets von vorn durchsucht, können logisch zusammenhängende Teile einer Datei physisch beliebig auf der Platte verteilt sein.
Der Aufbau und die Organisation der zu dieser Plattenverwaltung notwendigen Tabellen werden im folgenden vorgestellt. Sie können bei der Erstellung eines nutzereigenen BIOS von Hand codiert bzw.

automatisch durch das Systemprogramm GENDEF generiert werden.

## 7.2. Plattenparameterkopf (DPH)

Jedem Plattenlaufwerk ist ein (16 Byte-) Plattenparameterkopf zugeordnet, der Informationen über das Plattenlaufwerk enthält und einen Informations-Bereich für bestimmte BDOS-Operationen bereitstellt.
Der Plattenparameterkopf hat folgendes Format:

| Plattenparameterkopf | | | | | | | |
|---|---|---|---|---|---|---|---|
| XLT | 0000 | 0000 | 0000 | DIRBUF | DPB | CSV | ALV |
| 16b | 16b | 16b | 16b | 16b | 16b | 16b | 16b |

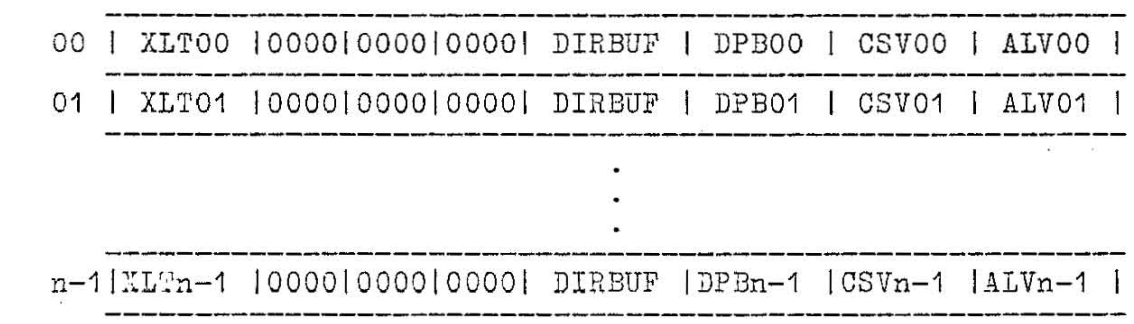
Bild 12: Plattenparameterkopf

Jedes Element ist ein 16-Bit-Wert. Die Elemente des DPH haben folgende Bedeutung:

Tabelle 11: Plattenparameterkopf (DPH)

| Byte | Bezeichnung | Bedeutung |
|---|---|---|
| 0,1 | XLT | Adresse der Umrechnungstabelle logischer-physischer Sektor. Enthält diese Tabelle den Wert 0000H, bedeutet das, die logische Sektoradresse ist gleich der physischen. Das gilt für alle logischen Plattenlaufwerke. |
| 2...7 | 0000 | Bytes, die für interne Nutzung durch BDOS reserviert sind. (der Initialisierungswert ist ohne Bedeutung) |
| 8,9 | DIRBUF | Adresse eines 128-Byte-Puffers für das Einlesen des Dateiverzeichnisses (alle DPH enthalten die gleiche Adresse) |
| 10,11 | DPB | Adresse des Plattenparameterblockes (siehe Tabelle 12). Jedes logische Plattenlaufwerk hat einen eigenen Plattenparameterblock. |
| 12,13 | CSV | Adresse eines Puffers, der für das Speichern eines Prüfvectors (zur Prüfung auf Plattenwechsel) erforderlich ist. Jedes logische Laufwerk benötigt einen solchen Puffer. |
| 14,15 | ALV | Adresse des Plattenbelegungsverzeichnisses, der die Plattenspeicherbelegung widerspiegelt. Eine 1 auf dem Bit n dieses Vectors bedeutet, daß der Block n auf der Platte von einer Datei belegt ist. Eine 0 bedeutet, daß der Block nicht belegt ist. Die ersten Blöcke und damit die ersten Bits sind durch das Dateiverzeichnis belegt. |

Bei n Laufwerken sind die DPH's in einer Tabelle angeordnet, deren erste 16-Byte-Reihe dem Laufwerk 0, die letzte Reihe dem Laufwerk n-1 entspricht.



