Max-Planck-Institut für Strömungsforschung
Bunsenstraße 10
D-3400 Göttingen

M. Pätzold
Telefon (0551)709-2714

Die Zettelsammlung wird ständig vervollständigt und überarbeitet. Viele Angaben sind mehrfach vorhanden, weil sie von unterschiedlichem Standpunkt betrachtet werden, z. B. als zum Prozessor, BIOS oder Betriebssystem gehörig.

Stand
26. April 1993

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Diese Sammlung von Bemerkungen zur AT-Hardware und zu MS-DOS wird nie abgeschlossen sein.
Ein AT-Personal-Computer nach IBM-Rezept [1] besteht aus mindestens den folgenden Komponenten:

- Dem Processor 80286.
- Dem Bus-Controller z. B. 82288.
- Dem Clock-Generator z. B. 82284.
- Dem Interrupt-Controller z. B. 8259A.
- Dem DMA-Controller.
- Den Data-Bus-Tranceivern.
- Den Address-Bus-Latches.
- Dem DRAM-Controller mit ECC.
- Dem Memory bis maximal 16MB.
- Dem optionalen Co-Processor 80287.
- Dem ROM, in dem sich das BIOS (Basic Input Output System) befindet.

Es sind neuerdings chip sets erhältlich, die es durch höhere Integration erlauben einen AT mit erheblich weniger Komponenten herzustellen (Baby-AT).
Im Folgenden werden einige Teile der Hardware und Software des bei uns verwendeten AT-Clones beschrieben. Es ist nur die Sammlung von Informationen aus den unterschiedlichsten Quellen und kann daher nicht vollständig sein. Es wird nur auf die unter DOS 3.3 gegebenen Möglichkeiten in "real address mode" eingegangen. Darüber hinausgehendes soll nur aufzeigen, was noch alles machbar ist (Queue, Tasking). Für die Rechner mit 80386 Prozessor werden die Erweiterungen angegeben. Manche Informationen sind mehrfach vorhanden, da sie in unterschiedlichem Zusammenhang betrachtet werden müssen.

Quellen:

- Operation Manual, Computer Innovations Comp. (Hardware Manual DOC NO. 880922 REV. 1.0)
- DOS Referenzhandbuch, IBM, 12/85 (8130063/1)
- MASM Language Reference, IBM, 1987 (00F8619)

[1] warum einfach wenn es auch umständlich geht?

- MS-DOS Programmer's Reference Manual, Microsoft (Markt & Technik)
- DOS Technical Reference Manual, IBM, 1986 (6183355)
- Technical Reference Manual, IBM, 1987 (6280059)
- The MS-DOS Encyclopedia, Microsoft, 1988 ISBN 1-55615-049-0
- System BIOS for IBM PC/XT/AT Computers and Compatibles, Phoenix Technologies Ltd., 1990 ISBN 0-201-51806-6

1.1 Allgemeines

Der AT ist ein Byte-orientierter 16-Bit Rechner mit 24-Bit Adresse. Es können mit einem Zugriff sowohl 8-Bit (Byte) als auch 16-Bit (Wort) Daten erreicht und vom Prozessor verarbeitet werden. Eine ungerade (odd) Adresse erreicht das ungerade Byte und eine gerade Adresse kann sowohl das gerade (even) Byte als auch das Wort erreichen. Auch ein Wortzugriff auf eine ungerade Adresse ist möglich, benötigt aber zwei Zyklen.

Im "real address mode" werden 20-Bit Adresse (1MB) und im "protected mode" werden 24-Bit Adresse (16MB) vom Prozessor erzeugt. Die physikalische Adresse im "real address mode" wird aus dem Inhalt eines Segment-Registers und dem eines Offset-Registers erzeugt, wobei der Inhalt des Segment-Registers um vier Bit nach links (multipliziert mit 16) in ein 20-Bit Register geschoben und der Offset addiert wird. Schreibweise z.B. CS:IP, wobei CS das Code Segment Register und IP der Instruction Pointer ist. Die physikalische Adresse im "protected mode" wird auch mit zwei Registern (low und high) dargestellt. Von diesen 32 Bit (2 x 16 Bit) sind allerdings nur 24 Bit als Adresse auf dem Bus vorhanden. Es können also 16 MByte adressiert werden von 1 GByte die im Programm unterschieden werden. Die Bits 30 und 31 des "Adressregisters" werden beim 80286 nicht erzeugt. Der virtuelle Adressbereich von 1 GByte wird mit Selektoren verwaltet.

Da MS-DOS nur 640KB verwalten kann, kann das restliche Memory nur von Programmen oder als virtuelle Disk mit device=vdisk.sys/e 384 im config.sys genutzt werden. 384 ist die Differenz zwischen den im Rechner vorhandenen 1024KB und den von MS-DOS benutzbaren 640KB.Die Option /e muß gegeben werden, um das vom DOS nicht benutzte Memory zu benutzen. Die virtuelle Platte kann z.B. Compiler beschleunigen. Bei PCTEX wird sie als letzte Option mit /Z=e (e = Drive Nr.) angegeben. Damit werden schnelle Zwischenspeicherungen ermöglicht.

Grundsätzlich können alle Teile direkt programmiert werden. Leider wird das durch weglassen von Informationen erschwert. Deshalb ist es für jeden günstiger die Ebene des Betriebssystems nicht zu verlassen. Falls das doch einmal notwendig sein sollte, müßte das BIOS ausreichend sein. Oder wollt ihr ein neues BS schreiben? Tanenbaum [2] hats gemacht!

Alle im folgenden angeführten Adressen oder Inhalte von Adressen oder Registern sind entweder revisionsabhängig oder werden durch die Systemkonfiguration bestimmt. Die Angaben über die Hardware Parameter sind den entsprechenden Manuals entnommen.

Die 20 Adressbits, das low-Data-Byte, die Interrupts des ersten Interrupt-Controllers und IRQ9 des zweiten Interrupt-Controllers befinden sich auf dem 8-Bit Steckplatz (XT). Die restlichen Adressen, das high-Data-Byte und die restlichen Interruptleitungen befinden sich nur auf dem zweiten Bus-Stecker (AT-Extension).

1.2 Prozessor Architektur

Im Bild 1.1 auf Seite 3 wird der Zusammenhang der vier internen Einheiten des 80286 aufgezeigt. Sie arbeiten von einander unabhängig und parallel zu einander. Diese Architektur wird von Intel mit 'pipelined' bezeichnet.

[2] Operating Systems 'Design and Implementation'; Prentice-Hall Int.; ISBN 0-13-637331-3

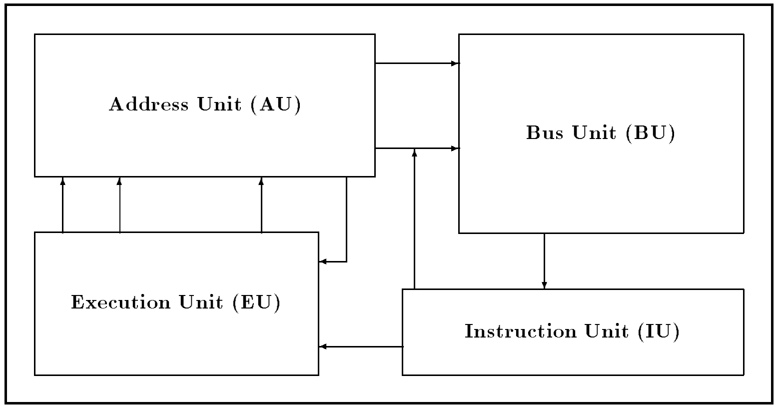


Abbildung 1.1: Internal Block Diagram 80286

1.2.1 Bus Unit

In Bild 1.2 auf Seite 4 wird die Funktionsweise der Bus Unit dargestellt.

Die Bus Unit vollzieht alle Busoperationen für die CPU, erzeugt die Adressen, die Daten und die Signale, die für den Zugriff auf externes Memory und I/O erforderlich sind. Die BU kontrolliert auch das Interface für die Prozessorerweiterungen (Coprocessor) und andere 'local bus masters'. Die meistens Bussignale kommen aus der BU oder sind für sie bestimmt.

Wenn der Bus nicht belegt ist und die BU nichts anderes zu tun hat, holt sie sich Instruktionen vom Memory. Dabei geht sie davon aus, daß das Programm sequential abgearbeitet wird, das heißt, daß die nächste Instruktion der vorhergehenden im Memory folgt. Wenn die Prefetchqueue das Ende des Segmentbereichs erreicht, wird keine weitere Instruktion geholt. Wenn das Programm an einer anderen Stelle weiter macht, wird die Queue zurückgesetzt und die Queue neu gefüllt. Die Queue ist 6-Byte lang.

1.2.2 Instruction Unit

In Bild 1.3 auf Seite 4 wird die Funktionsweise der Instruction Unit dargestellt.

Die Instruction Unit (IU) bekommt die Instruktionen aus der 'prefetch queue', dekodiert sie und stellt sie volldekodiert in der 'instruction queue' der Execution Unit zur verfügung. Die Instruction Queue ist 3 Instruktionen tief.

1.2.3 Execution Unit

In Bild 1.4 auf Seite 4 wird die Funktionsweise der Execution Unit dargestellt.

Dekodierte Instruktionen werden durch die EU von der IU geholt und ausgeführt. Die EU benutzt die BU zum Datentransfer vom und zum Memory oder I/O.

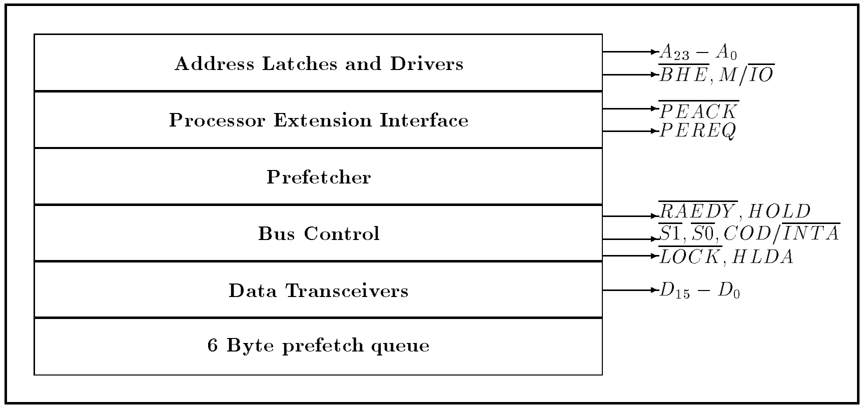


Abbildung 1.2: Bus Unit

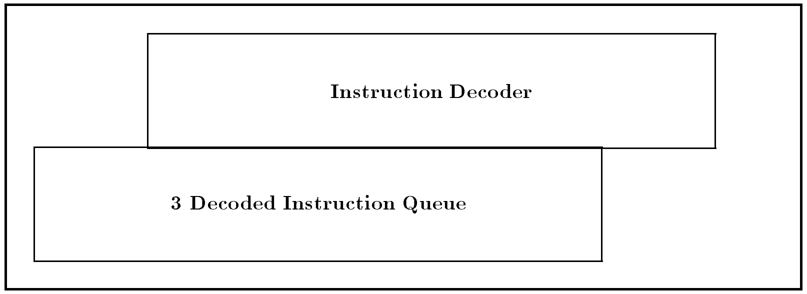


Abbildung 1.3: Instruction Unit

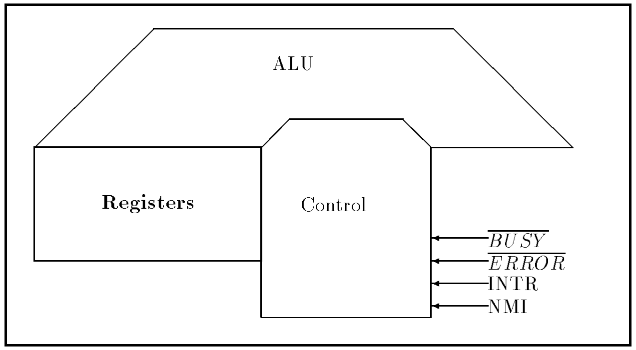


Abbildung 1.4: Execution Unit

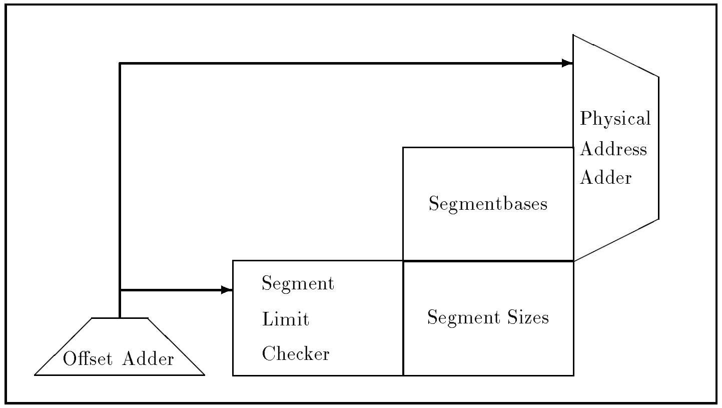


Abbildung 1.5: Address Unit

1.2.4 Address Unit

In Bild 1.5 auf Seite 5 wird die Funktionsweise der Adress Unit dargestellt.

Die AU bewerkstelligt das Memory Management und Protektion der CPU und übersetzt logische in physikalische Adressen für die BU. Ein Registercache in der AU enthält die Informationen für die vielfältigen memory translation und protection checks für jeden Buszyklus.

1.2.5 Instruction Pipelining

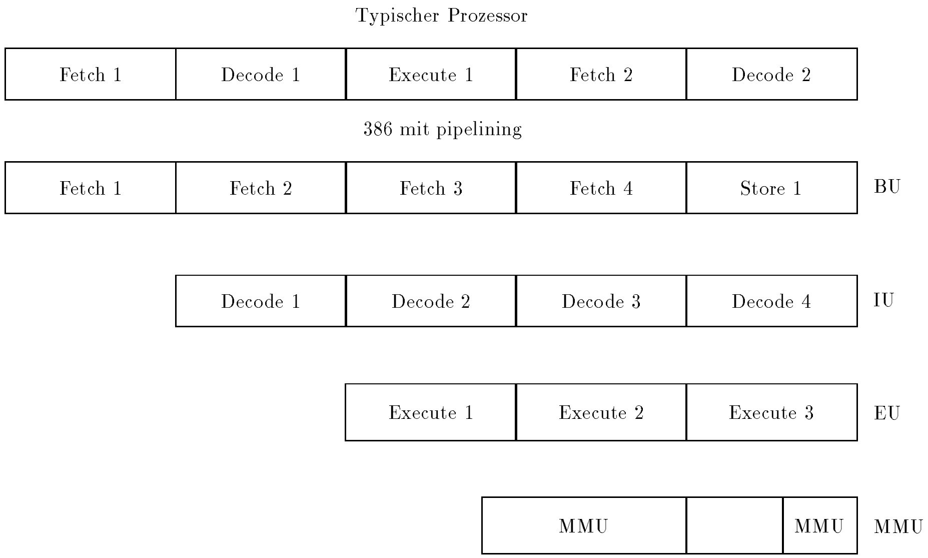
Durch die Aufteilung des Processors in die vorher aufgezeigten Funktionseinheiten können unterschiedliche Arbeiten gleichzeitig, **wirklich gleichzeitig**, bearbeitet werden. Beim abarbeiten eines Befehls werden typischerweise die folgenden Situationen entstehen:

1. Bereitstellen des Befehls oder der Daten (fetch).
2. Dekodieren (decode).
3. Ausführen (execute).
4. Neue Adresse berechnen (memory management).

Ist der erste Befehl an die **EU** abgegeben worden kann die **IU** den nächsten Befehl dekodieren (s. Bild 1.6 auf Seite 6).

1.3 Memory System des 80286

Im Protected Virtual-Address Mode kann der 80286-Prozessor 16 Megabyte physikalisches Memory adressieren. Dieser große Bereich kann in Bereiche aufgeteilt werden, die unterschiedliche Charakteristiken und Funktionen haben können. Diese Teile können



æ

Abbildung 1.6: Instruction Pipelining

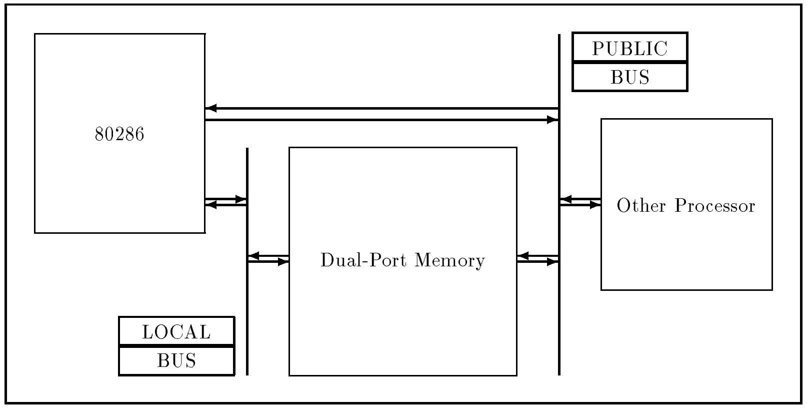


Abbildung 1.7: Dual-Port Memory

1. an einem local bus angeschlossen sein, also nur von einem Prozessor zu benutzen, oder
2. an einem public bus angeschlossen sein, so daß alle an diesen bus angeschlossenen Prozessoren dieses Memory nutzen können, oder
3. sowohl an einen local als auch einen public bus angeschlossen sein, also dual-ported.

Ein typisches 80286 System wird eine Kombination der drei Möglichkeiten benutzen.

1.3.1 Dual-Port Memory

In Bild 1.7 auf Seite 7 wird das Prinzip eines dual-ported Memory gezeigt. Für die Regelung der Zugriffsrechte steht z. B. der 8207 zur Verfügung.

1.3.2 System Memory Map

In Bild 1.8 auf Seite 8 wird die Systemmemory Map dargestellt. Der Memory Adress Bereich teilt sich auf in:

1. Private Local Memory.
2. Public System Memory.
3. Local EPROM.

Parallel dazu ist der I/O Adress Bereich vorhanden, der sich in die Bereiche

1. Private I/O und
2. Public I/O

aufteilt. In dem privaten Bereich befinden sich die eigenen Timer, Interrupt- und DMA-Kontroller, Coprozessor u.s.w.. In dem public Bereich befinden sich alle Interface für Bildschirme, Drucker, Plotter u.s.w..

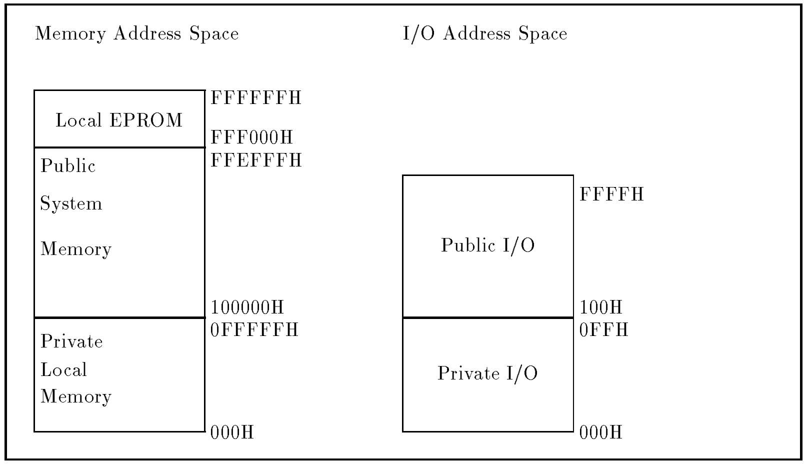


Abbildung 1.8: System Memory Map

1.3.3 Memory Organisation

Da der Prozessor Bytes und Worte, der 80386 Doppelworte, verarbeiten kann, muß das Memorysystem entsprechend ausgelegt sein. Der Zugriff erfolgt über die Adressen $A_1 - A_{19}$ und die Selectleitungen (Byteselect, Bankselect, Chipselect) $\overline{BHE}$ und A_0. Das gerade (even) Byte wird mit aktiver Leitung A_0, das ungerade (odd) Byte wird mit aktiver Leitung $\overline{BHE}$ erreicht. Sind beide aktiv, wird ein Wort erreicht. Wird ein ungerades Wort verlangt, muß zwischengespeichert und umgeordnet werden.

1.4 Pinbelegungen und Bus-Signale

Ein Steckplatz muß die XT-Belegung aufweisen, denn ohne die 8-Bit-Signale kann man nicht auskommen. In der Verlängerung kann ein Steckplatz auch die 16-Bit-Signale aufweisen. 32-Bit-Signale können in einer weiteren Verlängerung des Steckplatzes gefunden werden oder sich, wie beim EISA Steckplatz, in der "zweiten Etage" befinden. Alle Signale sind TTL-kompatibel und können zwei LS-TTL-Lasten treiben (pro Steckplatz hoffentlich, also aufpassen bei Eigenkontruktionen und Zusatzplatinen!!!). Bei reinen XT-PC's wird der Bus synchron mit 4,77 MHz betrieben. Synchron bedeutet eine feste Kopplung der Signale in Timing und Phase an den Bustakt. Die ersten AT's benutzten 6 MHz, die neueren dann 8 MHz. In den Chip-Sätzen wurde dann der Prozessortakt durch einen, meist eintellbaren, Faktor geteilt.

1.4.1 XT-Bus-Signale

In Bild 1.10 auf Seite 9 wird die Pinbelegung des XT-Teils eines Steckplatzes gezeigt.

Die Signale werden meistens durch Open-Collector-Treiber erzeugt. Die meisten sind nicht abgeschlossen und können bei ungünstigem Layout und Bestückung zu Reflexionen neigen, die dann zu nicht reproduzierbaren Abstürzen führen können. Oft hilft dann ein Abschuß mit $40\ldots60\Omega$ in Reihe mit $30\ldots70pF$. Manche Motherboards haben bei einigen Leitungen Pullup-Widerstände, die mit den Kondensatoren des Layout und den der Bestückung das Timing beeinflussen können.

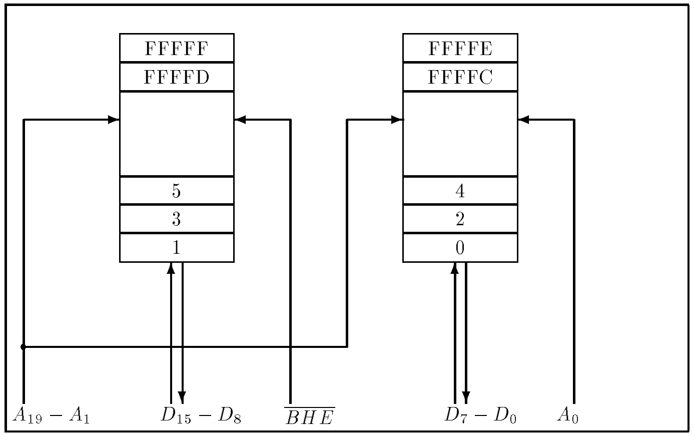


Abbildung 1.9: Memory Organisation

PIN NO.	SIGNAL NAME	I/O	PIN NO.	SIGNAL NAME	I/O
A1	-I/O CH CK	I	B1	Ground	
A2	SD7	I/O	B2	Reset Drive	O
A3	SD6	I/O	B3	+5Vdc	
A4	SD5	I/O	B4	IRQ9	I
A5	SD4	I/O	B5	-5Vdc	
A6	SD3	I/O	B6	DRQ2	I
A7	SD2	I/O	B7	-12Vdc	
A8	SD1	I/O	B8	0 WS	I
A9	SD0	I/O	B9	+12Vdc	
A10	-I/O CH RDY	I	B10	Ground	
A11	AEN	I	B11	-SMEMW	O
A12	SA19	I/O	B12	-SMEMR	O
A13	SA18	I/O	B13	-IOW	I/O
A14	SA17	I/O	B14	-IOR	I/O
A15	SA16	I/O	B15	-DACK3	O
A16	SA15	I/O	B16	DRQ3	I
A17	SA14	I/O	B17	-DACK1	O
A18	SA13	I/O	B18	DRQ1	I
A19	SA12	I/O	B19	Refresh	I/O
A20	SA11	I/O	B20	CLK	O
A21	SA10	I/O	B21	IRQ7	I
A22	SA9	I/O	B22	IRQ6	I
A23	SA8	I/O	B23	IRQ5	I
A24	SA7	I/O	B24	IRQ4	I
A25	SA6	I/O	B25	IRQ3	I
A26	SA5	I/O	B26	-DACK2	O
A27	SA4	I/O	B27	T/C	O
A28	SA3	I/O	B28	BALE	O
A29	SA2	I/O	B29	+5Vdc	
A30	SA1	I/O	B30	OSC	O
A31	SA0	I/O	B31	Ground	

Abbildung 1.10: Pinbelegung der Steckplätze (XT-Teil)

SA0-SA19 20 Adressleitungen. Bei I/O-Buszyklen werden nur die zehn niedrigsten benutzt. Sie werden entweder vom Prozessor oder dem DMA-Kontroller erzeugt. Sie werden mit der steigenden $\uparrow$ Flanke des Signals "BALE" gelatched und verbleiben dann den gesamten Zugriffszyklus über auf dem Bus. Frühestens 40 ns nach dem Ende des /COMMAND$\uparrow$-Signals. Die Leitungen sind offen.

SD0-SD7 Acht bidirektionale Datenleitungen (Systemdata). Die Leitungen floaten. Nur bei EISA sind $8,2K\Omega$ Pullup-Widerst!ande vorgesehen. Ein Zugriff auf nicht vorhandene I/O-Ports oder Speicheradressen ergeben dann immer FFh. Bei ISA ist das oft auch der Fall, aber nicht immer.

BALE "Bufferd Address Latch Enable". Dieses Signal zeigt die Gültigkeit der an A0-A19 liegenden Adresse während eines Speicherzugriffs an. Mit der fallenden Flanke sollen die SA0-19 übernommen werde. Bei DMA-Zyklen ist das Signal high. Die Signale SA0-19, AEN und /SBHE sind mindestens 29ns vor BALE$\downarrow$ auf den Bus zu legen.

IRQ2-IRQ7 Sechs maskierbare Interrupt-Leitungen. Bei ISA sind die Interrupts flankengetriggert ($IRQx \uparrow$). Bei EISA kann zwischen kompatibler Flanken- und inkompatibler Pegeltriggerung wählen. Die Pullup-Widerstände sind bei ISA $2,2k\Omega$, bei EISA $8,2K\Omega$.

DRQ1-DRQ2,DACK1-DACK3 DMA-Request- und -Acknowledge-Leitungen. Nur bei EISA ist ein 2K Pulldown vorgesehen. Der niedrigstpriorisierte Kanal kann unberücksichtigt bleiben, wenn die anderen Kanäle immer busy sind.

Bereits vor /DACK$\downarrow$ kommt AEN auf den Bus, generiert vom HOLDA der CPU. Dann erst reagiert der DMA-Kontroller mit /DACK$\downarrow$ und legt zusammen mit den Page-Registern die Adressen SA und LA an. Mindestens 100 ns vor dem I/O-COMMAND$\downarrow$ soll AEN auf high gehen. /IOR$\downarrow$ darf bereits 65 ns nach /DACK$\downarrow$ aktiv sein, /IOW$\downarrow$ erst 235 ns.

IO CH RDY Signal, das von langsamen Speicher- oder Peripheriekarten verwendet (gesetzt) wird, um Wait-States zu erzeugen. Darf maximal 10 Taktzyklen lang sein. ($2.5\mu s$) nach IBM Technical Reference. Für die Anforderung ist bei 8-Bit Zugriffen nach /COMMAND$\downarrow$ 362 ns, 398ns bei EISA, Zeit. Bei 16-Bit sind es nur 59 bzw. 86 ns.

IOR/W, SMEMR/W I/O- und Speicher-Read/Write-Strobes (activ low). SMEMR und SMEMW sind low, wenn das untere MB angesprochen wird. IOR/W sind mit mit $4,7k\Omega$ bei ISA bzw. $8,2k\Omega$ abgeschlossen.

OSC Ein 14,31818 MHz-Taktsignal, das von einigen (color graphics) Videoboards verwendet wird und nicht mit dem übrigen Bus synchronisiert ist. Es ist oft nur an einem gekennzeichneten Steckplatz vorhanden.

CLK Das Bus-Taktsignal (4,77 MHz beim PC). Dieser Takt ist mit den Read/Write Strobes synchronisiert und sollte auch nur dafür verwendet werden. Hat bei NEAT-Rechnern nichts mit dem Systemtakt zu tun.

AEN,T/C "Adress Enable" und "Terminal Count"=EOP (end of process). Diese Steuersignale werden für DMA-Zyklen gebraucht. AEN ist high, wenn der DMA Kontroller den Adressbus treibt. T/C ist ein Puls, der am Ende eines DMA-Transfers erzeugt wird.

IO CH CHK Alarmiert den Prozessor bei Paritäts- und anderen Fehlern einer Steckkarte durch einen nichtmaskierbaren Interrupt. Wird nicht benutzt, da scheinbar niemand weder die Hardware noch die Software beherrscht.

Reset DRV Zeigt einen System-Reset an. Wird von "power up", "low line-voltage" oder "RESET" erzeugt.

+5V,-5V,+12V,-12V und **Ground** Stromversorgungsleitungen.

0WS Wird von der Karte gesetzt, wenn der Befehl ohne waitstates ausgeführt werden kann. Bei manchen Rechnern nur am achten Steckplatz verfügbar.

Refresh Zeigt einen Refreshzyklus an, muss also zum Sperren benutzt werden!

PIN NO.	SIGNAL NAME	I/O	PIN NO.	SIGNAL NAME	I/O
C1	SBHE	I/O	D1	-MEM CS16	I
C2	LA23	I/O	D2	-I/O CS16	I
C3	LA22	I/O	D3	IRQ10	I
C4	LA21	I/O	D4	IRQ11	I
C5	LA20	I/O	D5	IRQ12	I
C6	LA19	I/O	D6	IRQ15	I
C7	LA18	I/O	D7	IRQ14	I
C8	LA17	I/O	D8	-DACK0	O
C9	-MEMR	I/O	D9	DRQ0	I
C10	-MEMW	I/O	D10	-DACK5	O
C11	SD8	I/O	D11	DRQ5	I
C12	SD9	I/O	D12	-DACK6	O
C13	SD10	I/O	D13	DRQ6	I
C14	SD11	I/O	D14	-DACK7	O
C15	SD12	I/O	D15	DRQ7	I
C16	SD13	I/O	D16	+5Vdc	
C17	SD14	I/O	D17	-Master	I
C18	SD15	I/O	D18	Ground	

Abbildung 1.11: Pinbelegung der Steckplätze (AT-Teil)

1.4.2 AT-Bus Signale

In Bild 1.11 auf Seite 11 ist die Pinbelegung des AT-Steckplatzteils aufgezeigt.

SD8-SD15 Die acht neuen Datenleitungen.

LA17-LA23 Die neuen latched Adressleitungen für den protected mode.

SBHE "System Bus High Enable". Zeigt an, daß das "high byte" (SD8-SD15) auf dem Datenbus ist.

IRQ10-IRQ12, IRQ14-IRQ15 Zusätzliche Interruptleitungen. IRQ13 ist für den numerischen Coprocessor reserviert.

DRQ0,DACK0 Auf dem AT wird der DMA-Kanal 0 nicht mehr für den Refresh benutzt.

DRQ5-DRQ7, DACK5-DACK7 Steuersignale für die neuen DMA-Kanäle.

MEMR/W Memory Read/Write Strobes im protected mode. Diese werden dann aktiviert, wenn SMEMR/W anliegen (nur im PC-Adressraum). Unklar!! Nicht mehr unklar. Sind low bei jedem Memoryzugriff (vgl. SMEMR/W).

Master Neues Signal, das einer Steckkarte die Eigenschaft des Bus-Masters überträgt. Dies dauert mehrere Buszyklen. Außerdem muß der Master den Bus periodisch (max. $15\mu s$) für den Refresh freigeben oder diesen selbst durchführen.

MEM CS16, I/O CS16 Zeigen dem Motherboard an, daß der Adapter zu 16-Bit Transfers in der Lage ist. Generell: 16 Bit Transfer!

1.4.3 Pinbelegung und Signale der EISA-Erweiterung

Der EISA-Standard hat dem AT-Steckplatz eine zweite "Etage" untergeschoben. AT- und XT-Karten können diese Kontakte nicht erreichen. Sie sind durch Kodierstege gegen zu tiefes einstecken geschützt. Die Leitungen der XT- oder AT-Anschlüsse werden zwischen den Anschlüssen der EISA-Erweiterung auf die Platine geführt.

BE0-BE3 "Byte Enable". Zeigen an, welche Byte-'Viertel' auf dem 32-Bit-Datenbus am aktuellen Buszyklus teilnehmen. Sie entsprechen den gleichnamigen Ausgängen der 80386 und 80486 Prozessoren.

M-IO Unterscheidet EISA-Speicher- von I/O-Zugriffen.

START Zeigt den Beginn eines EISA-Buszyklus an.

CMD Ermöglicht Timing-Kontrolle in einem EISA-Buszyklus.

MSBURST Zeigt an, ob ein Master Bursts ausführen kann.

SLBURST Zeigt an, ob ein Slave Bursts verarbeiten kann.

EX32, EX16 Identifizieren einen Slave als EISA-Board, das 32-Bit bzw. 16-Bit Zyklen verarbeiten kann. Liegt keines dieser Signale an, schaltet der Bus in einen ISA-Zyklus zurück.

EXRDY Am Ende eines Zyklus von einem Slave aktiviert.

MREQn Von einem potentiellen Bus-Master Nummer N signalisierte Bus-Anforderung.

MACKn Zeigt Master Nummer N an, daß ihm der Bus zugeteilt wurde.

D16-D31 Die neuen Datenleitungen.

LA2-LA15, LA24-LA31 Neue Adressbusleitungen. Wie LA17-LA23 sind diese nicht auf der Hauptplatine gelatched und erlauben so einen schnellen Adapterzugriff. LA1 und LA2 konnten durch die Einführung der BE-Leitungen entfallen.

EISA SIGNAL	PIN NO.	SIGNAL NAME	I/O	PIN NO.	SIGNAL NAME	I/O	EISA SIGNAL
CMD	A1	-I/O CH CK	I	B1	Ground	O	GND
START	A2	SD7	I/O	B2	Reset Drive	O	+5Vdc
EXRDY	A3	SD6	I/O	B3	+5Vdc		+5Vdc
EX32	A4	SD5	I/O	B4	IRQ9	I	Hersteller
GND	A5	SD4	I/O	B5	-5Vdc		Hersteller
Kodiersteg	A6	SD3	I/O	B6	DRQ2	I	Kodiersteg
EX16	A7	SD2	I/O	B7	-12Vdc		Hersteller
SLBURST	A8	SD1	I/O	B8	0 WS	I	Hersteller
MSBURST	A9	SD0	I/O	B9	+12Vdc		+12Vdc
W-R	A10	-I/O CH RDY	I	B10	Ground		M-IO
GND	A11	AEN	I	B11	-SMEMW	O	LOCK
reserviert	A12	SA19	I/O	B12	-SMEMR	O	reserviert
reserviert	A13	SA18	I/O	B13	-IOW	I/O	GND
reserviert	A14	SA17	I/O	B14	-IOR	I/O	reserviert
GND	A15	SA16	I/O	B15	-DACK3	O	BE3
Kodiersteg	A16	SA15	I/O	B16	DRQ3	I	Kodiersteg
BE1	A17	SA14	I/O	B17	-DACK1	O	BE2
LA31	A18	SA13	I/O	B18	DRQ1	I	BE0
GND	A19	SA12	I/O	B19	Refresch	I/O	GND
LA30	A20	SA11	I/O	B20	CLK	O	+5V
LA28	A21	SA10	I/O	B21	IRQ7	I	LA29
LA27	A22	SA9	I/O	B22	IRQ6	I	GND
LA25	A23	SA8	I/O	B23	IRQ5	I	LA26
GND	A24	SA7	I/O	B24	IRQ4	I	LA24
Kodiersteg	A25	SA6	I/O	B25	IRQ3	I	Kodiersteg
LA15	A26	SA5	I/O	B26	-DACK2	O	LA16
LA13	A27	SA4	I/O	B27	T/C	O	LA14
LA12	A28	SA3	I/O	B28	BALE	O	+5Vdc
LA11	A29	SA2	I/O	B29	+5Vdc		+5Vdc
GND	A30	SA1	I/O	B30	OSC	O	GND
LA9	A31	SA0	I/O	B31	Ground		LA10
LA7	C1	SBHE	I/O	D1	-MEM CS16	I	LA8
GND	C2	LA23	I/O	D2	-I/O CS16	I	LA6
LA4	C3	LA22	I/O	D3	IRQ10	I	LA5
LA3	C4	LA21	I/O	D4	IRQ11	I	+5V
GND	C5	LA20	I/O	D5	IRQ12	I	LA2
Kodiersteg	C6	LA19	I/O	D6	IRQ15	I	Kodiersteg
D17	C7	LA18	I/O	D7	IRQ14	I	D16
D19	C8	LA17	I/O	D8	-DACK0	O	D18
D20	C9	-MEMR	I/O	D9	DRQ0	I	GND
D22	C10	-MEMW	I/O	D10	-DACK5	O	D21
GND	C11	SD8	I/O	D11	DRQ5	I	D23
D25	C12	SD9	I/O	D12	-DACK6	O	D24
D26	C13	SD10	I/O	D13	DRQ6	I	GND
D28	C14	SD11	I/O	D14	-DACK7	O	D27
Kodiersteg	C15	SD12	I/O	D15	DRQ7	I	Kodiersteg
GND	C16	SD13	I/O	D16	+5Vdc		D29
D30	C17	SD14	I/O	D17	-Master	I	+5V
D31	C18	SD15	I/O	D18	Ground		+5V
MREQn							MACKn

Abbildung 1.12: EISA-Stekplatz

1.5 Die Erweiterungen des 286

Der 80286 hat die gleichen adressierungs modes und basic instructions wie der 8086. Die Befehle für virtual memory, multitasking und protection sind nur in dem "protected mode" wirksam. Im real address mode erzeugen diese Befehle den Interrupt 06H, invalid opcode, der bei IBM nicht benutzt (IRET) wird. Ist also kein Treiber für den Interrupt 06H vorhanden, wird die Benutzung dieser Befehle nicht erkannt. Über den Befehlssatz des 8086 hinaus sind im real address mode noch Instruktionen für subroutine linkage, parameter validation, index calculation und block I/O transfers vorhanden. Auf dem 80286 befindet sich auch eine MMU (memory management unit) mit der es möglich ist ein Programm bis zu 1Giga-Byte auszuweiten und in einem bis zu 16Mega-Byte großen Memory ablaufen zu lassen.

1.6 Die neuen chips

Die neue Rechnergeneration wird mit einem NEAT-Chipsatz (New Enhanced AT) von "G2", "Chips & Technology" oder "VLSI" ausgeliefert. Dadurch hat sich nicht nur die Zahl der Chips verringert sondern auch der Platzbedarf auf der Hauptplatine. Diese konnte dadurch verkleinert werden und hat trotzdem noch mehr Memory on board. Die angegebene Geschwindigkeit, z. B. 6/12MHz, bezieht sich nur auf den Prozessor und nicht auch auf die Busse.

Die vorhandene Hardwarekonfiguration muß mit der mitgelieferten Software in das erweiterte CMOS-RAM geschrieben werden. Auf die zu dem Chipsatz passende Sofwareversion ist unbedingt zu achten. In dem Menu sind zuerst die programmierbaren chips aufgeführt. Danach kann ein vordefinierter file eingelesen oder ein eigener erzeugt werden. Das shadow ram und das video ram können nur benutzt werden wenn das Memory nicht nach oberhalb 1M gemappt wird (relocate). Mit dem "power on reset" werden die Änderungen übernommen. Mit der Brücke "Reset 82C206" auf der Rechnerplatine lässt sich diese Information rückgängig machen, d.h. alles ist vergessen, auch die Type des Festplattenlaufwerks.