Die Marke DPBASE definiert den Offset der DPH-Tabelle relativ zum DS-Register.
Aufgabe der in Abschnitt 6 aufgeführten SELDSK-Subroutine ist es, den Offset des DPH vom Anfang des Operationssystems für das ausgewählte Laufwerk im Register BX bereitzustellen. Die folgende Befehlsfolge ist ein Beispiel für die SELDSK-Routine. Existiert das Laufwerk nicht, wird im Register BX der Wert 0000H übergeben.

```
NDISKS  EQU     4               ; Anzahl der Laufwerke
 ....
SELDSK:
        ; SELECT DISK N GIVEN BY CL
        MOV     BX,0000H        ; Bereitmachen für Fehler
        CMP     CL,NDISKS       ; N größer als max. Anzahl
                                ; Laufwerke?
        JNB     RETURN          ; Ja
        MOV     CH,0            ;
        MOV     BX,CX           ; BX = N
        MOV     CL,4            ;
        SHL     BX,CL           ; N = N*16
        MOV     CX,OFFSET       ; DPBASE
        ADD     BX,CX           ; DPBASE+N*16
RETURN: RET
```

Die Übersetzungsvektoren (XLT00 bis XLTn-1) liegen im BIOS und stimmen mit den logischen Sektornummern 0 bis n-1 überein. Der Plattenparameterblock für jedes Laufwerk ist komplizierter. Ein bestimmter DPB, der von einem oder mehreren DPH adressiert wird, hat die allgemeine Form

| SPT | BSH | BLM | EXM | DSM | DRM | AL0 | AL1 | CKS | OFF | PSH | PSM | DW |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16b | 8b | 8b | 8b | 16b | 16b | 8b | 8b | 16b | 16b | 8b | 8b | 8b |

wobei jedes Element ein Byte- oder Wortwert ist, gekennzeichnet durch "8b" oder "16b" unter dem Feld. Die Felder sind in Tabelle 12 definiert.

Tabelle 12: Plattenparameterblock

| Byte | Bezeichnung | Bedeutung |
|---|---|---|
| 0,1 | SPT | Anzahl von 128-Byte-Einheiten/Spur |
| 2 | BSH | Blockgrößenfaktor<br>Es bedeuten z.B.:<br>3 Blockgröße 1 KByte<br>4 " 2 "<br>5 " 4 "<br>6 " 8 "<br>7 " 16 " |
| 3 | BLM | Blockmaske<br>(Anzahl der Sätze/Block minus 1)<br>Es bedeuten z.B.:<br>7 Blockgröße 1 KByte<br>15 " 2 " |
| 4 | EXM | Extentmaske (abhängig von der Blockgröße und der Anzahl der Blöcke/Platte)<br>Es bedeuten z.B.:<br>0 1 Extent/Verzeichnis-Eintragung<br>1 2 Extents/Verzeichnis-Eintragung |
| 5,6 | DSM | Anzahl der Blöcke/Platte minus 1<br>z.B.: K5600.20 : 153 (Blockgröße 2K)<br>K5602.10 : 147 (Blockgröße 2K) |
| 7,8 | DRM | Anzahl der Verzeichnis-Eintragungen minus 1<br>z.B.: K5600.20 : 63<br>K5602.10 : 63 |
| 9,10 | AL0,AL1 | Erste 16 Bits des Plattenbelegungsverzeichnisses (Bit 0...15)<br>(dient zum Setzen der Bits im Plattenbelegungsverzeichnis für die durch das Verzeichnis belegten Blöcke) |
| 11,12 | CKS | Länge des Prüfvectors<br>(aus Anzahl der Verzeichnis-Eintragungen dividiert durch 4) |
| 13,14 | OFF | Anzahl der Systemspuren am Anfang der logischen Platte |

Tabelle 12: Plattenparameterblock (Fortsetzung)

<table>
<tr><th>Byte</th><th>Bezeichnung</th><th>Bedeutung</th></tr>
<tr><td>15</td><td>PSH</td><td>Faktor für die physische Sektorgröße<br><br>Sektorgröße — PSH<br>128 — 0<br>256 — 1<br>512 — 2<br>1024 — 3</td></tr>
<tr><td>16</td><td>PSM</td><td>Anzahl von 128-Byte-Einheiten im physischen Sektor minus 1<br><br>Sektorgröße — PSM<br>128 — 0<br>256 — 1<br>512 — 3<br>1024 — 7</td></tr>
<tr><td>17</td><td>DW</td><td>Laufwerkschreiber<br><br>7 6 5 4 3 2 1 0<br>| | |0|0| | | | |<br>0=SS, 1=DS<br>0=FM, 1=MfM<br>0=Sonderbehandlung f. Track0, 1=sonst<br>0=48tpi, 1=96tpi<br>0=Floppy Disk 1=Hard Disk<br>0=8" 1=5,25"</td></tr>
</table>

Die 8 Bits des Laufwerkschreibers (DW) werden zur Initialisierung der Anschlußsteuerung für das entsprechende Laufwerk benötigt.