Die chips können auch selber programmiert werden, wenn man die gewünschte Registeradresse nach Port 22H schreibt und dann Port 23H zum schreiben oder lesen der Daten benutzt.

1.6.1 Peripheral Controller

Der 82C206 Peripheral Controller enthält die Funktionen von zwei 8237 DMA Controllern, zwei 8259 Interrupt Controllern, einen 8254 Timer/Counter, eine MC146818 Real Time Clock und einen 74LS612 Memory Mapper. Über den IBM PC AT Standard hinaus erlaubt der Controller variable DMA wait states, getrennt für 8 und 16 Bit, und 8 oder 4 MHz System Clock. Für diese zusätzlichen Informationen sind 64 Byte RAM auf der real time clock zusätzlich vorhanden.

Die angegebenen Register beziehen sich auf *einen* chip-set. Bei anderen Sets können die Register anders bezeichnet sein. Herstellerangaben beachten!

- Register 01H (Konfiguration)
  - Bit7,6 — Anzahl der Waits bei einem I/O-Zugriff
    - 00 ein wait
    - 01 zwei wait
    - 10 drei wait
    - 11 vier wait (default)
  - Bit5,4 Anzahl der Waits bei 16-Bit-DMA-Zugriff
    - 00 ein wait (default)
    - 01 zwei wait
    - 10 drei wait

    - 11 vier wait
  - Bit3,2 Anzahl der Waits bei 8-Bit-DMA-Zugriff
    - 00 ein wait (default)
    - 01 zwei wait
    - 10 drei wait
    - 11 vier wait
  - Bit1 Normalerweise wird das MEMR des DMA-Controller um einen Clock-Zyklus gegenüber dem IOR verzögert. Falls eine DMA-Peripherie diese beiden Signale gleichzeitig haben möchte, kann man das durch dieses Bit (auf 1) bewirken.
  - Bit0 Standardmäßig wird der Takt SCLK noch durch zwei geteilt und dem DMA-Baustein zugeführt. Setzt man dieses Bit, so wird der Teiler übergangen und der SCLK-Takt direkt benutzt.

1.6.2 Bus-Controller

Der 82C211 Bus-Controller ist für den Buszugriff des 80286 zuständig. I/O-Zugriff, 8-Bit und 16-Bit Memory-Zugriff können mit unterschiedlichen wait states und comand-delay generiert werden. Ein "Port B Register" als Statusregister ist auch vorhanden. Das Bild 1.13 auf Seite 16 zeigt ein Beispiel.

- RA0 Register 60H (Taktrate)
  - Bit7,6 Revisionsnummer des 82C211
  - Bit5 Wenn dieses Bit von low nach high wechselt, wird ein Reset für die CPU erzeugt.
  - Bit4 Der CPU-Takt kommt von CLK2IN (0) oder BCLK (1). (Umschaltung von MUX1).
  - Bit3 reserviert.
  - Bit2 Bei einem Timeout des lokalen Speicherbus (Hauptplatine), kann ein NMI generiert werden, um diesen Fehler anzuzeigen. Das AWARD-BIOS behandelt diesen Fehler jedoch nicht, wodurch Dauer-NMI und Stack-Überlauf vorgezeichnet sind.
  - Bit1 reserviert.
  - Bit0 Flag, die anzeigt, daß ein Timeout beim Zugriff auf den lokalen Speicher aufgetreten ist. Dies kann z. B. passieren, wenn überhaupt kein Speicher an der betreffenden Adresse liegt. Ist Bit2 gesetzt, wird zusätzlich ein NMI erzeugt.
- RA1 Register 61H (Verzögerung der Steuerleitungen)
  - Bit7 Um die Zeit, in der die Speicher-Adresse auf dem Bus liegt, zu vergrößern, kann dieses Bit gesetzt werden. Damit sind eventuell Speicher mit einer langsameren Zugriffszeit verwendbar.
  - Bit6 Aktivieren (0) oder Abschalten (1) des Quick-Modus. Dabei wird nicht nur bei Zugriffen über den AT-Bus das Signal ALE korrekt bedient, sondern auch bei Speicherzugriffen über den lokalen Bus.
  - Bit5,4 Anzahl der Verzögerungen bei einem 16-Bit-Speicherzugriff über den AT-Slot-Bus.
  - Bit3,2 Anzahl der Zyklen zur Verzögerung bei einem 8-Bit-Speicherzugriff über den AT-Slot-Bus. Nach einem Reset wird standardmäßig ein Zyklus eingefügt.
  - Bit1,0 Anzahl der BCLK-Zyklen, die bei einem I/O-Zugriff auf den AT-Slot-Bus gewartet werden soll.
- RA2 Register 62H (Wait-States)
  - Bit7 reserviert (82C211). 387sx-Coprozessor vorhanden (82C811).

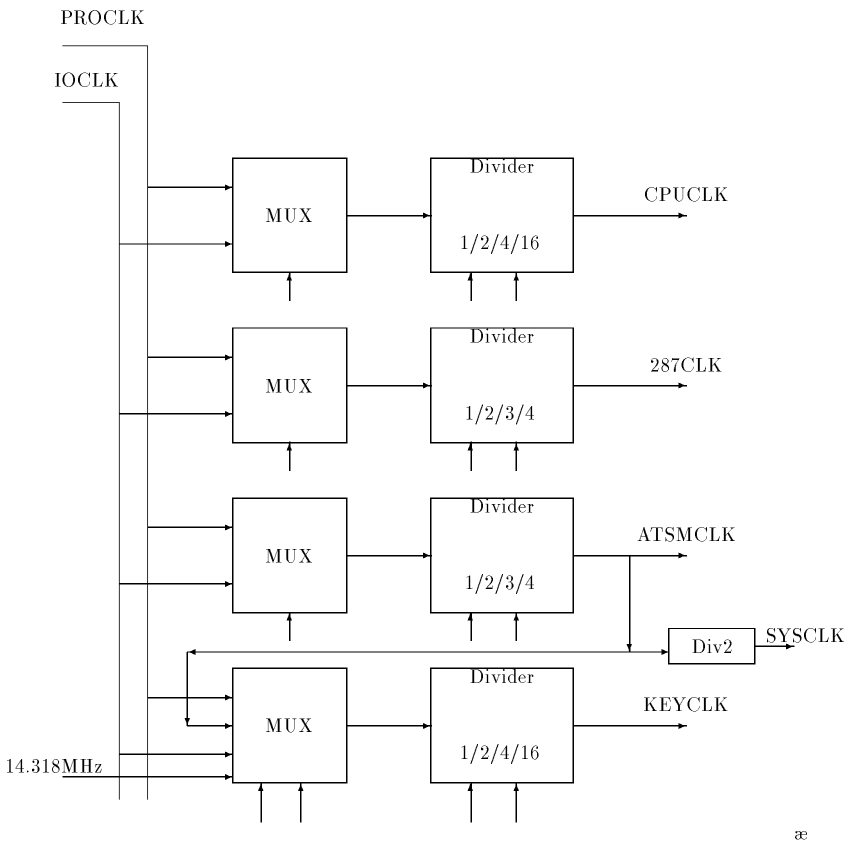


Abbildung 1.13: Blockschaltbild des Taktgenerators

- Bit6 reserviert (82C211). READY-Erzeugung vom Coprozessor (1) oder vom 82C811 (0).
- Bit5,4 Anzahl der Waits bei einem 16-Bit-Zugriff. Hier sollten normalerweise drei waits eingestellt sein.
- Bit3,2 Anzahl der zusätzlichen Waits (mindestens zwei) bei einem 8-Bit-Zugriff über den AT-Slot-Bus. Ohne Änderung werden hier fünf waits erzeugt.
- Bit1,0 Hiermit wählt man den Takt für den AT-Slot-Bus BCLK über den zweiten Multiplexer.
  * 00 BCLK = CLK2IN/2
  * 01 BCLK = CLK2IN
  * 10 BCLK = ATCLK
  * 11 nicht benutzt.

  Beim 82C811 für den SX:
  * 00 BCLK = CLKIN/3
  * 11 BCLK = CLK2IN

1.6.3 Memory-Controller

Der 82C212 Memory Kontroller führt alle Memorykontrollfunktionen aus und kann DRAMs im "page mode" verfügbar machen.

Memory Mapping

Der 82C212 besitzt eine Reihe von mapping Registern für viele Organisationsformen des Memory. Das das ROM (BIOS) überlappende RAM kann in den Bereich oberhalb 1MB gelegt werden. Das 1MB RAM kann dann in zwei Teilen adressiert werden, 0 bis 640K und 1M bis 1,384M (relocated). Wenn nicht relocated wird, kann shadow ram benutzt werden, sowohl für das BIOS-ROM als auch für das Video-ROM.
Paged oder interleaved refresh mode kann nur mit zwei Bänken von gleichen RAMs benutzt werden. Der Rechentest wird dann um ca. 5% verlängert wenn interleaved erlaubt ist. Bei manchen Karten mit eigenem Memory gibt es Probleme wenn interleave erlaubt ist (timing).

- RB0 Register 64H (Version)
  - Bit7 Identifikationsbit
  - Bit6,5 interne Revisions-Nummer
  - Bit4-0 reserviert
- RB1 Register 65H (Konfiguration)
  - Bit7 Schreibschutz für RAM von Adresse C0000H bis CFFFFH
  - Bit6 Schreibschutz für RAM von Adresse D0000H bis DFFFFH
  - Bit5 Schreibschutz für RAM von Adresse E0000H bis EFFFFH
  - Bit4 Schreibschutz für RAM von Adresse F0000H bis FFFFFH
  - Bit3 Aktiviert ROM auf Adresse C0000H bis CFFFFH
  - Bit2 Aktiviert ROM auf Adresse D0000H bis DFFFFH
  - Bit1 Aktiviert ROM auf Adresse E0000H bis EFFFFH
  - Bit0 Aktiviert ROM auf Adresse F0000H bis FFFFFH

  Ist das Aktivierungsbit 0, so wird bei einem Zugriff auf den entsprechenden Bereich das Signal CSROM generiert. Dies Signal steht nicht an den Steckplätzen zur Verfügung.

- RB2 Register 66H 512 KByte bis 640KByte)
  - Bit7 Aktiviert den Speicher von Adresse 80000H bis 9FFFFH als On-Board-Hauptspeicher.
  - Bit6-0 reserviert
  - Bit6 memory status (256K bis 512K) nur 82C812
  - Bit5 memory status (0K bis 256K) nur 82C812
- RB3-RB5 Register 67H bis 69H (RAM-Konfiguration)

Bit	67H	68H	69H
7	AC000 - AFFFF	DC000 - DFFFF	FC000 - FFFFF
6	A8000 - ABFFF	D8000 - DBFFF	F8000 - FBFFF
5	A4000 - A7FFF	D4000 - D7FFF	F4000 - F7FFF
4	A0000 - A3FFF	D0000 - D3FFF	F0000 - F3FFF
3	BC000 - BFFFF	CC000 - CFFFF	EC000 - EFFFF
2	B8000 - BBFFF	C8000 - CBFFF	E8000 - EBFFF
1	B4000 - B7FFF	C4000 - C7FFF	E4000 - E7FFF
0	B0000 - B3FFF	C0000 - C3FFF	E0000 - E3FFF

  Man achte auf die Verdrehung im Register 67H.

- RB6 Register 6AH (Speicher-Aktivierung Bank 0/1)
  - Bit7,6 Speicher-Chip-Typen für erste Bank
    * 00 kein Speicher
    * 01 256k Bit und 64K Bit zusammen 640K Byte.
    * 10 nur 256K Bit
    * 11 1M Bit
  - Bit5 Anzahl der RAM-Bänke
  - Bit4-0 reserviert
- RB7 Register 6BH (Konfiguration)
  - Bit7 Mit diesem Bit wird der page/interleave Modus ein (1) oder aus (0) geschaltet. Nur möglich bei Bestückung mit gerader Anzahl Bänke.
  - Bit6 Nur möglich mit 256K Chips. Gibt an ob der Bereich von 640K bis 1M oberhalb von 1M liegen soll (relocated).
  - Bit5 Anzahl der Waits beim Hauptspeicherzugriff. Beim 82C812 nur beim Schreibzugriff.
  - Bit4 EMS-Aktivierung.
  - Bit3,2 Anzahl der Waits beim Zugriff auf EMS-Speicher.
    * 00 kein wait
    * 01 ein wait
    * 10 zwei wait
    * 11 drei wait
  - Bit3 Nur beim 82C812 Anzahl der waits beim Lesezugriff
  - Bit1,0 Anzahl der Wait-Zyklen beim ROM-Zugriff
    * 00 kein wait
    * 01 ein wait
    * 10 zwei wait
    * 11 drei wait
- RB8 Register 6CH (Speicher-Aktivierung Bank 2/3)

  - Bit7,6 Speicher-Chip-Typen für erste Bank
    * 00 kein Speicher
    * 01 256k Bit und 64K Bit zusammen 640K Byte.
    * 10 nur 256K Bit
    * 11 1M Bit
  - Bit5 Anzahl der RAM-Bänke
  - Bit4-0 reserviert
- RB9 Register 6DH (EMS-Basisadresse)
  - Bit7-4 Diese Bits legen die Basisadresse des 64K Byte Segments fest, in den die einzelnen EMS-Seiten eingeblendet werden sollen. Die übrigen Kombinationen sind nicht benutzbar.
    * 0000 C0000H
    * 0001 C4000H
    * 0010 C8000H
    * 0011 CC000H
    * 0100 D0000H
    * 0101 D4000H
    * 0110 D8000H
    * 0111 DC000H
    * 1000 E0000H
  - Bit3-0 Basisadresse der EMS-Page-Ports. Alle anderen Kombinationen sind reserviert und dürfen nicht benutzt werden.
    * 0000 208H
    * 0001 218H
    * 0101 258H
    * 0110 268H
    * 1010 2A8H
    * 1011 2B8H
    * 1110 2E8H
- RB10 Register 6EH (EMS Adress-Erweiterung)
  - Bit7,6 Diese Bits definieren die Adress-Bits A21 und A22 für die Seite 3.
    * 00 0 bis 2 MByte
    * 01 2 bis 4 MByte
    * 10 4 bis 6 MByte
    * 11 6 bis 8 MByte
  - Bit5,4 dito für Seite 2
  - Bit3,2 dito für Seite 1
  - Bit1,0 dito für Seite 0
- RB12 Register 6FH (zusätzliche Funktionen)
  - Bit7-5 Diese drei Bits legen die Größe des EMS-Speichers fest.
    * 000 weniger als 1MByte
    * 001 1 MByte
    * 010 2 MByte
    * 011 3 MByte

  - ∗ 100 4 MByte
  - ∗ 101 5 MByte
  - ∗ 110 6 MByte
  - ∗ 111 7 MByte
- Bit4 reserviert beim 82C211. Externer EMS-Mapper aktiv beim 82C811.
- Bit3 reserviert beim 82C211. Abschalten des internen EMS-Mappers beim 82C811.
- Bit2 Eine 0 schaltet den internen RAS-Timeout-Counter ein. Man sollte den Timeout eigentlich immer erlauben.
- Bit1 Mit diesem Bit erfolgt die Umschaltung der Adressleitung A20. Mit diesem Bit geht die Umschaltung erheblich schneller als IBM das macht.

Shadow RAM

Bevor die entsprechenden Bit im ROM-enable und RAM-mapping Register gesetzt werden, muß das BIOS in das RAM kopiert werden. Damit wird die Ausführung von BIOS calls erheblich verkürzt. Wenn mehr als 1MB installiert ist, kann auf das RAM zwischen 640KB und 1MB nicht zugegriffen werden, wenn shadow RAM nicht erlaubt ist. Bei Zugriff auf Adressen oberhalb 1MB schaltet das BIOS normalerweise den Pozessor in protected mode. Die NEAT chips haben ein mapping register, damit geht es schneller und es muß danach nicht gebootet werden, um in den real mode zu kommen. Wird Video RAM erlaubt, wird der Bildschirmzugriff bei meiner Karte von 606 μ-Sekunden auf 182 μ-Sekunden verkürzt. Manche Video-Karten haben ein großes BIOS, dann muß mit dem mitgelieferten VSHADOW der Rest installiert werden weil das kleine video-shadow im Menue nur den unteren Teil kopiert.

EMS Adress Übersetzung

Das Expanded Memory System ist ein Schema mit dem ein 64KByte Block aus dem EPROM Bereich D0000H- DFFFFH (oder E0000-EFFFF) in den Bereich zwischen 1MByte und 8MByte gelegt werden kann. Die 64KB sind in Seiten zu 16KB aufgeteilt die in diesen Bereich gelegt werden können. Im EMS mode werden die Adress-bits A14 bis A22 mit dem EMS mapping Register übersetzt, deshalb ist ein umschalten in protected mode nicht nötig. Der 64KB Block kann von irgendwo nach irgendwo gelegt werden. Der Bereich darf natürlich nicht mit den RAM- oder ROM-Bereichen z. B. der Bildschirmkarte überlappen. Das I/O Port muß bei manchen EMS-Treibern bei 208/209h bleiben. Manchmal können die Parameter aber auch in der Zeile im CONFIG.SYS angegeben werden.

1.6.4 Data/Address-Buffer

Der 82C215 vermittelt zwischen local memory, system data und address bus. Er stellt auch die Konversionslogik für 16 bit nach 8 bit Übertragungen zur Verfügung.

1.7 Der 80386 Prozessor unter DOS

Viele der zusätzlichen Funktionen des 80386 oder SX können unter DOS mit den vorhandenen Werkzeugen nicht genutzt werden. Damit sind nicht nur die über den real mode hinausgehenden gemeint. Mit Microsoft's MASM V5.1 können einige Funktionen trotzdem genutzt werden. Die Programme sind dann aber nur auf Rechnern mit 80386 Prozessor lauffähig.

- Neue Instruktionen, wenn sie nicht den protected mode managen, können benutzt werden.
  - Bit scan (BSF und BFR)

  - Bit test (BT, BTC, BTR und BTS)
  - Move with sign and zero extend (MOVSX und MOVZX)
  - Set Byte on condition (SET condition)
  - Double-precision shift (SHLD und SHRD)
- 80286 Instruktionen, die auf 32-Bit Register erweitert wurden.
  - Integer multiply (IMUL)
  - Conversion (CWDE und CDQ)
  - String (CMPSD, LODSD, MOVSD, SCASD, STOSD, INSD, OUTSD)
  - 32-Bit Stack (PUSHAD, POPAD, PUSHFD, POPFD, IRETD)
- Doppelwort Integer können mit 32-Bit Registern bearbeitet werden. Einige Multiplikationen und Divisionen können mit 64-Bit Integers gemacht werden.
- 32-Bit Register können als Zeiger in 16-Bit Segmente benutzt werden. Auf eine gültige 16-Bit Adresse muß geachtet werden.
- Device driver und TSR-Programme, die 32-Bit Register benutzen, müssen diese mit PUSHAD retten und mit POPAD zurückschreiben, wenn sie unterbrochen werden.

1.8 POST Power On Self Test

Nachdem die Versorgungsspannung sich stabilisiert hat, kann das Resetsignal zurückgenommen werden. Damit wird das Segmentregister auf den Wert F0000H und das Offsetregister auf FFF0H gesetzt. An dieser Stelle im BIOS-ROM steht die Einsprungadresse für den POST.

1.9 Fehler und Bemerkungen

- In dem MS-DOS 3.1 programmer's reference manual von Markt & Technik sind beim Interrupt 21 Funktion 44 die Subfunktionen 2 und 3 falsch. Die Subfunktion 3 schreibt (Ver. 3.30). Stanford benutzt aber auch die Unterfunktion 2 zum Initialisieren der NI-IEEE488 Karte.
- Durch das Einhaken in einen bestehenden Interrupt können natürlich nur unterschiedliche Funktionen erkannt und benutzt werden. Der zuletzt eingehakte fischt sich die Funktion und bearbeitet sie. Das Weiterreichen an den Vorgänger und das Austragen scheint nicht bei allen Programmen zu funktionieren (s. TSR).
- Die COM1,COM2-Karte ME-91 von Meilhaus hat zum Umschalten die Adresse 300 belegt die aber in den vom NI5010 DECnet-Kontroller benutzten Bereich fällt.
- Die Bezeichnung der Schalter der Karte "SUN KWONG" ist falsch. Richtig:
  - SW1 — CH1 — J1
  - SW2 — CH2 — J2
  - SW3 — CH3 — J1
  - SW4 — CH4 — J2
  - SW5 — 1ST — IRQ4
  - SW6 — 1ST — IRQ3
  - SW7 — 2ND — IRQ4

- SW8 — 2ND — IRQ3

- MASM

  Den Zahlen oberhalb von 9 muß eine 0 vorangestellt werden da der ASM sonst von einem Register oder einem Symbol ausgeht und das ist nicht definiert. Das H für hex wird erst interpretiert, wenn das Vorhergehende als Zahl begriffen wurde.

- DBASE3+

  Dbase3+ funktioniert nicht in einem 6/12 MHz Rechner mit einem original 5 MHz Coprocessor. Fortran mit floating-point geht aber.

- OS/2

  Die 8250 UART müssen gegen die 16450 getauscht werden.

  Bei den Videokarten muß der auto-switch disabled sein beim booten. Multifunktionskarten sollten im EGA-Mode sein. Nur die Paradise AutoSwitch Mono EGA Card Model 02-17 ist OK.

- Linker

  Der Linker ermittelt zwar eine checksum und legt sie in den Bytes 12H und 13H im header des .exe-files ab, sie wird aber nicht benutzt.

- LPT-Fehler

  Manche, vorallem ältere Monochrome/Printer Adapter erzeugen kein IRQ7, deshalb funktionieren Gerätetreiber, die mit IRQ7 arbeiten, nicht. Entweder Karte oder Treiber tauschen!

- Inkompatibel

  1. POPF
  2. PUSH SP
  3. Shift Zähler
  4. Division durch 0 Exeption
  5. POP SS

æ

Kapitel 2

System Initialisierung

Die POST's sind nicht nur stark versionsabhängig sondern vor allem von dem BIOS-Hersteller.

2.1 Initialisation des 80286

Wenn das RESET-Signal nach low gegangen ist, befindet sich der 80286 in einem genau definierten Zustand. Die gültigen Felder der Segment und Description Register werden eingeschaltet. Alle privilege-level Felder werden nach zero gesetzt. Der real address mode wird eingeschaltet. Das CS base register wird mit FF0000H und der IP mit FFF0H geladen. Damit liegt die Adresse, bei der der Prozessor zu arbeiten anfängt, fest bei FFFFF0H. Dort befindet sich normalerweise eine JMP-Instruktion zum Anfang des Programms. In der Tabelle 2.1 auf der Seite 23 werden alle Register mit ihren Inhalten gezeigt, auch wenn sie für den real-address mode nicht zuständig sind. Da im real address mode nur 20 Bit, im protected mode aber 24 Bit generiert werden, muß das Boot-Programm für ein Protected Mode System sich die oberen 4 Bit freischalten und im Protected Mode erneut Booten.

Zwei feste Memorybereiche sind reserviert:

CPU Component	Content
Flags	0002H
MSW	FFF0H
IP	FFF0H
CS Selector	F000H
DS Selector	0000H
SS Selector	0000H
ES Selector	0000H
CS Base	FFF0H
DS Base	0000H
SS Base	0000H
ES Base	0000H
CS Limit	FFFFH
DS Limit	FFFFH
SS Limit	FFFFH
ES Limit	FFFFH
IDT Base	0000H
IDT Limit	03FFH

Tabelle 2.1: Registerinhalte des 80286 nach RESET

1. von FFFFF0h bis FFFFFFh der System-Initialisierungs-Bereich und
2. von 000000h bis 0003FFh der Interrupt-Tabellen-Bereich.

Ab diesem Punkt sind die Segmentregister gültig und die Bits der Zugriffsrechte 0. Der Prozessor fängt im real-address-mode an zu arbeiten mit PE=0. Die maskierbaren Interrupts sind nicht erlaubt und eine Prozessor Erweiterung (Coprozessor) wird nicht benutzt oder emuliert (EM=MP=0).

Die Kombination CS:IP, also die Startadresse, ergibt FFFF0h. Im real address mode werden die ersten vier Bit nicht benutzt, womit die Kompatibilität zum 8086 hergestellt ist. An der Startadresse befindet sich normalerweise eine JMP Anweisung zu der Systeminitialisierungsroutine. Dort kann dann auch in den protected mode umgeschaltet werden für ein richtiges Betriebssystem. Wird aber in ein anderes Segment im real mode umgeschaltet, werden die oberen vier Bit des CS Registers gelöscht. Damit werden nicht mehr die oberen 64KB des 16MB Adressraums sondern nur noch Segmente unterhalb 1MB angesprochen.

2.1.1 Real Address Mode

1. Allocate a stack.
2. Load programs and data into memory from secondary storage.
3. Initialize external devices and the Interrupt Vector Table.
4. Set registers and MSW bits to desired values.
5. Set FLAG bits to desired values — including the IF bit to enable interrupts — after insuring that a valid interrupt handler exists for each possible interrupt.
6. Execute (usually via an inter-segment JMP to the main system program).

2.1.2 Protected Mode

1. Load programs and associated descriptor tables.
2. Load valid GDT and IDT descriptor tables, setting the GDTR and IDTR to their correct value.
3. Set the PE bit to enter protected mode.
4. Execute an intra-segment JMP to clear the processor ques.
5. Load or construct a valid task state segment for the initial task to be executed in protected mode.
6. Load the LDTR selector from the task's GDT or 0000h if an LDT is not needed.
7. Set the stack pointer (SS,SP) to a valid location in a valid stack segment.
8. Mark all items not in the memory as not-present.
9. Set FLAGS and MSW bits to correct values for the desired system configuration.
10. Initialize external devices.
11. Ensure that a valid handler exists for each possible interrupt.
12. Enable interrupts.
13. Execute.

2.2 Power On Self Test

Der POST besteht eigentlich aus zwei Teilen:

1. Funktionen zum Testen der zentralen Hardware und
2. Funktionen zum Konfigurieren des Systems und der Hardware, die sich nicht auf der Mutterplatine befindet.

Die zentrale Hardware wird zuerst getestet und dann initialisiert in der folgenden Reihenfolge:

- Die CPU
- die Checksum des ROM BIOS
- das CMOS RAM
- den DMA Controller
- den Keyboard Controller
- die ersten 64K des System-RAM
- die Interrupt Controller
- die Interrupt Timer
- den Cache Controller

Fehler in diesen Tests werden durch einen "beep code" angezeigt und in das Port 80H geschrieben. Bei EISA wird das Port 300h, bei PS/2 680h benutzt. Compaq hat 84h und andere nehmen EBh.

Danach werden die Daten im CMOS-RAM mit der aktuellen Konfiguration verglichen und die Geräte initialisiert.

- CMOS RAM Konfigurationsdaten
- die CRT Kontroller
- RAM Memory oberhalb 64K
- das Keyboard
- das Vorhandensein des Laufwerks A
- das serielle Interface
- den Diskettenkontroller
- den Festplattenkontroller
- jede zusätzliche Hardware

Fehlermeldungen werden auf dem Bildschirm angezeigt.

Das BIOS-ROM hat einige herstellerabhängige festgelegte Eintrittspunkte: s. Tabelle 2.2 auf Seite 26.

Adresse	Description
FE05BH	POST entry point
FE2C3H	NMI handler entry point
FE3FEH	INT 13H fixed disk service entry point
FE401H	Fixed disk parameter table
FE6F2H	INT 19H boot load service entry point
FE6F5H	Configuration data table
FE729H	Baud rate generator table
FE739H	INT 14H serial communication service entry point
FE82EH	INT 16H keyboard service entry point
FE987H	INT 09H keyboard service entry point
FEC59H	INT 13H diskette service entry point
FEF57H	INT 0EH diskette hardware ISR entry point
FEFC7H	Diskette controller parameter table
FEFD2H	INT 17H printer service entry point
FF045H	INT 10H functions 0-FH entry point
FF065H	INT 10H video support service entry point
FF0A4H	INT 1DH MDA/CGA video parameter table
FF841H	INT 12H memory size service entry point
FF84DH	INT 11H equipment list service entry point
FF859H	INT 15H system service entry point
FFA6EH	Characterfont for 320x200 and 640x200 graphics (lower 128 characters)
FFE6EH	INT 1AH time-of-day service entry point
FFEA5H	INT 08H system timer ISR entry point
FFEF3H	Initial interrupt vector offset loaded by POST
FFF53H	IRET instruction for dummy interrupt handler
FFF54H	INT 05H print screen service entry point
FFFF0H	Power-up entry point
FFFF5H	ASCII date ROM was built – 8 characters
FFFFEH	System model ID

Tabelle 2.2: BIOS-ROM Entries

Adress Bereich	Beschreibung
C0000H – C7FFFH	Video expansion ROM (1)
C8000H – DFFFFH	General expansion ROM (2)
E0000H	System expansion ROM (3)

Tabelle 2.3: Vogegebene Adressbereiche für expansion ROM

CMOS RAM	Size	Beschreibung
0FH	1 Byte	shutdown code
		00H = Power-on or soft reset
		01H = Memory size pass
		02H = Memory test pass
		03H = Memory test fail
		04H = POST end; boot system
		05H = JMP dword pointer with EOI
		06H = Protected tests pass
		07H = Protected tests fail
		08H = Memory size fail
		09H = INT 15H block move
		0AH = JMP dword pointer without EOI

Tabelle 2.4: Shutdown Code

2.3 I/O expansion ROM

Ein I/O expansion ROM ist eine zusätzliche Erweiterung des BIOS und wird als Teil eines I/O-Subsystems in die Maschinenumgebung integriert. I/O ROMs ersetzen oft einen Teil des BIOS für bestimmte Funktionen wie z. B. ESDI fixed disk BIOS oder VGA video adapter.

POST sucht nach expansion ROM in drei vorgegebenen Adressbereichen: s. Tabelle 2.3 auf Seite 26.

1. Der POST durchsucht den Adressbereich C0000h bis C7FFFh in Schritten zu 2K nach einem Video expansion ROM.
2. Der POST durchsucht den Adressbereich C8000h bis DFFFFh in Schritten zu 2K nach einem General expansion ROM.
3. Der POST durchsucht die Adresse E0000h nach einem expansion ROM nur an dieser Adresse. Ein vorhandenes ROM muß die ganzen 64K aufweisen.

Die Suche nach einem expansion ROM ist nur erfolgreich, wenn die ersten beiden Worte (4 Byte) aussehen wie folgt:

I/O ROM Byte	Wert
0	55H
1	AAH
2	ROM-Länge in 512-Byte Blöcken
3	Entry für ROM-Initialisierung (über FAR CALL)

Wird ein ROM erkannt, kann es eine Initialisierungsroutine durchlaufen um seine spezielle Hardware und Software zu initialisieren. Diese fängt bei Byte 3 an und wird vom POST mit einem FAR CALL angesprungen.

2.4 Re-Entering Real Mode

Einige POST Teile erfordern den protected mode. Danach kann der Prozessor nur noch durch RESET in den real mode kommen. Dann wird aber der POST neu aufgesetzt. Deshalb liest er den Platz 0FH des CMOS RAM und springt dem code entsprechend zu dem richtigen Entry.

Der POST benutzt nur die codes 01H, 02H, 03H, 04H, 06H, 07H und 08H. Die codes 05H und 0AH sind erheblich flexibler. Die Kontrolle geht zurück an die Routine, deren Adresse im Doppelwort 40:67H abgespeichert wurde. Der Shutdown code 05H entleert den Keyboardbuffer und erzeugt ein EOI-Signal [1] vor dem Sprung zu der angegebenen Routine. Bei einem schutdown code von 0AH wird nur zu der Routine gesprungen. Die codes 05H und 0AH sind für Betriebssystemerweiterungen, z. B. durch I/O-ROM's, vorgesehen.

Wenn der POST die Initialisierungen beendet hat, benutzt er den INT 19h zum booten. Dabei gilt:

1. Wenn ein Disketten Laufwerk vorhanden **und** eine bootfähige Diskette eingelegt ist überträgt der INT 19h den Inhalt des Bootsektors in das Memory beginnend mit der Adresse 0000:7C00h.

2. Wenn eine Festplatte vorhanden ist und sich **keine** bootfähige Diskette im Laufwerk befindet überträgt der INT 19h den Inhalt des Bootsektors in das Memory beginnend mit der Adresse 0000:7C00h.

3. Wenn **weder** auf der Diskette **noch** auf der Festplatte ein Bootsektor zu finden ist ruft der INT 19h den INT 18h. Dieser schreibt auf den Bildschirm: **NO BOOT DEVICE AVAILABLE**.

   Der Vektor des INT 18h kann von einem Netzwerk ROM beim Initialisieren überschrieben werden. Dann kann ein 'diskless' Rechner über das Netz booten.

2.5 NMI nonmaskable interrupt

Wenn der 80286 einen NMI bekommt, wird kein INTA (interrupt acknowledge) erzeugt. Mit Bit 7 im Adressregister Port (70H) des CMOS RAM kann der NMI unterdrückt werden. Der NMI benutzt den Vektor des Interrupt Nr. 2H. Der Vektor wird mit F000:E2C3 initialisiert.

Die NMI ISR zeigt nur eine run-time error message und hält den Prozessor an.

Ein NMI kann nur durch booten gelöscht werden.

2.5.1 NMI parity error

Es können zwei Typen von Paritätsfehlern auftreten:

1. Fehler in Memory auf der Mutterplatine

2. Fehler in Memory auf einer Karte

Manche BIOS-Hersteller (Phoenix) versuchen die fehlerhafte Stelle im Memory zu ermitteln. Dann wird sie auf dem Bildschirm gezeigt. Bei solchen kann man durch eintippen von **S** fortfahren.

2.5.2 DMA NMI

Wenn ein DMA Interface oder Device den Bus länger als 7.8 μs benutzt, wird eine entsprechende Zeile auf dem Bildschirm erscheinen und ein NMI erzeugt.

2.5.3 Coprozessor

Wenn der Coprozessor eine Ausnahmesituation erkennt setzt er den IRQ13, der den INT 75h benutzt. Ist eine Behandlungsroutine vorhanden die den NMI nicht behandeln kann, wird der INT 71h aufgerufen der den INT 02h benutzt.

Der INT 75h Vektor wird vom POST mit dem selben Eintrittspunk wie der INT 02h initialisiert (F000:E2C3h).

[1] End of interrupt

2.6 Error und Beep Codes

Die Error-Codes werden in das Port 80h geschrieben und bei manchen Herstellern auch durch einen "Beep-Code" zu Gehör gebracht.

Port 80	AMI	AWARD	PHOENIX	Beep-Code
01	NMI disabled register test	Prozessor Test status verify	Test CPU Register	none
02	286 Register Test over	Bestimmt den Typ des Test fails if keyboard	CMOS write/read Fehler	1-1-3
03	ROM checksum	clear 8042 interface	ROM checksum	1-1-4
04	8259 init	reset 8042	programmable intervall timer fail	1-2-1
05	CMOS pending interrupt disabled	get 8042 manufacturing status	DMA init fails	1-2-2
06	Video disabled and timer counting	initialize chips DMA,8259,CMOS	DMA page register read/write fails	1-2-3
07	8253 ch 2 test	test 2	n/a	
08	ch 2 delta count	ini CMOS	RAM refresh verify	1-3-1
09	ch 1 delta count	EPROM checksum	1st 64k RAM	none
0A	ch 0 delta count	ini video interface	1st 64k failure chip or data line	1-3-3
0B	parity status cleared	ch 0 8254	1st 64k logic	1-3-4
0C	refresh, timer	ch 1 8254	1st 64k address	1-4-1
0D	refresh link toggling	ch 2 8254	1st 64k parity	1-4-2
0E	refresh period	CMOS shutdown		
0F		extended CMOS		
10	confirmed refresh on 64 k memory test	DMA ch 0	bit 0 1st 64k failure	2-1-1
11	address line	DMA ch 1	bit 1 1st 64k	2-1-1
12	64k base memory	DMA page register	bit 2 1st 64K	2-1-3
13	interrupt vectors	keyboard controller	bit 3 1st 64k	2-1-4
14	keyboard controller	mem refresh toggle	bit 4 1st 64k	2-2-1
15	CMOS read/write	first 64k	bit 5 1st 64k	2-2-2
16	CMOS checksum	interrupt vectors	bit 6 1st 64k	2-2-3
17	monochrome set	set up video I/O	bit 7 1st 64k	2-2-4
18	colormode set	video memory	bit 8 1st 64k	2-3-1
19	video ROM detect	8259 mask bits ch1	bit 9 1st 64k	2-3-2
1A	video ROM control	8259 mask bits ch2	bit A 1st 64k	2-3-3
1B	display memory read/write	CMOS battery level	bit B 1st 64k	2-3-4
1C	alternate display memory	CMOS checksum	bit C 1st 64k	2-4-1

Bei manchen neueren Boards sind 2 LED Anzeigen vorhanden. Was nutzen die aber ohne Tabelle?

Port 80	AMI	AWARD	PHOENIX	Beep
1D	video retrace check	set config from CMOS	bit D 1st 64k	2-4-2
1E	global equipment byte set for video	size system memory compare with CMOS	bit E 1st 64k	2-4-3
1F	mode set call for mono/color	test base memory 64k to top of mem	bit F 1st 64k	2-4-4
20	video test	8259 interrupt bits	slave DMA gerister	3-1-1
21	video passed	NMI bits	master DMA register	3-1-2
22	power on message	8259 interrupt	master interrupt mask register	3-1-3
23		protected, virtual, page mode	slave interrupt mask register	3-1-4
24		size extended memory		
25		all mem exept 1st 64k	interrupt vector loading	
26		protected mode exeptions		
27		set up cache control or shadow RAM	keyboard controller	3-2-4
28		set up cache controller or special keboard	CMOS power failure and checksum calc in progress	
29		reserved	CMOS config validation	
2A		initialize keyboard		
2B		initialize Floppy drive/controller	ini Screen	3-3-4
2C		detect and initialize serial ports	sreen retrace	3-4-1
2D		detect and initialize parallel ports	detect video ROM	3-4-2
2E		initialize hard drive/controller	screen running	
2F		detect and initialize coprocessor		
30		reseved	screen operable	
31	virtual mode memory	detect and ini ROM's	mono monitor operable	
32	Processor in virtual mode		color monitor (40) operable	
33			color monitor (80) operable	
34	memory address line		timer tick interrupt [2]	4-2-1
35	memory below 1MB calculation		shutdown test	4-2-2
36	memory size computation		gate A20 failure	4-2-3
37	memory test		unexpected interrupt	4-2-4
38	memory ini below 1MB		RAM test bis 1MB	4-3-1
39	memory ini over 1MB			
3A	display memory size		interval timer ch 2	4-3-3
3B	test memory below 1MB		time-of-day clock test	4-3-4
3C	passed		serial port test	4-4-1
3D	above 1MB passed		parallel port test	4-4-2

[2]

Port 80	AMI	AWARD	PHOENIX	Beep
3E	shutdown to real mode		coprocessor test	4-4-3
3F	shutdown successful			
40	disable gate A20			
41	gate A20 disabled		systemboard select failure	low-1-1-2
42	DMA controller test		extended CMOS failure	low-1-1-3

Die Unterschiede in der Fülle der Fehler- und Status-Anzeigen und die Änderung der Reihenfolge sind doch beachtlich. Wer früher sein Memory und Display einsatzbereit hat, kann früher sinnvollere Aussagen auf dem Bildschirm darstellen. Die hier angegebenen Daten sind, wie alle anderen auch, immer von der Version abhängig und dürfen nicht als absolutes "Muß" angesehen werden!

[2] nur in manufacturing loop mit jumper, sonst auf screen.