Obwohl diese Tabellenwerte beim Aufbau eines nutzereigenen BIOS automatisch durch GENDEF erzeugt werden, lohnt es sich, den Inhalt jedes Feldes zu kontrollieren, um Werte, wenn notwendig, korrigieren zu können.

Erläuterungen zu Tabelle 12:

Für einen vorgegebenen BLS-Wert (Blockzuteilungsgröße) ergeben sich folgende dezimale BSH und BLM-Werte:

Tabelle 13: BSH- und BLM-Werte für BLS-Werte

| BLS | BSH | BLM |
|---|---|---|
| 1.024 | 3 | 7 |
| 2.048 | 4 | 15 |
| 4.096 | 5 | 31 |
| 8.192. | 6 | 63 |
| 16.384 | 7 | 127 |

Der Wert von EXM hängt von BLS und davon ab, ob der DSM-Wert kleiner oder größer als 255 ist, wie es in der Tabelle 14 gezeigt wird.

Tabelle 14: Maximale EXM-Werte

| BLS | DSM<256 | DSM>255 |
|---|---|---|
| 1.024 | 0 | N/A |
| 2.048 | 1 | 0 |
| 4.096 | 3 | 1 |
| 8.192 | 7 | 3 |
| 16.384 | 15 | 7 |

Der DSM-Wert ist die maximale Datenblockanzahl, die durch das entsprechende Laufwerk unterstützt wird, gemessen in BLS-Einheiten. Das Produkt aus BLS*(DSM+1) liefert die Gesamtzahl der Bytes, die auf dem Plattenlaufwerk abgespeichert werden können. Sie muß innerhalb der Kapazität der physischen Platte liegen, wobei die reservierten Betriebssystem-Spuren nicht einberechnet sind. Der eingetragene Wert von DSM ist um eins kleiner als die Gesamtzahl der Verzeichnis-Eintragungen, die einen 16-Bit-Wert annehmen kann. Die Werte von AL0 und AL1 werden immer durch DRM festgelegt. Die zwei Werte können zusammen als 16-Bit-Kette betrachtet werden (siehe Bild 13)

| AL0 | | | | | | | | AL1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |

Bild 13: AL0,AL1

wobei Position 00 dem höchstwertigen Bit des mit AL0 markierten Bytes und Position 15 dem niederwertigen Bit des mit AL1 markierten Bytes entsprechen. Jede Bitposition reserviert einen Datenblock für eine Anzahl von Verzeichnis-Eintragungen. Damit wird eine Gesamtzahl von 16 Datenblöcken, die für Verzeichnis-Eintragungen vorgesehen sind, ermöglicht. (Die Bits werden ausgehend von 00 bis 15 gefüllt.) Jede Verzeichnis-Eintragung belegt

32 Bytes, woraus sich Tabelle 15 ergibt:

Tabelle 15: BLS und Zahl der Verzeichnis-Eintragungen

| BLS | Verzeichnis-Eintragungen |
|---|---|
| 1.024 | 32 mal Anzahl der gesetzten Bits |
| 2.048 | 64 mal " " " " |
| 4.096 | 128 mal " " " " |
| 8.192 | 256 mal " " " " |
| 16.384 | 512 mal " " " " |

Beispiel:

Wenn DRM=127 (128 Verzeichnis-Eintragungen) und BLS=1024 sind, dann kommen auf einen Block 32 Verzeichnis-Eintragungen, was 4 reservierte Blöcke erfordert. In diesem Fall sind die 4 höherwertigen Bits von AL0 gesetzt, es ergeben sich die Werte AL0=F0H und AL1=00H.