Port 80	AWARD
EE	unexpected processor exeption
CC	NMI handler shutdown
FF	boot attempt

Tabelle 2.5: AWARD Port80 Erweiterungen

Port 80	AMI Fortsetzung
4E	address line test passed
4F	processor in real mode after shutdown
50	DMA page register test
51	start DMA unit-1 base register test
52	DMA unit-1 channel passed
53	DMA channel 2 base register test passed
54	test F/F latch for unit 1
55	F/F latch test for both units passed
56	DMA unit 1 and 2 programs passed
57	8259 initialization over
58	8259 mask register check
59	master 8259 mask register passed
5A	check timer and keyboard interrupt level
5B	timer interrupt passed
5C	test keyboard interrupt
5D	Error! timer/keyboard interrupt not in proper level
5E	8259 interupt controller error
5F	8259 interrupt controller test passed
70	keyboard test
71	keyboard BAT test passed
72	keyboard test passed
73	keyboard global data setup passed
74	start floppy setup
75	floppy setup passed
76	start hard disk setup
77	hard disk setup passed
79	initialize timer data area
7A	verify CMOS battery power
7B	CMOS battery verification complete
7D	analyze diagnostics memory test results
7E	CMOS memory size update passed
7F	check option ROM at C000
80	keyboard sensed to enable setup
81	optional ROM control passed
82	printer global data initialization
83	RS-232 global data initialization
84	80287 check/test
85	display any soft error messages
86	give control to system ROM at E000
87	system ROM E000:0 check complete
00	Boot at INT19

2.7 Initialisierung des 80386

Kleine Unterschiede müssen beachtet werden.

Register	State(hex)
EFLAGS [1]	XXXX0002
EIP	0000FFF0
CS [2]	0F000
DS [3]	0000
SS	0000
ES [4]	0000
PS	0000
GS	0000
IDTR(base)00000000	
IDTR(limit)03FF	
DR7	0000

Tabelle 2.6: Registerinhalte nach RESET (386)

[1] [2] [3]. [4] æ

[1] Die oberen 14 Bit sind nicht definiert. Die Flags sind gelöscht.
[2] Der unsichtbare Teil des CS Registers ist 0FFFFF0000 mit einem Limit von 0FFFF.
[3] Der unsichtbare Teil der Register DS un ES enthalten 0 mit Limit 0FFFF
[4] siehe DS

Kapitel 3

Registers

Es sind fünfzehn 16-Bit Register vorhanden, die sich in folgende Gruppen aufteilen lassen:

General-Register

$\boxed{\text{Register AX}}$ oder $\boxed{\text{Reg. AH}}$ und/oder $\boxed{\text{Reg. AL}}$

$\boxed{\text{Register DX}}$ oder $\boxed{\text{Reg. DH}}$ und/oder $\boxed{\text{Reg. DL}}$

$\boxed{\text{Register CX}}$ oder $\boxed{\text{Reg. CH}}$ und/oder $\boxed{\text{Reg. CL}}$

Segment-Register

$\boxed{\text{Register CS}}$

$\boxed{\text{Register DS}}$

$\boxed{\text{Register SS}}$

$\boxed{\text{Register ES}}$

Base-Register

$\boxed{\text{Register BX}}$ oder $\boxed{\text{Reg. BH}}$ und/oder $\boxed{\text{Reg. BL}}$

$\boxed{\text{Register BP}}$

Index-Register

$\boxed{\text{Register SI}}$

$\boxed{\text{Register DI}}$

Status-Register

$\boxed{\text{Register F}}$

$\boxed{\text{Register MSW}}$

Control-Register

$\boxed{\text{Register IP}}$

$\boxed{\text{Register SP}}$

Benutzung der Register

AX	word multiply, word divide, word I/O
AL	byte multiply, byte divide, byte I/O, translate, decimal arithmetic
AH	byte multiply, byte divide
BX	translate
CX	string operations, loops
CL	variable shift and rotate
DX	word multiply, word divide, indirect I/O
SP	stack operations
SI	string operations, source
DI	string operations, destination

Tabelle 3.1: Benutzung der Register

Mit X sind die 16-Bit Register gemeint. Sie können aber auch als 8-Bit Register mit L oder H für low oder high anstelle des X angesprochen werden. Bei ungerader Adresse und X Kennzeichnung wird kein ODD-ADDRESS-TRAP (DEC) erzeugt, sondern der Zyklus nur verlängert, weil die Wortgrenze überschritten wird.

Bei bestimmten Instruktionen wird den Registern eine bestimmte Bedeutung zugeordnet, s. Tabelle 3.1.

3.1 General Registers

Als allgemeine Register sind AX, BX, CX, DX, SP, BP, SI und DI vorhanden. Mache davon werden aber meistens speziell verwendet. Deshalb werden sie hier auch gesondert betrachtet. Es bleiben als allgemein verwendbare Register nur A, D und C vorhanden. Sie müssen mit X für 16-Bit (z.B. AX), oder mit L (low z.B. DL) oder H (high z.B. CH) für Byte angegeben werden.

Das AX (Accumulator) Register wird oft zum speichern von temporären Daten benutzt, da der Prozessor mit Daten im Akkumulator schneller arbeitet als mit Daten im Memory. Bei Divisions Instruktionen ist vor der Operation der Dividend und danach der Quotient in AX. Vor einer Multiplikations ist einer der Faktoren in AX und danach das Resultat. Bei I/O Instruktionen werden die Daten über AX transferiert. Der 80386 hat auch ein 32-Bit EAX Register. Das A-Register kann also in vier Formen angesprochen werden:

1. mit EAX für 32-Bit oder
2. mit AX für 16-Bit (nur der untere Teil) oder
3. mit AH für die 8 oberen Bits des unteren 16-Bit Teils oder
4. mit AL für die unteren 8-Bit.

Das DX (Data) Register wird oft zum Speichern von temporären Daten benutzt. Bei Doppelwort Divisionen ist in DX das obere Wort vor der Division und der Rest danach. Bei Wortmultiplikationen ist das obere Wort des Doppelwortresultats in DX. Bei I/O Operationen steht in DX die Portnummer. Der 80386 hat auch ein 32-Bit EDX Register.

Das CX (Count) Register wird für loop oder repeat Instruktionen mit dem Zähler geladen. Die Richtung für autoincrement oder autodecrement hängt von DF (direction-flag-bit) im Flag-Register ab. Es kann auch für temporäre Daten benutzt werden. Der 80386 hat auch ein 32-Bit ECX Register.

3.2 Segment Registers

Vier 16-Bit Register mit vorgegebener Bedeutung stehen als Segment-Register zur Verfügung. Sie zeigen modulo 16 auf die Basisadresse eines Segments von 64KB innerhalb der 1MB physikalischen Adresse. Ein Pointer zeigt dann auf das gewünschte Byte.

CS = Code Segment wird automatisch zum Instruction prefetch benutzt. Bildet zusammen mit dem IP den "Program-Counter".

DS = Data Segment alle Data Reference, ausser wenn relative zum Stack oder bei string destination.

SS = Stack Segment für alle Stack push und pop, jede memory reference, die BP als Basis-Register benutzt. Das Stack Segment Register zeigt modulo 16 auf die Basisadresse des Stack.

ES = Extra Segment alternate Data Segment und destination of string operation. Der 80386 hat zwei zusätzliche Extra Segmentregister FS und GS.

3.3 Base Registers

Als Base-Register sind BX und BP (Base-Pointer) vorhanden. Sie werden bei entsprechenden Adressierungsarten automatisch benutzt.

Das BX (Base) Register kann als Pointer benutzt werden. Es kann auch zum Speichern von temporären Daten benutzt werden. Der 80386 hat auch ein 32-Bit EBX Register.

Das BP (Base Pointer) Register kann zum speichern von Daten oder als Pointer benutzt werden. Meistens wird es als Zeiger zu der Basis eines Stapels genommen. Dann wird das SS Register als Segment Register angenommen. Der 80386 hat auch ein 32-Bit EBP Register.

3.4 Index Registers

Als Index-Register sind SI und DI vorhanden. Sie werden als Source und Destination Index, auch in Verbindung mit ES, bei Stringmanipulationen verwendet. Der Counter ist dann in CX. Die Richtung des Counters hängt von DF im F-Register ab. Der 80386 hat auch ein 32-Bit ESI Register und ein 32-Bit EDI Register.

3.5 Status Registers

Als Statusregister sind das Flagregister und das Machine Status Word vorhanden.

3.5.1 Flag-Register

Die Flags CF, ZF und SF hängen von der Operation ab, d. h. beziehen sich auf 8 oder 16 bit.
F ist das Flag-Register mit folgenden Bits:

XX	NT	IO	PL	OF	DF	IF	TF	SF	ZF	XX	AF	XX	PF	XX	CF

Bit00=CF carry flag – set on high bit carry or borrow

Bit02=PF parity flag – set if low byte of result contain even one's

Bit04=AF set on carry from or borrow to low four bits when performing BCD arithmetic.

Bit06=ZF zero flag – set if result is zero

Bit07=SF set equal to high-order bit of result(1 if negative).

Bit08=TF single step flag – once set, a single step interrupt occurs after the next instruction executes. TF is cleared by the single step interrupt.

Bit09=IF interrupt enable flag – when set, maskable interrupts will cause the CPU to transfer control to an interrupt vector specified location.

Bit10=DF direction flag – causes string instructions to auto decrement the appropriate Index registers when set. Clearing DF causes auto increment.

Bit11=OF overflow flag – set if result is too-large or too-small number to fit into destination operand.

Bit12+13 IOPL I/O privileg level

- 00 Level 0 Kernal
- 01 Level 1 System Services
- 10 Level 2 Custom Extensions
- 11 Level 3 Applications

Bit14=NT nested task flag

Der 80386 hat auch ein 32-Bit Eflags Register.

3.5.2 Machine Status Word

Im Machine Status Word (MSW) -Register steht z.B. PE = protection enable. Wenn es mal gesetzt wurde, kann es nur durch "Reset" geändert werden. Daraus folgt aber ⇒ CS:IP wird F000:FFF0 = hardware boot.

XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	TS	EM	MP	PE

- Bit00=PE= protection enable
- Bit01=MP= monitor processor extension
- Bit02=EM= processor extension emulated
- Bit03=TS= task switch

3.6 Control Registers

Als Controlregister sind Instruktion- und Stack-Pointer Register vorhanden.

3.6.1 Instruction Pointer

Im Instruction Pointer (IP) steht der 16-Bit Offset der Adresse der nächsten ausführbaren Anweisung. Er ist nur in Verbindung mit CS zu sehen. Der IP wird beim Programmablauf automatisch erhöht und erzeugt bei Überschreiten von FFFF=64KB eine Trap. Deshalb sind nur Teile zu 64kB nutztbar. Sollen größere Programme benutzt werden, so muß das CS-Register vor erreichen der 64kB-Grenze des IP neu gesetzt werden. Dann wird mit setzen des IP auf den Anfang des neuen Teils das Programm dort fortgesetzt.

- Segment code1
- Segment data
- Segment code2
- Segment code3
- Segment stack

Jedes Segment kann also durch den IP 64kB groß sein. Sie werden im Memmory hintereinander gelegt. Da die Erhöhung um eins im CS ein weiterschalten der Adresse um 16 bedeutet, kann die Lücke zwischen zwei folgenden Segmenten nur 15 Byte oder kleiner sein. Kompiler und Linker bilden sich dabei aber meistens etwas ein, so daß die Lücke erheblich größer sein wird.

Der 80386 hat auch ein 32-Bit EIP Register. Welcher Kompiler, Linker, Betriebssystem nutzt das und ist zu bezahlbaren Preisen zu bekommen?

3.6.2 Stack Pointer

Im SP steht der Pointer, der auf den Platz im augenblicklich benutzten Stack Segment zeigt. Er ist ein Wortpointer und wird deshalb automatisch bei Benutzung der Stackbefehle um 2 erhöht oder erniedrigt. Stack Segmente können genauso umgeschaltet werden wie Code oder Data Segmente. Der 80386 hat auch ein 32-Bit ESP Register.

3.7 80386 only

3.8 80486 only

3.8.1 Kontrollregister

Statusregister ist bitkompatibel mit den Vorgängern. Zusätzlich ist Bit 18 mit *alignment check* belegt. Ist das Bit gesetzt, wird ein Interrupt (11H) erzeugt, wenn ein Wortzugriff mit ungerader Adresse oder ein 32-Bit-Befehl mit einer Adresse, die nicht durch 4 teilbar ist, erfolgt.

Bit0 carry flag

Bit1 1

Bit2 parity flag

Bit3 R

Bit4 auxiliary carry

Bit5 R

Bit6 zero flag

Bit7 sign flag

Bit8 trap flag

Bit9 interrupt enable

Bit10 direction flag

Bit11 overflow

Bit12-13 I/O privileg level

Bit14 nested task

Bit15 R

Bit16 resume flag

Bit17 virtual mode

Bit18 alignment check (in Verbindung mit CR0 Bit18)

Bit19-31 reserviert

Kontrollregister0 **Bit0** protection enable

Bit1 monitor processor

Bit2 emulate coprocessor

Bit3 task switched

Bit4 reserviert

Bit5 numeric exception

Bit6-15 reserviert

Bit16 write protect

Bit17 R

Bit18 alignment mask (in Verbindung mit Statusregister Bit18)

Bit19-28 reserviert

Bit29 writers transparent (WT)

Bit30 cache enable (CE)

Bit31 paging enable

CE=0,WT=0 schaltet das Füllen des Cache und die Funktion zum Kennzeichnen des Cache-Inhalts als ungültig aus. Speicherlesezugriffe, die in diesem Modus auftreten und auf Speicherzellen gerichtet sind, deren Inhalt bereits im Cache vorliegt, werden dennoch aus dem Cache statt aus dem Arbeitsspeicher bedient. Um sicherzugehen, daß der Cache völlig abgeschaltet ist, muß er, nachdem er ausgeschaltet wurde, also auch noch geleert werden.

CE=0,WT=1 verhindert ein Füllen des Cache, erlaubt aber der Cache-Logik, beim Schreibzugriff auf eine im Cache vorliegende Speicherzelle oder bei Zugriffen anderer Geräte (DMA-Controller) den Cache-Inhalt zu aktualisieren. Dieser Modus erweist sich besonders dann als nützlich, wenn der Cache zeitweilig ausgeschaltet ist, aber die Daten im Cache korrekt bleiben müssen.

CE=1,WT=0 general protection fault.

CE=1,WT=1 ist der normale Betriebsmodus des Cache. In diesem Modus wird der Cache bei Lesezugriffen auf Speicherzellen, die noch nicht im Cache vorliegen, gefüllt und bei Zugriffen auf Speicherzellen, die im Cache vorliegen, aktualisiert. Alle zu schreibenden Daten werden unabhängig von eine Treffer oder Fehlzugriff im Cache auch über den externen Bus in die Speicherbausteine des Arbeitsspeichers geschrieben (write-through).

Kontrollregister1 reserviert

Kontrollregister2 Bits 0 — 31 Seitenfehler - Linearadreßregister. Enthält die 32-Bit-Adresse, die den letzten Speicherseitenfehler verursachte.

Kontrollregister3 Bits 8 — 31 Seitenverzeichnis - Basisregister. enthält den Satz der CE- und WT-Cache Kontrollbits, die benutzt werden, wenn eine Speicherseitentabelle ungültig ist.

Debug-Register haben die selbe Bedeutung wie bei dem 80386. Zusätzlich ist im DR7 ein Global-Debug-Bit eingerichtet worden, um den Schutz der Debug-Register vor unerlaubtem Zugriff zu verbessern. Wenn dieses Bit gesetzt ist, verursacht jeder Zugriff auf die Debug-Register einen Interrupt 1H.

Testregister TR3 Cache Data, erlaubt Daten in den Cache schreiben und auszulesen.

TR4 Cache Status

TR5 Cache Kontroll

1. schreiben in den Cache
2. lesen aus dem Cache
3. schreiben in den Cache-F2ullbuffer
4. lesen aus dem Cache-Lesebuffer
5. leeren des Cache

TR6 TLB

TR7 TLB

æ

Kapitel 4

Instruction Set

Der Instruktionssatz besteht aus sieben Gruppen: Data transfer, arithmetic, shift/rotate/logical, string manipulation, control transfer, high level instructions und processor control.

Eine 80286 Instruktion kann sich auf keinen, einen oder zwei Operanden beziehen wobei der Operand sich in einem Register, in der Intruktion selbst oder im Memory befinden kann.

Die Bedeutung der Befehle in der folgenden Liste stammt von IBM. Sie sind wenn nicht direkt falsch, so doch zumindest interpretationsbedürftig. Die richtige Bedeutung kann im Abschnitt "Die Befehle" nachgelesen werden. Sie stammen von Intel. Die müssen ja wissen, was sie eingebaut haben!

Intel sagt leider nicht alles. Bei den Befehlen AAD und AAM ist die Konstante 0AH die Basis für die Division bzw. Multiplikation.

4.1 Liste der Befehle

4.1.1 Data Transfer Instructions

General Purpose

MOV	move byte or word
PUSH	push word onto stack
POP	pop word off stack
PUSHA	push all Registers on stack
POPA	pop all registers from stack
XCHG	exchange byte or word
XLAT	translate byte

Input/Output

IN	input byte or word
OUT	output byte or word
INS	input bytes or word string
OUTS	output bytes or word string

Address Objekt

LEA	load effective address
LDS	load pointer using DS
LES	load pointer using ES

Flag Transfer

LAHF	load AH register from flags
SAHF	store AH register in flags
PUSHF	push flags onto stack
POPF	pop flags off stack

4.1.2 Arithmetic Instructions

Addition

ADD	add byte or word
ADC	add byte or word with cary
INC	increment byte or word by 1
AAA	ASCII adjust for addition
DAA	decimal adjust for addition

Subtraction

SUB	subtract byte or word
SBB	subtract byte or word with borrow
DEC	decrement byte or word by 1
NEG	negate byte or word
CMP	compare byte or word
AAS	ASCII adjust for subtraction
DAS	decimal adjust for subtraction

Multiplication

MUL	multiply byte or word unsigned
IMUL	integer multiply byte or word
AAM	ASCII adjust for multiply

Division

DIV	divide byte or word unsigned
IDIV	integer divide byte or word
AAD	ASCII adjust for division
CBW	convert byte to word
CWD	convert word to doubleword

4.1.3 String Instructions

MOVS	move byte or word string
INS	input bytes or word string
OUTS	output bytes or word string
CMPS	compare byte or word string
SCAS	scan byte or word string
LODS	load byte or word string
STOS	store byte or word string
REP	repeat
REPE/REPZ	repeat while equal/zero
REPNE/REPNZ	repeat while not erqual/not zero

4.1.4 Shift/Rotate/Logical Instructions

Logicals

NOT	'not' byte or word
AND	'and' byte or word
OR	'inclusive or' byte or word
XOR	'exclusive or' byte or word
TEST	'test' byte or word (Ergebnis sind die flags)

Shifts

SHL/SAL	shift logical/arithmetic left byte or word
SHR/SAR	shift logical/arithmetic right byte or word

Rotates

ROL/ROR	rotate left/right byte or word
RCL/RCR	rotate through carryt left/right byte or word

4.1.5 Program Transfer Instructions

Conditional Transfer

JA/JNBE	jump if above/not below nor equal
JAE/JNB	jump if above or equal/not below
JB/JNAE	jump if below/not above nor equal
JBE/JNA	jump if below or equal/not above
JC	jump if carry
JE/JZ	jump if equal/zero
JG/JNLE	jump if greater/not less nor equal
JGE/JNL	jump if greater or equal/not less
JL/JNGE	jump if less/not greater nor equal
JLE/JNG	jump if less or equal/not greater
JNC	jump if not carry
JNE/JNZ	jump if not equal/not zero
JNO	jump if not overflow
JNP/JPO	jump if not parity/parity odd
JNS	jump if not sign
JO	jump if overflow
JP/JPE	jump if parity/parity even
JS	jump if sign

Unconditional Transfer

CALL	call procedure
RET	return from procedure
JMP	jump

Iteration Controls

LOOP	loop
LOOPE/LOOPZ	loop if equal/zero
LOOPNE/LOOPNZ	loop if not equal/not zero
JCXZ	jump if register CX = 0

Interrupts

INT	interrupt
INTO	interrupt if overflow
IRET	interrupt return

4.1.6 Processor Control Instructions

Flag Operations

STC	set carry flag
CLC	clear carry flag
CMC	complement carry flag
STD	set direction flag (autodecrement)
CLD	clear direction flag (autoincrement)
STI	set interrupt enable flag
CLI	clear interrupt enable flag

External Synchronisation

HLT	halt until interrupt or reset
WAIT	wait for busy not active
ESC	escape to extension processor
LOCK	lock bus during next instruction

No Operation

NOP	no operation

Execution Environment Control

LMSW	load machine status word
SMSW	store machine status word

4.1.7 High Level Instructions

ENTER	format stack for procedure entry
LEAVE	restore stack for procedure exit
BOUND	detects values outside prescribed range (ACHTUNG PRINT SCREEN!)

4.1.8 Data Types

Der 80286 unterstützt direkt die folgenden Datentypen:

Integer: A signed binary numeric value contained in an 8-bit byte or a 16-bit word. All operations assume a 2's complement representation.

Ordinal: An unsigned binary numeric value contained in an 8-bit byte or a 16-bit word.

Pointer: A 32-bit quantity, composed of a segment selector component and an offset component. Each component is a 16-bit word.

String: A contiguous sequens of bytes or words. A string may contain from 1 byte to 64k bytes.

ASCII: A byte representation of alphanumeric and control characters using the ASCII standard of character representaion.

BCD: A byte (unpacked) representation of the decimal digits 0 - 9.

Packed BCD: A byte (packed) representation of two decimal digits 0 - 9 storing one digit in each nibble of the byte.

Floating Point: A signed 32, 64 or 80-bit real number representation (80287).

4.2 Die Befehle

Im Folgenden werden die Befehle des 80286 im einzelnen erläutert. Innerhalb jeder Operation können mehrere Instruktionen möglich sein in Abhängigkeit von den Operanden.

Die Erläuterungen folgen einem Schema dessen Teile und die darin verwendeten Abkürzungen im folgenden beschrieben werden. Die Erläuterungen beziehen sich nur auf den real address mode.

4.2.1 Bedeutung der Abkürzungen

In der Spalte Opcode:

/n: einstellige Oktalzahl. Ein ModRM byte und möglicherweise ein immediate und displacement field folgt dem Opcode.

/r: ModRM-Byte mit register operand und effective address operand dem möglicherweise ein immediate und displacement field folgt.

cb: Ein Byte signed displacement im Bereich -128 und +127 folgt dem opcode. Es wird auf 16 Bit erweitert und modulo 65536 zu dem offset der folgenden Instruktion addiert um den neuen Wert des IP zu ergeben.

cw: Ein Wort displacement wird modulo 65536 zu dem offset der folgenden Instruktion addiert um den Wert des neuen IP zu ergeben.

cd: Ein Zwei-Wort pointer der den neuen CS:IP Wert ergibt. Zuerst steht der offset gefolt von dem selector.

db: Ein Byte operand zu der Instruktion der dem opcode und dem ModRM Byte folgt.

dw: Ein Wort operand zu der Instruktion der dem opcode und dem ModRM Byte folgt.

dq: Ein Doppel-Wort operand zu der Instruktion der dem opcode und dem ModRM Byte folgt.

+rb: Ein Register code zwischen 0 und 7 der zu dem hexadezimalen Byte links von dem Pluszeichen addiert wird um dem opcode zu ergeben. AL = 0, CL = 1, DL = 2, BL = 3, AH = 4, CH = 5, DH = 6, BH = 7.

+rw: Ein Register code zwischen 0 und 7 der zu dem hexadezimalen Byte links von dem Pluszeichen addiert wird um dem opcode zu ergeben. AX = 0, CX = 1, DX = 2, BX = 3, SP = 4, BP = 5, SI = 6, DI = 7.

pm: a clock count that applies when the instuction executes in Protected Mode.

In der Spalte Instruction:

cb: Offset zu einer Zielinstruktion in dem Bereich 128 Byte vor Ende der Instruktion bis 127 Byte nach dem Ende der Instruktion.

cw: Offset zu einer Zielinstruktion innerhal des gleichen Segments.

cd: Zieladresse, typischerweise in einem anderen Code Segment.

db: Operand einer Instruktion mit dem Wert zwischen -128 und +127.

dw: Operand einer Instruktion der als Wort der Instruktion folgt.

eb: Byte Operand.

ed: Pointer Operand, memory based.

ew: Wort Operand.

eq: Doppel-Wort Operand.

m: Memory location.

mb: memory-based byte-sized operand.

mw: memory-based byte-sized operand.

rb: Eins der Byte Register:
AL = 0, CL = 1, DL = 2, BL = 3, AH = 4, CH = 5, DH = 6, BH = 7.

rw: Eins der Wort Register:
AX = 0, CX = 1, DX = 2, BX = 3, SP = 4, BP = 5, SI = 6, DI = 7.

rq: Eins der Extended Register: EAX = 0, ECX = 1, EDX = 2, EBX = 3, ESP = 4, EBP = 5, ESI = 6, EDI = 7.

xb: Eine Byte memory Variable ohne Base- oder Index-Register.

xw: Eine Wort memory Variable ohne Base- oder Index-Register.

In der Spalte Clocks:

Hier wird die Anzahl der Clock-Zyklen angegeben die diese Form der Instruktion für die Ausführung benötigt. Die Zeit für einen Zyklus kann errechnet werden in dem man eine mikrosekunde durch die Zahl der MHz teilt mit der der Prozessor läuft. Ein 10-MHz Prozessor, der an einen 20-MHz Kristall angeschlossen ist, benötigt 100 ns für jeden Zyklus.
Wenn die Adressierungsform base plus index plus displacement benutzt wird, muß eine clock hinzugerechnet werden. Für jeden 16-Bit memory based operand der an einer physikalischen odd Adresse beginnt müssen zwei clock addiert werden. Für jeden wait state bei jedem memory read muß jeweils ein clock dazugerechnet werden. Wait states in memory writes oder instruction fetch müssen nicht zu einer längeren Ausführungszeit führen.
Bei einigen Instruktionsformen sind zwei Zahlen angegeben, eine ungelabelte und eine mit label. In Klammern stehen die Zahlen für den 386DX.

mem: Die Instruktion hat einen Operanden der entweder ein Reister oder eine Memoryvariable sein kann. Für jeden indizierten Zugriff muß eine clock hinzugerechnet werden.

noj: Die Instruktion kann einen abhängigen Sprung oder Interrupt ausführen. Die gelabelte Zahl gilt für den nichtausgeführten Sprung.

pm: Dieser Label ist für die Ausführung im protected mode.

In der Spalte **Description** wird nur eine kurze Erläuterung gegeben.
In der Zeile **Flags Modified** werden die Flags angegeben, die der Instruktion entsprechend gesetzt werden. Werden sie immer auf einen bestimmten Wert gesetzt, so ist =0 oder =1 hinter dem Namen der Flag.
In der Zeile **Flags Undefined** stehen die Flags die für diese Instruktion bedeutungslos sind.
Alle bisher nicht erwähnten Flags werden nicht geändert.
In der Zeile **Operation** wird die Instruktion detailiert beschrieben.
In der Zeile **Real Address Mode Exeption** werden die möglichen Exeptions angegeben. Der häufigste Fehler wird ein Wortzugriff auf das letzte Byte in einem Segment sein (Interrupt Nr. 13 General Protection).

4.2.2 AAA ASCII Adjust AL After Addition

Opcode	Instruction	Clocks	Description
37	AAA	3(4)	ASCII adjust AL after addition

Flags Modified Auxiliary carry, carry.

Flags Undefined Overflow, sign, zero, parity.

Operation

AAA sollte nur nach einer ADD Instruktion benutzt werden. ADD hinterläßt ein Byte Resultat im AL Register. Die niederwertigen 4-Bit des Operanden der ADD Instruktion sollten im Bereich von 0 bis 9 (BCD digit) sein. In diesem Fall wird die AAA Instruktion den Inhalt des AL Registers den richtigen Dezimal Stellenwert zuordnen. Wenn die Addition einen Dezimalüberlauf erzeugt, wird das AH Register erhöht und die Carryflag und die Auxiliarycarryflag gesetzt. Wird kein Dezimalüberlauf erzeugt wird die carry und auxiliary flag nach 0 gesetzt und das AH Register bleibt unverändert. In jedem Fall wird das obere Nibble nach 0 gesetzt. Um AL nach ASCII zu konvertieren kann die Intruktion OR AL,30H nach AAA benutzt werden.

Real Address Mode Exeption

keine.

4.2.3 AAD ASCII Adjust AX Before Division

Opcode	Instruction	Clocks	Description
D5 0A	AAD	14(19)	ASCII adjust AX before division

Flags Modified Sign, zero, parity.

Flags Undefined Overflow, auxiliary carry, carry.

Operation

AAD wird benutzt um zwei unpacked BCD digit für eine Division vorzubereiten.

Real Address Mode Exeption

keine.

4.2.4 AAM ASCII Adjust AX After Multiply

Opcode	Instruction	Clocks	Description
D4 0A	AAM	16(17)	ASCII adjust AX after multiply

Flags Modified Sign, zero, parity.

Flags Undefined Overflow, auxiliary carry, cary.

Operation
AAM sollte nur nach einer Multiplikation von zwei unpacked BCD digits erfolgen. Da das Ergebnis der Multiplikation unter Einhundert ist, passt es in das Register AL. AAM teilt das Resultat in AL durch zehn, legt den Quotienten in AH und den Rest in AL ab.

Real Address Mode Exeption
keine.

4.2.5 AAS ASCII Adjust AL After Subtraction

Opcode	Instruction	Clocks	Description
3F	AAS	3(4)	ASCII adjust AL after subtraction

Flags Modified Auxiliary carry, carry.

Flags Undefined Overflow, sign, zero, parity.

Operation
AAS sollte nur nach einer Subtraktion erfolgen die ein Byteresultat im Register AL hinterläßt. Die niederwertigen Nibbles müssen in dem Bereich zwischen 0 und 9 sein. Dann wird die AAS Instruktion das richtige Resultat in dem Register AL ergeben. Erfolgt ein Übertrag, so wird das Register AH dekrementiert und die auxiliary carry und die carry Flag gesetzt. Das obere Nibbel wird nach 0 gesetzt.

Real Address Mode Exeption
keine.

4.2.6 ADC/ADD Integer Addition

Opcode	Instruction	Clocks	Description
10 /r	ADC eb,rb	2,mem=7(2/7)	Add with carry byte register into EA byte
11 /r	ADC ew,rw	2,mem=7(2/7)	Add with carry word register into EA word
12 /r	ADC rb,eb	2,mem=7(2/6)	Add with carry EA byte into byte register
13 /r	ADC rw,ew	2,mem=7(2/6)	Add with carry EA word into word register
14 db	ADC AL,db	3(2)	Add with carry immediate byte into AL
15 dw	ADC AX,dw	3(2)	Add with carry immediate word into AX
80 /2 db	ADC eb,db	3,mem=7(2/7)	Add with carry immediate byte into EA byte
81 /2 dw	ADC ew,dw	3,mem=7(2/7)	Add with carry immediate word into EA word
83 /2 db	ADC ew,db	3,mem=7(2/7)	Add with carry immediate byte into EA word
00 /r	ADD eb,rb	2,mem=7(2/7)	Add byte register into EA byte
01 /r	ADD ew,rw	2,mem=7(2/7)	Add word register into EA word
02 /r	ADD rb,eb	2,mem=7(2/7)	Add EA byte into byte register
03 /r	ADD rw,ew	2,mem=7(2/7)	Add EA word into word register
04 db	ADD AL,db	3(2)	Add immediate byte into AL
05 dw	ADD AX,dw	3(2)	Add immediate word into AX
80 /0 db	ADD eb,db	3,mem=7(2/7)	Add immediate byte into EA byte
81 /0 dw	ADD ew,dw	3,mem=7(2/7)	Add immediate word into EA word
83 /0 db	ADD ew,db	3,mem=7(2/7)	Add immediate byte into EA word

Bei dem 386 clocks 2, mem=6-7.

Flags Modified Overflow, sign, zero, auxiliary carry, parity, carry.

Flags Undefined keine.

Operation
ADD und ADC erzeugen eine Addition von zwei Integerwerten. ADC berücksichtigt auch das vorherige Carrybit. Das Ergebnis wird in dem ersten Operanden abgelegt. Wird ein Byte und ein Wort verwendet, wird das Byte zuerst auf Wortlänge ausgeweitet (sign-extended).

Real Address Mode Exeption
Interrupt 13 bei einem Wortoperand mit Offset 0FFFFH.

4.2.7 AND Logical AND

Opcode	Instruction	Clocks	Description
20 /r	AND eb,rb	2,mem=7(2/7)	Logical-AND byte register into EA byte
21 /r	AND ew,rw	2,mem=7(2/7)	Logical-AND word register into EA word
22 /r	AND rb,eb	2,mem=7(2/7)	Logical-AND EA byte into byte register
23 /r	AND rw,ew	2,mem=7(2/7)	Logical-AND EA word into word register
24 db	AND AL,db	3(2)	Logical-AND immediate byte into AL
25 dw	AND AX,dw	3(2)	Logical-AND immediate word into AX
80 /4 db	AND eb,db	3,mem=7(2/7)	Logical-AND immediate byte into EA byte
81 /4 dw	AND ew,dw	3,mem=7(2/7)	Logical-AND immediate word into EA word

Bei dem 386 clocks 2, mem=6-7.

Flags Modified Overflow=0, sign, zero, parity, carry=0.

Flags Undefined Auxiliary carry.

Operation
Jedes Bit im Ergebnis ist 1 wenn die jeweiligen Bits der Operanden 1 sind.

Real Address Mode Exeption
Interrupt 13 bei einem Wortoperand mit Offset 0FFFFH.

4.2.8 ARPL Adjust RPL Field of Selector

Opcode	Instruction	Clocks	Description
63 /r	ARPL ew,rw	10,mem=11(pm=20/21)	Adjust RPL of EA word not less than RPL of rw

Flags Modified zero.

Flags Undefined keine.

Operation
Die ARPL-Instruktion hat zwei Operanden. Der erste ist eine 16-Bit memory variable oder ein Wort-Register mit dem Wert des Selektors. Der zweite Operand ist ein Wort-Register. Wenn das RPL-Feld (die unteren zwei Bits) des ersten Operanden kleiner ist als das RPL-Feld des zweiten Operanden wird die Zero-Flag nach 1 gesetzt und das RPL-Feld des ersten Operanden wird erhöht.
Die ARPL-Instruktion wird nur in Betriebsystem-Software benutzt!

Real Address Mode Exeption
Interrupt 6. ARPL wird im real address mode nicht erkannt.

4.2.9 BOUND Ceck Array Index Against Bounds

Opcode	Instruction	Clocks	Description
62 /r	BOUND rw,md	noj=13(10)	INT5 if rw not within bounds

Flags Modified keine.

Flags Undefined keine.

Operation

Die BOUND-Instruktion wird benutzt um festzustellen ob der in einem Register (erster Operand) angegebene array index Wert gleich oder größer dem Wert in der ersten Adresse (zeiter Operand) ist. Der index muß auch kleiner oder gleich dem Wert in der nächsten Adresse sein. Werden diese Bedingungen nicht eingehalten, wird ein Interrupt 5 erzeugt (Print Screen).

Wenn die beiden Grenzwerte vor das Array gelegt werden können sie einfach mit dem Offset -4 zur Array-Adresse angesprochen werden.

Real Address Mode Exeption

Interrupt 5 wenn der Test einen Fehler ergibt. Interrupt 13 für einen zweiten Operanden mit einem Offset 0FFFDH oder größer. Interrupt 6 wenn der zweite Operand ein Register ist.

4.2.10 CALL Call Procedure

Opcode	Instruction	Clocks	Description
E8 cw	CALL cw	7	Call near, offset relative to next instruction
FF /2	CALL ew	7,mem=11	Call near, offset absolute at EA word
9A cd	CALL cd	13,pm=26	Call inter-segment,immediate 4-byte address
9A cd	CALL cd	41	Call gate, same privilege
9A cd	CALL cd	82	Call gate, more privilage, no parameters
9A cd	CALL cd	86+4X	Call gate, more privilege, X parameters
9A cd	CALL cd	177	Call via Task State Segment
9A cd	CALL cd	182	Call via task gate
FF /3	CALL ed	16,mem=29	Call inter-segment, address at EA doubleword
FF /3	CALL ed	44	Call gate, same privilege
FF /3	CALL ed	83	Call gate, more privilege, no parameters
FF /3	CALL ed	90+4X	Call gate, more privilege, X parameters
FF /3	CALL ed	180	Call via Task State Segment
FF /3	CALL ed	185	Call via task gate

Für jedes Byte der nächsten ausgeführten Instruktion muß ein clock dazugerechnet werden.

Flags Modified Keine, nur wenn eine Task umgeschaltet wird.

Flags Undefined keine.

Operation

Die Call Instruktion läßt die in dem Operanden angegebene Prozedure ausführen. Wenn die Prozedure mit einer RETURN Instruktion beendet wird, wird das vorhergehende Programm mit der Instruktion nach dem CALL fortgesetzt.

Die cw-Form der Instruktion (Opcode E8) addiert modulo 65536 den 2-Byte Operanden zu dem Offset der der Callinstruktion folgenden Instruktion und setzt den IP auf den resultierenden Offset-Wert. Der

Offset der der Call-Instruktion folgenden Instruktion wird auf den Stack gepushed. Das CS-Register wird nicht verändert.

Bei der ew-Form wird der 2-Byte Operand aus dem angegebenen Memory entnommen.

Bei der cd-Form wird der 4-Byte Operand als Pointer benutzt.

Bei der ed-Form wird der long pointer aus dem angegebenen Memory entnommen. Bei den Formen mit long pointer wird vor dem Offset noch das Segment auf den Stack gepushed.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperand mit Offset 0FFFFH.

4.2.11 CBW Convert Byte into Word

Opcode	Instruction	Clocks	Description
98	CBW	2(3)	Convert byte into word (AH = top bit of AL)

Flags Modified keine.

Flags Undefined keine.

Operation

CBW convertiert das signed Byte in AL zu einem signed Wort in AX. Das höchste Bit in AL wird in alle Bits des AH geschrieben.

Real Address Mode Exeption

keine.

4.2.12 CLC Clear Carry Flag

Opcode	Instruction	Clocks	Description
F8	CLC	2(2)	Clear carry flag.

Flags Modified carry = 0.

Flags Undefined keine.

Operation

CLC setzt die Carry-Flag nach 0.

Real Address Mode Exeption

keine.

4.2.13 CLD Clear Direction Flag

Opcode	Instruction	Clocks	Description
FC	CLD	2(2)	Clear direction flag, SI and DI will increment.

Flags Modified direction = 0.

Flags Undefined keine.

Operation

CLD setzt das Richtungsbit in dem Flag-Register nach 0. Stringbefehle incrementieren danach die Index-Register (SI und/oder DI).

Real Address Mode Exeption

keine.

4.2.14 CLI Clear Interrupt Flag

Opcode	Instruction	Clocks	Description
FA	CLI	3(8)	Clear interrupt flag, interrupts disabled.

Flags Modified Interrupt = 0.

Flags Undefined keine.

Operation

CLI cleared die Interrupt enable flag wenn CPL mindestens gleich IOPL ist. Die Einschränkung gilt nur im protected mode.

Real Address Mode Exeption

keine.

4.2.15 CLTS Clear Task Switched Flag

Opcode	Instruction	Clocks	Description
0F 06	CLTS	2(6)	Clear task switched flag.

Flags Modified task switched = 0.

Flags Undefined keine.

Operation

CLTS cleared die task switched flag im MSW nur im protected mode im Level 0.

Real Address Mode Exeption

keine.

4.2.16 CMC Complement Carry Flag

Opcode	Instruction	Clocks	Description
F5	CMC	2(2)	Complement carry flag.

Flags Modified carry.

Flags Undefined keine.

Operation

CMC dreht das Carry-Bit um, aus 0 wird 1 oder aus 1 wird 0.

Real Address Mode Exeption

keine.