Der CKS-Wert wird folgendermaßen festgelegt:
Wenn das Plattenspeichermedium auswechselbar ist, dann ist CKS=(DRM+1)/4, wobei DRM die letzte Verzeichnis-Eintragungszahl ist. Wenn das Medium fest (fixed) ist, dann ist CKS=0. (In diesem Fall werden keine Verzeichnis-Eintragungen geprüft.)
Das OFF-Feld legt die Zahl der Spuren fest, die am Anfang der physischen Platte übersprungen werden. Beim Ruf der SETTRK-Subroutine wird dieser Wert automatisch addiert. Er kann für das Überspringen von reservierten Operationssystemspuren oder zum Unterteilen einer großen Platte in kleiner segmentierte Abschnitte benutzt werden.
Verschiedene DPH können den gleichen DPB adressieren, wenn ihre Laufwerkcharakteristika identisch sind. Weiterhin kann der DPB dynamisch gewechselt werden, wenn ein neues Laufwerk adressiert wird. Das geschieht durch einfaches Wechseln der Adresse im DPH, da BDOS die DPB-Werte in einen lokalen Bereich kopiert, wenn die SELDSK-Subroutine aufgerufen wird.
Bei Rückkehr zum DPH eines bestimmten Laufwerkes ist zu beachten, daß die beiden Adreßwerte CSV und ALV erhalten bleiben. Beide Adressen verweisen auf einen Bereich des dem BIOS folgenden nicht initialisierten Speichers. Jedes Laufwerk muß seinen eigenen Bereich erhalten und die Größe jedes Bereiches wird durch die Werte im DPB festgelegt.
Die Größe des durch CSV adressierten Bereiches beträgt CKS Bytes. Das ist ausreichend für die Verzeichnis-Prüf-Operation des entsprechenden Laufwerkes. Wenn CKS=(DRM+1)/4 ist, müssen (DRM+1)/4 Byte bei Verwendung reserviert werden. Ist CKS=0, wird kein Speicher reserviert.
Die Größe des durch ALV adressierten Bereiches ist durch die Maximalanzahl von Datenblöcken festgelegt, die für das ausgewählte Laufwerk gestattet sind. Sie berechnet sich aus (DSM/8)+1.

## 7.3. Tabellengenerierung mittels GENDEF

Das Systemprogramm GENDEF ist ein Hilfsmittel zur Anpassung des SCP 1700 auf nicht standardmäßig unterstützte Externspeicher. Es dient zur Generierung von Steuertabellen für den Zugriff zu Folienspeichern. GENDEF verarbeitet eine Datei

name.DEF

mit Definitionsanweisungen für Folienspeicher und erzeugt daraus eine Assemblerquelldatei

name.LIB

welche Assemblersprachanweisungen enthält, die die Tabellen definieren, die zur Unterstützung einer bestimmten Laufwerkkonfiguration notwendig sind. Das GENDEF-Kommando hat die Form:

GENDEF name Parameterliste

Die Parameterliste besteht aus keinen oder mehreren Symbolen, die in Tabelle 16 definiert sind.

Tabelle 16: GENDEF wahlfreie Parameter

| Parameter | Wirkung |
|---|---|
| ¤C | Erzeugen von Kommentarzeilen |
| ¤0 | Erstellen des DPBASE Offsets ¤ |
| ¤C0 | (alle obengenannten) |

### Erläuterungen zur Tabelle 16:

Parameter ¤C weist GENDEF an, eine Kommentarzeile zu erzeugen. Sie ist der Ausgabe des Systemprogrammkommandos STAT DSK: ähnlich, die die Charakteristiken jeder definierten Platte beschreibt.
Normalerweise ist DPBASE folgendermaßen definiert:

DPBASE EQU ¤

was ein MOV CX,OFFSET DPBASE in der vorn beschriebenen SELDSK Subroutine erfordert. Der ¤0-Parameter erzeugt die Definitionsanweisung

DPBASE EQU OFFSET ¤

wodurch in SELDSK ein MOV CX,DPBASE möglich ist. Die Datei name.DEF kann mittels des SCP 1700-Texteditors erzeugt werden. Sie besteht aus Definitionsanweisungen, die in folgender Reihenfolge einzugeben sind:

```
DISKS     n
DISKDEF   0,...
DISKDEF   1,...
 ...
DISKDEF   n-1
ENDEF
```

Jede Anweisung steht auf einer einzelnen Zeile, wahlweise können Kommentare zwischen den Schlüsselworten, Zahlen und Begrenzungszeichen eingefügt sein.
Die Anweisung DISKS definiert die Zahl der im System zu konfigurierenden Laufwerke, wobei n eine Integerzahl zwischen 1 und 16 ist. Es folgt eine Reihe von DISKDEF-Anweisungen, die die Charakteristiken aller logischen Platten von 0 bis n-1 definieren, die den logischen Laufwerken A bis F zugeordnet sind. Es ist zu beachten, daß die Anweisungen DISKS und DISKDEF feste interne Datentabellen generieren, die im vorhergehenden Abschnitt beschrieben sind. Das bedeutet, daß sie im nichtabarbeitbaren Teil des BIOS abgelegt werden müssen.
Die Anweisung ENDEF (Ende der Plattendefinition) generiert die erforderlichen nichtinitialisierten RAM-Bereiche, die hinter den initialisierten RAM im BIOS liegen.

Die Anweisung DISKDEF hat die Form:

DISKDEF dn,typ[,format]

Erläuterungen:

dn - logische Laufwerksnummer (0...n-1)
typ - Typ des angeschlossenen Laufwerks:
K5600.20 (5,25", 96tpi, MFM)
K5602.10 (8", FM)
K5600.10 (5,25", 48tpi, MFM)
MF3200 (8", FM)
MF6400 (8", MFM)
format - Formatierung der Disketten für das angeschlossene Laufwerk:
H - "SCP-Hausformat"
S - "CP/M-Standardformat" (Nur für 8"-Disketten)

Wenn 'format' nicht angegeben wurde, wird "H" angenommen.

Folgende Diskettenformate werden erzeugt:

| typ | format | |
|---|---|---|
| | H | S |
| K5600.20 | Spur0: 16 Sektoren (FM) a' 128 Byte<br>Spur 1-79: 16 Sektoren (MFM) a' 256 Byte<br>3 Systemspuren<br>154 Datenblöcke a 2 KByte<br>64 Verzeichniseinträge<br>306 KByte Anwenderkapazität | nicht angebbar |
| K5602.10 (FM) oder<br><br>MF3200 (FM) | Spur0: 26 Sektoren a' 128 Byte<br>Spur1-76: 4 Sektoren a' 1024 Byte<br>3 Systemspuren<br>148 Datenblöcke a' 2KByte<br>64 Verzeichniseinträge<br>294 KByte Anwenderkapazität | Spur0-76: 26 Sektoren a' 128 Byte<br><br>Skewfaktor:6<br>2 Systemspuren<br>243 Datenblöcke a' 1KByte<br>64 Verzeichniseinträge<br>241 KByte Anwenderkapazität |
| K5600.10 | Spur0: 16 Sektoren (FM) a' 128 Byte<br>Spur1-39: 16 Sektoren a' 256 Byte<br>3 Systemspuren<br>74 Datenblöcke a' 2 KByte<br>64 Verzeichniseinträge<br>146 KByte Anwenderkapazität | nicht angebbar |
| MF6400 | Spur0: 26 Sektoren (FM) a' 128 Byte<br>Spur1-76: 8 Sektoren (MFM) a' 1024 Byte<br>2 Systemspuren<br>300 Datenblöcke a' 2 KByte<br>128 Verzeichniseinträge<br>596 KByte Anwenderkapazität | Spur0-76: 26 Sektoren (FM) a' 128 Byte<br>Skewfaktor: 6<br><br>2 Systemspuren<br>243 Datenblöcke a' 1 KByte<br>64 Verzeichniseinträge<br>241 KByte Anwenderkapazität |

## 7.4. GENDEF-Ausgabe

GENDEF gibt ein Verzeichnis der Anweisungen, die in der DEF-Datei enthalten sind, auf dem Systembildschirm aus (CONTROL-P kann verwendet werden, um bei Bedarf eine gedruckte Liste zu erhalten). Jede Quellzeile ist numeriert, und irgendwelche Fehler werden unterhalb der fehlerhaften Zeile mit einem "?" angezeigt. Die Ursache der Fehler ist in den Tabellen 17 und 18 aufgelistet.

Tabelle 17: GENDEF-Fehlermitteilungen (bei Quellanweisungen)

| Mitteilung | Bedeutung |
|---|---|
| Bad Val | In der DISKS-Anweisung wurden mehr als 16 Platten definiert oder in einer DISKDEF-Anweisung wurde ein falscher typ- oder format-Parameter angegeben. |
| Convert | Ziffer kann nicht konvertiert werden, sie muß eine Binär-, Oktal-, Dezimal- oder Hexadezimalkonstante im Sinne von ASM 86 sein. |
| Duplic | Doppeldefinition eines Plattenlaufwerks |
| Missing | Eine Parameterangabe fehlt. |
| No Stmt | Das Anweisungsschlüsselwort wurde nicht erkannt. |
| Numeric | Parameter der DISKS-Anweisung bzw. 1. Parameter der DISKDEF-Anweisung ist keine Zahl. |
| Sequence | Anweisungsreihenfolge falsch |

Tabelle 18: GENDEF Ein- und Ausgabe-Fehlermitteilungen

| Mitteilung | Bedeutung |
|---|---|
| Cannot Close ".LIB"File | Das Schließen der LIB-Datei war nicht erfolgreich, z.B. auf Grund des Hardware-Schreibschutzes. |
| "LIB" Disk Full | Kein Speicherplatz für die LIB-Datei. |
| No Input File Present | Die angegebene DEF-Datei wurde nicht gefunden. |
| No "LIB" Directory Space | Die LIB-Datei kann nicht erstellt werden, da bereits zu viele Dateien auf der LIB-Platte sind. |

# 8. SCP 1700 Anfangsladen und Anpassungsverfahren

## 8.1. Allgemeines

Dieser Abschnitt beschreibt die Komponenten der SCP 1700-Standard-Lieferplatte und die Verfahren zur Anpassung des SCP 1700 an nicht standardgemäße Hardware.
SCP 1700 wird auf einer 5 1/4" Diskette geliefert. Die ersten drei Spuren sind für den Anfangslader reserviert, während der Rest der Diskette die Dateiverzeichnisinformation, das Betriebssystem und die Programm- und Datendateien enthält.
Die prinzipiellen Komponenten des Liefersystems sind:

- der Anfangslader (LDSCP.CMD)
- das SCP 1700 System (SCP.SYS)

Von der BOOT-Routine des Monitors wird das Anfangsladeprogramm (LDSCP.CMD) von den ersten Spuren der Diskette in den Hauptspeicher gelesen und ihm die Steuerung übergeben.