4.2.17 CMP Compare two Operands

Opcode	Instruction	Clocks	Description
3C db	CMP AL,db	3(2)	Compare immediate byte from AL.
3D dw	CMP AX,dw	3(2)	Compare immediate word from AL.
80 /7 db	CMP eb,db	3,mem=6(2/5)	Compare immediate byte from EA byte.
38 /r	CMP eb,rb	2,mem=7(2/5)	Compare byte register from EA byte.
83 /7 db	CMP ew,db	3,mem=6(2/5)	Compare immediate byte from EA word.
81 /7 dw	CMP ew,dw	3,mem=6(2/5)	Compare immediate word from EA word.
39 /r	CMP ew,rw	2,mem=7(2/5)	Compare word register from EA word.
3A /r	CMP rb,eb	2,mem=6(2/6)	Compare EA byte from byte register.
3B /r	CMP rw,rw	2,mem=6(2/6)	Compare EA word from word register.

Bei dem 386 clocks 2, mem=5-6.

Flags Modified overflow, sign, zero, auxiliary carry, carry, parity.

Flags Undefined keine.

Operation

CMP subtrahiert den zweiten Operanden von dem ersten ohne das Ergebnis zu speichern. Nur die Flags werden geändert und mit einem folgenden abhängigen Sprung ausgewertet. Wird ein Wortoperand mit einem direkt folgenden Byte verglichen, so wird das Byte vorher auf ein Wort ausgeweitet (signed-extended).

Real Address Mode Exeption

Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.18 CMPS/CMPSB/CMPSW Compare String Operands

Opcode	Instruction	Clocks	Description
A6	CMPS mb,mb	8	Compare bytes ES:[DI] from [SI].
A6	CMPSB	8	Compare bytes ES:[DI] from DS:[SI].
A7	CMPSW	8	Compare words ES:[DI] from DS:[SI].

Flags Modified overflow, sign, zero, auxiliary carry, parity, carry.

Flags Undefined keine.

Operation
CMPS vergleicht das Byte oder Wort auf das SI zeigt mit dem Byte oder Wort auf das DI zeigt. Das Ergenis sind die Flags. Nach dem Vergleich werden SI und DI verändert. Die Richtung hängt von der direction flag ab (0 increment; 1 decrement). Die Zahl hängt vom Operanden ab (1 für Byte; 2 für Wort). Die REP Instruktion kann benutzt werden um CX-Bytes oder -Worte zu vergleichen.

Real Address Mode Exeption
Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.19 CWD Convert Word to Doubleword

Opcode	Instruction	Clocks	Description
99	CWD	2(2)	Convert word to doubleword (DX:AX = AX).

Flags Modified keine.

Flags Undefined keine.

Operation
CWD convertiert das signed Wort in AX zu einem signed Doppelwort in DX:AX. Das höchste Bit in AX wird in alle Bits von DX geschrieben.

Real Address Mode Exeption
keine.

4.2.20 DAA Decimal Adjust AL After Addition

Opcode	Instruction	Clocks	Description
27	DAA	3(4)	Decimal adjust AL after addition.

Flags Modified sign, zero, auxiliary carry, parity, carry.

Flags Undefined overflow.

Operation
DAA sollte nur nach einer ADD Instruktion, die ein zwei-BCD-digit Byte im AL hinterläßt, verwendet werden.
Wenn die niedrigen vier Bit des AL größer als neun sind oder die auxiliary flag gesetzt ist, wird AL um sechs erhöht und die auxiliary flag gesetzt. Andernfalls wird die auxiliary flag gelöscht. Ist AL jetzt größer als 9FH oder ist die carry flag gesetzt, wird AL um 60H erhöht und die carry flag gesetzt. Andernfalls wird die carry flag gelöscht.

Real Address Mode Exeption
keine.

4.2.21 DAS Decimal Adjust AL After Subtraction

Opcode	Instruction	Clocks	Description
2F	DAS	3(4)	Decimal adjust AL after subtraction.

Flags Modified sign, zero, auxiliary carry, parity, carry.

Flags Undefined overflow.

Operation
DAS sollte nur nach einer Subtraktion, die ein zwei-BCD-digit Byte im AL hinterläßt, verwendet werden. Wenn die niedrigen vier Bit des AL größer als neun sind oder die auxiliary flag gesetzt ist, wird AL um sechs erniedrigt und die auxiliary flag gesetzt. Andernfalls wird die auxiliary flag gelöscht. Ist AL jetzt größer als 9FH oder ist die carry flag gesetzt, wird AL um 60H erniedrigt und die carry flag gesetzt. Andernfalls wird die carry flag gelöscht.

Real Address Mode Exeption
keine.

4.2.22 DEC Decrement by 1

Opcode	Instruction	Clocks	Description
FE /1	DEC eb	2,mem=7(2/6)	Decrement EA byte by 1.
FE /1	DEC ew	2,mem=7(2/6)	Decrement EA word by 1.
48 +rw	DEC rw	2(2)	Decrement word register by 1.

Flags Modified overflow, sign, zero, auxiliary carry, parity.

Flags Undefined keine.

Operation
Von dem Operanden wird 1 abgezogen. Die carry flag wird nicht erzeugt. Wenn die carry flag benutzt werden soll, muß die SUB Instruktion benutzt werden.

Real Address Mode Exeption
Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.23 DIV Unsigned Divide

Opcode	Instruction	Clocks	Description
F6 /6	DIV eb	14,mem=17(14/17)	Unsigned divide AX by EA byte.
F7 /6	DIV ew	22,mem=25(22/25)	Unsigned divide DX:AX by EA word.

Flags Modified keine.

Flags Undefined overflow, sign, zero, auxiliary carry, parity, carry.

Operation
Als Dividend is AX oder AL vorgegeben. Nur der Divisor wird als Operand angegeben. Ist der Operand ein Byte, wird AX durch das Byte geteilt. Der Quotient wird in AL und der Rest in AH gespeichert. Ist der Operand ein Wort, wird DX:AX durch das Wort geteilt. Der quotient wird in AX und der Rest in DX gespeichert.

Real Address Mode Exeption
Interrupt 0 wenn der Quotient zu groß für das angegebene Register (AL oder AX) ist, oder der Divisor ist 0. Interrupt 13 für einen Wortoperanden mit einem Offset von 0FFFFH.

4.2.24 ESC

Wird einer Instruktion oder einem Memory-Operanden für die Benutzung durch einen Coprozessor vorangestellt. Der MASM von Microsoft fügt eine ESC-Instruktion automatisch ein, wenn sie für die 8087-Familie benötigt wird.

4.2.25 ENTER Make Stack Frame for Procedure Parameters

Opcode	Instruction	Clocks	Description
C8 dw 00	ENTER dw,0	11(10)	Make stack frame for procedure parameters.
C8 dw 01	ENTER dw,1	15(12)	Make stack frame for procedure parameters.
C8 dw db	ENTER dw,db	12+4db(15+4(n-1))	Make stack frame for procedure parameters.

Flags Modified keine.

Flags Undefined keine.

Operation

ENTER wird benutzt um für blockstrukturierte Hochsprachen die Stackgröße festzulegen. Der erste Operand gibt an wieviel dynamischer Speicher benutzt werden soll. Der zweite den lexical nesting level in dem Hochsprachen-Programm. Er bestimmt wieviel Pointers des Vorgängers in den neuen Stack kopiert werden sollen. Dabei wird BP als der neue stack frame pointer benutzt.

Wenn der zweite Operand 0 ist wird BP gepushed, BP wird zu SP gesetzt und der erste Operand von SP abgezogen.

Real Address Mode Exeption

keine.

4.2.26 HLT Halt

Opcode	Instruction	Clocks	Description
F4	HLT	2(5)	HALT.

Flags Modified keine.

Flags Undefined keine.

Operation

Der Prozessor hört auf Instruktionen abzuarbeiten. Aus diesem Zustand kommt er nur mit einem erlaubten Interrupt oder mit Reset zurück. Wird der Interrupt benutzt, so wird das Programm bei dem gesicherten CS:IP (der Instruktion nach dem HLT) fortgesetzt.

Real Address Mode Exeption

keine.

4.2.27 IDIV Signed Divide

Opcode	Instruction	Clocks	Description
F6 /7	IDIV eb	17,mem=20	Signed divide AX by EA byte (AL=Quo,AH=Rem).
F7 /7	IDIV ew	25,mem=28	Signed divide DX:AX by EA word (AX=Quo,DX=Rem).

Flags Modified keine.

Flags Undefined overflow, sign, zero, auxiliary carry, parity, carry.

Operation

Als Dividend is AX oder AL vorgegeben. Nur der Divisor wird als Operand angegeben. Ist der Operand ein Byte, wird AX durch das Byte geteilt. Der Quotient wird in AL und der Rest in AH gespeichert. Ist der Operand ein Wort, wird DX:AX durch das Wort geteilt. Der quotient wird in AX und der Rest in DX gespeichert.

Real Address Mode Exeption

Interrupt 0 wenn der Quotient zu groß für das angegebene Register (AL oder AX) ist, oder der Divisor ist 0. Interrupt 13 für einen Wortoperanden mit einem Offset von 0FFFFH.

4.2.28 IMUL Signed Multiply

Opcode	Instruction	Clocks	Description
F6 /5	IMUL eb	13,mem=16	Signed multiply (AX=AL X EA byte).
F7 /5	IMUL ew	21,mem=24	Signed multiply (DXAX=AL X EA word).
6B /r db	IMUL rw,db	21,mem=24	Signed multiply imm. byte into word reg.
69 /r dw	IMUL rw,ew,dw	21,mem=24	Signed multiply (rw=EA word X imm. word).
6B /r db	IMUL rw,ew,db	21,mem=24	Signed multiply (rw=EA word X imm. byte).

Flags Modified overflow, carry.

Flags Undefined sign, zero, auxiliary carry, parity.

Operation

Wenn der Operand ein einzelnes Byte ist, wird dieses mit AL multipliziert und das 16-Bit Ergebnis in AX abgelegt. Carry und overflow werden nach 0 gesetzt wenn AH die Erweiterung des Vorzeichens von AL ist. Andernfalls werden sie nach 1 gesetzt.

Wenn der Operand ein einzelnes Wort ist, wird es mit AX multipliziert und das 32-Bit Ergebnis in DX:AX abgelegt. Carry und overflow werden nach 0 gesetzt wenn AH die Erweiterung des Vorzeichens von AL ist. Andernfalls werden sie nach 1 gesetzt.

Wenn drei Operanden vorhanden sind, wird der zweite (Wort bei EA) mit dem dritten (direkt folgend) multipliziert und das 16-Bit Ergebnis in den ersten (Wortregister) abgelegt. Carry und overflow werden nach 0 gesetzt wenn das Ergebnis in das signed Wort (-32768 bis +32767) paßt. Andernfalls werden sie nach eins gesetzt.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit dem Offset 0FFFFH.

4.2.29 IN Input from Port

Opcode	Instruction	Clocks	Description
E4 db	IN AL,db	5	Input byte from immediate port into AL.
EC	IN AL,DX	5	Input byte from port DX into AL.
E5 db	IN AX,db	5	Input word from immediate port into AX.
ED	IN AX,DX	5	Input word from port DX into AX.

Flags Modified keine.

Flags Undefined keine.

Operation

In tranferiert Datenbytes oder Datenworte von dem im zweiten Operanden angegebenen Port in das als ersten Operanden angegebene (AL oder AX) Register. Die 16-Bit Portadresse wird in DX angegeben. Beschränkt man sich auf eine 8-Bit Adresse, so kann das Byte der Intruktion direkt folgen. Die oberen Bit der Adresse sind dann null. Intel hat die Adressen von 0F8H bis 0FFH reserviert. Sie sollen nicht benutzt werden!

Real Address Mode Exeption

keine.

4.2.30 INC Increment by 1

Opcode	Instruction	Clocks	Description
FE /0	INC eb	2,mem=7	Increment EA byte by 1.
FE /0	INC ew	2,mem=7	Increment EA word by 1.
40+rw	INC rw	2	Increment word register by 1.

Flags Modified overflow, sign, zero, auxiliary carry, parity.

Flags Undefined keine.

Operation

INC addiert 1 zu dem Operanden. Die carry flag wird nicht verändert. Soll die carry flag benutzt werden, muß die ADD Instruktion genommen werden.

Real Address Mode Exeption

Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.31 INS/INSB/INSW Input from Port to String

Opcode	Instruction	Clocks	Description
6C	INS eb,DX	5	Input byte from port DX into ES:[DI].
6D	INS ew,DX	5	Input word from port DX into ES:[DI].
6C	INSB	5	Input byte from port DX into ES:[DI].
6D	INSW	5	Input word from port DX into ES:[DI].

Flags Modified keine.

Flags Undefined keine.

Operation
INS transferiert Daten vom im Register DX angegebenen Port zum Memory auf das ES:DI zeigt. ES, DX und DI sind vorgegeben und müssen benutzt werden. DI wird automatisch verändert, inkrement (DF=0) oder dekrement (DF=1), um 1 bei Byteoperanden oder 2 bei Wortoperanden. INS kann mit REP benutzt werden, wobei dann in CX die Zahl der Transfers steht. Intel hat die Ports 0F8H bis 0FFh reserviert. Sie sollen nicht benutzt werden.
Nicht jedes Gerät kann die mögliche Transfergeschwindigkeit zur Verfügung stellen (ca. 1MB oder 1MW)!

Real Address Mode Exeption
Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.32 INT/INTO Call to Interrupt Procedure

Opcode	Instruction	Clocks	Description
CC	INT 3	23	Interrupt 3 (trap to debugger).
CC	INT 3	40	Interrupt 3, protected mode, same privilege.
CC	INT 3	78	Interrupt 3, protected mode, more privilege.
CC	INT 3	167	Interrupt 3, protected mode, via task gate.
CD db	INT db	23	Interrupt, numbered by immediate byte.
CD db	INT db	40	Interrupt, protected mode, same privilege.
CD db	INT db	78	Interrupt, protected mode, more privilege.
CD db	INT db	167	Interrupt, protected mode, via task gate.
CE	INTO	24,noj=3	Interrupt 4 if overflow flag is 1.

Flags Modified Alle, wenn eine Task umgeschaltet wird. Die IF ist ausgeschaltet.

Flags Undefined keine.

Operation
Die INT Instruktion erzeugt einen Aufruf einer Interrupt-Procedure. Der direkt folgende Operand gibt zwischen 0 und 255 den Index in die Interrupt Descriptor Table an. Im real address mode ist die IDT ein Array aus 4-Byte long Pointers mit der festen Adresse 00000H.
Die INTO Instruktion ist mit INT identisch, nur ist der Vektor fest auf 4 vergeben und die Instruktion wird nur ausgeführt, wenn die overflow flag gesetzt ist.
Die ersten 32 Interrupts sind von Intel für das System reserviert. Einige sind für Routinen für die Behandlung interner Fehler vorgesehen. Die meisten können mit der INT Instruktion nicht ausgeführt werden.
Interrupts wirken wie CALLs ausser daß das Flagregister vor der Returnadresse auf den Stack gepushed wird. IRET stellt das alte Flagregister wieder her.
In real address mode INT pushed die flags, CS und IP, in der Reihenfolge, reseted die Trap Flag und springt dann zu der Routine, die durch den Index angegeben wurde.

Real Address Mode Exeption
Keine. Ist der SP = 1, 3 oder 5 vor einer INT oder INTO Instruktion, wird der Prozessor angehalten.

4.2.33 IRET Interrupt Return

Opcode	Instruction	Clocks	Description
CF	IRET	17,pm=31	Interrupt return (far return and pop flags).
CF	IRET	55	Interrupt return, lesser privilege.
CF	IRET	169	Interrupt return, different task (NT=1).

Flags Modified Das ganze Flagregister wird vom Stack genommen.

Flags Undefined keine.

Operation

In real address mode wird IP, CS und Flags vom Stack gepopped und die unterbrochene Routine fortgesetzt.

Real Address Mode Exeption

Interrupt 13 wenn der Stack mit einem Offset von 0FFFFH gepopped wird.

4.2.34 JCOND Jump Short if Condition Met

Opcode	Instruction	Clocks	Description
77 cb	JA cb	7,noj=3	Jump short if above (CF=0 and ZF=0).
73 cb	JAE cb	7,noj=3	Jump short if above or equal (CF=0).
72 cb	JB cb	7,noj=3	Jump short if below (CF=1).
76 cb	JBE cb	7,noj=3	Jump short if below or equal (CF=1 or ZF=1).
72 cb	JC cb	7,noj=3	Jump short if carry (CF=1).
E3 cb	JCXZ cb	8,noj=4(9 5)	Jump short if CX register is zero (CF or ZF)=0.
74 cb	JE cb	7,noj=3	Jump short if equal (ZF=1).
7F cb	JG cb	7,noj=3	Jump short if greater (ZF=0 and SF=OF).
7D cb	JGE cb	7,noj=3	Jump short if greater or equal (SF=OF).
7C cb	JL cb	7,noj=3	Jump short if less (SF xor OF)=1.
7E cb	JLE cb	7,noj=3	Jump short if less or equal ((SF xor OF) or ZF)=1.
76 cb	JNA cb	7,noj=3	Jump short if not above (CF=1 or ZF=1).
72 cb	JNAE cb	7,noj=3	Jump short if not above nor equal (CF=1).
73 cb	JNB cb	7,noj=3	Jump short if not below (CF=0).
77 cb	JNBE cb	7,noj=3	Jump short if not below nor equal (CF=0 and ZF=0).
73 cb	JNC cb	7,noj=3	Jump short if not carry (CF=0).
75 cb	JNE cb	7,noj=3	Jump short if not equal (ZF=0).
7E cb	JNG cb	7,noj=3	Jump short if not greater ((SF xor OF) or ZF)=1.
7C cb	JNGE cb	7,noj=3	Jump short if not greater nor equal (SF xor OF)=1.
7D cb	JNL cb	7,noj=3	Jump short if not less (SF=OF).
7F cb	JNLE cb	7,noj=3	Jump short if not less nor equal (ZF=0 and SF=OF).
71 cb	JNO cb	7,noj=3	Jump short if not overflow (OF=0).
7B cb	JNP cb	7,noj=3	Jump short if not parity (PF=0).
79 cb	JNS cb	7,noj=3	Jump short if not sign (SF=0).
75 cb	JNZ cb	7,noj=3	Jump short if not zero (ZF=0).
70 cb	JO cb	7,noj=3	Jump short if overflow (OF=1).
7A cb	JP cb	7,noj=3	Jump short if parity (PF=1).
7A cb	JPE cb	7,noj=3	Jump short if parity even (PF=1).
7B cb	JPO cb	7,noj=3	Jump short if parity odd (PF=0).
78 cb	JS cb	7,noj=3	Jump short if sign (SF=1).
74 cb	JZ cb	7,noj=3	Jump short if zero (ZF=1).

Flags Modified keine.

Flags Undefined keine.

Operation

Abhängige Sprünge, bis auf JCXZ, testen die Flags, die von vorhergehenden Instruktionen gesetzt wurden. In der Liste oben beziehen sich "less" und "greater" auf signed Integers, "above" und below" auf unsigned Integers.

JCXZ benutzt das CX Register für den Test auf zero.

Real Address Mode Exeption

keine.

4.2.35 JUMP Jump

Opcode	Instruction	Clocks	Description
EB cb	JMP cb	7	Jump short.
EA cd	JMP cd	180	Jump to task gate.
E9 cw	JMP cw	7	Jump near.
EA cd	JMP cd	11,pm=23	Jump far (4-byte immediate address).
EA cd	JMP cd	38	Jump to call gate, same privilege.
EA cd	JMP cd	175	Jump via task state segment.
FF /4	JMP ew	7,mem=11	Jump near to EA word (absolute offset).
FF /5	JMP ed	15,pm=26	Jump far (4-byte EA in memory doubleword).
FF /5	JMP ed	41	Jump to call gate, same privilege.
FF /5	JMP ed	178	Jump via task state segment.
FF /5	JMP ed	183	Jump to task gate.

Für jedes Byte der nächsten ausgeführten Instruktion muß eine clock dazugerechnet werden.

Flags Modified keine wenn die task nicht umgeschaltet wird.

Flags Undefined keine.

Operation

Ist der Operand ein Byte, so wird es sign-extended zu dem IP addiert. Das CS wird nicht geändert.

Ist der Operand ein Wort, so wird es modulo 65536 zu dem IP addiert. Das CS wird nicht geändert.

Ist der Operand indirekt angegeben, wird der IP auf den neuen Wert gesetzt. Das CS wird nicht geändert.

Ist der Operand ein Pointer, wird CS mit dem Selektor und IP mit dem Offset geladen.

Real Address Mode Exeption

Interrupt 6 wenn der indirekt inter-segment Operand ein Register ist.

4.2.36 LAHF Load Flags into AH Register

Opcode	Instruction	Clocks	Description
9F	LAHF	2(2)	Load:AH = flags SF,ZF,xx,AF,xx,PF,xx,CF.

Flags Modified keine.

Flags Undefined keine.

Operation

Das low Byte des Flags Registers wird in das Register AH geladen.

Real Address Mode Exeption

keine.

4.2.37 LAR Load Access Rights Byte

Opcode	Instruction	Clocks	Description
0F 02 /r	LAR rw,ew	14,mem=16	Load: high (rw)=access rights byte, selector ew.

Flags Modified zero.

Flags Undefined keine.

Operation

Access Rights werden im real address mode nicht benutzt.

Real Address Mode Exeption

Interrupt 6; LAR ist im real address mode nicht enthalten.

4.2.38 LDS/LES Load Doubleword Pointer

Opcode	Instruction	Clocks	Description
C5 /r	LDS rw,ed	7,pm=21	Load EA doubleword into DS and word register.
C4 /r	LES rw,ed	7,pm=21	Load EA doubleword into ES and word register.

Flags Modified keine.

Flags Undefined keine.

Operation

Das erste Wort auf das der Pointer zeigt, wird als Offset in das angegebene Register geladen. Das zweite Wort wird als Selektor in das in der Intruktion (DS oder ES) verwendete Segment Register geladen.

Real Address Mode Exeption

Interrupt 13 für einen Operanden mit Offset 0FFFFH oder 0FFFDH. Interrupt 6 wenn der source Operand ein Register ist.

4.2.39 LEA Load Effective Address Offset

Opcode	Instruction	Clocks	Description
8D /r	LEA rw,m	3	Calculate EA offset given by m, place in rw.

Flags Modified keine.

Flags Undefined keine.

Operation

Der Offset der effektiven Adresse des zweiten Operanden wird in das im ersten Operanden angegebene Register geladen.

Real Address Mode Exeption

Interrupt 6 wenn der zweite Operand ein Register ist.

4.2.40 LEAVE High Level Procedure Exit

Opcode	Instruction	Clocks	Description
C9	LEAVE	5	Set SP to BP, then POP BP.

Flags Modified keine.

Flags Undefined keine.

Operation

LEAVE ist das Komplement zu ENTER. BP wird nach SP geladen. Der alte frame pointer wird dann nach BP gepopped. Mit RET nn werden dann die nn Argumente des Aufrufs übergangen.

Real Address Mode Exeption

Interrupt 13 bei einem Wortoperand mit Offset 0FFFFH.

4.2.41 LGDT/LIDT Load Global/Interrupt Descriptor Table Register

Opcode	Instruction	Clocks	Description
0F 01 /2	LGDT m	11(11)	Load m into global descriptor table register.
0F 01 /3	LIDT m	12(11)	Load m into interrupt descriptor table register.

Flags Modified keine.

Flags Undefined keine.

Operation

Das Global oder das Interrupt Descriptor Table Register werden mit den 6 Byte geladen auf die der Operand zeigt. Das Limit Feld wird mit dem ersten Wort geladen. Das Base Feld wird mit den nächsten 3 Byte geladen. Das letzte Byte wird nicht berücksichtigt.

LGDT uns LIDT werden nur in Betriebssystemsoftware benutzt.

Real Address Mode Exeption

Interrupt 13 bei einem Wortoperand mit Offset 0FFFFH. Interrupt 6 wenn der source Operand ein Register ist.

Diese Instruktionen sind im real address mode nur erlaubt, um den protected mode zu initialisieren.

4.2.42 LLDT Load Local Descriptor Table Register

Opcode	Instruction	Clocks	Description
0F 00 /2	LLDT ew	17,mem=19(20)	Load selector ew into local descriptor table register.

Flags Modified keine.

Flags Undefined keine.

Operation

LLDT wird nur in Betriebssystemsoftware benutzt.

Real Address Mode Exeption

Interrupt 6; LLDT ist im real address mode nicht vorhanden.

4.2.43 LMSW Load Machine Status Word

Opcode	Instruction	Clocks	Description
0F 01 /6	LMSW ew	3,mem=6(11/14)	Load EA word into MSW.

Flags Modified keine.

Flags Undefined keine.

Operation

Das MSW wird mit dem Operanden geladen. Diese Instruktion wird zum Umschalten in den protected mode benutzt. In den real mode kann man nicht zurück setzen.

Real Address Mode Exeption

Interrupt 13 bei einem Operanden mit Offset 0FFFFH.

4.2.44 LoadAll

Dieser Befehl wird von MASM nicht unterstützt. Mit diesem undokumentierten Befehl kann **jede** Speicherstelle im extended Memory erreicht werden.

Opcode	Instruction	Clocks	Description
0F 05	nicht bekannt	195	Restaurieren des im Memory abgelegten Prozessor Zustands.

Flags Modified alter Zustand

Flags Undefined alter Zustand

Operation Vom Segment 80H werden die internen Register und Caches zurückgeladen. Ist die Adressleitung A20 freigeschaltet, lässt sich damit im Memory oberhalb 1 MB arbeiten ohne in den protected mode zu schalten.

Auslegung des Segments 80H:

Adresse	Verwendung	Adresse	Verwendung
00 - 05	nicht belegt	06 - 07	MSW
08 - 15	nicht belegt	16 - 17	TR
18 - 19	Flags	1A - 1B	IP
1C - 1D	LDT	1E - 1F	DS
20 - 21	SS	22 - 23	CS
24 - 25	ES	26 - 27	DI
28 - 29	SI	2A - 2B	BP
2C - 2D	SP	2E - 2F	BX
30 - 31	DX	32 - 33	CX
34 - 35	AX	36 - 3B	ES-Cache
3C - 41	CS-Cache	42 - 47	SS-Cache
48 - 4D	DS-Cache	4E - 53	GDTR-Cache
54 - 59	LDT-Cache	5A - 5F	IDTR-Cache
60 - 65	TSS-Cache	66 - ... DOS-BIOS?	

Auslegung der Descriptor-Cache:

0 — 2 24-Bit Basisadresse des Segments. Das LSB befindet sich an der niedrigsten Adresse, das MSB an der höchsten.

3 Zugriffsrechte für die CS-, DS-, ES- und SS-Descriptor-Caches; Null für die GDTR-, LDTR- und IDTR-Caches.

4 — 5 16-Bit-Segmentgröße.

4.2.45 LOCK Assert Bus LOCK Signal

Opcode	Instruction	Clocks	Description
F0	LOCK	0(0)	Assert BUSLOCK signal for the next instruction.

Flags Modified keine.

Flags Undefined keine.

Operation

LOCK wird vor einer Instruktion angegeben, um das BUS-LOCK Signal des 80286 für die Zeit der Instruktion zu setzen. Damit werden andere Prozessoren ausgeschlossen, Daten zu verändern, an denen der Prozessor gerade arbeitet. LOCK kann für die Instruktionen MOVS, INS und OUTS angegeben werden (ca. 33ms für 64K Byte oder Worte bei 10MHz). Die Instruktion XCHG setzt BUS LOCK immer.

Real Address Mode Exeption

Diese Instruktion selbst nicht, aber die darauf folgende.

4.2.46 LODS/LODSB/LODSW Load String Operand

Opcode	Instruction	Clocks	Description
AC	LODS mb	5	Load byte [SI] into AL.
AD	LODS mw	5	Load word [SI] into AX.
AC	LODSB	5	Load byte DS:[SI] into AL.
AD	LODSW	5	Load word DS:[SI] into AX.

Flags Modified keine.

Flags Undefined keine.

Operation

LODS schreibt das Wort oder Byte auf das SI zeigt in das Register AX oder AL. Danach wird SI automatisch geändert. Die Änderung hängt von der direction flag (0 = increment; 1 = decrement) ab. Bei einer Byteoperation wird um 1, bei einer Wortoperation wird um 2 erhöht.

Real Address Mode Exeption

Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.47 LOOP/LOOPcond Loop Control with CX Counter

Opcode	Instruction	Clocks	Description
E2 cb	LOOP cb	8,noj=4(11)	DEC CX; jump short if CX≠ 0.
E1 cb	LOOPE cb	8,noj=4(11)	DEC CX; jump short if CX≠ 0 and equal (ZF=1).
E0 cb	LOOPNE cb	8,noj=4(11)	DEC CX; jump short if CX≠ 0 and not equal(ZF=0).
E0 cb	LOOPNZ cb	8,noj=4(11)	DEC CX; jump short if CX≠ 0 and ZF=0.
E1 cb	LOOPZ cb	8,noj=4(11)	DEC CX; jump short if CX≠ 0 and zero (ZF=1).

Bei dem 386 muß für jedes Byte der nächsten ausgeführten Instruktion ein clock hinzugerechnet werden.

Flags Modified keine.

Flags Undefined keine.

Operation

LOOP decrementiert zuerst das CX Register. Dann wird der Zustand mit dem angegebenen überfrüft. Ist der angegebene erreicht, wird der Operand zum IP addiert.

Real Address Mode Exeption

keine.

4.2.48 LSL Load Segment Limit

Opcode	Instruction	Clocks	Description
0F 03 /r	LSL rw,ew	14,mem=16	Load:rw=segment limit, selector ew.

Flags Modified keine.

Flags Undefined keine.

Operation

LSL ist nur im protected mode gültig.

Real Address Mode Exeption

Interrupt 6. LSL ist im real address mode nicht vorhanden.

4.2.49 LTR Load Task Register

Opcode	Instruction	Clocks	Description
0F 00 /3	LTR ew	17,mem=19(23/27)	Load EA word into task register.

Flags Modified keine.

Flags Undefined keine.

Operation

LTR ist nur im protected mode gültig.

Real Address Mode Exeption

Interrupt 6. LTR ist im real address mode nicht vorhanden.

4.2.50 MOV Move Data

Opcode	Instruction	Clocks	Description
88 /r	MOV eb,rb	2,mem=3	Move byte register into EA byte.
89 /r	MOV ew,rw	2,mem=3	Move word register into EA word.
8A /r	MOV rb,eb	2,mem=5	Move EA byte into byte register.
8B /r	MOV rw,ew	2,mem=5	Move EA word into word register .
8C /0	MOV ew,ES	2,mem=3	Move ES into EA word.
8C /1	MOV ew,CS	2,mem=3	Move CS into EA word.
8C /2	MOV ew,SS	2,mem=3	Move SS into EA word.
8C /3	MOV ew,DS	2,mem=3	Move DS into EA word.
8E /0	MOV ES,mw	5,pm=19	Move memory word into ES.
8E /0	MOV ES,rw	2,pm=17	Move word register into ES.
8E /2	MOV SS,mw	5,pm=19	Move memory word into SS.
8E /2	MOV SS,rw	2,pm=17	Move word register into SS.
8E /3	MOV DS,mw	5,pm=19	Move memory word into DS.
8E /3	MOV DS,rw	2,pm=17	Move word register into DS.
A0 dw	MOV AL,xb	5	Move byte variable (offset dw) into AL.
A1 dw	MOV AX,xw	5	Move word variable (offset dw) into AX.
A2 dw	MOV xb,AL	3	Move AL into byte variable (offset dw).
A3 dw	MOV xw,AX	3	Move AX into word register (offset dw).
B0+ rb db	MOV rb,db	2	Move immediate byte into byte register.
B8+ rw dw	MOV rw,dw	2	Move immediate word into word register.
C6 /0 db	MOV eb,db	2,mem=3	Move immediate byte into EA byte.
C7 /0 dw	MOV ew,dw	2,mem=3	Move immediate word into EA word.

Flags Modified keine.

Flags Undefined keine.

Operation

Der zweite Operand wird in den ersten Operanden geschrieben. Wird SS beschrieben, sind Interrupts bis zum Ende der nächsten Instruktion nicht erlaubt.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.51 MOVS/MOVSB/MOVSW Move Data from String to String

Opcode	Instruction	Clocks	Description
A4	MOVS mb,mb	5	Move byte [SI] to ES:[DI]
A5	MOVS mw,mw	5	Move word [SI] to ES:[DI]
A4	MOVSB	5	Move byte DS:[SI] to ES:[DI]
A5	MOVSW	5	Move word DS:[SI] to ES:[DI]

Flags Modified keine.

Flags Undefined keine.

Operation
MOVS schreibt das Byte oder Wort auf das [SI] zeigt in das Byte oder Wort auf das ES:[DI] zeigt. Danach werden SI und DI geändert. Die Änderung hängt von der direction flag ab (0 = increment; 1 = decrement). Wurde ein Byte bewegt, wird um eins geändert. Wurde ein Wort bewegt, wird um zwei geändert. MOVS kann auch mit REP benutzt werden, wobei dann die Anzahl der Operationen im Register CX ist.

Real Address Mode Exeption
Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.52 MUL Unsigned Multiplication of AL or AX

Opcode	Instruction	Clocks	Description
F6 /4	MUL eb	13,mem=16	Unsigned multiply (AX=AX x EA byte).
F7 /4	MUL ew	21,mem=24	Unsigned multiply (DXAX=AX x EA word).

Flags Modified overflow, carry.

Flags Undefined sign, zero, auxiliary carry, parity.

Operation
Wenn der Operand ein Byte ist wird AL damit multipliziert. Das Resultat ist dann in AX. Carry und overflow werden nach 0 gesetzt, wenn AH 0 ist; andernfalls werden sie nach 1 gesetzt.
Wenn der Operand ein Wort ist wird AX damit multipliziert. Das Resultat ist dann in DX:AX. Carry und overflow werden nach 0 gesetzt, wenn DX 0 ist; andernfalls werden sie nach 1 gesetzt.

Real Address Mode Exeption
Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.53 NEG Two's Complement Negation

Opcode	Instruction	Clocks	Description
F6 /3	NEG eb	2,mem=7	Two's complement negate EA byte.
F7 /3	NEG ew	2,mem=7	Two's complement negate EA word.

Flags Modified overflow, sign, zero, auxiliary carry, parity, carry.

Flags Undefined keine.

Operation
Das Zweierkomplement des Operanden ersetzt den Operanden. Der Operand wird von zero subtrahiert. Die carry flag wird nach 1 gesetzt. Wenn der Inputoperand zero war, wird die carry flag nach 0 gesetzt.

Real Address Mode Exeption
Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.54 NOP No Operation

Opcode	Instruction	Clocks	Description
90	NOP	3	No Operation.

Flags Modified keine.

Flags Undefined keine.

Operation
NOP ist eine 1-Byte Füllinstruktion. Es wird nur der IP erhöht.

Real Address Mode Exeption
keine.

4.2.55 NOT One's Complement Negation

Opcode	Instruction	Clocks	Description
F6 /2	NOT eb	2,mem=7	Reverse each bit of EA byte.
F7 /2	NOT ew	2,mem=7	Reverse each bit of EA word.

Flags Modified keine.

Flags Undefined keine.

Operation
Der Operand wird invertiert. Jede 1 wird 0 und jede 0 wird 1.

Real Address Mode Exeption
Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.56 OR Logical Inclusive OR

Opcode	Instruction	Clocks	Description
08 /r	OR eb,rb	2,mem=7	Logical OR byte register into EA byte.
09 /r	OR ew,rw	2,mem=7	Logical OR word register into EA word.
0A /r	OR rb,eb	2,mem=7	Logical OR EA byte into byte register.
0B /r	OR rw,ew	2,mem=7	Logical OR EA word into word register.
0C db	OR AL,db	3	Logical OR immediate byte into AL.
0D dw	OR AX,dw	3	Logical OR immediate word into AX.
80 /1 db	OR eb,db	3,mem=7	Logical OR immediate byte into EA byte.
81 /1 dw	OR ew,dw	3,mem=7	Logical OR immediate word into EA word.

Flags Modified overflow=0, sign, zero, parity, carry=0.

Flags Undefined auxiliary carry.

Operation
Die Instruktion erzeugt das inklusive OR der beiden Operanden. Jedes Bit im Resultat wird nach 0 gesetzt wenn in beiden Operanden die entsprechenden Bits 0 sind. Das Resultat wird in den ersten Operanden geschrieben.

Real Address Mode Exeption
Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.57 OUT Output to Port

opcode	Instruction	Clocks	Description
E6 db	OUT db,AL	3	Output byte AL to immediate port number db
E7 db	OUT db,AX	3	Output word AX to immediate port number db
EE	OUT DX,AL	3	Output byte AL to port number DX
EF	OUT DX,AX	3	Output word AX to port number DX

Flags Modified keine.

Flags Undefined keine.

Operation

OUT schreibt das als zweiten Operanden angegebene Register in das als ersten Operanden angegebene Port. Das Register kann AL für Byte oder AX für Wort sein. Wird DX als Port angegeben, so kann die Adresse maximal 65535 sein. Wird ein Byte als Port angegeben, so werden die höheren Bits mit zero aufgefüllt.

Intel hat die Adressen von 0F8H bis 0FFH reserviert. Sie sollen nicht benutzt werden.

Real Address Mode Exeption

keine.

4.2.58 OUTS/OUTSB/OUTSW Output String to Port

Opcode	Instruction	Clocks	Description
6E	OUTS DX,eb	5	Output byte [SI] to port number DX
6F	OUTS DX,ew	5	Output word [SI] to port number DX
6E	OUTSB	5	Output byte DS:[SI] to port number DX
6F	OUTSW	5	Output word DS:[SI] to port number DX

Flags Modified keine.

Flags Undefined keine.

Operation

OUTS schreibt Bytes oder Worte vom Memory auf das SI zeigt zum Port das in DX angegeben wurde. Danach wird SI automatisch geändert. Die Änderung hängt von der direction flag (0 = increment; 1 = decrement) ab. Für Bytes wird um 1 und für Worte um 2 geändert.

OUTS kann auch mit REP benutzt werden. Der Zähler ist dann in CX.

Intel hat die Adressen von 0F8H bis 0FFH reserviert. Sie sollen nicht benutzt werden.

Nicht jedes Gerät kann die mit diesem Befehl verbundene Transferrate (5+ clocks) verarbeiten.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.59 POP Pop a Word from the Stack

Opcode	Instruction	Clocks	Description
1F	POP DS	5,pm=20	Pop top of stack into DS
07	POP ES	5,pm=20	Pop top of stack into ES
17	POP SS	5,pm=20	Pop top of stack into SS
8F /0	POP mw	5	Pop top of stack into memory word
58+rw	POP rw	5	Pop top of stack into word register

Flags Modified keine.

Flags Undefined keine.

Operation

Das Wort am oberen Ende des Stack auf das SS:SP zeigt ersetzt den angegebenen Operanden. SP wird um 2 erhöht um auf das neue Ende zu zeigen. Eine POP SS Instruktion verbietet jeden Interrupt bis zum Ende der nächsten Instruktion. Deshalb ist eine POP SP Instruktion danach erlaubt.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.60 POPA Pop All General Registers

Opcode	Instruction	Clocks	Description
61	POPA	19	Pop in order: DI,SI,BP,SP,BX,DX,CX,AX

Flags Modified keine.

Flags Undefined keine.

Operation

Die Register DI, SI, BP, SP, BX, DX, CX, AX werden vom Stack zurückgeladen. Der Wert des SP wird weggelegt (wohin?) und nicht in den SP geladen!

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.61 POPF Pop from Stack into the Flags Register

Opcode	Instruction	Clocks	Description
9D	POPF	5	Pop top of stack into flags register

Flags Modified Das ganze flags register wird vom stack geladen.

Flags Undefined keine.

Operation

Das Flagregister wird mit dem Wort geladen auf das SS:SP zeigt. Der SP wird um 2 erhöht.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

Im real mode werden NT und IOPL nicht verändert.