## 8.2. Die Anfangsladefunktion

Das LDSCP -Programm ist eine einfache Version des SCP 1700, die genügend Dateibearbeitungsmöglichkeiten enthält, um SCP.SYS von der Systemplatte in den Speicher zu lesen. Wenn der LDSCP seine Arbeit beendet, erhält das SCP.SYS-Programm die Steuerung und fährt fort, die Bedienereingabekommandos zu bearbeiten.
Sowohl dem LDSCP als auch dem SCP.SYS sind Standard CMD-Kopfsätze vorangestellt. Der 128 Byte LDSCP Kopfsatz enthält einen einzelnen Bereichsdescriptor

| G-Form | G-Länge | A-Base | G-Min | G-Max |
|---|---|---|---|---|
| 1 | xxxxxxxxx | 0500 | xxxxxxx | xxxxxxx |
| 8Bit | 16Bit | 16Bit | 16Bit | 16Bit |

Bedeutung

G-Form = 1
: Das Programm enthält nur einen Code-Bereich (8080-Speichermodell)

xxxxx
: Diese Felder werden ignoriert

A-Base
: Dieses Wort enthält die Basisparagraphenadresse für den LDSCP. Ab dieser Adresse wird vom Monitor der LDSCP auf dem Hauptspeicher abgespeichert.

Der LDSCP.CMD kann an jeder Paragraphengrenze, die nicht mit SCP 1700 überlappt, geladen und ausgeführt werden. LDSCP.CMD selbst besteht aus drei Teilen:
- dem Loader CCP (LDCCP)
- dem Loader Basic Disk System (LDBDOS)
- dem Loader Basic-System (LDBIOS)

Im Bild 14 wird die Organisation des LDSCP dargestellt:

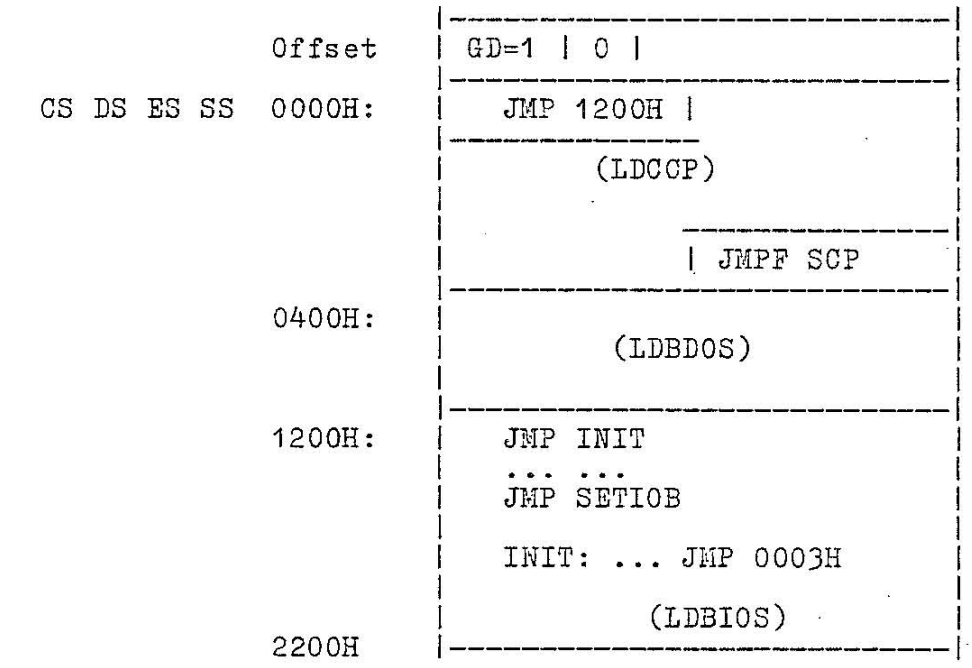



Bild 14: Organisation des LDSCP.CMD

GD=1 ist der Gruppendescriptor für die oben beschriebene LDSCP Codegruppe, unmittelbar gefolgt von einem "0"-Gruppenbegrenzer. Das vollständige LDSCP-Programm, außer dem Kopfsatz, wird von der BOOT-Routine des Monitors beginnend an der Paragraphenadresse 0500H entsprechend dem A-Feld, eingelesen. Nach Beendigung des Lesens übergibt der Monitor die Steuerung zum Platz 05000H, wo das LDSCP-Programm die Arbeit beginnt.
Der Befehl JMP 1200H am Beginn des LDCCP überträgt die Steuerung an den Anfang des LDBIOS, der die Steuerung dann an die INIT-Subroutine übergibt. Die Subroutine, bei INIT beginnend initialisiert die Geräte, bringt eine Meldung auf dem CONSOLE-Gerät und kehrt zum LDCCP-Programm beim Byteoffset 003H zurück. Der LDCCP-Modul eröffnet die SCP.SYS-Datei, lädt das SCP 1700-System in den Speicher und übergibt die Steuerung an SCP 1700 durch den Befehl JMPF SCP am Ende der LDCCP-Ausführung als Beendigung des Anfangsladens.
Die Dateien LDCCP.H86 und LDBDOS.H86 sind im Liefersatz des SCP 1700 enthalten, so daß der Nutzer auch ein an andere Hardware angepaßtes LDBIOS in die Konstruktion des benötigten LDSCP selbst einfügen kann. BIOS.A86 enthält einen Schalter zur bedingten Assemblierung, genannt "Loader bios", der, wenn gesetzt, das benötigte LDBIOS produziert.
Man benutzt ASM86 zur Assemblierung des LDBIOS.A86 Programms:

```
ASM86 LDBIOS
```

und zur Herstellung der LDBIOS.H86-Maschinencodedatei.
Man verkettet die drei LDSCP-Module mit PIP

```
PIP LDSCP.H86=LDCCP.H86,LDBDOS.H86,LDBIOS.H86
```

um eine Maschinencodedatei des LDSCP-Programms herzustellen.

Obwohl das Standard-LDSCP-Programm am Offset 2200H endet, kann das veränderte LDBIOS von dieser letzten Adresse abweichen mit der Einschränkung, daß der LDSCP auf die ersten Spuren paßt und keine SCP 1700-Felder überschreibt. Mit Hilfe des Dienstprogrammes GENCMD wird das abarbeitungsfähige Programm (CMD-Datei) erstellt:

```
GENCMD LDSCP 8080 CODE[A500]
```

dabei wird eine Datei LDSCP.CMD mit einem Kopfsatz, der ein 8080-Speichermodell mit einer absoluten Paragraphadresse von 500H, bzw. Byteadresse 5000H, erstellt.

Falls man zu einem lauffähigen SCP 1700-System zugreifen kann, kopiert das Kommando

```
LDCOPY LDSCP
```

den LDSCP auf die Systemspuren. Man hat jetzt eine Platte mit einem LDSCP-Programm, welches das zum Lesen der SCP.SYS-Datei in den Speicher benötigte LDBIOS enthält.
Der LDSCP ist statisch umlagerbar und seine Anfangsadresse wird durch den Wert von A-Base im Kopfsatz bestimmt.

## 8.3. Die Organisation des SCP.SYS

Die SCP.SYS-Datei durch das LDSCP-Programm gelesen, besteht aus dem CCP, BDOS und BIOS im CMD-Dateiformat mit einem 128-Byte Kopfsatz ähnlich dem des LDSCP-Programms.

| G-Form | G-Länge | A-Base | G-Min | G-Max |
|---|---|---|---|---|
| 1 | xxxxxxxx | 0104 | xxxxx | xxxxx |
| 8Bit | 16Bit | 16Bit | 16Bit | 16Bit |

Die vollständige SCP.SYS-Datei auf der Platte wird im Bild 15 dargestellt

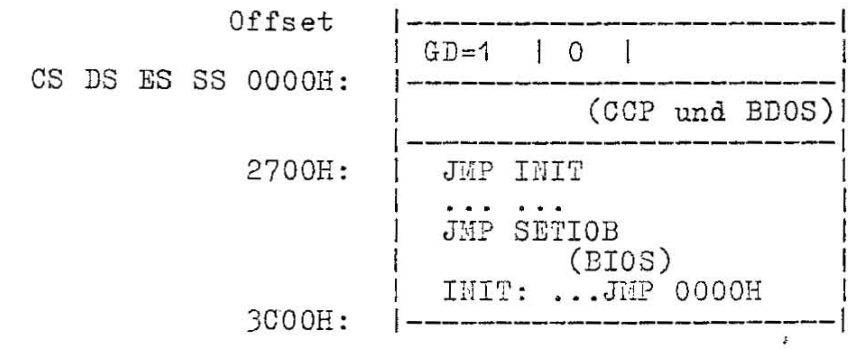



Bild 15: Organisation des SCP.SYS

wobei GD=1 der Gruppendescriptor ist, der den A-Basiswert enthält, und von einem "0"-Begrenzer gefolgt wird.
Die SCP.SYS-Datei wird durch den LDSCP, beginnend mit der durch A-Base gegebenen Adresse (Byteadresse 0104OH), gelesen und die