4.2.62 PUSH Push a Word onto the Stack

Opcode	Instruction	Clocks	Description
06	PUSH ES	3	Push ES
0E	PUSH CS	3	Push CS
16	PUSH SS	3	Push SS
1E	PUSH DS	3	Push DS
50+rw	PUSH rw	3	Push word register
FF /6	PUSH mw	5	Push memory word
68 dw	PUSH dw	3	Push immediate word
6A db	PUSH db	3	Push immeditate sign-extended byte

Flags Modified keine.

Flags Undefined keine.

Operation
Der SP wird um 2 ernidriegt und der Operand wird in das memory geschrieben auf das SS:SP zeigt.

Real Address Mode Exeption
Keine. Der Prozessor hält an, wenn der SP = 1 ist.

4.2.63 PUSHA Push All General Registers

Opcode	Instruction	Clocks	Description
60	PUSHA	17	Push in order: AX,CX,DX,BX, original SP,BP,SI,DI.

Flags Modified keine.

Flags Undefined keine.

Operation
PUSHA sichert die Register auf dem Stack. Der SP wird um 16 ernidriegt.

Real Address Mode Exeption
Der Prozessor stell den Betrieb ein wenn vor der Ausführung der PUSHA Instruktion der SP = 1, 3 oder 5 ist. Ist SP = 7, 9, 11, 13 oder 15 wird Interrupt 13 erzeugt.

4.2.64 PUSHF Push Flags Register onto the Stack

Opcode	Instruction	Clocks	Description
9C	PUSHF	3	Push flags register

Flags Modified keine.

Flags Undefined keine.

Operation
Der SP wird um 2 ernidriegt und das Flagregister auf den Stack geschrieben auf den SS:SP zeigt.

Real Address Mode Exeption
Keine. Der Prozessor stellt den Betrieb ein, wenn der SP = 1 ist.

4.2.65 RCL/RCR/ROL/ROR Rotate Instuctions

Opcode	Instruction	Clocks	Description
D0 /2	RCL eb,1	2,mem=7	Rotate 9-bits (CF,EA byte) left once
D2 /2	RCL eb,CL	5,mem=8	Rotate 9-bits (CF,EA byte) left CL times
C0 /2 db	RCL eb,db	5,mem=8	Rotate 9-bits (CF,EA byte) left db times
D1 /2	RCL eb,1	2,mem=7	Rotate 17-bits (CF,EA word) left once
D3 /2	RCL eb,CL	5,mem=8	Rotate 17-bits (CF,EA word) left CL times
C1 /2 db	RCL eb,db	5,mem=8	Rotate 17-bits (CF,EA word) left db times
D0 /3	RCR eb,1	2,mem=7	Rotate 9-bits (CF,EA byte) right once
D2 /3	RCR eb,CL	5,mem=8	Rotate 9-bits (CF,EA byte) right CL times
C0 /3 db	RCR eb,db	5,mem=8	Rotate 9-bits (CF,EA byte) right db times
D1 /3	RCR eb,1	2,mem=7	Rotate 17-bits (CF,EA word) right once
D3 /3	RCR eb,CL	5,mem=8	Rotate 17-bits (CF,EA word) right CL times
C1 /3 db	RCR eb,db	5,mem=8	Rotate 17-bits (CF,EA word) right db times
D0 /0	ROL eb,1	2,mem=7	Rotate 8-bit EA byte left once
D2 /0	ROL eb,CL	5,mem=8	Rotate 8-bit EA byte left CL times
C0 /0 db	ROL eb,db	5,mem=8	Rotate 8-bit EA byte left db times
D1 /0	ROL eb,1	2,mem=7	Rotate 16-bit EA word left once
D3 /0	ROL eb,CL	5,mem=8	Rotate 16-bit EA word left CL times
C1 /0 db	ROL eb,db	5,mem=8	Rotate 16-bit EA word left db times
D0 /1	ROR eb,1	2,mem=7	Rotate 8-bit EA byte right once
D2 /1	ROR eb,CL	5,mem=8	Rotate 8-bit EA byte right CL times
C0 /1 db	ROR eb,db	5,mem=8	Rotate 8-bit EA byte right db times
D1 /1	ROR eb,1	2,mem=7	Rotate 16-bit EA word right once
D3 /1	ROR eb,CL	5,mem=8	Rotate 16-bit EA word right CL times
C1 /1 db	ROR eb,db	5,mem=8	Rotate 16-bit EA word right db times

Flags Modified overflow (only for single rotates), carry.

Flags Undefined overflow for multi-bit rotates.

Operation
Jede rotate Instruktion verschiebt die Bits in einem Register oder angegebenen Memory. Die Bits am Ende kommen auf der anderen Seite herein. Bei den RCL und RCR Instruktionen wird das Register um die carry flag verlängert (9 bzw. 17 Bits). Bei den ROL und ROR Instruktionen ist die carry flag nicht Teil des Registers sondern nimmt nur den Wert des über die Registergrenze geschobenen Bits an. Die Instruktion wird sooft widerholt wie in dem zweiten Operanden angegeben, der entweder der Instruktion folgt oder sich im CL Register befindet. Um Die maximale Zeit der Ausführung zu begrenzen läßt der 80286 nur Zähler bis 31 zu. Ist der Zähler größer, werden nur die 5 nidriegsten Bit berücksichtigt.

Real Address Mode Exeption
Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.66 REP/REPE/REPNE Repeat Following String Operations

Opcode	Instruction	Clocks	Description
F3 6C	REP INS eb,DX	5+4⋆CX	Input CX bytes from port DX into ES:[DI]
F3 6D	REP INS ew,DX	5+4⋆CX	Input CX words from port DX into ES:[DI]
F3 6C	REP INSB	5+4⋆CX	Input CX bytes from port DX into ES:[DI]
F3 6D	REP INSW	5+4⋆CX	Input CX words from port DX into ES:[DI]
F3 A4	REP MOVS mb,mb	5+4⋆CX	Move CX bytes from [SI] to ES:[DI]
F3 A5	REP MOVS mw,mw	5+4⋆CX	Move CX words from [SI] to ES:[DI]
F3 A4	REP MOVSB	5+4⋆CX	Move CX bytes from [SI] to ES:[DI]
F3 A5	REP MOVSW	5+4⋆CX	Move CX words from [SI] to ES:[DI]
F3 6E	REP OUTS DX,eb	5+4⋆CX	Output CX bytes from [SI] to port DX
F3 6E	REP OUTS DX,ew	5+4⋆CX	Output CX words from [SI] to port DX
F3 6E	REP OUTSB	5+4⋆CX	Output CX bytes from DS:[SI] to port DX
F3 6E	REP OUTSW	5+4⋆CX	Output CX words from DS:[SI] to port DX
F3 AA	REP STOS mb	4+3⋆CX	Fill CX bytes at ES:[DI] with AL
F3 AB	REP STOS mw	4+3⋆CX	Fill CX words at ES:[DI] with AX
F3 AA	REP STOSB	4+3⋆CX	Fill CX bytes at ES:[DI] with AL
F3 AB	REP STOSW	4+3⋆CX	Fill CX words at ES:[DI] with AX
F3 A6	REPE CMPS mb,mb	5+9⋆N	Find nonmatching bytes in ES:[DI] and [SI]
F3 A7	REPE CMPS mw,mw	5+9⋆N	Find nonmatching words in ES:[DI] and [SI]
F3 A6	REPE CMPSB	5+9⋆N	Find nonmatching bytes in ES:[DI] and DS:[SI]
F3 A7	REPE CMPSW	5+9⋆N	Find nonmatching words in ES:[DI] and DS:[SI]
F3 AE	REPE SCAS mb	5+8⋆N	Find non-AL byte starting at ES:[DI]
F3 AE	REPE SCAS mw	5+8⋆N	Find non-AX word starting at ES:[DI]
F3 AE	REPE SCASB	5+8⋆N	Find non-AL byte starting at ES:[DI]
F3 AE	REPE SCASW	5+8⋆N	Find non-AX word starting at ES:[DI]
F2 A6	REPNE CMPS mb,mb	5+9⋆N	Find matching bytes in ES:[DI] and [SI]
F2 A7	REPNE CMPS mw,mw	5+9⋆N	Find matching words in ES:[DI] and [SI]
F2 A6	REPNE CMPSB	5+9⋆N	Find matching bytes in ES:[DI] and DS:[SI]
F2 A7	REPNE CMPSW	5+9⋆N	Find matching words in ES:[DI] and DS:[SI]
F2 AE	REPNE SCAS mb	5+⋆N	Find AL, starting at ES:[DI]
F2 AF	REPNE SCAS mw	5+⋆N	Find AX, starting at ES:[DI]
F2 AE	REPNE SCASB	5+⋆N	Find AL, starting at ES:[DI]
F2 AF	REPNE SCASW	5+⋆N	Find AX, starting at ES:[DI]

Flags Modified Keine.

Flags Undefined keine.

Operation

REP wird den Stringinstruktionen voran gestellt und wirkt mit CX als Zähler oder der zero flag als Bedingung.

Die REP Instruktionen können nur in Verbindung mit String-Operationen benutzt werden.

1. Test das CX Register. Wenn es null ist, wird die nächste Instruktion ausgeführt.
2. Bestätigung anstehender Interrupts.
3. Einmalige Ausführung der String-Operation.
4. Dekrement CX mit eins. Keine flags werden geändert.

5. Wenn die String-Operation SCAS oder CMPS ist, wird die zero flag getestet. Wenn die Repeat-Bedingung nicht mehr gegeben ist, wird die nächste Instruktion ausgeführt. Ausgang für REPE und ZF=0 (der letzte Vergleich war nicht gleich) oder für REPNE und ZF=1 (der letzte Vergleich war gleich).
6. Zurück zur nächste Wiederholung.

Nicht alle angesprochenen Geräte können der Abarbeitungsgeschwindigkeit der Instruktionen folgen!

Real Address Mode Exeption
Keine für REP.

4.2.67 RET Return from Procedure

Opcode	Instruction	Clocks	Description
CB	RET	15,pm=25	Return to far caller, same privilege
CB	RET	55	Return, lesser privilege, switch stacks
C3	RET	11	Return to near caller, same privilege
CA dw	RET dw	15,pm=25	RET (far), same privilege, pop dw bytes !!
CA dw	RET dw	55	RET (far), lesser privilege, pop dw bytes !!
C2 dw	RET dw	11	RET (near), same privilege, pop dw bytes !! pushed before Call

Flags Modified keine.

Flags Undefined keine.

Operation
RET gibt die Kontrolle an die Returnadresse, die sich auf dem Stack befindet, ab. Diese Adresse wird gewöhnlich von einer CALL Instruktion auf den Stack geschrieben. Dann wird RET mit der Instruktion nach CALL fortfahren.
In der dw-Form können nach dem Rücksprung dw-Byte (Argumente des CALL) von dem Wortstack entlassen werden. In diesem Fall ist von dem sonst durchgängigen Byte zu Byte und Wort zu Wort abgegangen worden!
Bei einem Intra-Segment return werden 2 Byte vom Stack in den IP geladen. Das CS Register wird nicht geändert.
Bei einem Inter-Segment return werden 4 Byte vom Stack genommen. Die ersten 2 Byte sind der Offset und die zweiten 2 Byte das Segment (der Selektor).

Real Address Mode Exeption
Interrupt 13 wenn die Stackadresse von 0FFFFH nach 0 geht.

4.2.68 SAHF Store AH into Flags

Opcode	Instruction	Clocks	Description
9E	SAHF	2	Store AH into flags SF ZF xx AF xx PF xx CF

Flags Modified Sign, zero, auxiliary carry, parity, carry.

Flags Undefined keine.

Operation
Die Flagbits werden entsprechend den Bits im AH Register gesetzt.

Real Address Mode Exeption
keine.

4.2.69 SAL/SAR/SHL/SHR Shift Instructions

Opcode	Instruction	Clocks	Description
D0 /4	SAL eb,1	2,mem=7	Multiply EA byte by 2, once
D2 /4	SAL eb,CL	5,mem=8	Multiply EA byte by 2, CL times
C0 /4 db	SAL eb,db	5,mem=8	Multiply EA byte by 2, db times
D1 /4	SAL ew,1	2,mem=7	Multiply EA word by 2, once
D3 /4	SAL eb,CL	5,mem=8	Multiply EA word by 2, CL times
C1 /4 db	SAL eb,db	5,mem=8	Multiply EA word by 2, db times
D0 /7	SAR eb,1	2,mem=7	Signed divide EA byte by 2, once
D2 /7	SAR eb,CL	5,mem=8	Signed divide EA byte by 2, CL times
C0 /7 db	SAR eb,db	5,mem=8	Signed divide EA byte by 2, db times
D1 /7	SAR ew,1	2,mem=7	Signed divide EA word by 2, once
D3 /7	SAR ew,CL	5,mem=8	Signed divide EA word by 2, CL times
C1 /7 db	SAR ew,db	5,mem=8	Signed divide EA word by 2, db times
D0 /5	SHR eb,1	2,mem=7	Undigned divide EA byte by 2, once
D2 /5	SHR eb,CL	5,mem=8	Undigned divide EA byte by 2, CL times
C0 /5 db	SHR eb,db	5,mem=8	Undigned divide EA byte by 2, db times
D1 /5	SHR ew,1	2,mem=7	Undigned divide EA word by 2, once
D3 /5	SHR ew,CL	5,mem=8	Undigned divide EA word by 2, CL times
C1 /5 db	SHR ew,db	5,mem=8	Undigned divide EA word by 2, db times

Flags Modified Overflow (nur für einzelshift), carry, zero, parity, sign.

Flags Undefined Auxiliary carry; auch overflow für multibitshift.

Operation

SAL verschiebt die Bits nach oben. Das höchste Bit wird in die carryflag geschoben und das unterste nach null gesetzt.

SAR und SHR schieben die Bits nach unten. Das unterste Bit wird in die carryflag geschoben. Der Operand wird also durch zwei geteilt. Bei SAR bleibt das Vorzeichenbit (oberstes) erhalten. Bei SHR wird das oberste Bit nach null gesetzt.

Es wird so oft geschoben, wie in dem zweiten Operanden angegeben. Der 80286 begrenzt die Zahl auf die letzten 5 Bit (max. 31). Der 8086 benutzt 8 Bit.

Es wird keine flag verändert, wenn der zweite Operand (Zähler) null ist.

Real Address Mode Exeption

Interrupt 13 fr̈ einen Wortoperanden bei 0FFFFH.

4.2.70 SBB Integer Subtraction with Borrow

Opcode	Instruction	Clocks	Description
18 /r	SBB eb,rb	2,mem=7	Subtract with borrow byte register from EA byte
19 /r	SBB ew,rw	2,mem=7	Subtract with borrow word register from EA word
1A /r	SBB rb,eb	2,mem=7	Subtract with borrow EA byte from byte register
1B /r	SBB rw,ew	2,mem=7	Subtract with borrow EA word from word register
1C db	SBB AL,db	3	Subtract with borrow imm. byte from AL
1D dw	SBB AX,db	3	Subtract with borrow imm. word from AX
80 /3 db	SBB eb,db	3,mem=7	Subtract with borrow imm. byte from EA byte
81 /3 dw	SBB ew,dw	3,mem=7	Subtract with borrow imm. word from EA word
83 /3 db	SBB ew,db	3,mem=7	Subtract with borrow imm. byte from EA word

Flags Modified Overflow, sign, zero, auxiliary carry, parity, cary.

Flags Undefined keine.

Operation

Der zweite Operand wird zu der carry-flag addiert und dann vom ersten Operanden abgezogen. Der erste Operand wird durch das Resultat ersetzt und die flags entsprechend gesetzt. Wenn ein Byte von einem Wortoperanden absezogen werden soll, so wird es mit dem Vorzeichen expandiert.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset FFFFH.

4.2.71 SCAS/SCASB/SCASW Compare String Data

Opcode	Instruction	Clocks	Description
AE	SCAS mb	7	Compare bytes AL - ES:[DI], advance DI
AF	SCAS mw	7	Compare words AX - ES:[DI], advance DI
AE	SCASB	7	Compare bytes AL - ES:[DI], advance DI
AF	SCASW	7	Compare words AX - ES:[DI], advance DI

Flags Modified Overflow, sign, zero, auxiliary carry, parity, carry.

Flags Undefined keine.

Operation

SCAS subtrahiert ein Byte oder ein Wort von AL oder AX. Das Resultat wird nicht aufgehoben, nur die flags werden gesetzt. Der Operand muß zum ES Register adressierbar sein.

Nach der Operation wird das DI Register automatisch geändert, entsprechend der direction flag.

SCAS kann mit REP verwendet werden z. B. in einem Block mit CX Bytes oder Worten.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.72 SGDT/SIDT Store Global/Interrupt Descriptor Table Register

Opcode	Instruction	Clocks	Description
0F 01 /0	SGDT m	11	Store Global Descriptor Table register to m
0F 01 /1	SIDT m	12	Store Interrupt Descriptor Table register to m

Flags Modified keine.

Flags Undefined keine.

Operation

Die 6 Byte des descriptor table register werden in das Memory geladen, auf das der Operand zeigt. Das LIMIT Feld belegt das erste Wort, die nächsten 3 Byte sind das BASE Feld und das lezte Byte hat zur Zeit keine Bedeutung.

SGDT und SIDT werden nur in Betriebssystemsoftware benutzt.

Real Address Mode Exeption

Diese Instruktionen sind im real address mode erlaubt, um die power-up Test zu unterstützen und die Initialisierung des protected mode zu ermöglichen.

Interrupt 6 wenn der Operand ein Register ist. Interrupt 13 bei einem Wortoperanden mit Offset 0FFFFH.

4.2.73 SLDT Store Local Descriptor Table Register

Opcode	Instruction	Clocks	Description
0F 00 /0	SLDT ew	2,mem=3	Store Local Descriptor Table register to EA word

Flags Modified keine.

Flags Undefined keine.

Operation

Die 2 Byte des local descriptor table Registers werden im Memory bei der angegebenen Adresse abgelegt. Dieses Register ist ein Selector, der auf die global descriptor table zeigt.

SLDT wird nur in Betriebssystemsoftware benutzt.

Real Address Mode Exeption

Interrupt 6; SLDT ist im real address mode nicht gültig.

4.2.74 SMSW Store Machine Status Word

Opcode	Instruction	Clocks	Description
0F 01 /4	SMSW ew	2,mem=3	Store Machine Status Word to EA word

Flags Modified keine.

Flags Undefined keine.

Operation

Das MSW wird im angegebenen Register oder Memorywort abgelegt.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.75 STC Set Carry Flag

Opcode	Instruction	Clocks	Description
F9	STC	2	Set carry flag

Flags Modified Carry=1

Flags Undefined keine.

Operation Die Carryflag wird nach 1 gesetzt.

Real Address Mode Exeption

keine.

4.2.76 STD Set Direction Flag

Opcode	Instruction	Clocks	Description
FD	STD	2	Set direction flag so SI and DI will decrement

Flags Modified Direction=1

Flags Undefined keine.

Operation Das Richtungsbit im Flagregister wird nach 1 gesetzt. Folgende Stringoperationen zählen die Indexregister (SI und/oder DI) herunter.

Real Address Mode Exeption

keine.

4.2.77 STI Set Interrupt Enable Flag

Opcode	Instruction	Clocks	Description
FB	STI	2	Set interrupt enable flag

Flags Modified Interrupt=1

Flags Undefined keine.

Operation Das Interrupt-enable-bit wird nach 1 gesetzt. Der Prozessor reagiert auf Interruptanforderung der Geräte.

Real Address Mode Exeption

keine.

4.2.78 STOS/STOSB/STOSW Store String Data

Opcode	Instruction	Clocks	Description
AA	STOS mb	3	Store AL to byte ES:[DI], advance DI
AB	STOS mw	3	Store AL to word ES:[DI], advance DI
AA	STOSB	3	Store AL to byte ES:[DI], advance DI
AB	STOSW	3	Store AL to word ES:[DI], advance DI

Flags Modified keine.

Flags Undefined keine.

Operation

STOS transferiert AL oder AX in das Memory auf das ES:DI zeigt. Der Operand muß von ES adressiert werden können. Danach wird DI automatisch geändert. Ist die directionflag = 0, wird incrementiert, andernfalls decrementiert. Bei Byteoperanden wird um eins, bei Wortoperanden um zwei geändert.

STOS kann mit REP für CX Operationen benutzt werden.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.79 STR Store Task Register

Opcode	Instruction	Clocks	Description
0F 00 /1	STR ew	2,mem=3	Store Task Register to EA word

Flags Modified keine.

Flags Undefined keine.

Operation

Der Inhalt des Task Registers wird in das Wortregister oder das angegebene Memory geschrieben.

Real Address Mode Exeption

Interrupt 6; STR ist im real address mode nicht erlaubt.

4.2.80 SUB Integer Subtraction

Opcode	Instruction	Clocks	Description
28 /r	SUB eb,rb	2,mem=7	Subtract byte register from EA byte
29 /r	SUB ew,rw	2,mem=7	Subtract word register from EA word
2A /r	SUB rb,eb	2,mem=7	Subtract EA byte from byte register
2B /r	SUB rw,ew	2,mem=7	Subtract EA word from word register
2C db	SUB AL,db	3	Subtract immediate byte from AL
2D dw	SUB AX,dw	3	Subtract immediate word from AX
80 /5 db	SUB eb,db	3,mem=7	Subtract immediate byte from EA byte
81 /5 dw	SUB ew,dw	3,mem=7	Subtract immediate word from EA word
83 /5 db	SUB ew,db	3,mem=7	Subtract immediate byte from EA word

Flags Modified Overflow, sign, zero, auxiliary carry, parity, carry.

Flags Undefined keine.

Operation

Der zweite Operand wird vom ersten Operanden abgezogen. Der erste Operand wird von dem Resultat ersetzt. Ein Byte Operand wird vorher mit dem Vorzeichen expandiert.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.81 TEST Logical Compare

Opcode	Instruction	Clocks	Description
84 /r	TEST eb,rb	2,mem=6	AND byte register into EA byte for flags only
84 /r	TEST rb,eb	2,mem=6	AND EA byte into byte register for flags only
85 /r	TEST ew,rw	2,mem=6	AND word register into EA word for flags only
85 /r	TEST rw,ew	2,mem=6	AND EA word into word register for flags only
A8 db	TEST AL,db	3	AND immediate byte into AL for flags only
A9 dw	TEST AX,dw	3	AND immediate word into AX for flags only
F6 /0 db	TEST eb,db	3,mem=6	AND immediate byte into EA byte for flags only
F7 /0 dw	TEST ew,dw	3,mem=6	AND immediate word into EA word for flags only

Flags Modified Overflow=0, sign, zero, parity, carry=0.

Flags Undefined Auxiliary carry.

Operation
Jedes Bit des Resultats ist eins, wenn die entsprechenden Bits der Operanden jeweils eins sind. Das Resultat wird nicht gespeichert, nur die flags werden entsprechend gesetzt.

Real Address Mode Exeption
Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.82 VERR,VERW Verify a Segment for Reading or Writing

Opcode	Instruction	Clocks	Description
0F 00 /4	VERR ew	14,mem=16	Set ZF = 1 if seg. can be read, selector ew.
0F 00 /5	VERW ew	14,mem=16	Set ZF = 1 if seg. can be written, selector ew.

Flags Modified zero.

Flags Undefined keine.

Operation
Die Instruktionen erwarten in dem Wortregister oder dem Memory-Operanden einen Selector-Wert. Sie stellen fest ob das angegebene Segment von dem augenblicklichen Privileglevel aus erreichbar ist.

Real Address Mode Exeption
Interrupt 6; VERR und VERW sind im real mode nicht vorhanden.

4.2.83 WAIT Wait until BUSY Pin is inactive (HIGH)

Opcode	Instruction	Clocks	Description
9B	WAIT	3	Wait until busy-pin is inactive (HIGH).

Flags Modified keine.

Flags Undefined keine.

Operation
Die Instruktion setzt das Abarbeiten von 80286-Instruktionen aus bis der BUSY-Pin inactiv wird. Der BUSY-Pin wird von dem 80287 NPX benutzt. Damit wird sichergestellt, daß die numerische Instruktion abgearbeitet wurde und ein eventueller Fehler festgestellt werden kann. Für die Coprozessoren der 8087-Familie steht der Befehl **FWAIT** zur Verfügung.

Real Address Mode Exeption
Interrupt 7 wenn das TS Bit im MSW gesetzt ist. Interrupt 16 wenn der 80287 einen nichtmaskierten Fehler feststellt.

4.2.84 XCHG Exchange Memory/Register with Register

Opcode	Instruction	Clocks	Description
86 /r	XCHG eb,rb,	3,mem=5	Exhange byte register with EA byte
86 /r	XCHG rb,eb,	3,mem=5	Exhange EA byte with byte register
87 /r	XCHG ew,rw,	3,mem=5	Exhange word register with EA word
87 /r	XCHG rw,ew,	3,mem=5	Exhange EA word with word register
90+rw	XCHG AX,rw,	3	Exhange word register with AX
90+rw	XCHG rw,AX,	3	Exhange with word register

Flags Modified keine.

Flags Undefined keine.

Operation

Die beiden Operanden werden ausgetauscht. Die Reihenfolge ist unwesentlich. Unabhängig von LOCK oder IOPL wird BUSLOCK gesetzt.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.85 XLAT Table Look-up Translation

Opcode	Instruction	Clocks	Description
D7	XLAT mb	5	Set AL to memory byte DS:[BX + unsigned AL]
D7	XLATB	5	Set AL to memory byte DS:[BX + unsigned AL]

Flags Modified keine.

Flags Undefined keine.

Operation

Der Index in AL, der auf einen Platz in der von DS:BX adressierten Tabelle zeigt, wird von dem Tabelleninhalt überschrieben. BX wird nicht geändert.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.2.86 XOR Logical Exclusive OR

Opcode	Instruction	Clocks	Description
30 /r	XOR eb,rb	2,mem=7	Exclusive-OR byte register into EA byte
31 /r	XOR ew,rw	2,mem=7	Exclusive-OR word register into EA word
32 /r	XOR rb,eb	2,mem=7	Exclusive-OR EA byte into byte register
33 /r	XOR rw,ew	2,mem=7	Exclusive-OR EA word into word register
34 db	XOR AL,db	3	Exclusive-OR immediate byte into AL
35 dw	XOR AX,dw	3	Exclusive-OR immediate word into AX
80 /6 db	XOR eb,db	3,mem=7	Exclusive-OR immediate byte into EA byte
81 /6 dw	XOR ew,dw	3,mem=7	Exclusive-OR immediate word into EA word

Flags Modified overflow=0, sign, zero, parity, carry=0.

Flags Undefined auxiliary carry.

Operation

Jedes Bit des Resultats ist eins, wenn die entsprechenden Bits der Operanden unterschiedlich sind. Es ist null, wenn sie gleich sind. Das Resultat ersetzt den ersten Operanden.

Real Address Mode Exeption

Interrupt 13 für einen Wortoperanden mit Offset 0FFFFH.

4.3 80386 only

Hier stehen die über den Befehlssatz des 286 hinausgehenden Befehle.

4.3.1 ADC/ADD Integer Addition

Opcode	Instruction	Clocks	Description
15 dq	ADC EAX,dq	2	Add with carry immediate dword into EAX
81 /2 dq	ADC eq,dq	2,mem=7	Add with carry immediate word into EA word
83 /2 dq	ADC eq,db	2,mem=7	Add with carry immediate byte into EA word
11 /r	ADC eq,rq	2,mem=7	Add with carry word register into EA word
13 /r	ADC rq,eq	2,mem=6	Add with carry EA word into word register
01 /r	ADD eq,rq	2,mem=7	Add word register into EA word
03 /r	ADD rq,eq	2,mem=7	Add EA word into word register
05 dq	ADD EAX,dq	2	Add immediate word into EAX
81 /0 dq	ADD eq,dq	2,mem=7	Add immediate word into EA dword
83 /0 db	ADD eq,db	2,mem=7	Add immediate byte into EA dword

4.3.2 AND Logical AND

Opcode	Instruction	Clocks	Description
21 /r	AND eq,rq	2,mem=7	Logical-AND word register into EA dword
23 /r	AND rq,eq	2,mem=7	Logical-AND EA dword into dword register
25 dq	AND EAX,dq	2	Logical-AND immediate dword into EAX
81 /4 dq	AND eq,dq	2,mem=7	Logical-AND immediate dword into EA dword
83 /4 db	AND ew,eb	2,mem=7	AND sign-extended immediate byte with EA word
83 /4 db	AND eq,eb	2,mem=7	AND sign-extended immediate byte with EA dword

4.3.3 BOUND Ceck Array Index Against Bounds

Opcode	Instruction	Clocks	Description
62 /r	BOUND rq,mq	10	INT5 if rw not within bounds

4.3.4 BSF Bit Scan Forward

Opcode	Instruction	Clocks	Description
0F BC	BSF r16,r/m16	11+3n	Bit scan forward on r/m word
0F BC	BSF r32,r/m32	11+3n	Bit scan forward on r/m dword

4.3.5 BSR Bit Scan Reverse

Opcode	Instruction	Clocks	Description
0F BD	BSR r16,r/m16	9+3n	Bit scan reverse on r/m word
0F BD	BSR r32,r/m32	9+3n	Bit scan reverse on r/m dword

4.3.6 BT Bit Test

Opcode	Instruction	Clocks	Description
0F A3	BT r/m16,r16	3/12	Save bit in carry flag
0F A3	BT r/m32,r32	3/12	Save bit in carry flag
0F BA	BT r/m16,imm8	3/6	Save bit in carry flag
0F BA	BT r/m32,imm8	3/6	Save bit in carry flag

4.3.7 BTC Bit Test and Complement

Opcode	Instruction	Clocks	Description
0F BB	BTC r/m16,r16	6/13	Save bit in carry flag and complement
0F BB	BTC r/m32,r32	6/13	Save bit in carry flag and complement
0F BA /7 ib	BTC r/m16,imm8	6/8	Save bit in carry flag and complement
0F BA /7 ib	BTC r/m32,imm8	3/6	Save bit in carry flag and complement

4.3.8 BTR Bit Test and Reset

Opcode	Instruction	Clocks	Description
0F B3	BTR r/m16,r16	6/13	Save bit in carry flag and reset
0F BB	BTR r/m32,r32	6/13	Save bit in carry flag and reset
0F BA /6 ib	BTR r/m16,imm8	6/8	Save bit in carry flag and reset
0F BA /6 ib	BTR r/m32,imm8	3/6	Save bit in carry flag and reset

4.3.9 BTS Bit Test and Set

Opcode	Instruction	Clocks	Description
0F AB	BTS r/m16,r16	6/13	Save bit in carry flag and set
0F AB	BTS r/m32,r32	6/13	Save bit in carry flag and set
0F BA /5 ib	BTS r/m16,imm8	6/8	Save bit in carry flag and set
0F BA /5 ib	BTS r/m32,imm8	3/6	Save bit in carry flag and set

4.3.10 CALL Call Procedure

Opcode	Instruction	Clocks	Description
E8 cw	CALL rel16	7+m	Call near, displacement relative to next instruction
FF /2	CALL r/m16	7+m/10+m	Call near, register indirect/memory indirect
9A cd	CALL prt16:16	17+m,pm=34+m	Call intersegment, to full pointer given
9A cd	CALL ptr16:16	pm=52+m	Call gate, same privilege
9A cd	CALL ptr16:16	pm=86+m	Call gate, more privilege, no parameter
9A cd	CALL ptr16:16	pm=94+4x+m	Call gate, more privilege, x parameter
9A cd	CALL ptr16:16	ts	Call to task
FF /3	CALL m16:16	22+m,pm=38+m	Call intersegment, address at r/m dword
FF /3	CALL m16:16	pm=56+m	Call gate, same privilege
FF /3	CALL m16:16	pm=90+m	Call gate, more privilege, no parameters
FF /3	CALL m16:16	pm=98+4x+m	Call gate, more privilege, x parameters
FF /3	CALL m16:16	5+ts	Call to task
E8 cd	CALL rel32	7+m	Call near, displacement relative to next instruction
FF /2	CALL r/m32	7+m/10+m	Call near, indirect
9A cp	CALL ptr16:32	17+m,pm=34+m	Call intersegment to full pointer given
9A cp	CALL ptr16:32	pm=52+m	Call gate, same privilege
9A cp	CALL ptr16:32	pm=86+m	Call gate, more privilege, no parameters
9A cp	CALL ptr32:32	pm=94+4x+m	Call gate, more privilege, x parameters
9A cp	CALL ptr16:32	ts	Call to task
FF /3	CALL m16:32	22+m,pm=38+m	Call intersegment, address at r/m dword
FF /3	CALL m16:32	pm=56+m	Call gate, same privilege
FF /3	CALL m16:32	pm=90+m	Call gate, more privilege, no parameter
FF /3	CALL m16:32	pm=98+4x+m	Call gate, more privilege, x parameters
FF /3	CALL m16:32	5+ts	Call to task

Die Werte für **ts** sind in folgender Tabelle:

Old Task			Newtask			
	386 TSS		386 TSS		286 TSS	
	VM = 0		VM = 1			
			via task gate?			
	N	Y	N	Y	N	Y
386 VM = 0	392	401	309	321	285	294
286 TSS	310	316	229	238	285	294

4.3.11 CBW/CWDE Convert Byte into Word/Word to Doubleword

Opcode	Instruction	Clocks	Description
98	CBW	3	Convert byte into word (AH = top bit of AL)
98	CBWE	3	Convert word into dword (EAX = top bit of AX)

4.3.12 CMP Compare two Operands

Opcode	Instruction	Clocks	Description
3D dq	CMP EAX,dq	2	Compare immediate dword from EAX.
83 /7 db	CMP eq,db	2,mem=5	Compare immediate byte from EA dword.
81 /7 dq	CMP eq,dq	2,mem=5)	Compare immediate dword from EA dword.
3B /r	CMP rq,rq	2,mem=6	Compare EA dword from dword register.
39 /r	CMP eq,rq	2,mem=5	Compare dword register from EA dword.

4.3.13 CMPS/CMPSB/CMPSW/CMPSD Compare String Operands

Opcode	Instruction	Clocks	Description
A6	CMPS mb,mb	10	Compare bytes ES:[(E)DI] from [(E)SI].
A7	CMPS mw,mw	10	Compare words ES:[(E)DI] from [(E)SI].
A7	CMPS mq,mq	10	Compare dwords ES:[(E)DI] from [(E)SI].
A6	CMPSB	10	Compare bytes ES:[(E)DI] from DS:[SI].
A7	CMPSW	10	Compare words ES:[(E)DI] from DS:[SI].
A7	CMPSD	10	Compare dwords ES:[(E)DI] from DS:[SI].

4.3.14 CDQ Convert Doubleword to Quadword

Opcode	Instruction	Clocks	Description
99	CDQ	2	EDX:EAX←sign-extended of EAX

4.3.15 DEC Decrement by 1

Opcode	Instruction	Clocks	Description
FF /1	DEC r/m32	2/6	Decrement r/m dword by 1
48 +rw	DEC r32	2	Decrement dword register by 1

4.3.16 DIV Unsigned Divide

Opcode	Instruction	Clocks	Description
F7 /6	DIV EAX,r/m32	38/41	Unsigned divide EDX:EAX by r/m dword.

4.4 80486 only

4.4.1 BSWAP Byte Swap

Der BSWAP-Befehl hilft bei der Konvertierung des Datenformats *big-endian* in *little-endian*. Die Intel-Prozessoren speichern Daten, die mehr als ein Byte beanspruchen, mit dem niederwertigeren Byte in der numerisch niedrigeren Adresse. Diese Speicherschema wird als Little-Endian-Format bezeichnet. Motorola Prozessoren, wie die 680X0-Reihe, speichern dagegen Daten im Big-Endian-Format, das heißt das höherwertige Byte unter der numerisch niedrigen Adresse.

4.4.2 CMPXCHG Compare and Exchange

Kombination des CMP- und XCHG-Befehls.

4.4.3 INVD Invalidate Data Cache

Cache-Reset. Markiere Cache Daten als ungültig.

4.4.4 INVLPG Invalidate TLB Entry

Markiere **T**ranslation **L**ookaside **B**uffer Eintrag als ungültig.

4.4.5 WBINVD Write Back and invalidate Data Cache

Cache-Reset intern und extern.

4.4.6 XADD Exchange and Add

Kombination des XCHG- und ADD-Befels. æ

Kapitel 5

Coprocessors

Mit den Prozessoren der 8087 Familie können mathematische Berechnungen in real, packed BCD oder long integers sehr viel schneller ausgeführt werden, als mit den Prozessoren der 8086 Familie selbst.

5.1 Allgemeines

Der 8087 hat acht 80-Bit Datenregister und sieben 16-Bit Controlregister. Die Datenregister sind als Stack angeordnet, d. h. wird ein neuer Wert eingeschrieben, so wird der alte in ein Register mit höherer Bezeichnung geschoben. Die Datenregister können auch einzeln angesprochen werden, **ST0** – **7**, wobei die 0 entfallen kann, da dann immer der "top of stack" angenommen wird. Das 10-Byte temporäre real Format der Datenregister entspricht **IEEE**, wird aber von höheren Sprachen nicht genutzt.

Die Controlregister sind: Control Word, Status Word, Tag Word, 32-Bit Instruction Pointer und 32-Bit Operand Pointer.

Wird die **.8087** Direktive benutzt, wird von MASM automatisch ein WAIT Befehl vor jeder Instruktion eingefügt zur Synchrinisierung des 8087 mit dem 8088 oder 8086. Die 80287 und 80387 benötigen den WAIT Befehl nicht und es sollte die **.286** oder **.386** Direktive angegeben werden, wenn das Programm **nicht** auf einem 8087 laufen soll.

5.1.1 Real-Number Encoding

Bei der Darstellung der Realzahlen herrscht ein geordnetes Durcheinander.

IEEE Format

Wird nichts angegeben, so benutzt der Microsoft Assembler das IEEE-Format. Das ist eine erhebliche Änderung gegenüber den vorhergehenden Versionen von MASM. Das IEEE-Format wird in Tabelle 5.1 und Tabelle 5.2 dargestellt.

Bit	Bedeutung
31	Sign; 0 = positiv
24 – 30	Exponent
0 – 23	Mantisse

Tabelle 5.1: Short real number

Bit	Bedeutung
63	Sign; 0 = positiv
62 – 52	Exponent
51 – 0	Mantisse

Tabelle 5.2: Long real number

Bit	Bedeutung
0 – 22	Mantisse
23	Sign
24 – 31	Exponent; Bias ist 81H

Tabelle 5.3: Short real number

Bit	Bedeutung
0 – 54	Mantisse
55	Sign
56 – 63	Exponent; Bias ist 401H

Tabelle 5.4: Long real number

Microsoft Binary Format

Wird die MASM Direktive **.MSFLOAT** angegeben, wird das Microsoft Binary Format benutzt.

10-Byte IEEE Format

Dieses Format wird auch von Microsoft's MASM unterstützt, aber nicht von den anderen Sprachen, weil es selten vorkommen wird, daß jemand den Coprozessor selbst programmieren möchte und auf die Unterstützung der 4- und 8-Byte Fromate verzichtet. Auffällig ist die separate Darstellung des "integer part".

5.1.2 Control Word

Bit0 IM-Bit, Invalid Operation, 1=Masked, 0=unmasked.

Bit1 DM-Bit, Denormalized Operand,

Bit2 ZM-Bit, Zero Divide,

Bit3 OM-Bit, Overflow,

Bit4 UM-Bit, Underflow,

Bit5 PM-Bit, Precision,

Bit7 IE-Bit, 8087 only, Interrupt enable Bit.

Bit8-9 PC-Bits, Precision control.

- 00 = 24-bit mantisse
- 10 = 53-bit mantisse
- 11 = 64-bit mantisse

Bit10-11 RC-Bits, Rounding control.

- 00 = round to nearest even (default).
- 01 = round down toward -infinity.
- 10 = round up toward +infinity.
- 11 = chop by truncating toward 0.

Bit12 IC-Bit, 0 = projective (default on 8087 and 80287)

Bit12 IC-Bit, 1 = affine, 8087 and 80287 only, 80387 uses affine regardless of setting.