Steuerung wird dem INIT-Eintrittspunkt an der Offset-Adresse 2700H übergeben. Jede zusätzliche Initialisierung, die nicht vom LDSCP ausgeführt wird, wird in der INIT-Subroutine durchgeführt, und nach der Beendigung erfolgt durch INIT ein JMP 0000H zum Beginn des CCP. Die aktuelle Ladeadresse von SCP.SYS wird vollständig durch die im A-Basisfeld gegebene Adresse bestimmt. Sie kann verändert werden, wenn man die Ausführung des SCP 1700 in einer anderen Speicherregion wünscht.
Ähnlich dem LDSCP-Programm, kann man das BIOS durch Veränderung von BIOS.A86, das sich im Liefersatz des SCP 1700 befindet, modifizieren. Das benötigte BIOS, das die speziellen I/O-Driver enthält, wird erstellt und mit

ASM86 BIOS

assembliert zu BIOS.H86, das den BIOS-Maschinencode enthält. Das neue BIOS wird mit dem SCP.H86 des Liefersatzes verkettet:

PIP SCP.H86=SCIPX.H86,BIOS.H86

Die entstandene CCP, BDOS und BIOS HEX-Datei wird dann zu einer CMD-Datei konvertiert, durch

GENCMD SCP 8080 CODE[A104]

Es entsteht ein CMD-Speicherabbild mit der A-Basis=104H. Schließlich wird die SCP-Datei durch

REN SCP.SYS=SCP.CMD

umbenannt und auf der Systemdiskette plaziert.
Das LDSCP-Programm liest die SCP.SYS-Datei mit dem angepaßten BIOS in den Speicher ab der Adresse 01040H. Die Steuerung wird an das SCP 1700 übergeben. SCP 1700 bleibt im Speicher bis die nächste Anfangsladeoperation es wieder einliest.
Man kann die Zwei-Schritt-BOOT-Operation vermeiden, wenn man eine Diskette zur Verfügung hat, die genügend Platz zur Aufnahme des vollständigen SYS.SCP auf den Systemspuren bietet.
In diesem Fall bringt der Anfangslader das Speicherabbild des SCP 1700 in den Speicher ab der durch A-Base angegebenen Adresse und übergibt die Steuerung an den INIT-Eintrittspunkt mit dem Offset 2700H.

## Abkürzungsverzeichnis

| | |
|---|---|
| ACS | Arbeitsplatzcomputersystem |
| ASM | Assembler ASM86 |
| BDOS | Basic Disk Operating System |
| BIOS | Basic I/O-System |
| BLM | Blockmaske (Anzahl der Sätze/Block) |
| BLS | Blockzuteilungsgröße |
| BS | Betriebssystem |
| BSH | Blockgrößenfaktor |
| CCP | Console Command Processor |
| CKS | Größe des Prüfvektors |
| CMD | Dateityp für abarbeitungsfähige Programme |
| CP/M | Name eines Betriebssystems der Firma Digital Research |
| CR | Carriage Return (Wagenrücklauf) |
| DDT | Debug-Programm |
| DMA | Direkte Hauptspeicheradresse |
| DPB | Platten Parameterblock |
| DRM | Anzahl der Verzeichniseintragungen minus 1 |
| DSM | Anzahl der Blöcke/Diskette minus 1 |
| EOV | Marke End of Volume |
| EXM | Extentmaske |
| FCB | Dateisteuerblock |
| FM | Aufzeichnungsformat "einfache Dichte" |
| GENCMD | Generierprogramm für CMD-Dateien |
| GENDEF | Generierprogramm für Plattenparameterblöcke |
| I/O | E/A (Ein-/Ausgabe) |
| LF | Line feed (Zeilenschaltung) |
| MCB | Speichersteuerblock |
| MFM | Aufzeichnungsformat "doppelte Dichte" |
| MMS | Mikrorechner-Modulsystem |
| R/O | Read/Only |
| R/W | Read/Write |
| SCP | Single User Control Programm |
| SP | Stackpointer |
| STAT | Dienstprogramm STAT |

## Sachwortverzeichnis