Bit	Bedeutung
0 – 62	Mantisse
63	integer part
64 – 78	Exponent
79	Sign

Tabelle 5.5: 10-Byte IEEE Format

5.1.3 Status Word

Bit0 IE-Bit, Invalid Operation.

Bit1 DE-Bit, Denormalized Operand.

Bit2 ZE-Bit, Zero Divide.

Bit3 OE-Bit, Overflow.

Bit4 UE-Bit, Underflow.

Bit5 PE-Bit, Precision.

Bit6 SF-Bit, stack flag, 80387 only.

Bit7 ES-Bit, error summary on 80287/387, interrupt request on 8087.

Bit8 C0, condition codes.

Bit9 C1

Bit10 C2

Bit11-13 ST-Bits, points to current top of stack.

Bit14 C3

Bit15 B-Bit, busy, 1 = exeption control unit active.

5.2 Daten-Transfer

Die 8087-Familie hat unterschiedliche Befehle für folgende Arten des Datentransfers:

1. Transfer zwischen Memorie und Registern oder unterschiedlichen Registern.
2. Konstanten in Register laden.
3. Transfer von Kontroll Daten von und zum Memorie.

Der Befehl stellt selbst fest, ob der Wert im Memorie eine integer, BCD oder real Zahl ist. Im Coprozessor ist der Wert immer auf die 10-Byte temporäre real Zahl expandiert.

Daten werden zu den Stackregistern mit LOAD Befehlen transferiert. Wird kein Register angegeben, so wird gepushed. STORE Befehle popen den Stack. Manche Befehle ändern den Stack nicht, sondern kopieren nur die Daten.

5.2.1 Real Transfers

FLD mem pushed eine Kopie von mem nach ST. Der Operand muß 4-, 8- oder 10-Byte lang sein. Es wird automatisch in das temporäre real Format überführt.

FLD ST (num) pushed eine Kopie des angegebenen Registers nach ST.

FST mem kopiert ST nach mem ohne den Stack zu ändern.

FST ST(num) kopiert ST in das angegebene Register. Der Wert wird überschrieben.

FSTP mem popped eine Kopie des ST ins mem.

FSTP ST(num) popped ST in das angegebene Register.

FXCH ST(num) Tauscht die Werte in ST mit ST(num). Wenn kein Operand angegeben wird, wird ST(0) mit ST(1) getauscht.

5.2.2 Integer Transfers

FILD mem pushed eine Kopie von mem nach ST. Es muss ein 2-, 4- oder 8-Byte integer memory Operand sein.

FIST mem Kopiert ST nach mem.

FISTP mem popped ST ins mem.

5.2.3 Packed BCD Transfers

FBLD mem pushed eine Kopie von mem nach ST. Es wird nicht geprüft, ob der 10-Byte Wert ein gültiger BCD Wert ist.

FBSTP mem popped ST nach mem. Der Wert wird wenn nötig gerundet und nach BCD Darstellung gewandelt.

5.2.4 Loading Constants

Die folgenden Befehle laden den 'top of stack' mit den angegebenen Konstanten.

FLDZ pushed eine 0 nach ST.

FLD1 pushed eine 1 nach ST.

FLDPI pushed den Wert von π nach ST.

FLDL2E pushed den Wert von $log_2 e$ nach ST.

FLDL2T pushed den Wert von $log_2 10$ nach ST.

FLDLG2 pushed den Wert von $log_{10} 2$ nach ST.

FLDLN2 pushed den Wert von $log_e 2$ nach ST.

5.2.5 Transfering Control Data

Die Store-Befehle, vom Coprozessor zum Memory, sind in zwei Formen vorhanden: Die "Wait-Form" überprüft die Fehler und wartet auf die Fehlerbehandlung. Die "no-wait-Form" beginnt immer mit **FN**.

FLDCW mem2byte loads control word. Memory nach Coprozessor.

F(N)STCW mem2byte stores control word. Coprozessor nach Memory.

F(N)STSW mem2byte stores status word.

FLENV mem14byte loads environment.

F(N)STENV mem14byte stores environment.

FRSTOR mem94byte restores state.

F(N)SAVE mem94byte saves state.

80287/387 only

FSTSW und FNSTSW kann direkt mit Register AX benutzt werden.

80387 only

Im 32-bit Mode werden 32-bit Adressen benutzt. FSAVE benötigt daher 98 Byte (108 nach Reference) und FSTENV 18 Byte (28 nach Reference).

5.3 Arithmetische Befehle

5.3.1 Addition

FADD Addiert ST und ST(1) und popped nach ST. Beide Operanden werden zerstört.

FADD ST(num),ST Addiert die beiden Werte in den Registern und ersetzt ST(num) mit dem Ergebnis.

FADD ST,ST(num) Addiert die beiden Werte in den Registern und ersetzt ST mit dem Ergebnis.

FADD mem Addiert die Realzahl im Memory zu dem Wert in ST.

FIADD mem Addiert die Integerzahl im Memory zu dem Wert in ST.

FADDP ST(num),ST Addiert die beiden Werte in den Registern und popped das Ergebins nach ST(num).

5.3.2 Normal Subtraction

FSUB Subtrahiert ST von ST(1) und popped das Ergebnis in ST. Beide Operanden sind zerstört.

FSUB ST(num),ST Subtrahiert ST von ST(num), das Ergebnis wird in ST(num) abgelegt.

FSUB ST,ST(num) Subtrahiert ST(num) von ST, das Ergebnis geht nach ST.

FSUB mem Subtrahiert die Realzahl im Memory von ST. Das Ergebis ersetzt ST.

FISUB mem Subtrahiert die Integerzahl im Memory von ST. Das Ergebnis ersetzt ST.

FSUBP ST(num),ST Subtrahiert ST von ST(num) und popped das Ergebnis nach ST(num).

5.3.3 Reversed Subtraction

FSUBR Subtrahiert ST(1) von ST und popped das Ergebnis in ST. Beide Operanden sind zerstört.

FSUBR ST(num),ST Subtrahiert ST(num) von ST, das Ergebnis wird in ST(num) abgelegt.

FSUBR ST,ST(num) Subtrahiert ST von ST(num), das Ergebnis geht nach ST.

FSUBR mem Subtrahiert ST von der Realzahl im Memory. Das Ergebis ersetzt ST.

FISUBR mem Subtrahiert ST von der Integerzahl im Memory. Das Ergebnis ersetzt ST.

FSUBRP ST(num),ST Subtrahiert ST(num) von ST und popped das Ergebnis nach ST(num).

5.3.4 Multiplication

FMUL Multipliziert ST mit ST(1) und popped das Ergebnis nach ST.

FMUL ST(num),ST Multipliziert die beiden Werte der Register und ersetzt ST(num) mit dem Ergebnis.

FMUL ST,ST(num) Multipliziert die beiden Werte der Register und ersetzt ST(num) mit dem Ergebnis.

FMUL mem Multipliziert eine Realzahl im Memory mit ST. Das Ergebnis wird in ST abgelegt.

FIMUL mem Multipliziert eine Integerzahl im Memory mit ST. Das Ergebnis wird in ST abgelegt.

FMULP ST(num),ST Multipliziert die beiden Register und popped das Ergebnis nach ST(num).

5.3.5 Normal Division

FDIV dividiert ST(1) mit ST und popped das Ergebnis nach ST.

FDIV ST(num),ST dividiert ST(num) mit ST und ersetzt ST(num) mit dem Ergebnis.

FDIV ST,ST(num) dividiert ST mit ST(num) und ersetzt ST mit dem Ergebnis.

FDIV mem dividiert ST mit der Realzahl im Memory. Das Ergebnis ersetzt ST.

FIDIV mem dividiert ST mit der Integerzahl im Memory. Das Ergebnis ersetzt ST.

FDIVP ST(num),ST dividiert ST(num) mit ST und popped das Ergebnis nach ST(num).

5.3.6 Reversed Division

FDIVR dividiert ST mit ST(1) und popped das Ergebnis nach ST.

FDIVR ST(num),ST dividiert ST mit ST(num) und ersetzt ST(num) mit dem Ergebnis.

FDIVR ST,ST(num) dividiert ST(num) mit ST und ersetzt ST mit dem Ergebnis.

FDIVR mem dividiert die Realzahl im Memory mit ST. Das Ergebnis ersetzt ST.

FIDIVR mem dividiert die Integerzahl im Memory mit ST. Das Ergebnis ersetzt ST.

FDIVRP ST(num),ST dividiert ST mit ST(num) und popped das Ergebnis nach ST(num).

5.3.7 Andere Operationen

Die folgenden Befehle benutzen als Ziel den "top of stack" (ST). Das Ergebis der Operation ersetzt den Wert in ST.

FABS setzt das Vorzeichen von ST positiv.

FCHS dreht das Vorzeichen um.

FRNDINT rundet ST zu einer Integerzahl.

FSQRT erzeugt in ST den Wurzelwert des alten Wertes von ST.

FSCALE addiert den Wert in ST(1) zu dem Exponenten von ST. Es wird also multipliziert mit $2^{ST(1)}$. Weil das Exponentenfeld vom Typ integer ist, muß der Wert in ST(1) auch integer sein. Bei den 8087 und 80287 ist das Ergebnis nicht definiert wenn ST(1) nicht in dem Bereich $-2^{15} \leq X < 2^{15}$ ist oder in dem Bereich 0 ¡ X ¡ 1 ist.

FPREM der Wert von ST wird durch den Wert von ST(1) geteilt. Der Rest geht nach ST. ST(1) wird nicht geändert. Da diese Instruktion durch widerholtes abziehen gelöst wird, kann sie bei großen Unterschieden der Operanden recht lange dauern. Sie wird oft von trigonometrischen Funktionen benutzt. $remainder$ = ST - ST(1) $*$ $quotient$.

C3	C2	C1	C0	Meaning
?	1	?	?	incomplete reduction
0	0	0	0	$quotient$MOD8=0
0	0	0	1	$quotient$MOD8=4
0	0	1	0	$quotient$MOD8=1
0	0	1	1	$quotient$MOD8=5
1	0	0	0	$quotient$MOD8=2
1	0	0	1	$quotient$MOD8=6
1	0	1	0	$quotient$MOD8=3
1	0	1	1	$quotient$MOD8=7

FXTRACT teilt die Zahl in den Exponenten und die Mantisse. Die Mantisse wird gepushed. In ST steht also der Wert der Mantisse und in ST(1) der Exponent ohne Bias.

80387 only

Bei dem 80387 ist der Befehl **FPREM1** zusätzlich vorhanden. Er richtet sich nach IEEE Standard.

5.4 Beispiel Quadratische Gleichung

	.DATA	
a	DD	3.0
b	DD	7.0
c	DD	2.0
posx	DD	0.0
negx	DD	0.0

	.CODE	
fld1		;get constants 2 and 4
fadd	st,st	;2 at bottom
fld	st	;copy it
fmul	a	; = 2a
fmul	st(1),st	;= 4a
fxch		;exchange
fmul	c	;= 4ac
fld	b	;load b
fmul	st,st	;= b^2
fsubr		;= $b^2 - 4ac$
		;kein Test auf negativ kann Fehler bringen
fsqrt		;= $\sqrt{b^2 - 4ac}$
fld	b	;load b
fchs		;make it negative
fxch		;exchange
fld	st	;copy square root
fadd	st,st(2)	;plus version = $-b + \sqrt{b^2 - 4ac}$
fxch		;exchange
fsubp	st(2),st	;minus version = $-b - \sqrt{b^2 - 4ac}$
fdiv	st,st(2)	;divide plus version
fstp	posx	;store it
fdivr		;divide minus version
fstp	negx	;store it

Mit dem Befehl **FTST** kann zu den entsprechenden Fehlerroutinen verzweigt werden.

5.5 Transcendental Instructions

F2XM1 errechnet $2^x - 1$, wobei x der Wert im top of stack ist. Der Wert muß zwischen 0 und .5 einschließlich sein. Der Wert $2^x - 1$ kann mit größerer Genauigkeit errechnet werden als der Wert 2^x.

FYL2X errechnet $Ylog_2X$, wobei X in ST und Y in ST(1) ist. Der Stack wird gepopped, so daß X und Y zerstört werden. Das Ergebnis ist in ST. X muß positiv sein.

FYL2XP1 errechnet $Ylog_2(X+1)$, wobei X in ST und Y in ST(1) ist. Der Stack wird gepopped, so daß X und Y zerstört werden. Das Ergebnis ist in ST. Der absolute Wert von X muß zwischen 0 und $\sqrt{\frac{2}{2}}$ sein. Diese Instruktion ist gegenüber FYL2X genauer, wenn es sich um Werte nahe 1 handelt.

FPTAN errechnet den Tangens des Wertes in ST. Das Ergebnis ist das Verhältnis $\frac{Y}{X}$, wobei Y den Wert in ST ersetzt und X auf den Stack gepushed wird, so daß ST Y und ST(1) X enthält. Der Wert, der errechnet wird, muß kleiner $\frac{\pi}{4}$ sein bei 8087 und 80287. Bei einem 80387 muß der Betrag des Wertes kleiner 2^{63} sein. Das Ergebnis kann zur Berechnung anderer trigonometrischer Funktionen benutzt werden.

FPATAN errechnet den Arcustangens des Verhältnis $\frac{y}{x}$, wobei X in ST und Y in ST(1) ist. Der Stack wird gepopped, wobei X und Y zerstört werden. Das Ergebnis ist in ST. X und Y müssen positive Zahlen unterhalb Unendlich sein. Y muß kleiner X sein. Das Ergebnis kann zur Berechnung anderer trigonometrischer Funktionen benutzt werden.

80387 only.

Die folgenden trigonometrischen Funktionen sind nur mit dem 80387 verfügbar.

FSIN errechnet den Sinus des Radian-Wertes in ST. Das Ergebnis ist in ST wenn der Betrag von ST $<2^{63}$ ist. Andernfalls wird das C2-Bit im Statuswort gesetzt und die Berechnung findet nicht statt.

FCOS errechnet den Cosinus des Radian-Wertes in ST. Das Ergebnis ist in ST wenn der Betrag von ST $<2^{63}$ ist. Andernfalls wird das C2-Bit im Statuswort gesetzt und die Berechnung findet nicht statt.

FSINCOS errechnet den Sinus und den Cosinus des Wertes in ST. Der Cosinus ist in ST und der Sinus in ST(1). Der alte ST(1)-Wert ist dann (pushed) in ST(2). ST kann mit FPREM oder FPREM1 auf den benötigten Betrag reduziert werden.

5.6 Controlling the Coprocessor

Ein N an der zweiten Stelle bezeichnet "no-wait".

F(N)INIT setzt den Coprozessor zurück und die Ausganszustände in den Control- und Statusregistern. Er sollte immer am Anfang und am Ende eines Programms benutzt werden, damit keine Zufallsergebnisse durch alte Zahlen entstehen können.

F(N)CLEX setzt alle Ausnahme flags und die busy flag im Status Wort zurück. Setzt auch die error-status flag des 80287 und 80387 oder dir interrupt-request flag des 8087 zurück.

FINCSTP addiert eins zu dem Stackpointer im Status Wort.

FDECSTP subtrahiert eins von dem Stackpointer im Status Wort.

FREE ST(num) erklärt das angebene Register als leer.

FNOP tut nichts. (Kopiert ST zu sich selbst, verbraucht also nur Rechenzeit).

8087 only.

Der 8087 hat auch noch die Befehle **FDISI, FNDISI, FENI, FNENI**. Diese Befehle erlauben oder verbieten den Interrupt. Das **N** in der zweiten Stelle steht für NOWAIT. Die 287 und 387 erlauben zwar diese Befehle, ignorieren sie aber.

80287/80387 only.

Mit dem Befehl **FSETPM** wird der Coprozessor in den protected mode geschaltet. Dadurch werden die Adressen in den Instruktion und Operand Pointern als Segment Selektor und nicht als Segment Adresse verstanden. Die Direktive **.286P** oder **.386P** muß angegeben worden sein.

5.7 Controlling Program Flow

Der Coprozessor hat einige Instruktionen, die im Status-Register Controll-Flags setzen. Diese müssen in das Memory gebracht werden und können dann mit "conditional jumps" bearbeitet werden. Die 80287 und 80387 erlauben einen direkten Transfer in das Register AX. Die Vergleichs-Befehle benutzen die Controll-Flags C3, C2 und C0. C1 wird nicht benutzt.

nach FCOM	nach FTEST	C3	C2	C0
ST >source	ST is positive	0	0	0
ST <source	ST is negative	0	0	1
ST = source	St is 0	1	0	0
not comparable	ST is NAN[1]	1	1	1

5.7.1 Compare

FCOM compares ST to ST(1)

FCOM ST(num) compares ST to ST(num)

FCOM mem compares ST to mem. Der Memory-Operand kann eine 4- oder 8-Byte Real-Zahl sein.

FICOM mem compares ST to mem. Der Memory-Operand kann eine 2- oder 4-Byte Integer-Zahl sein.

FTST compares ST to 0.

5.7.2 Compare and Pop

FCOMP compares ST to ST(1) and pops ST off the register stack.

FCOMP ST(num) compares ST to ST(num) and pops ST off the register stack.

FCOMP mem compares ST to mem and pops ST off the register stack. Der Memory-Operand kann eine 4- oder 8-Byte Real-Zahl sein.

FICOMP mem compares ST to mem and pops ST off the register stack. Der Memory-Operand kann eine 2- oder 4-Byte Integer-Zahl sein.

FCOMPP compares ST to ST(1) and pops ST an ST(1) off the register stack.

80387 only

Unordered compare instructions are available with the 80387. The **FUCOM, FUCOMP and FUCOMPP** instructions are like **FCOM, FCOMP and FCOMPP** exept that the unordered versions do not cause invalid operation exeptions if one of the operands is a quiet NAN.

5.8 Testing Control Flags after Other Instructions

FXAM setzt den Wert der Control-Flags nach dem Type der Zahl in ST.

C3	C2	C1	C0	Interpretation
0	0	0	0	+unnormal
0	0	0	1	+NAN
0	0	1	0	-unnormal
0	0	1	1	-NAN
0	1	0	0	+normal
0	1	0	1	+infinity
0	1	1	0	-normal
0	1	1	1	-infinity
1	0	0	0	+0
1	0	0	1	empty
1	0	1	0	-0
1	0	1	1	empty
1	1	0	0	+denormal
1	1	0	1	empty
1	1	1	0	-denormal
1	1	1	1	empty

5.9 Weitek 4167 Coprozessor

Der Weitek 4167 ist für 486-Systeme vorgesehen. Der software compatible 3167 wird in 386-Systemen verwendet. Programme für 80386/3167 laufen ohne Änderung auf i486/4167. Unix System V unterstützt den 4167. Auch die MS-DOS Protekted-Mode Umgebungen von Phar Lap, IGC oder AI Architects unterstützen den 4167. Green Hills (Phone 818 246 5555) und Silicon Valley Software (SVC) (Phone 408 725 8890) haben Fortran, Pascal und C Compiler für UNIX. Von Metaware (Phone 408 429 6382) gibt es einen C Compiler für DOS. Von Weitek gibt es mit den Bestellnummern Diagnostics, Macros (4800-xx67-02 (UNIX) -03 (DOS)) und transcendental Libraries für DOS (4800-xx67-04) und UNIX (4800-xx67-01).

5.9.1 Registers

Der 4167 enthält 32 single-precision Register ws0 bis ws31, die auch als Paare wd0 bis wd30 benutzt werden können. Dann wird für das MSW das gerade Register und für das LSW das ungerade Register benutzt. Jedes 486'er Doppelwort Register kann als Daten- oder Operanden-Register benutzt werden. Der 4167 besitzt auch ein 32-bit Processor Context Register (PCR) mit dem die Rundungs- Modes und Ausnahmebehandlungen und die control settings und die status flags behandelt werden können. W0 hat in Memory-to-Memory Instruktionen eine besondere Bedeutung.

5.9.2 Process Context Register

Bit 31-28 Modeselect

Bit 27-24 Modefield

Bit 23-16 Exeption Mask

Bit 15-8 Condition Code

Bit 7-0 Accumulated Exeption

5.9.3 Modeselect

Die Modeselect-Bits werden nur bei der Initialisierung benötigt. Sind sie 1100 beim Laden des Context Registers, werden nur die EM, CC und AE Felder berücksichtigt. Sind sie 0000, werden zusätzlich auch die Rundungsbits gesetzt.

Bit 31-28 Immer 0

5.9.4 Modefield

Mit diesen Bits werden die Rundungsoptionen nach IEEE gesetzt.

Bit 27-26 RND

- 00 = round toward nearest value (RN)
- 01 = round toward zero (RZ)
- 10 = round toward positiv infinity (RP)
- 11 = round toward negativ infinity (RM)

Bit 25 IRND

- 0 = integer rounding based on RND
- 1 = integer rounding always toward zero

Bit 24 Fast Mode (immer 1)

5.9.5 Exeption Mask

Bit 23 DM: data chain exeption mask

Bit 22 UOM: undefined opcode exeption mask

Bit 21 PM: precision exeption mask

Bit 20 UM: underflow exeption mask

Bit 19 OM: overflow exeption mask

Bit 18 ZM: zero divide exeption mask

Bit 17 (immer 1)

Bit 16 IM: invalid operation exeption mask

5.9.6 Condition Code

Bit 15 not defined

Bit 14 Z

Bit 13-11 not defined

Bit 10 C2

Bit 9 not defined

Bit 8 C0

Z	C2	C0	Meaning
1	0	0	equal
0	0	1	less than
0	0	0	greater than
1	1	1	unordered

5.9.7 Accumulated Exeption

Bit 7 DE: data chain exeption

Bit 6 UOE: undefined opcode exeption

Bit 5 PE: precision exeption

Bit 4 UE: underflow exeption

Bit 3 OE: overflow exeption

Bit 2 ZE: zero divide exeption

Bit 1 EE: enable exeption (contains the value of INTR)

Bit 0 IE: invalid operation exeption

5.9.8 Instruction Set

Die meisten 4167-Instruktionen beziehen sich auf zwei 4167-Register oder ein 4167-Register und dem i486-Datenbus. Ein 4167-Befehl hat die Form:

OPCODE source2/destination, soource1

Source1 und Source2/Destination sind immer Operanden-Adressen. Das Ergebis wird immer nach Source2 geladen. Source2/Destionation ist immer eins der 32 internen Register des 4167. Source1 kann entweder ein 4167-Register, eine Konstante oder ein i486-Register sein.

Die folgenden Tabellen zeigen den Instruktionssatz des 4167. Alle Registernamen beginnen mit 'w' gefolgt von 's' für single oder 'd' für double; 'x' bedeutet entwerde s oder d. Die Registernamen enden mit 't' für to oder 'f' für from. Für die meisten Befehle gilt wxt für Destination-Register und wxf für Source-Register.

Data Movement

WFLD	wst,wsf	load: wst=wsf
WFLD	wst,data	load: wst=486 data
WFLD	wdt,wdf	load: wdt=wdf
WFLDCTX	ereg	load: CTX=486 E-register
WFPOP	wst	pop wst from the 486 stack
WFPOP	wdt	pop two double words from the 486 stack to wdt
WFLDSD	wst,addr,count	block move: wst array=486 memory
WFST	ereg,wst	store: 486 E-register $*a$ wst
WFST	ereg,wst,opcode	store: 486 ereg = ereg $*$ wst
WFSTCTX	ereg	store: 486 E-register $*-*$ CTX
WFSTCTX	ereg,opcode	store: 486 ereg - ereg $**$ CTX
WFPUSH	wst	push wst onto the 486 stack
WFPUSH	wdt	push wdt (two double words) onto the 486 stack
WFSTSD	wst,addr,count	block move: 486 memory=wst array
WFSTRL	EAX	store revision level to EAX

Format conversion

WFLOAT	wxt,wsf	convert integer wsf to floating wxt
WFLOAT	wxt,data	convert integer 486 data to floating wxt
WFIX	wst,wxf	convert floating wxf to integer wst
WFIX	wst,data	convert floating (486 data) to integer wst
WFCVT	wst,wdf	convert wdf to wst
WFCVT	wst,data	convert dbl-precision (486 data and wsl) to wst
WFCVT	wdt,wsf	convert wsf to wdt
WFCVT	wdt,data	convert single-precision 486 data to wdt

Four-function Arithmetic

WFADD	wxt,wxf	add: wxt = wxt + wxf
WFADD	wxt,data	add: wxt = wxt + (486 data)
WFSUBR	wxt,wxf	reversed subtract: wxt = wxf - wxt
WFSUBR	wxt,data	reversed subtract: wxt = (486 data) - wxt
WFSUB	wxt,wxf	subtract: wxt = wxt - wxf*
WFSUB	wxt,data	subtract: wxt = wxt - (486 data)*
WFMUL	wxt,wxf	multiply: wxt = wxt x wxf
WFMUL	wxt,data	multiply: wxt = wxt x (486 data)
WFMUL	wxt,wxf	negative multiply: wxt = -wxt x wxf
WFMUL	wxt,data	negative multiply: wxt = -wxt x (486 data)
WFAMUL	wxt,wxf	absolute multiply: wxt = —wxt x wxf—
WFAMUL	wxt,data	absolute multiply: wxt = —wxt x (486 data)—
WFMAC	wst,wsf	multiply and accumulate: ws2 = ws2 + wst x wsf
WFMAC	wst,data	multiply and accumulate: ws2 = ws2 + wst x (486 data)
WFMACD	wst,wsf	multiply and accumulate: wd2 = wd2 + wst x wsf*
WFMACD	wst,data	multiply and accumulate: wd2 = wd2 + wst x (486 data)*
WFMACD	wdt,wdf	multiply and accumulate: wd2 = wd2 x wdf*
WFDIVR	wxt,wxf	reversed divide: wxt = wxf / wxt
WFDFVR	wxt,data	reversed divide: wxt = (486 data) / wxt
WFSORT	wxt,wxf	square root: wxt = sqrt(wxf)*
WFSQRT	wxt,data	square root: wxt = sqrt(data)*

Sign Manipulation

WFNEG	wxt,wxf	negate: wxt = -wxf
WFNEG	wxt,data	negate: wxt = -(486 data)
WFABS	wxt,wxf	absolute value: wxt = —wxf—
WFABS	wxt,data	absolute value: wxt = —486 data—

Compare and Test

WFCMPR	wxt,wxf	reversed compare: set CTX for (wxf - wxt)
WFCMPR	wxt,data	reversed compare: set CTX for (486 data) - wxt
WFCMPRT	wxt,wxf	reversed compare with trap: set CTX flags for (wxf - wxt)
WFCMPRT	wxt,data	reversed compare with trap: set CTX for (486 data) - wxt
WFTST	wxf	test: set CTX flags for
WFTST	data	test: set CTX flags for
WFTST	data,ws1	test: set CTX flags for double precision (486 data, ws1) -0
WFTSTT	wxf	test with trap: set CTX flags for (wxf -0)
WFTSTT	data	test with trap: set CTX flags for (486 data) -0
WFTSTT	data,ws1	test with trap: set CTX flags for (486 data, ws1) - 0

Paging Directives

WFSPAGE	force next wfld/wfst to single precision page
WFDPAGE	force next wfld/wfst to double precision page

Diese Befehle gelten für den 4167 und 3167 aber nicht für den 1167!

5.9.9 Instruction Format

Der 4167 ist ein memory-mapped Device. Die physikalische Adresse reicht von C000 0000 bis C000 FFFF. Eine Adresse in diesem Bereich selektiert den Co-Prozessor, gibt den Ort der Source1 und Source2/Destination und gibt die Instruktion an, die der Co-Prozessor ausführen soll.

Format:

Bit2 bis 6 Source2/Destination

Bit0-1 und 7-9 Source1 (Bit0 = lsb, Bit9 = msb)

Bit10-15 Opcode

Bit16-31 Coprocessor select field

Coprocessor select field Die oberen 16 Bit der physikalischen Adresse geben einen Coprozessor Befehl an. Die oberen Bits müssen in den Breich C000 bis C1FF fallen, um einen 4167-Befehl anzugeben. Um mit zukünftigen Devices kompatible zu sein, soll der Programmierer nur C000 verwenden um einen 4167 Befehl anzugeben.

Opcode field Diese 6 Bits geben den Befehl an.

Operand fields Die jeweils 5 Bit des Source1- und Source2/Destination- Feldes geben das Register an. Wenn Source1 gleich 0 ist, werden die Daten über den Systemdaten Bus transferiert. Wegen der i486 Block-Move Funktion wurde das Source1 Feld geteilt.

Beispiel:

Zwei single-precision Zahlen in den Registern WS1 und WS2 sollen addiert und das Ergebis in WS2 abgelegt werden.

C000 0000 Coprozessor select

0000 Opcode = ADD.S

01 Source1 = WS1

08 Source2/Destination = WS2

$A_{31} \ldots A_0$ = C000 0009

Ein i486 Move-Befehl, der die physikalische Adresse C000 0009 generiert, führt die Floatingpoint Addition aus.

5.9.10 Instruction Encoding

Opcode	Binary	Hex	Base address	Definition
ADD.S	000000	00	0000	single precision add
LOAD.S	000001	01	0400	load SRC1 into SRC2/Dest
MUL.S	000010	02	0800	single precision multiply
STOR.S	000011	03	0C00	transfer SRC2 to 386 data bus
SUB.S	000100	04	1000	single precision subtract
DIV.S	000101	05	1400	single precision divide
MULN.S	000110	06	1800	single precision multiply and negate
FLOAT.S	000111	07	1C00	integer to single floating point
CMPT.S	001000	08	2000	single precision compare SRC1 to SRC2 (trap)
TSTT.S	001001	09	2400	single precision compare with zero
NEG.S	001010	0A	2800	single precision negate
ABS.S	001011	0B	2C00	single precision absolute value
CMP.S	001100	0C	3000	single precision compare SRC1 to SRC2
TST.S	001101	0D	3400	single precision compare with zero
AMUL.S	001110	0E	3800	single precision multiply absolute value
FIX.S	001111	0F	3C00	single precision FP to integer
CVTS.D	010000	10	4000	double to single conversion
CVTD.S	010001	11	4400	single to double conversion
MAC.S	010010	12	4800	multiply/accumulate SP input and SP result
ADD.D	100000	20	8000	double precision add
LOAD.D	100001	21	8400	load SRC1 into SRC2/Dest
MUL.D	100010	22	8800	double precision multiply
STOR.D	100011	23	8C00	transfer SRC2 to 386 data bus
SUB.D	100100	24	9000	double precision subtract
DIV.D	100101	25	9400	single precision divide
MULN.D	100110	26	9800	single precision multiply and negate
FLOAT.D	100111	27	9C00	integer to double floating point
CMPT.D	101000	28	A000	double precision compare SRC1 to SRC2 (trap)
TSTT.D	101001	29	A400	single precision compare with zero (tarp)
NEG.D	101010	2A	A800	double precision negate
ABS.D	101011	2B	AC00	double precision absolute value
CMP.D	101100	2C	B000	doube precision compare SRC1 to SRC2
TST.D	101101	2D	B400	single precision compare with zero
AMUL.D	101110	2E	B800	double precision multiply absolute value
FIX.D	101111	2F	BC00	double precision FP to integer
LDCTX	110000	30	C000	load 386 data bus into PCR
STCTX	110001	31	C400	transfer PCR to 386 data bus
MACD.S	110010	32	C800	multiply/accumulate SP input and DP result

5.9.11 Beispiele für UNIX m4 macro

Definitionen:

```
define  (WTL, 0xC0000000)

define  (ADDS, 0x00)           s. Tabelle oben!
```


Mit den Operationen shift left (≪), OR (|), AND (&) und divide (/) werden die Instruktionsfelder des Intruktionswortes richtig gefüllt. Für eine single-precision Addition: wfadd (1, 2)

```
define  (wfadd, ‘movb %al, WTL |[ADDS<<10]|[$1/4<<7]|[$2<<2]|[$1&0x3]’)
```

WTL und ADDS sind oben definiert. ADDS wird mit shift left auf Bit10 positioniert. Das erste Argument wird durch 4 geteilt und das Ergebnis nach Bit7 gebracht. Das zweite Argument wird um 2 Bit nach links geschoben. Aus dem ersten Argument werden die rechten beiden Bits ausmaskiert. Alle Teile werden zu der Adresse WTL mit OR eingebracht.

Double precision memory-to-register Operationen müssen speziell behandelt werden. Zuerst muß die niederwertige Hälfte des Operanden vom 386-Register mit der LOAD Instruktion in den Weitek mit R1 als destination und R0 als source gebracht werden. Danach wird die double precision memory-toregister Operation ausgeführt. Das folgende Beispiel zeigt eine Multiplikation:

```
movl    LEAST, %eax     ;MOVE LSW of memory operand to register EAX

wdfld   (0, 1)          ;LOAD EAX into WTL register R1

movl    MOST, %eax      ;MOVE MSW of memory operand to register EAX

wdfmul  (0, 4)          ;MULTIPLY operand in registers (R1, R0) by operand in

                        ;registers (R4, R5) and store result in registers (R4,

R5)
```

Dabei wurde wdfld (0, 1) definiert als:

```
define  (LOADD, 0x21)

define  (wdfld, ‘movl %eax, WTL|[LOADD<<10]|[$1/4<<7]|[$2<<2]|[$1&0x3]‘)
```

Und wdfmul (0, 4) wurde definiert als:

```
define  (MULD, 0x22)

define  (wdfmul, ‘movl %eax, WTL|[MULD<<10]|[$1/4<<7]|[$2<<2]|[$1&0x3]‘)
```

Das folgende Beispiel kopiert zweiundreizig 32-Bit Worte aus einem Memoryarray mit der Startadresse VECTOR in die 32 Register des Weitek angefangen bei R0 bis zum R31.

```
smovl:

movl    32, %ecx        ;Load the array length in ECX

movl    VECTOR, %esi    ;load the address of the first element of the array

                        ;in source pointer ESI

movl    STADR, %edi     ;load destination pointer EDI with address specifying

                        ;a floating point LOAD to register R0

rep     smovl           ;execute vector transfer
```

Dabei wurden folgende Definitionen benutzt:

```
define  (LOADS, 0x01)

define  (STADR, ‘WTL|[LOADS<<10]|[$1/4<<7]|[$2<<2]|[$1&0x3]’)
```

Beispiel einer 4X4- Matrix Transformation. Die Matrixkoeffizienten sind in den Weitekregistern R16 - R31. x, y, z und w sind die Werte in den Adressen X, Y, Z und W. Die Ergebnisse werden auch dorthin zurückgeschrieben.

$$[xyzw]\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = [x'y'z'w']$$

```
movl    X, %eax         ;load x into 386 eax register

wfld    (0, 4)          ;load x into R4

movl    Y, %eax         ;load y into 386 eax register

wfld    (0, 5)          ;load y into R5

movl    Z, %eax         ;load z into 386 eax register

wfld    (0, 6)          ;load z into R6

movl    W, %eax         ;load w into 386 eax register

wfld    (0, 7)          ;load w into R7

wfld    (16, 2)         ;move a11 into R2

wfmul   (4, 2)          ;R2 = a11 * x
```



```
wfmac   (5, 17)         ;R2 = a11 * x + a21 * y

wfmac   (6, 18)         ;R2 = a11 * x + a21 * y + a31 * z

wfmac   (7, 19)         ;R2 = a11 * x + a21 * y + a31 * z + a41 * w

wfst    ( 2)            ;store x’

movl    %eax, X         ;store x’ into memory location X

wfld    (20, 2)         ;move a12 into R2

wfmul   (4, 2)          ;R2 = a12 * x

wfmac   (5, 21)         ;R2 = a12 * x + a22 * y

wfmac   (6, 22)         ;R2 = a12 * x + a22 * y + a32 * z

wfmac   (7, 23)         ;R2 = a12 * x + a22 * y + a32 * z + a42 * w

wfst    ( 2)            ;store y’

movl    %eax, Y         ;store y’ into memory location Y

wfld    (24, 2)         ;move a13 into R2

wfmul   (4, 2)          ;R2 = a13 * x

wfmac   (5, 25)         ;R2 = a13 * x + a23 * y

wfmac   (6, 26)         ;R2 = a13 * x + a23 * y + a33 * z

wfmac   (7, 27)         ;R2 = a13 * x + a23 * y + a33 * z + a43 * w

wfst    ( 2)            ;store z’

movl    %eax, Z         ;store z’ into memory location Z

wfld    (28, 2)         ;move a14 into R2

wfmul   (4, 2)          ;R2 = a14 * x

wfmac   (5, 29)         ;R2 = a14 * x + a24 * y

wfmac   (6, 30)         ;R2 = a14 * x + a24 * y + a34 * z

wfmac   (7, 31)         ;R2 = a14 * x + a24 * y + a34 * z + a44 * w

wfst    ( 2)            ;store w’

movl    %eax, W         ;store w’ into memory location W
```

5.9.12 Exeption Handling

5.10 80860 RISC Processor

Zur Zeit sind diese Prozessoren als 860XP und 860XR vorhanden.

Der Prozessor enthält eine 32-Bit Ganzzahl-Arithmetik mit paging und Busverwaltung, zwei getrennte Cache-Speicher für Daten und Instruktionen und eine Gleitkommaeinheit mit Kontrollogik, Addierer, Multiplizierer und dreidimensionaler Graphik. Daten- und Instruktionsbus sind getrennt (Harvard Architektur). Der Datenbus hat nach außen eine Breite von 64 Bit, intern eine Breite von 32 Bit. Ganzzahl-Einheit, Gleitkomma-Multiplizierer und -Addierer arbeiten parallel, es können also drei Befehle gleichzeitig ausgeführt werden. Bei 40 MHz Takt werden 120 MOPS, über 100 MIPS und 21 Linpak MFLOPS erreicht.

Die Ganzzahlverarbeitung verfügt über ein vierstufiges Befehlsfließband. Ein fünftoriges Gleitkomma-Register ermöglicht es, Lade- und Speicherinstruktionen parallel zu Gleitkomma-Befehlen abzuarbeiten. Der Gleitkomma-Addierer kann pro Zyklus ein 32- oder 64-Bit Ergebnis errechnen, der Multiplizierer abwechselnd pro Zyklus je ein 32-Bit, dann ein 64-Bit Ergebnis. Der Date-Cache ist 8 KByte, der Befehls-Cache 4 KByte groß. Sie sind durch 32-Bit Leitungen verbunden. Die Paging-Verwaltung besteht aus einer assoziativen Vierweg-Seitentabelle mit 64 Einträgen.

Die vergleichsweise winzige Grafikeinheit kann bis zu 21 Millionen verschieden beleuchtete Pixel pro Sekunde mit Beseitigung verdeckter Oberflächen erzeugen, indem sie den Gleitkomma-Datenpfad mitbenutzt.

5.11 80860-Befehlssatz

5.11.1 RISC-Kern

Load and store instruction

id.x	load integer
st.x	store integer
fld.y	fp load
pfld.z	pipelined fp load
fst.y	fp store
pst.d	pixel store

Register-to-register moves

ixfr	integer to fp register
fxfr	fp to integer register

Integer arithmetic instructions

ddu	add unsigned
adds	add signed
subu	subtract unsigned
subs	subtract signed

Shift instructions

shl	shift left
shr	shift right
shra	shift right arithmetic
shrd	shift right double

Logical instructions

and	logical and
andh	logical and high
andnot	logical and not
andnoth	logical and not high
or	logical or
orh	logical or high
xor	logical exclusive or
xorh	logical exclusive or high

Control-transfer instructions

trap	software trap
intovr	software trap on integer overflow
br	branch direct
bri	branch indirect
bc	branch on cc
bc.t	branch on cc taken
bnc	branch on not cc
bnc.t	branch on noy cc taken
bte	branch if equal
btne	branch if not equal
bla	branch on lcc and add
call	subroutine call
calli	indirect subroutine call

System-control instructions

flush	cache flush
ld.c	load from control register
st.c	store to control register
lock	begin interlocked sequence
unlock	end interlocked sequence

5.11.2 FPU

Floating-point multiplier instructions

fmul.p	fp multiply
pfmul.p	pipelined fp multiply
pfmul3.dd	three-stage pipelined fp multiply
fmlow.p	fp multiply low
frcp.p	fp reciprocal
frsqr.p	fp reciprocal square root

Floating-point adder instructions

fadd.p	fp add
pfadd.p	pipelined fp add
fsub.p	fp subtract
pfsub.p	pipelined fp subtract
pfgt.p	pipelined fp greater than compare
pfeq.p	pipelined fp equal compare
fix.p	fp-to-integer conversion
pfle.p	pipelined fp less than or equal to
famou.p	fp adder move
pfamou.p	pipelined fp adder move
pfix.p	pipelined fp-to-integer conversion
ftrunc.p	fp-to-integer truncation
pftrunc.p	pipelined fp-to-integer truncation

Dual-operation instructions

fam.p	pipelined fp add and multiply
pfsm.p	pipelined fp subtract and multiply
pfmam	pipelined fp multiply with add
pfmsm	pipelined fp multiply with subtract

Long-integer instructions

fisub.z	long-integer subtract
pfisub.z	pipelined long-integer subtract
fiadd.z	long-integer add
pfiadd.z	pipeliend long-integer add

Graphics instructions

fzchks	16-bit z-buffer check
pfzchks	pipelined 16-bit z-buffer check
fzchkl	32-bit z-buffer check
pfzchkl	pipelined 32-bit z-buffer check
faddp	add with pixel merge
pfaddp	pipelined add with pixel merge
faddz	add with z merge
pfaddz	pipelined add with z merge
form	or with merge register
pform	pipelined or with merge register

Assembler pseudo-operation

mov	integer register-to-register move
fmov.q	fp register-register move
pfmov.q	pipelined fp register-register move
nop	core no operation
fnop	fp no operation
pfle.p	pipelined fp less than or equal to

5.12 80960 RISC Processor

FPU, MMU, CACHE, Interrupt-Controller integriert. 66 MIPS, 32 Bit. æ

Kapitel 6

Input/Output

Die Register der peripheren Geräte werden über den Adressbus selectiert und mit den Leitungen IOW oder IOR beschrieben oder gelesen. Periphere Geräte sind nicht nur die Floppy sondern auch der Interrupt Controller. Der vorbelegte Adressraum für eine Port-Adresse geht von 0H bis 3FFH (s. S. 64 Hardware Manual). Adressen über den Port-Bereich hinaus können mit den Befehlen IN oder OUT in Verbindung mit dem Inhalt von DX, also 64kB, erreicht werden. Keine Adresse darf in einer Anwendung mehrfach vergeben werden. Das gilt auch für die IRQ-Leitungen. Die Steckerbelegung ist auf Seite 66 und 67, die Zuordnung zu den Geräten auf Seite 55 im Hardware Manual zu sehen. In der Regel sind die Geräte mit der kurzen (8-Bit) Steckerleiste ausgeführt, d. h. die Zahl der möglichen zusätzlichen Interrupts ist klein weil meistens ein serielles und ein paralleles Port schon vergeben sind. Allerdings ist auf dem Platz des IRQ2 des XT der IRQ9 gelegt, sodaß Geräte die eine Belegung des IRQ2 zulassen benutzt werden können, wenn die Software eine Selektion des IRQ2 zulässt. Das MS-DOS weist dem Hardware Interrupt Request 9 den Vector des IRQ2 zu (revector).

Für die Kommunikation mit den peripheren Geräten stehen zwei Arten der Programmierung zur Verfügung, fixed oder variable Port. Fixed Port ist ein Byte ≤ 255, z. B. auch "COM1", mit $A_{15} - A_8$ der Adresse gleich null. Die "variable port" Zahl steht in DX und kann also bis zu 64kB groß werden, z. B. in einer Loop. Die Richtung ergibt sich aus dem Befehl, IN oder OUT. Mit X oder L als Extension wird Word oder Byte angegeben. H, das ungerade Byte, ist nicht vorgesehen. Ist bei einem Wort-Befehl das Port ungerade, so wird ein zusätzlicher Buszyclus benötigt, da die Wort-Grenze überschritten wird. Bei DEC würde ein ODD-ADDRESS-TRAP erzeugt werden.

6.1 I/O Address Map

Achtung! Die meisten Karten kodieren nur **zehn** Bit als I/O-Register aus. Deshalb werden sie alle 400h Portadressen gespiegelt. Manche Motherboards erzeugen nicht mehr als zehn Bit Portadressen. Mit dem Debugger kann das nachgeprüft werden. Mit dem Befehl i und der Portadresse wird der Inhalt angezeigt. Durch Vergleich mehrerer aufeinander folgender Ports kann die Spiegelung mit großer Wahrscheinlichkeit nachgewiesen werden.

Es folgt eine Liste der Adressen in Hex der vordefinierten Device. Von den 32 für den Interrupt-Controller reservierten Adressen werden nur die ersten beiden benutzt. Die Adressen 000H bis 0FFH sind für Systemboard I/O reserviert. Die Adressen 100H bis 3FFH stehen am I/O channel zur Verfügung. Diese Liste müßte ständig ergänzt werden. Die COM1,COM2-Karte ME-91 von Meilhaus hat zum Umschalten die Adresse 300 belegt die aber in den vom NI5010 DECnet-Kontroler benutzten Bereich fällt.

000-01F DMA Controller 1

- 000— DMA Ch. 0 Address Register Port Address

- 001— DMA Ch. 0 Wordcount Register Port Address
- 002— DMA Ch. 1 Address Register Port Address
- 003— DMA Ch. 1 Wordcount Register Port Address
- 004— DMA Ch. 2 Address Register Port Address (low 8 bit und next 8 bit) (Diskette)
- 005— DMA Ch. 2 Wordcount Register Port Address (max 64k -1) (Diskette)
- 006— DMA Ch. 3 Address Register Port Address
- 007— DMA Ch. 3 Wordcount Register Port Address
- 008— DMA Status Register Port Address
  - write command
    * Bit0=1 DMA freigeben
    * Bit1=1 Adresse für Kanal 0
    * Bit2=1 DMA-Controller sperren
    * Bit3=1 komprimiertes Timing verwenden
    * Bit4=1 rotierende Priorität
    * Bit5=1 erweiterte Schreibauswahl
    * Bit6 Bedeutung des DREQ-Signals 0 = active low; 1 = active high.
    * Bit7 Bedeutung des DACK-Signals 0 = active low; 1 = active high.
  - read status
  - Bit0=1 Kanal 0 Terminal count
  - Bit1=1 Kanal 1 Terminal count
  - Bit2=1 Kanal 2 Terminal count
  - Bit3=1 Kanal 3 Terminal count
  - Bit4=1 Kanal 0 request
  - Bit5=1 Kanal 1 request
  - Bit6=1 Kanal 2 request
  - Bit7=1 Kanal 3 request
- 009— Anforderungsregister
  - Bit00-01 Anforderungsbit des Kanals (0-3) auswählen
  - Bit02 Anforderungsbit-Bit löschen =0 oder setzen=1
- 00A— DMA Init Port (mask-register)
  - Bit00-01 Maskierungsbit für Kanäle 0-3 auswählen
    * 00 channel 0 select
    * 01 channel 1 select
    * 10 channel 2 select
    * 11 channel 3 select
  - Bit02 Maskierungsbit löschen=0 oder setzen=1
  - Bit 7 - 3 = 0 reserved
- 00B—(write) mode byte
  - Bit00-01 Kanal 0-3 auswählen
    * 00 channel 0 select
    * 01 channel 1 select
    * 10 channel 2 select
    * 11 channel 3 select
  - Bit02-03 Funktion auswählen

    * 00 Überprüfung (verify)
    * 01 schreiben
    * 10 lesen
    * 11 nicht verwendet (illegal)
  - Bit04 Auto-Initialisierung freigeben=1
  - Bit05 Adressen inkrementieren=0 oder dekrementieren=1
  - Bit06-07 Modus
    * 00 nachfragen
    * 01 einzeln
    * 10 block
    * 11 kaskadieren
- 00B—(read) DMA Status port
- 00C—(write) clear byte pointer flip-flop (slave)
- 00C— first/last f/f (DMA Status port)
  - 046— DMA read opcode
  - 04A— DMA write opcode
- 00D— read status register/write command register
  - 06H— master clear
- 00E— DMA channel 0 - 3 clear mask register
- 00F— Masken-Schreib-Register
  - Bit 0 = channel 0 mask bit
  - Bit 1 = channel 1 mask bit
  - Bit 2 = channel 2 mask bit
  - Bit 3 = channel 3 mask bit
  - Bits 4 - 7 = 0 reserved
- 019— DMA scratch register

020-03F INT Controller 1, programmable interrupt controller (PIC)

20 operation control register (ICW1, OCW2, OCW3)

- 20 write initialization command word (ICW1)
  - Bit0 = 1 ICW4 needed
  - Bit1 = 0 cascade mode
  - Bit1 = 1 single mode — no ICW3 needed
  - Bit2 = 0 successive interrupt vectors are separated by eight bytes
  - Bit2 = 1 successive interrupt vectors are separated by four bytes
  - Bit3 = 0
  - Bit4 = 1
  - Bit7-5 = 0 only used in 80/85 mode
- 20 write operation command word (OCW3)
  - Bit1-0 =
    * 00 = no operation
    * 01 = no operation
    * 10 = read IRR on next read port 20h
    * 11 = read ISR on next read port 20h

  - Bit2 = 0 no poll command
  - Bit3 = 1
  - Bit4 = 0
  - Bit6-5 =
    * 00 = no operation
    * 01 = no operation
    * 10 = reset special mask
    * 11 = set special mask
  - Bit7 = 0
- 20 read interrupt request/in-service register programmed by OCW3
  - IRR Bits7 — 0 = 1 active request for the corresponding interrupt line
  - ISR Bits7 — 0 = 1 the corresponding interrupt line is currently being serviced

21 interrupt mask register (ICW2, ICW3, ICW4, OCW1)

- 21 write
  - ICW2: Bits 7 — 3 = Addresslines A0 - A3 of base vector adress for interrupt controller. Bits 2 — 0 = 0
  - ICW3: Bits 7 — 0 = 1 slave controller attached to corresponding interrupt pin.
  - ICW4:
    * Bit0 = 0 80/85 mode
    * Bit0 = 1 8086/8088 mode
    * Bit1 = 0 normal EOI
    * Bit1 = 1 auto EOI
    * Bit3-2 = 2h bufferd mode/slave
    * Bit3-2 = 3h bufferd mode/master
    * Bit4 = 1 special fully-nested mode
    * Bit7-5 = 0
- 21 read/write (OCW1)
  - Bit7 = 0 enable parallel printer interrupt
  - Bit6 = 0 enable diskette interrupt
  - Bit5 = 0 enable fixed disk interrupt
  - Bit4 = 0 enable serial port 1 interrupt
  - Bit3 = 0 enable serial port 2 interrupt
  - Bit2 = 0 enable video interrupt
  - Bit1 = 0 enable keyboard interrupt
  - Bit0 = 0 enable timer interrupt
- 21 write (OCW2)
  - Bit7 — 5 =
    * 000 = rotate in automatic EOI mode (clear)
    * 001 = non-specific EOI
    * 010 = no operation
    * 011 = specific EOI
    * 100 = rotate in automatic EOI mode (set)
    * 101 = rotate on non-specific EOI command
    * 110 = set priority command
    * 111 = rotate on specific EOI command

  - Bit4 = 0
  - Bit3 = 0
  - Bit2-0 = interrupt request to which the command applies

022—03F reserved for use by Intel 8259 PIC

022— NEAT Registeraddressport

023— NEAT read/write port

24 NEAT Datenregister Headland (HTK320), UMC (82C480A), PCCHIP (CHIP5/6)

28 Neat Indexregister Headland (HTK320)

040-05F Timer

- 40 channel 0 (time of day)
- 40 Counter Status (read after Read-Back Command)
  - Bit7 = 0 = OUT Pin is low; = 1 = OUT Pin is high.
  - Bit6 = 1 control register contents are moved into CE
  - Bit5-4 = Read/Load Control
    * 00 = counter latch command
    * 01 = read/load least significant byte
    * 10 = read/load most significant byte
    * 11 = read/load LSB then MSB
  - Bit3-1 mode control
    * 000 = interrupt on terminal count (TC)
    * 001 = hardware retriggerable one-shot
    * 010 = rate generator
    * 011 = square wave generator
    * 100 = software triggered strobe
    * 101 = hardware triggered strobe
  - Bit0 = 0 = 16-bit binary countdown; = 1 = 4-decade BCD countdown.
- 41 dynamic-memory refresh
  - 12H— start timer
- 42 channel 2 (speaker)
- 43 operation control register (write)
  - Bit0 =0 16-Bit-Binärzähler, =1 BCD-Zähler.
  - Bit1-3 = Modus 0 bis 5. 0 unterbrechung bei Zählerüberlauf; 1 progammierbarer Pulsgenerator; 2 Taktgenerator; 3 Rechtecksignalgenerator; 4 software-gesteuerte Auslösung; 5 hardware-gesteuerte Auslösung.
  - Bit4-5 = 00 Zählerstand zum Lesen übernehmen; 01 nur MSB lesen/laden; 10 nur LSB lesen/laden; 11 erst LSB, dann MSB lesen/laden.
  - Bit6-7 = Zähler 0 bis 2 (40-42). 3 is reserved for Read-Back.
- 43 Timer status (read)
  - Bit7-6 = select counter; 11 = reserved for Read-Back.
  - Bit5-4 = 00 = counter latch command – specified by these bits being set to 0.
- 43 Read Back Command
  - Bit7-6 = 11 = counter read back command – specified by these bits being set to 0.

  - Bit5 = 0 = latch count of selected counters
  - Bit4 = 0 = latch status of selected counters
  - Bit3 = 1 = select counter 2
  - Bit2 = 1 = select counter 1
  - Bit1 = 1 = select counter 0
- 44 Timer micellaneous
- 47 control word register for counter 0
- 48 Fail-safe Timer (Counter 0) Timer 2
- 49 Reserved
- 4A Counter 2
- 4B Control word register (see 43)

060-06F Keyboard

- 60 8042 Keyboard scan code/control port (data in/out bei AT)

  read scan code

  write control port
    * DDH Adressleitung 20 sperren
    * DFH Adressleitung 20 freigeben
- 61 Port B read/write diagnostic register
  - Bit 7 = 1 parity error detected
  - Bit 6 = 1 an expansion board is asserting IOCHK (NMI request).
  - Bit 5 = 1 timer 2 output ON
  - Bit 4 = 1 toggle with each refresh request
  - Bit 3 = 0 channel check enabled
  - Bit 2 = 0 parity check enabled
  - Bit 1 = 1 speaker data enabled
  - Bit 0 = 1 timer 1 Ch 2 gate to speaker enabled
- 64 status register (read)
  - Bit 7 Parity Error — 0 = odd, 1 = even, normal is odd
  - Bit 6 Receive time out (general time out)
  - Bit 5 Transmit time out (Auxiliary output buffer full)
  - Bit 4 Inhibit switch
  - Bit 3 Command/Data
  - Bit 2 System flag
  - Bit 1 Input buffer full
  - Bit 0 Output buffer full
- 64 command register (write)
  - 20—3F read commands
  - 60—7F write commands
  - die unteren 5 Bits geben die RAM Adresse an!

44 - 47 Watchdog Timer bei PS/2

60 - 63 PC PIO 8255

64 - 7F Redundante dekodierung des 8255 beim PC

66 reset beim PC 512

070-07F Real-Time Clock, NMI Mask

- 070— I/O Address of CMOS Address Port
  - Bit 7 = 1 NMI disable
  - Bit 7 = 0 NMI enable
  - Bit 6 - 0 = CMOS RAM address
- 071— I/O Address of CMOS Data Port
  - 00—0D Time Portion of RTC
  - 0E—3F RAM Portion of RTC
  - 40—4F Additional Standby RAM
  - 50—68 reserved
  - 69—7F Chip Select Control Register
- 73
- 75
- 77
- 78 auxiliary control port des intelboards 303
  - Bit7 = 1 = force bad parity
  - Bit6 = 0 = force cache misses
  - Bit5 = 1 = video BIOS shadow
  - Bit4 = 1 = system BIOS shadow
  - Bit3 = reserved
  - Bit2 = 1 = A20 enabled
  - Bit1 = 1 = AT32 I/O enabled
  - Bit0 = reserved
- 79 auxiliary control port des intelboards 303
  - bit7 = 1 = above 16M enabled
  - bit6 = 0 = enable speaker
  - bit5 = 1 = slow CPU speed
  - bit4 = 0 = 512 - 640 K enabled
  - bit3 = 0 = diagnostic error
  - bit2 = reserved
  - bit1 = reserved
  - bit0 = reserved
- 7B
- 7D
- 7F

78 - 7A Maus beim PC 512

080-09F DMA Page Register

- 80 diagnostic-checkpoint (vgl. beep codes)
- 81 DMA Channel 2 (page register) (top 4 bit of 20 bit address)
- 82 DMA Channel 3 (used by BIOS)
- 83 DMA Channel 1

- 87 DMA Channel 0
- 89 DMA Channel 6
- 8A DMA Channel 7
- 8B DMA Channel 5
- 8F DMA Channel 4, cascade to channels 5 - 7 (refresh cycle page register)

90 Arbitration Register beim PS/2

91 Card feedback beim PS/2

092 PS/2-kompatibles "RESET"-port.

94 System setup beim PS/2

96 Adapter setup beim PS/2

0A0-0BF INT Controller 2

- A0 operation control register
- A1 interrupt mask register
  - Bit 7 = 0 reserved
  - Bit 6 = 0 enable fixed disk interrupt
  - Bit 5 = 0 enable coprocessor exception interrupt
  - Bit 4 = 0 enable mouse interrupt
  - Bit 3 = 0 reserved
  - Bit 2 = 0 reserved
  - Bit 1 = 0 enable redirect cascade
  - Bit 0 = 0 enable real time clock interrupt

0C0-0DF DMA Controller 2

- 0C0 CH0 base and current address
- 0C2 CH0 base and current word count
- 0C4 CH1 base and current address
- 0C6 CH1 base and current word count
- 0C8 CH2 base and current address
- 0CA CH2 base and current word count
- 0CC CH3 base and current address
- 0CE CH3 base and current word count
- 0D0 read status register/write command register
  - Bit 7 = 1 channel 7 request
  - Bit 6 = 1 channel 6 request
  - Bit 5 = 1 channel 5 request
  - Bit 4 = 1 channel 4 request
  - Bit 3 = 1 terminal count on channel 7
  - Bit 2 = 1 terminal count on channel 6
  - Bit 1 = 1 terminal count on channel 5
  - Bit 0 = 1 terminal count on channel 4
- 0D2 write request register

- 0D4 write single mask register
  - Bits 7 - 3 = 0 reserved
  - Bit 2 = 0 clear mask bit
  - Bit 2 = 1 set mask bit
  - Bits 1 - 0 = select channel
    - 00 = select channel 4
    - 01 = select channel 5
    - 10 = select channel 6
    - 11 = select channel 7
- 0D6 write mode register
- 0D8 clear byte pointer flip-flop
- 0DA read temporary register/write master clear
- 0DC clear mask register
- 0DE write all mask register bits

E0-E1 NEAT Index- und Daten-Register des ACT (A27C300)

0F0— Clear Arithmetic Processor Busy with data=0

0F1— Reset Arithmetic Processor with data=0

0F8-0FF Arithmetic Processor; werden vom 286 automatisch benutzt!

100 - 107 POS beim PS/2

130-133 SCSI-Adapter secondary (AHA-1542)

134-137 SCSI-Adapter secondary (AHA-1542)

170-177 Fixed Disk 1 (alternate)

- 170 fixed disk 1 data register
- 171 fixed disk 1 error register
- 172 fixed disk 1 sector count
- 173 fixed disk 1 sector number
- 174 fixed disk 1 cylinder low
- 175 fixed disk 1 cylinder high
- 176 fixed disk 1 drive/head register
- 177 fixed disk 1 status register

1CE-1CF (ATIVGA)

1ED Kontroll-Register des GCK113 (Headland)

Bit	Bedeutung
0-2	control register select
3-7	reseved

1EF Port-Register (Data I/O)

Kontrol-Register 0

Bit	Bedeutung
0	EMS context select
1	EMS enable
2	remap 384KByte
3	enable E0000 shadow
4	enable F0000 shadow
5-6	number of banks installed
7	RAM type (256/1M)

Kontrol-Register 1

Bit	Bedeutung
0-2	RAS active timeout
3-5	RAS precharge
6	mixed DRAM indicator
7	page mode enable

Kontrol-Register 2

Bit	Bedeutung
0-1	CAS active timeout
2-3	CAS precharge time
4	0 wait in page mode
5	0 wait in non page mode
6-7	CAS active to data ready

Kontrol-Register 3

Bit	Bedeutung
0-7	Speichergröße modulo 64K

Kontrol-Register 4

Bit	Bedeutung
0	reserved
1	page mode interleave on
2-7	version indication

1EE Map Address Register

Bit	Bedeutung
0-4	page selection
5	context selection
6	page write protection
7	autoincrement enable

1EC Map Register

Bit	Bedeutung
0-6	A16-A21 als Offset
7-8	bank select
9	map enable

1F0-1F8 Fixed Disk 0

- 1F0— read/write Data
- 1F1— get error register/set precompensation cylinder
  - 01H data address mark not found
  - 02H track 0 not found on recal
  - 04H aborted command

  - 10H id not found
  - 40H error data ecc
  - 80H error bad block
- 1F2— get/set sector count
- 1F3— get/set sector number
- 1F4— get/set cylinder low
- 1F5— get/set cylinder high (2 Bits)
- 1F6— get/set size/drive/head
- 1F7— get status register/set command register

Status:
  - 01H status error
  - 02H status index
  - 04H ecc correction successful
  - 08H status drq
  - 10H seek complete
  - 20H write fault
  - 40H ready
  - 80H busy

command:
  - 01H cmd modifier no retries
  - 02H cmd modifier ecc mode
  - 08H cmd modifier buffer mode
  - 10H recal
  - 20H read
  - 30H write
  - 40H verify
  - 50H format track
  - 60H initialize
  - 70H seek
  - 90H diagnostic
  - 91H set drive parameters
- 1F8—

200-207 Game I/O (201 control) (alternativ DEPCA - Ethernet card)

208-209 Neatemm port

210-213 Bus Erweiterung

214-216 Empfänger Karte

233-234 SCSI-Adapter secondary (AHA-1542)

234-237 SCSI-Adapter secondary (AHA-1542)

238-23F Serial Port 4 IRQ5 bei einer 4-fach Karte.

240— Data Translation card 16Bit

240-24F DSP Technology CAMAC Interface card

IGC 250, 252, 254, 256, 258, 25A

278-27F Parallel Printer Port 2

- 278— output port
- 279— status port
  - Bit 7 = 0 busy
  - Bit 6 = 0 achnowledge
  - Bit 5 = 1 out of paper
  - Bit 4 = 1 printer is selected
  - Bit 3 = 0 error
  - Bit 2 = 0 IRQ has occured
  - Bit 1 - 0 = reserved
- 27A— control port
  - Bit 7 - 6 = reserved
  - Bit 5 = 1 enable reading parallel port
  - Bit 4 = 1 enable IRQ
  - Bit 3 = 1 select printer
  - Bit 2 = 0 initialize printer
  - Bit 1 = 1 automatic line feed
  - Bit 0 = 1 strobe

290 AGS

292 AGS

294 AGS

296-297 AGS

290-29B GRX

2B0 GRX

292 GRX

294 GRX

2B6-2B7 GRX

2B0-2DF EGA (alternate)

2E0 EGA/VGA (alternate)

2E1— GPIB Adapter 0 (read=DATA IN; write=BYTE OUT)

2E2-2E3 Data Acquisition Adapter 0

2E8-2EF Serial Port 4

2E8-2ED VGA

2EA-2ED 8514A

2F2-2F7 Rearm of shared Interrupt (IRQ9,IRQ3-IRQ7)

2F8-2FF Serial Port2 IRQ3

- 2F8— data port (transmit buffer bei DLAB=0; receive buffer bei DLAB=0)
- 2F8— divisor latch, low byte bei DLAB=1
- 2F9— interrupt enable register
  - Bit 0 = 1 received data available interrupt enable when DLAB = 0
  - Bit 1 = 1 transmitter holding register empty interrupt enable
  - Bit 2 = 1 receiver line status interrupt enable
  - Bit 3 = 1 modem status interrupt enable
  - Bits 4 - 7 = 0 reserved
- 2FA— Interrupt ID Register of 8250 (secondary) . Information about a pending interrupt is stored here. When ID register is address, the highest priority interrupt is held and no other interrupts are acknoladged until the CPU services that interrupt.
  - Bit 7 - 3 = 0 reserved
  - Bit 2 - 1 = these bits identify the pending interrupt with the highest priority: 11 receiver line status interrupt; priority = highest 10 received data available; priority = second 01 transmitter holding register; priority = third 00 modem status interrupt; priority = fourth
  - Bit 0 = 0 interrupt pending, and contents of register can be used as a pointer to the appropriate interrupt service routine
  - Bit 0 = 1 no interrupt pending
- 2FB— line control register
  - Bit 7 = 0 receiver buffer, transmitter holding or interrupt enable register access
  - Bit 7 = 1 divisor latch access (DLA)
  - Bit 6 = 1 set break enabled. Serial output is forced to spacing state and remains there.
  - Bit 5 = stick parity
  - Bit 4 = even parity select
  - Bit 3 = 1 even number of ones are sent and checked in the data word bits and parity bit.
  - Bit 3 = 0 odd number of ones are sent and checked.
  - Bit 2 = 0 1 stop bit.
  - Bit 2 = 1 0 stop bits.
  - Bits 1 - 0 = specify the number of bits per character:
    * 00 = 5 bit word length
    * 01 = 6 bit word length
    * 10 = 7 bit word length
    * 11 = 8 bit word length
- 2FC— modem control register
  - Bits 7 - 5 = 0 reserved
  - Bit 4 = 1 loopback mode for diagnostic testing of serial port. Output of transmitter shift register is looped back to receiver shift register input. When in this mode, transmitted data is received immediately so that the CPU can verify the transmit data/receive data serial paths.
  - Bit 3 = 1 enable OUT2 interrupt
  - Bit 2 = 1 force OUT1 active
  - Bit 1 = 1 force request to send active
  - Bit 0 = 1 force data terminal ready active
- 2FD— line status register
  - Bit 7 = 0 reserved

- Bit 6 = 1 transmitter shift and holding register empty
- Bit 5 = 1 transmitter holding register empty. The controller is ready to accept a new character to send.
- Bit 4 = 1 Break interrupt. The received data input is held in the zero bit state longer than the time of start bit + data bits + parity bit + stop bits.
- Bit 3 = 1 Framing error. The stop bit that follows the last parity or data bit is a zero bit.
- Bit 2 = 1 Parity error. Character has wrong even or odd parity.
- Bit 1 = 1 Overrun error. A character was sent to the receiver buffer before the previous character in the buffer could be read. This destroys the previous character.
- Bit 0 = 1 Data ready. A complete incomming character has been received and sent to the receiver buffer register.

• 2FE— modem status register
  - Bit 7 = 1 data carrier detect
  - Bit 6 = 1 ring indicator
  - Bit 5 = 1 data set ready
  - Bit 4 = 1 clear to send
  - Bit 3 = 1 delta to carrier detect
  - Bit 2 = 1 trailing edge ring indicator
  - Bit 1 = 1 delta data set ready
  - Bit 0 = 1 delta clear to send

• 2FF— reserved

300-31F Prototype Card (NI5010-Ethernet Card)

320-324 Fixed disk adapter

• 320— fixed disk adapter register (8 or 16 bit)

• 322— fixed disk adapter control register
  - Bit 7 = 1 reset
  - Bit 6 = 1 reserved (exept during reset)
  - Bit 5 = 1 16-Bit mode (must match bit 2)
  - Bit 5 = 0 b-bit mode
  - Bit 4-3 = 0 reserved
  - Bit 2 = 1 16-bit mode (must match bit 5)
  - Bit 2 = 0 8-bit mode
  - Bit 1 = 1 enable interrupt through PIC (hardware interrupt)
  - Bit 1 = 0 enable interrupt through interrupt status register (port 324)
  - Bit 0 = 1 DMA mode
  - Bit 0 = 0 PIO mode

• 322— fixed disk adapter status register
  - Bit 7-6 = 0 reserved
  - Bit 5 = 1 16-bit mode
  - Bit 5 = 0 8-bit mode
  - Bit 4 = 1 data transfer request by adapter
  - Bit 3 = 1 direction is adapter to system
  - Bit 3 = 0 direction is system to adapter
  - Bit 2 = 1 busy
  - Bit 1 = 1 interrupt request (notification)

  - Bit 0 = 1 transfer in progress

• 324— fixed disk adapter attention register
  - Bit 7 = 1 command control block
  - Bit 6 = 1 command specify block
  - Bit 5 = 1 sense summary block
  - Bit 4 = 1 data transfer request by system
  - Bit 3 = 0 reserved
  - Bit 2 = 0 drive 0 select
  - Bit 2 = 1 drive 1 select
  - Bit 1 = 0 reserved
  - Bit 0 = 1 abort current command

• 324— fixed disk adapter interrupt status register
  - Bit 7 = 1 termination error, bits 0-6 indicate what the error is
  - Bit 6 = 1 invalid command
  - Bit 5 = 1 command reject
  - Bit 4-3 = 0 reserved
  - Bit 2 = 0 drive 0 select
  - Bit 2 = 1 drive 1 select
  - Bit 1 = 1 error recovery procedure invoked
  - Bit 0 = 1 equipment check

330-333 SCSI-Adapter (AHA-1542)

334-337 SCSI-Adapter secondary (AHA-1542)

338-33F Serial Port 3 IRQ2

348-357 DCA 3278

360-363 PC Network (low address)

36C-36F Reserved

368-36B PC Network (high address)

36C-36F Reserved

372 Diskette controller digital output register

• Bit 7 - 6 reserved
• Bit 5 = 1 enable drive motor 1
• Bit 4 = 1 enable drive motor 0
• Bit 3 = 0 allow interrupts
• Bit 2 = 0 controller reset
• Bit 1 = 0 reserved
• Bit 0 = 0 select drive 0
• Bit 0 = 1 select drive 1

374 Diskette controller status register

• Bit 7 = 1 data register is ready
• Bit 6 = 1 transfer is from controller to system

- Bit 6 = 0 transfer is from system to controller
- Bit 5 = 1 non-DMA mode
- Bit 4 = 1 diskette controller busy
- Bit 3 = reserved
- Bit 2 = reserved
- Bit 1 = 1 drive 1 busy
- Bit 0 = 0 drive 0 busy

375–376 diskette controller data register (alternate)

377 diskette controller digital input register

- Bit 7 = diskette change
- Bit 6 - 1 reserved
- Bit 0 = 0 high density select

378-37F Parallel Printer 2, Data Port

- 378— output port
- 379— status port
  - Bit 7 = 0 busy
  - Bit 6 = 0 acknowledge
  - Bit 5 = 1 out of paper
  - Bit 4 = 1 printer is selected
  - Bit 3 = 0 error
  - Bit 2 = 0 IRQ has occured
  - Bit 1-0= reserved
- 37A— control port
  - Bit 7 - 6 = reserved
  - Bit 5 = 1 enable reading parallel port
  - Bit 4 = 1 enable IRQ
  - Bit 3 = 1 select printer
  - Bit 2 = 0 initialize printer
  - Bit 1 = 1 automatic line feed
  - Bit 0 = 1 strobe

380-38F SDLC, Bisynchronus 2

390-393 Cluster (adapter 0)

3A0-3AF Bisynchronus 1

3B0-3BF Monochrome Display and Printer Adapter

HGA 3B0-3B1, 3B4-3B5, 3B8, 3BF

- 3B0–3B3 miscellaneous video register
- 3B4— MDA, EGA, VGA CRTC index register to port 3B5H
- 3B5— other MDA, EGA, VGA CRTC registers
- 3B8— MDA Mode control register
- 3BA— VGA input status register 1

- 3BA— EGA, VGA feature control register
- 3BC— Output port of printer 1 (Monocrome)
- 3BD— Status port
  - Bit2 = 0 IRQ has occured
  - Bit3 = 0 error
  - Bit4 = 1 printer is selected
  - Bit5 = 1 paper out
  - Bit6 = 0 acknowledge
  - Bit7 = 0 busy
- 3BE— Command port
  - Bit0 = 1 strobe signal
  - Bit1 = 1 autofeed
  - Bit2 = 0 initialize printer
  - Bit3 = 1 select printer
  - Bit4 = 1 enable IRQ
  - Bit5 = 1 enable reading parallel port

3BF Hercules configuration switch register

- Bit0 = 0 = prevents graphics mode
- Bit0 = 1 = allows graphics mode
- Bit1 = 0 = disables upper 32K of graphics mode buffer
- Bit1 = 1 = enables upper 32K of graphics mode buffer at B800:0000h.

3BC-3BF LPT1

3BC Datenregister für Druckerschnittstelle (lesen und schreiben)

3BD Statusregister

- Bit0 = 0 Jumper 1 eingesetzt
- Bit1 = 0 Jumper 2 eingesetzt
- Bit2 = 0 Jumper 3 eingesetzt
- Bit3 = 0 error
- Bit4 = 1 selected
- Bit5 = 1 paper out
- Bit6 = 0 ACK
- Bit7 = 0 busy

3BE Drucker Steuerregister (schreiben und lesen)

- Bit0 = 1 strobe active
- Bit1 = 1 automatischer Zeilenvorschub
- Bit2 = 0 Druckerinitialisierung
- Bit3 = 1 Druckerauswahl
- Bit4 = 1 Unterbrechungsfreigabe durch ACK-Signal

3BF-3C2 (SVGA) (VGA)

3C4-3CA (SVGA) (VGA)

3CC-3CF (SVGA) (VGA)

3C0-3CF EGA, CGA und VGA

- 3C0H VGA attribute controller index register (ATIVGA)
- 3C0H other video attribute controller register

- 3C1H other video attribute controller register (ATIVGA)
- 3C2H EGA, VGA, CGA miscellaneous output register (ATIVGA)
- 3C2H EGA, VGA, CGA input status register 0
- 3C3H video subsystem enable
- 3C4H CGA, EGA, VGA sequencer index register (ATIVGA)
- 3C5H other CGA, EGA, VGA sequencer register (ATIVGA)
- 3C6H VGA video DAC state register (8514A)
- 3C6H VGA video DAC PEL register (8514A)
- 3C7H VGA video DAC state mask (8514A) (ATIVGA)
- 3C8H video DAC PEL address, read mode (8514A) (ATIVGA)
- 3C8H video DAC PEL address, write mode (8514A) (ATIVGA)
- 3C9H VGA video DAC registers (8514A) (ATIVGA)
- 3CAH CGA/EGA, VGA graphics 2 position register (ATIVGA)
- 3CAH CGA/EGA, VGA feature control register (ATIVGA)
- 3CCH VGA miscellaneous output register (ATIVGA)
- 3CCH VGA graphics 1 position register (ATIVGA)
- 3CCH EGA memory mode register (ATIVGA)
- 3CDH VGA feature control register
- 3CEH VGA graphics controller index register (ATIVGA)
- 3CFH other VGA graphics controller register (ATIVGA)
- 3D0–3D3 Video registers
- 3D4H video CRTC index register (ATIVGA)
- 3D5H other CGA, EGA, VGA registers (ATIVGA)
- 3D6H video register
- 3D7H video register
- 3D8H CGA, EGA, VGA mode control register
- 3D9H CGA, EGA, VGA palette register
- 3DAH CGA, EGA, VGA register (ATIVGA)
- 3DB–3DF video registers
- 3D4H VGA CRT controller index register (COLOR)
- 3D5H other VGA CRT controller registers (COLOR)
- 3DAH VGA input status register 1 (COLOR)
- 3DAH VGA feature control register (COLOR)

3D0-3DF CGA

3D4-3D7 (SVGA)

3D4-3D5 (VGA)

3D8 (VGA)

3DA (SVGA) (VGA)

3E8-3EF Serial Port 3

3F0-3F7 Floppy Diskette Controller

3F0 Diskette controller status register A

- Bit 7 = 1 interrupt pending
- Bit 6 = 0 second drive installed
- Bit 5 = 1 step
- Bit 4 = 1 track 0

- Bit 3 = 1 head 1 select
- Bit 2 = 0 index
- Bit 1 = 0 write protect
- Bit 0 = 0 Data received by controller

- 3F1 D/S/P diagnostic register (status B)
  - Bit 7 = 0 reserved
  - Bit 6 = 0 reserved
  - Bit 5 = 0 select drive 0
  - Bit 4 = write data
  - Bit 3 = read data
  - Bit 2 = 1 write enable
  - Bit 1 = 1 enable drive motor 1
  - Bit 0 = 1 enable drive motor 0
- 3F2 diskette controller digital output port (motor drive)
  - Bit 7 = 0 reserved (enable drive 3 motor)
  - Bit 6 = 0 reserved (enable drive 2 motor)
  - Bit 5 = 1 enable drive 1 motor
  - Bit 4 = 1 enable drive 0 motor
  - Bit 3 = 0 enable diskette interrupts and DMA
  - Bit 2 = 0 controller reset
  - Bit 1 = 0 reserved (In Verbindung mit Bit 0 - select drive 0 - 3)
  - Bit 0 = 0 select drive 0
  - Bit 0 = 1 select drive 1
- 3F4 status port
  - Bit 7 = 1 data register is ready for transfer (request for master)
  - Bit 6 = 1 transfer is from controller to system
  - Bit 6 = 0 transfer is from system to controller
  - Bit 5 = 1 non-DMA mode
  - Bit 4 = 1 diskette controller busy
  - Bit 3 = 0 reserved (drive 3 busy)
  - Bit 2 = 0 reserved (drive 2 busy)
  - Bit 1 = 1 drive 1 busy
  - Bit 0 = 0 drive 0 busy
- 3F5–3F6 diskette controller data registers
- 3F5 data port

0F seek

E6 read

C5 write

08 tell status

07 go to cyl 0

03 accept params

  09 number of sectors per track

  2A intersector gap size

  FF data length (sector size)

  DF first parameter

02 second parameter

- 3F6 disk control port
- 3F7 fixed disk digital input register
  - Bit 7 = 1 diskette changed
  - Bit 6 = 1 write gate
  - Bit 5 = head select 3/reduced write current
  - Bit 4 = head select 2
  - Bit 3 = head select 1
  - Bit 2 = head select 0
  - Bit 1 = select drive 1
  - Bit 0 = select drive 0
  - Bits 6 - 0 apply to the fixed disk currently selected.!!!
- 3F7 diskette data transfer rate select register
  - Bit 7 - 2 = reserved
  - Bit 1 - 0 = transfer rate
    * 00 = 500 Kbs
    * 01 = 300 Kbs
    * 10 = 250 Kbs
    * 11 = reserved

3F8-3FF Serial Port 1 IRQ4 Port Offsets from Base Address

Offset	Name	Bedeutung
00H	DATA	Data Register
01H	IER	Interrupr enable register
00H	Baud0	Divisor low
01H	Baud1	Divisor high
02H	IID	interrupt identifier register
03H	LCR	line control register
04H	MCR	modem control register
05H	LSR	line status register
06H	MSR	modem status register

Die erste Bedeutung der Register 00h und 01h gelten bei DLABbit in LCR = 0. Die zweite bei DLABbit in LCR = 1.

Bit Rate Divisor Table for 8250.

45,5	2532
50	2304
75	1536
110	1047
134,5	857
150	768
300	384
600	192
1200	96
1800	64
2000	58
2400	48
4800	24
9600	12
19200	6
38400	3

- 3F8— Data Port (transmitter and receiver holding register)
  - Bits 7 - 0 = data bits 7 - 0 when Divisor Latch Access Bit (DLAB) = 0
- 3F8— receiver buffer register
  - Bits 7 - 0 = data bits 7 - 0 when DLAB = 0
- 3F8— divisor latch, low byte
  - Bits 7 - 0 = bits 7 - 0 of divisor, when DLAB = 1
- 3F9— divisor latch, high byte
  - Bits 7 - 0 = bits 15 - 8 of divisor, when DLAB = 1
- 3F9— interrupt enable register
  - Bits 7 - 4 = reserved
  - Bit 3 = 1 modem status interrupt enable
  - Bit 2 = 1 receiver line status interrupt enable
  - Bit 1 = 1 transmitter holding register empty interrupt enable
  - Bit 0 = 1 received data available interrupt enable when DLAB = 0
- 3FA— Interrupt ID Register of 8250 (primary). Information about a pending interrupt is stored here. When ID register is addressed, the highest priority interrupt is held and no other interrupts are acknowledged until the CPU services that interrupt.
  - Bits 7 - 3 = 0 reserved
  - Bit 2 - 1 = these bits identify the pending interrupt with the highest priority:
    * 11 receiver line status interrupt; priority = highest.
    * 10 received data available; priority = second.
    * 01 transmitter holding register; priority = third.
    * 00 modem status interrupt; prority = fourth.
  - Bit 0 = 0 interrupt pending, and contents of register can be used as a pointer to the appropriate interrupt service routine.
  - Bit 0 = 1 no interrupt pending.
- 3FB— Line control register
  - Bit 7 = 0 receiver buffer, transmitter holding, or interrupt enable register access
  - Bit 7 = 1 Divisor latch access DLAB
  - Bit 6 = 1 set break enable. Serial output is forced to spacing state and remains there.
  - Bit 5 = stick parity
  - Bit 4 = even parity select
  - Bit 3 = 1 even number of ones are sent and checked in the data word bits and parity bit.
  - Bit 3 = 0 odd number of ones are sent and checked.
  - Bit 2 = 0 1 stop bit
  - Bit 2 = 1 0 stop bits
  - Bits 1 - 0 specify the number of bits per character
    * 00 5 bit word length
    * 01 6 bit word length
    * 10 7 bit word length
    * 11 8 bit word length
- 3FC— Modem Control Register
  - Bit 7 - 5 = 0 reserved

- Bit 4 = 1 loopback mode for diagnostic testing of serial port. Output of transmitter shift register is looped back to receiver shift register input. When in this mode, transmitted data is immediately received so that the CPU can verify the transmit data/receive data serial port paths.
- Bit 3 = 1 enable out2 interrupt
- Bit 2 = 1 force out1 active
- Bit 1 = 1 force request to send active
- Bit 0 = 1 force data terminal ready active

- 3FD— Line status register
  - Bit 7 = 0 reserved
  - Bit 6 = 1 transmitter shift and holding register empty
  - Bit 5 = 1 transmitter holding register empty. The controller is ready to accept a new character to send.
  - Bit 4 = 1 break interrupt. The received data input is held in the zero bit state longer than the time of start bit + data bits + parity bit + stop bits.
  - Bit 3 = 1 framing error. The stop bit that follows the last parity or data bit is a zero bit.
  - Bit 2 = 1 parity error. Character has incorrect even or odd parity.
  - Bit 1 = 1 overrun error. A character was sent to the receiver buffer before the previous character in the buffer could be read. This destroys the previous cgaracter.
  - Bit 0 = 1 data ready. A compete incomming character has beeb received and sent to the receiver buffer register.
- 3FE— Modem Status Register
  - Bit 7 = 1 data carrier detect
  - Bit 6 = 1 ring indicator
  - Bit 5 = 1 data set ready
  - Bit 4 = 1 clear to send
  - Bit 3 = 1 delta data carrier detect
  - Bit 2 = 1 trailing edge ring indicator
  - Bit 1 = 1 delta data set ready
  - Bit 0 = 1 delta clear to send
- 3FF— reserved

400— Auger-Interface 137 ohne IRQ und DMA

401—4FF EISA-DMA extended mode programming registers (DEC).

40A DMA chaining mode register[Ch 0-3] (write)

- Bit4 signal to generate
  - = 0 generate IRQ13
  - = 1 generate TC
- Bit3-2 Buffer chaining mode
  - 00 = disable
  - 01 = enable for programming
  - 10 = illegal
  - 11 = programming complete, gebin chaining
- Bit1-0 DMA channel select
  - 00 = Ch 0

  - 01 = Ch 1
  - 10 = Ch 2
  - 11 = Ch 3

40A Channel interrupt status register (read)

Bit7 = 1 channel 7 has interrupt

Bit6 = 1 channel 6 has interrupt

Bit5 = 1 channel 5 has interrupt

Bit4 = 0 reserved

Bit3 = 1 channel 3 has interrupt

Bit2 = 1 channel 2 has interrupt

Bit1 = 1 channel 1 has interrupt

Bit0 = 1 channel 0 has interrupt

40B DMA extended mode register[Ch 0-3]

- Bit7 = 0 stop register enabled
- Bit6 = 0 EOP an output; = 1 EOP an input for this channel.
- Bit5-4 DMA cycle timing mode
  - 00 = ISA-compatible
  - 01 = type A
  - 10 = type B
  - 11 = type C Burst DMA
- Bit3-2 Addressing mode
  - 00 = 8-bit I/O by bytes
  - 01 = 16-bit I/O by words (addr shifted)
  - 10 = 32-bit I/O by bytes
  - 11 = 16-bit I/O by bytes
- Bit1-0 DMA channel select
  - 00 = Ch 0
  - 01 = Ch 1
  - 10 = Ch 2
  - 11 = Ch 3

40C Chain buffer expiration control register (read)

- Bit7 = 0 activate IRQ13; = 1 issue TC Ch 7
- Bit6 = 0 activate IRQ13; = 1 issue TC Ch 6
- Bit5 = 0 activate IRQ13; = 1 issue TC Ch 5
- Bit4 = 0 reserved
- Bit3 = 0 activate IRQ13; = 1 issue TC Ch 3
- Bit2 = 0 activate IRQ13; = 1 issue TC Ch 2
- Bit1 = 0 activate IRQ13; = 1 issue TC Ch 1
- Bit0 = 0 activate IRQ13; = 1 issue TC Ch 0

461 Extended NMI control/status port (read/write)

- Bit7 = 1 fail safe timer active, NMI requested.
- Bit6 = 1 32-bit bus timeout NMI requested.
- Bit5 = 1 NMI I/O port status software generated NMI.
- Bit3 = 1 enable NMI from 32-bit bus timeout.
- Bit2 = 1 enable NMI from timeout of fail-save timer.
- Bit1 = 1 enable NMI from software (writing to 462).
- Bit0 = 1 bus reset asserted (RSTDRV).

462 Software NMI generation port (write)

464 32-Bit bus master status latch [slots 1-8](read)

- Bit5 = 0 last EISA bus master granted SLOT6
- Bit4 = 0 last EISA bus master granted SLOT5
- Bit3 = 0 last EISA bus master granted SLOT4
- Bit2 = 0 last EISA bus master granted SLOT3
- Bit1 = 0 last EISA bus master granted SLOT2
- Bit0 = 0 last EISA bus master granted SLOT1

481 8-bit DMA channel 2 upper page register

482 8-bit DMA channel 3 upper page register

483 8-bit DMA channel 1 upper page register

487 8-bit DMA channel 0 upper page register

489 16-bit DMA channel 6 upper page register

48A 16-bit DMA channel 7 upper page register

48B 16-bit DMA channel 5 upper page register

48F refresh cycle page register

4CB DMA extended mode register(Ch 4-7)

4D0 Edge/level control register (ELCR) IRQ0-7

- Bit7 = 1 level-sensitive IRQ7
- Bit6 = 1 level-sensitive IRQ6
- Bit5 = 1 level-sensitive IRQ5
- Bit4 = 1 level-sensitive IRQ4
- Bit3 = 1 level-sensitive IRQ3
- Bit2 reserved
- Bit1 reserved
- Bit0 reserved

4D1 Edge/level control register (ELCR) IRQ8-15

- Bit7 = 1 level-sensitive IRQ15
- Bit6 = 1 level-sensitive IRQ14

- Bit5 = 1 level-sensitive IRQ13 (preconfigured)
- Bit4 = 1 level-sensitive IRQ12
- Bit3 = 1 level-sensitive IRQ11
- Bit2 = 1 level-sensitive IRQ10
- Bit1 = 1 level-sensitive IRQ9
- Bit0 reserved

4D4 DMA chaining mode register(Ch 4-7)

4D4 Chaining mode status register (read)

- Bit7 = 1 chaining mode enabled
- Bit6 = 1 chaining mode enabled
- Bit5 = 1 chaining mode enabled
- Bit4 = 0 reserved
- Bit3 = 1 chaining mode enabled
- Bit2 = 1 chaining mode enabled
- Bit1 = 1 chaining mode enabled
- Bit0 = 1 chaining mode enabled

5F0-5F7 fixed disk diagnostic register

6E1 GPIB Adapter 0 (read= Interrupt Status 1; write=Interrupt mask1)

6E2-6E3 Data Acquisition Adapter 1

6E8-6E9 (VGA)

6F2-6F7 Rearm of shared Interrupt (IRQ10-IRQ15)

6F8 CGA

790-793 Cluster Adapter 1

AE1 GPIB Adapter 0 (read=Interrupt Status 2; write=Interrupt mask 2)

AE2-AE3 Data Acquisition Adapter 2

AE8-AE9 (VGA)

B90-B93 Cluster Adapter 2

0zC80-C81 EISA product ID; drei-5-bit Werte; z = slot number

0zC82-C83 Product number and revision.

0zC84 Expansion board control register

- Bit2 = IOCHKRST I/O check reset (write only)
- Bit1 = IOCHKERR = 0 = no error
- Bit0 = ENABLE = 0 = disable the expansion board

EE1 GPIB Adapter 0 (read=serial poll status; write=serial poll mode)

EE2-EE3 Data Acquisition Adapter 3

EE8-EE9 (VGA)

1000-1001 MHGAD, MHGAC8, MHGAD8

1008 MHGAD, MHGAC8, MHGAD8

100F MHGAD, MHGAC8, MHGAD8

12E1 GPIB Adapter 0 (read=address status; write=address mode)

12E8-12E9 (VGA)

1390-1393 Cluster Adapter 3

16E1 GPIB Adapter 0 (read=CMD pass through; write=auxiliary mode)

16E8-16E9 (VGA)

1AE1 GPIB Adapter 0 (read=address 0; write=address 0/1)

1AE8-1AE9 (VGA)

1EE1 GPIB Adapter 0 (read=address 1; write=end of string)

1EE8-1EE8 (VGA)

22E1— GPIB Adapter 1

22E8-22E9 (VGA)

2390-2393 Cluster Adapter 4

23C6 CGA

3220-3227 Serial port 3

3228-322F Serial port 4

42E1— GPIB Adapter 2

42E8-42E9 (VGA)

46E8-46E9 (VGA)

4AE8 (8514A)

4AE8-4AE9 (VGA)

62E1— GPIB Adapter 3

82E1— GPIB Adapter 4

82E8-82E9 (VGA)

86E8-86E9 (VGA)

8AE8-8AE9 (VGA)

8EE8-8EE9 (VGA)

92E8-92E9 (VGA)

96E8-96E9 (VGA)

9AE8-9AE9 (VGA)

9EE8-9EE9 (VGA)

A2E1— GPIB Adapter 5

A2E8-A2E9 (VGA)

A6E8-A6E9 (VGA)

AAE8-AAE9 (VGA)

AEE8-AEE9 (VGA)

AFFF Plane 0 - 3 system latch (video register)

B2E8-B2E9 (VGA)

B6E8-B6E9 (VGA)

BAE8-BAE9 (VGA)

BEE8-BEE9 (VGA)

C2E1— GPIB Adapter 6

C2E8-C2E9 (VGA)

C6E8-C6E9 (VGA)

CAE8-CAE9 (VGA)

CEE8-CEE9 (VGA)

D2E8-D2E9 (VGA)

D6E8-D6E9 (VGA)

DAE8-DAE9 (VGA)

DEE8-DEE9 (VGA)

E2E1— GPIB Adapter 7

E2E8-E2E9 (VGA)

E6E8-E6E9 (VGA)

EAE8-EAE9 (VGA)

EEE8-EEE9 (VGA)

F2E8-F2E9 (VGA)

F6E8-F6E9 (VGA)

FAE8-FAE9 (VGA)

FEE8-FEE9 (VGA)

6.2 EISA Extensions I/O Address Range

Bei Verwendung von Erweiterungskarten muß auf die Auskodierung der Adressen geachtet werden. Normal werden nur 10-Bit der Adresse genommen, so daß es gegebenenfalls zu Kollisionen kommnen kann.

400—407 DMA Ch 0-3 (odd locations) high byte of Count Register (even locations reserved).

40A—40C DMA Controller 1. Channels 0 - 3 EISA register extensions.

461—462 Extended NMI, Reset Control Register, Software NMI Register.

464 Last EISA Bus Master granted.

480—48F DMA Page Address Register, high byte 480, 484-6, 488, 48C-F reserved.

4C6 DMA Ch 5 high byte of Count Register.

4CA DMA Ch 6 high byte of Count Register.

4CE DMA Ch 7 high byte of Count Register.

4D0–4D1 INT Edge/Level Control Register.

4D4—4D7 DMA Chaining Mode, Extended Mode Registers. 4D5, 4D7 reserved.

4E0—4FF DMA Stop Registers. 4E3, 4E7, 4EB, 4EF, 4F0-3, 4F7, 4FF reserved.

C00—C05 Motherboard Configuration Registers.

C80—C84 Motherboard ID, Motherboard Enable.

Die folgenden Register sind nur bei der DECstation 425 vorhanden und sind nicht Teil des ISA- oder EISA-Standards.

6.2.1 System Configuration Register 0 (0C00)

Das Register wird nach 0 gesetzt mit dem Reset. Es wird zur Kontrolle der Konfiguration benutzt.

Bit7 Read-Only Bit. 1= Configurations-Stecker eingesteckt, 0= nicht gesteckt. Entspricht dem "reset 206" bei den NEAT's. Benutzt dann beim Booten die ursprüngliche Konfiguration einschließlich Password.

Bit6 1 = Configuration EEPROM und Mapping SRAM können geändert werden. Sollte gleich wieder zurückgesetzt werden.

Bit5-4 Hardware-Triggered Slowdown Mode.

- 00 = disabled
- 01 = active for floppy access (ca. 2 Sekunden)
- 10 = always active
- 11 = invalid

Bit3-2 i860 System IRQ select. Der erlaubte Interrupt sollte auf "level-trigerred mode" gesetzt sein.

- 00 = IRQ15
- 01 = IRQ11
- 10 = IRQ10

- 11 = IRQ5

Bit1-0 Legt fest, wie das eingebaute serielle Interface benutzt werden soll. Der erlaubte Interrupt sollte auf "edge-triggered mode" gesetzt sein.

- 00 = disabled
- 01 = COM1, 03F8, IRQ4
- 10 = COM2, 02F8, IRQ3
- 11 = invalid

6.2.2 System Configuration Register 1 (0C01)

Das Register wird bei Reset auf 0 gesetzt.

Bit7 Reserved

Bit6 Das Mapping SRAM muß vorher initialisiert werden. Mit Bit6 = 1 wird caching und die DRAM Konfiguration der unteren 256 MB Systemspeicher erlaubt.

Bit5 parallel port extended mode. 0 = disabled, 1 = enabled.

Bit4 CPU Cache disabled wenn = 1.

Bit3-2 Console Interface System Interrupt Select (front control panel).

- 00 = IRQ15
- 01 = IRQ11
- 10 = IRQ10
- 11 = IRQ5

Bit1-0 Parallel Port Select. Für das einbgebaute parallel Interface sollte der "edge-triggered mode" gesetzt sein.

- 00 = disabled
- 01 = LPT1, 0378, IRQ7
- 10 = LPT2, 0278, IRQ7
- 11 = LPT3, 03BC, IRQ5

6.2.3 System Configuration Register 2 (0C02)

Bit7 860 R/T Clock Interrupt Request = 1.

Bit6 Flash EPROM Program Enable = 1. Read only = 0.

Bit5 0 = Reset 860, 1 = Start 860 from Start Address.

Bit4-0 reserved

6.2.4 System Configuration Register 3 (0C03)

Used to address the 8KB EISA Configuration EEPROM.

Die Bits 7 - 0 entsprechen den Bits 12 - 5 des EEPROM. Mit dieser "Page" können jeweils 32 Byte über C20 — C3F angesprochen werden.

6.2.5 System Configuration Register 4 (0C04)

Software, die auf dem 486 läuft, stellt im Memory für den 860 eine Message zur Verfügung und setzt dieses Bit. Der 860 bearbeitet die Message und setzt danach das Bit zurück damit es erneut benutzt werden kann.

Bit7 860 Interrupt Request.

Bit6-0 reserved, mask off during read.

6.2.6 System Configuration Register 5 (0C05)

Dieses Bit wird von dem 860 für den 486 wie das Register 4.

Bit7-4 reserved, mask off during read.

Bit3 486 Interrupt Request

Bit2-0 reserved, mask off during read.

6.2.7 System Control Port A (0092)

Diese Port ist vergleichbar dem in einem PS/2 System.

Bit7-6 HDU LED Bits.

Bit5-4 reserved, mask off during read.

Bit3 Security Lock Latch. Wenn das Bit gesetzt ist, ist Lesen und Schreiben der CMOS RAM Bytes 38 — 3F (Password) nicht erlaubt.

Bit2 reserved

Bit1 alternate gate A20.

Bit0 CPU reset.

6.3 DOS Memory Map

Es folgt eine Liste der Memorybereiche die an festen oder variablen Adressen beginnen.

- 0000:0000 Interrupt vector table
- 0040:0000 ROM communication area
- 0050:0000 DOS communication area
- 0070:0000 DOS interface to ROM I/O routinen (io.sys)
- XXXX:0000 DOS interrupt handlers, service routinen (INT21 funktionen)
- ????????? DOS buffers, control areas und installierte device drivers
- XXXX:0000 Residenter Teil von command.com, INT22, INT23, INT24, reloader
- XXXX:0000 External commands

- XXXX:0000 User stack für .COM files
- XXXX:0000 Transienter Teil von command.com
- bis 9FFFF freier Arbeitsspeicher
- A000:0000 256kB EGA-Karte oder
- B000:0000 MDA/Hercules-Karte oder
- B800:0000 CGA-Karte
- C000:0000 ROM-Bereich, ab FE00:0000 BIOS
- D000:0000 ROM- und RAM-Bereich der DEPCA (oder E000:0000).
- C000 0000 — C000 FFFF Weitek 4167 Co-Processor (3 GByte)

Es ist sehr sinnvoll mehrere Bereiche für unterschiedliche Bildschirmkarten vorzuhalten, wenn das System nur einen bedienen kann!

Nachdem der POST die Interrupt Vektoren für das BIOS gesetzt hat können zusätzliche ROM Module die Kontrolle erhalten und ihre Vektoren setzen.

Beim POST (power on self test) werden die Adressen C8000H bis E0000H in 2k-Schritten nach einem erlaubten Adapter ROM abgesucht. Ein Adapter ROM ist erlaubt wenn:

- Byte 0 = 55H
- Byte 1 = AAH
- Byte 2 = Anzahl der 512 Byte Blöcke des ROM
- Byte 3 = Entry für ein CALL FAR

Es ist gültig wenn die Checksumme modulo 100H gleich null ist.

Beim POST (power on self test) werden die Adressen E0000H bis EFFFFH in 64k-Schritten nach einem erlaubten ROM abgesucht. Ein ROM ist erlaubt wenn:

- Byte 0 = 55H
- Byte 1 = AAH
- Byte 2 = nicht benutzt
- Byte 3 = Entry für ein CALL FAR

Es ist gültig wenn die Checksumme modulo 100H gleich null ist. Diese Checksumme steht an der Adresse EFFFFH. Es ist also nur ein zusätzliches ROM (BASIC) möglich. FFFFH ist 64k. Soll ich noch was über IBM sagen?

6.4 ROM-Communication Area

Die ROM-Communication Area wird von den ROM-Routinen beschrieben und von dem System benutzt. Programme sollten diesen Bereich nicht direkt adressieren sondern sich auf die Systemaufrufe beschränken. Die folgenden Adressen werden auch in der Form 0040:00XX geschrieben.

ROM-BIOS DATA

- 400-407 Base-Address der vorhandenen RS232 Adapter.
- 408-40F Base-Address der vorhandenen Printer Adapter.
- 410-411 Flags der installierten Hardware. (Interrupt Nr. 11H)
  - Bit 15 - 14 = number of printer adapters
  - Bit 13 - 12 = reseved
  - Bit 11 - 9 = number of asynchronous adapters (RS232)
  - Bit 8 = reserved
  - Bit 7 - 6 = number of diskette drives
    - 00 = 1 diskette drive
    - 01 = 2 diskette drives
  - Bit 5 - 4 = initial video mode
    - 00 = EGA/VGA/PGA
    - 01 = 40x25 color
    - 10 = 80x25 color
    - 11 = 80x25 black and white
  - Bit 3 = reseved
  - Bit 2 = pointing device installed
  - Bit 1 = 1 math coprocessor installed
  - Bit 0 = diskette available for boot
- 412— Initialisierungs Flag. (Manufacturing test port)
- 413-414 Base Memory size in K Byte. (Interrupt Nr. 12H)
- 415-416 Hersteller error flag. (Manufacturing test port)

KEYBOARD DATA

- 417— Keyboard shift state und status flag.
  - Bit 7 = 1 insert active
  - Bit 6 = 1 caps lock active
  - Bit 5 = 1 numlock active
  - Bit 4 = 1 scroll lock active
  - Bit 3 = 1 Alt pressed
  - Bit 2 = 1 Ctrl pressed
  - Bit 1 = 1 left shift pressed
  - Bit 0 = 1 right shift pressed
- 418— Extended keyboard shift flags.

  - Bit 7 = 1 Insert pressed
  - Bit 6 = 1 Caps lock pressed
  - Bit 5 = 1 Num lock pressed
  - Bit 4 = 1 Scroll lock pressed
  - Bit 3 = 1 Ctrl-Num lock pressed
  - Bit 2 = 1 Sys Req pressed
  - Bit 1 = 1 Left Alt pressed
  - Bit 0 = 1 Left Ctrl pressed
- 419— Alternate key pad entry.
- 41A-41B Pointer to head of keyboard buffer (41E).
- 41C-41D Pointer to tail of keyboard buffer. Wenn head gleich tail ist der buffer leer.
- 41E-43D Buffer für scan code entries. (Interrupt Nr. 16H)

DISKETTE DATA

- 43E— seek status; Bits 0 - 3 = Drive 0 - 3; Bit = 0 = recalibration needed. (IBM-BIOS?)
  - Bit 7 = 1 diskette harware interrupt occurred
  - Bit 6 - 4 = not used
  - Bit 3 - 2 = reserved
  - Bit 1 = recalibrate drive 1
  - Bit 0 = recalibrate drive 0
  - Phoenix Techn. BIOS !
- 43F— motor status; Bits 0 - 3 = Drive 0 - 3; Bit = 1 = motor on. (IBM-BIOS ?)
  - Bit 7 = 1 current operation is write or format
  - Bit 7 = 0 current operation is read or verify
  - Bit 6 = reserved
  - Bit 5 - 4 = drive select status
    - 00 = drive 0 selected
    - 01 = drive 1 selected
    - 10 = reserved
    - 11 = reserved
  - Bit 1 = drive 1 motor is on
  - Bit 0 = drive 0 motor is on
  - Phoenix Techn. BIOS !
- 440— motor time out count (25H ticks).
- 441— return code status byte
  - Bit7 timed out (drive not ready)
  - Bit6 seek to track failed
  - Bit5 controller chip failed
  - Bit4 CRC error in data

  - Bit3 DMA error
  - Bit2 sector not found
  - Bit1 address mark not found
  - Bit0 invalid command requested
  - Bit 4 - 0 =
    * 01H = illegal function requested
    * 02H = address mark not found
    * 03H = write protect error
    * 04H = sector not found
    * 06H = drive door was opened
    * 08H = DMA overrun error
    * 09H = DMA boundary error
    * 0CH = media type unknown
    * 10H = CRC failed on diskette read

- 442-448 diskette status bytes

VIDEO DISPLAY

- 449— current display mode (type)
  - 00 = 40 column text
  - 01 = 40 column text, 16 Farben
  - 02 = 80 column text
  - 03 = 80 column text, 16 Farben
  - 04 = medium resolution graphics, 4 Farben
  - 05 = medium resolution graphics
  - 06 = high resolution graphics, 2 Farben
  - 07 = monochrome adapter
  - 08 = low resolution graphics, 16 Farben
  - 09 = medium resolution graphics, 16 Farben
  - 10 = high resolution graphics, 4 Farben
  - 13 = medium resolution graphics, 16 Farben
  - 14 = high resolution graphics, 16 Farben
  - 15 = special high resolution graphics, 4 Farben
- 44A-44B number of columns on screen
- 44C-44D length of regen(eration) buffer in bytes (current page size)
- 44E-44F starting address in regen buffer (current page address)
- 450-45F cursor position (2 bytes,col-row) for each of up to 8 pages
- 460-461 current cursor mode (starting and ending scan lines)
- 462— current page number
- 463-464 base address of active display card (port 3D4 or 3B4)

- 465— current setting of the 3X8 register (mode select)
- 466— current palette setting (color card)

POST (Power-on self test) and BIOS WORK DATA

- 467-468 pointer to ROM initialisation routine
- 469-46A pointer to I/O ROM Segment
- 46B— flag indicating an interrupt happend

In Peter Norton's Programmer's Guide findet man "these bytes are used for cassette tape control".
TIMER DATA

- 46C-46D low word of timer count
- 46E-46F high word of timer count
- 470— timer has rolled over since last read

SYSTEM DATA

- 471— BIOS break, Bit 7=1 if Ctrl break key has been pressed
- 472-473 Reset flag, word=1234H if keyboard reset underway (Ctrl Alt Del)
  - 1234H = Bypass memory test (warm boot)
  - 4321H = preserve memory
  - 64H = Burn in mode
- 473— Reserved for IBM; This byte is set to 24H when a stock dividend is declared (Peter Norton).

FIXED DISK DATA

- 474— fixed disk status
  - 00H = no error
  - 01H = invalid function request
  - 02H = address mark not found
  - 03H = write protect error
  - 04H = sector not found
  - 05H = reset failed
  - 07H = drive parameter activity failed
  - 08H = DMA overrun on operation
  - 09H = data boundary error
  - 0AH = bad sector flag detected
  - 0BH = bad trsck detected
  - 0DH = invalid number of sectors on format
  - 0EH = control data address mark detected
  - 0FH = DMA arbitration level out of range
  - 10H = uncorrectable ECC or CRC error

- 11H = ECC corrected data error
- 20H = general controller failure
- 40H = seek operation failed
- 80H = time-out
- AAH = drive not ready
- BBH = undefined error occured
- CCH = write fault on selected drive
- E0H = status error / error register is 0
- FFH = sense operation failed

- 475— count of fixed disk drives
- 476— head control byte
- 477— port Offset

TIME-OUT VARIABLES

- 478-47B time out counter for printers (je 1 byte). 1 reserviert.
- 47C-47F time out counter for RS232 (je 1 byte). 2 reserviert.

Zusätzliche KEYBOARD DATA

- 480-481 Address of keyboard buffer start
- 482-483 Address of end of buffer

EGA/PGA DISPLAY WORK AREA

- 484— rows on the active screen (minus 1) (24/25)
- 485-486 bytes per character (height)
- 487— mode options (video control bits)
  - Bit 7 = clear RAM
  - Bit 5,6 = Memory on video hardware
    - 00 = 64K
    - 01 = 128K
    - 10 = 192K
    - 11 = 256K
  - Bit 4 = not used
  - Bit 3 = 0 EGA-compatible mode active
  - Bit 2 = 1 wait for display enable
  - Bit 1 = 0 color or ECD attached to EGA-compatible adapter
  - Bit 1 = 1 monochrome monitor attached to EGA-compatible adapter
  - Bit 0 = 0 translate cursor video modes 0 - 3 when using ECD monitor in 350-line mode
  - Bit 0 = 1 inhibit cursor translation
- 488— feature bit switches (EGA/VGA)

  - Bit 7 - 4 = feature connector bits 3 - 0
  - Bit 3 - 0 = option switches 3 - 0
- 489— VGA control Bits
  - Bit 7 = 200 lines
  - Bit 5,6 = reserved
  - Bit 4 = 400 lines
  - Bit 3 = no palette load
  - Bit 2 = mono monitor
  - Bit 1 = gray scaling
  - Bit 0 = reserved
- 48A— index to the display combination code table (DCC)

Zusätzliche MEDIA DATA

- 48B— last diskette data rate selected
  - Bit 7 + 6 last data transfer rate sent to controller
    - 00 500 KBS
    - 01 300 KBS
    - 10 250 KBS
    - 11 reserved
  - Bit 5 + 4 reserved (last step rate selected)
  - Bit 3 + 2 data transfer rate that operation started in
    - 00 500 KBS
    - 01 300 KBS
    - 10 250 KBS
    - 11 reserved
  - Bit 1 reserved
  - Bit 0 adapter is DSP or COMBO (dual)
- 48C— fiexed disk status register
- 48D— fiexed disk error register
- 48E— fixed disk interrupt flag
- 48F— combo fixed disk/diskette card (bit 0=1 in 48B) drive A low nibble, drive B high nibble
  - Bit 7 + 3 reserved
  - Bit 6 + 2 multi data rate capability
  - Bit 5 + 1 multi data rate format capability
  - Bit 4 + 0 80 track capability (supports change line)
- 490— drive 0 media state
  - Bit 7 + 6 data transfer rate
    - 00 500 KBS
    - 01 300 KBS

    * 10 250 KBS
    * 11 reserved
  - Bit 5 media double stepping required (360K media/1.2 MB drive)
  - Bit 4 media determined/established
  - Bit 3 reserved
  - Bit 2 multi data rate capability determined
  - Bit 1 multi data rate format capability
  - Bit 0 80 track capability
  - Bit 2 - 0 = Definitions on return to user:
    * 111 = 720K media in 720H or 1.44 MB drive; or 1.44 MB media in 1.44 MB drive
    * 101 = known 1.2 MB media in 1.2 MB drive
    * 100 = known 360K media in 1.2 MB drive
    * 011 = known 360K media in 360K drive
    * 010 = trying 1.2 MB media in 1.2 MB drive
    * 001 = trying 360K media in 1.2 MB drive
    * 000 = trying 360K media in 360K drive

- 491— drive 1 media state
- 492— drive 0 operation start state
- 493— drive 1 operation start state
- 494— drive 0 present cylinder
- 495— drive 1 present cylinder

KEYBOARD FLAGS

- 496— keyboard mode state and type flags
  - Bit 7 = 1 Read ID in progress
  - Bit 6 = 1 Last code was first ID
  - Bit 5 = 1 Forced Num lock
  - Bit 4 = 1 101/102 keyboard used
  - Bit 3 = 1 Right Alt active
  - Bit 2 = 1 Right Ctrl active
  - Bit 1 = 1 Last code was E0H
  - Bit 0 = 1 Last code was E1H
- 497— keyboard LED flags
  - Bit 7 Error flag for keyboard command
  - Bit 6 LED update in progress
  - Bit 5 RESEND received from keyboard
  - Bit 4 ACK received from keyboard
  - Bit 3 reserved
  - Bit 2 current status of Caps lock LED
  - Bit 1 current status of Num lock LED

  - Bit 0 current status of Scroll lock LED

REAL TIME CLOCK DATA

- 498-499 Offset address of user wait flag
- 49A-49B Segment address of user wait flag
- 49C-49D low word of user wait flag
- 49E-49F high word of user wait flag
- 4A0— wait active flag (01=busy, 80=posted, 00=post acknowledged)
  - Bit 7 = 1 wait time elapsed
  - Bit 6 - 1 = reserved
  - Bit 0 = 1 INT 15H, AH = 86H occured

NETWORK ADAPTER AREA

- 4A1— reserved for network adapters
- —4A7

EGA/PGA PALETTE POINTER

- 4A8-4B1 pointer to EGA parameter control block
- 4B2— reserved

ANDERE DATEN

- 4B0–4B5 reserved
- 4B6–4B8 reserved for POST
- 4C0–4CD reserved
- 4CE— count of days since 1 - 1 - 80
- 4CF–4FF reserved

PRINT SCREEN DATA AREA

- 500— Address 40:100 = 50:000
- 500— print screen status byte
  - 00= ready/ok
  - 01=busy
  - FF=error

6.5 Intra-Application Communication Area

- 4F0-4FF Kann für Kommunikation unterschiedlicher Anwenderprogramme benutzt werden. Verwendet wird die ICA von einigen asynchronen Kommunikationspaketen von IBM, dem Volkswriter von Lifetree und dem TimeMark von Peter Norton. Bei gleichzeitiger Anwendung unterschiedlicher Programme (TSR) sollten Schutzmechanismen, Checksumme und/oder Signatur, benutzt werden um fremde Daten zu erkennen.

6.6 DOS Communication Area

Die DOS Communication Area wird wie folgt benutzt:

- 0050:0000 Print screen status flag.
- 0050:0001 Wird von BASIC benutzt.
- 0050:0004 Single-drive mode status byte.
  - 00 acting as drive A
  - 01 acting as drive B
- 0050:0010-0021 Wird von BASIC benutzt.
  - 0050:0010-0011 default Data Segment des Basic.
  - 0050:0012-0015 Pointer zur clock-tick Interruptroutine.
  - 0050:0016-0019 Pointer zur break-key Routine.
  - 0050:001A-001D Pointer zur Disketten Error Routine.
- 0050:0022-002F Wird von DOS beim initialisieren von Disketten benutzt.
- 0050:0030-0033 Wird von dem MODE Kommando benutzt.

Alle anderen Bytes innerhalb der 256 Byte (neu 512) die bei 0050:0000 beginnen sind für DOS reserviert.

6.7 CMOS-RAM

Die 64 Byte CMOS-RAM befinden sich auf dem RTC-Chip MC146818 und können mit den I/O Adressen 70H und 71H erreicht werden. Bei dem NEAT chip set wird ein erweiterter Speicher benutzt um die erforderlichen Angaben für die unterschiedlichen Konfigurationen unterzubringen.

Zuerst muß die gewünschte CMOS-RAM Adresse in die I/O Adresse 70H geschrieben werden. Nach einiger Zeit, die von der Maschinenkonfiguration abhängt, steht der Inhalt in der I/O Adresse 71H zur Verfügung. Er kann dann gelesen oder beschrieben werden. Für manche Aufgaben, z. B. Zeit, Datum oder Equipment, stehen besondere Interrupts zur Verfügung.

00 Sekunden

01 Alarm Sekunden

02 Minuten

03 Alarm Minuten

04 Stunden

05 Alarm Stunden

06 Wochentag

07 Datum

08 Monat

09 Jahr

0A Status Register A

- Bit 7 Update in Progress (UIP). Es kann nur gelesen werden. Ist das Bit gesetzt, ist ein Update-Zyklus unterwegs oder wird gleich beginnen. Ist es nicht gesetzt, ist keiner unterwegs und wird auch nicht in den nächsten 244 μs anfangen. Nur in diesem Fall sind Zeit-, Kalender- und Alarm-Daten gültig.
- Bit 6 - 4 Selektorbits für 22-stufigen Teiler (DV2-DV0). Das System initialisiert 010, was 32.768 kHz entspricht. Nur Bit 6 kann geändert werden.
  - 010 Operation Mode, Divider running.
  - 110 Reset Mode, Divider in reset state.
- Bit 3 - 0 Rate Selection (RS3 - RS0). Teiler Ausgangsfrequenz Selektion wird vom System mit 0110 entsprechend 1.024 kHz gesetzt.

RS value	Periodic Interrupt Rate
00H	none
01H	3.90625 ms
02H	7.8125 ms
03H	122.070 μs
04H	244.141 μs
05H	488.281 μs
06H	976.562 μs
07H	1.953125 ms
08H	3.90625 ms
09H	7.8125 ms
0AH	15.625 ms
0BH	31.25 ms
0CH	62.5 ms
0DH	125 ms
0EH	250 ms
0FH	500 ms

0B Status Register B

- Bit 7 Wenn dieses Bit gesetzt ist können die 14 Byte des Zeitzählers ausgelesen werden. Nur wenn dieses Bit nicht gesetzt ist, werden die Zähler jede Sekunde erhöht.
- Bit 6 Periodic Interrupt enable (PIE). Wenn dieses Bit gesetzt ist, ist ein Interrupt mit der Rate die von den rate (RS) und divider (DV) Bits bestimmt wird erlaubt. Es kann mit dem Interrupt 15H Funktion 83H gesetzt werden. Das System initialisiert dieses Bit mit 0.
- Bit 5 Alarm Interrupt Enable (AIE). Wenn dieses Bit gesetzt ist, wird nach Ablauf der Zeit der Interrupt 4AH (Interval Interrupt) ausgeführt. Es kann mit dem Interrupt 1AH Funktion 06H gesetzt werden. Das System initialisiert dieses Bit mit 0.
- Bit 4 Update-Ended Interrupt Enable (UIE). Wenn dieses Bit gesetzt ist, ist ein update-ended Interrupt erlaubt. Das System initialisiert dieses Bit mit 0.
- Bit 3 Square Wave Enabled (SQWE). Wird dieses Bit gesetzt ist die Square Wave Frequenz erlaubt die mit den RS Bits im Status Register A selektiert wurde. Das System initialisiert dieses Bit mit 0. Das Bit ist nicht in allen Chip-Sätzen vorhanden! Nicht benutzt bei DECstation 425.
- Bit 2 Date Mode (DM). Wenn dieses Bit gesetzt ist werden Zeit und Datum Einträge von der Update-Routine als binär betrachtet, sonst als BCD. Das System initialisiert dieses Bit mit 0.
- Bit 1 24/12 - Ist das Bit gesetzt wird der Eintrag für die Stunden als 24-Stunden-Eintrag verstanden, sonst als 12-Stunden. Das System initialisiert dieses Bit mit 1.
- Bit 0 Daylight Savings Enabled (DSE). Ist das Bit gesetzt, so ist umschalten Sommerzeit $\Longleftrightarrow$ Winterzeit erlaubt, sonst Standardzeit. Wenn das Bit gesetzt ist, wird z. B. am ersten Sonntag im April die Zeit von 1:59:59 auf 3:00:00 gesetzt und am letzten Sonntag im Oktober

von 1:59:59 auf 1:00:00 gesetzt. Die für einen Rechner gültigen Zeiten müssen den Unterlagen des Herstellers entnommen werden. Das System initialisiert dieses Bit mit 0.

0C Status Register C

- Bit 7 - 4 IRQF, PF, AF, UF - Diese Flag-Bits werden gesetzt wenn im Status-Register B AIE, PIE und UIE gesetzt werden.
  - Bit 4 wird nach jedem Update-Zyklus gesetzt und durch Lesen des Registers C gelöscht.
  - Bit 5 wird gesetzt, wenn die Alarmzeit der augenblicklichen entspricht und durch Lesen des Registers C gelöscht.
  - Bit 6 wird gesetzt, wenn der Teiler den Wert der periodischen Interrupt-Rate erreicht hat und wird durch Lesen des Registers C gelöscht.
  - Bit 7 wird gesetzt, wenn PF und PIE oder AF und AIE oder UF und UIE gesetzt sind.

$$IRQF = PF \cdot PIE + AF \cdot AIE + UF \cdot UIE$$

- Bit 3 - 0 reserved.

0D Status Register D

- Bit 7 Valid RAM Bit (VRB) oder Valid RAM Data and Time (VRT). Dieses Bit wird von dem power-sense pin abgeleitet. Es ist nicht gesetzt wenn die Baterie (Akku) für die real time clock nicht in Ordnung ist und damit die Daten im CMOS-RAM ungültig sind.
- Bit 6 - 0 reserved.

0E Diagnostic Status

- Bit 7 Power status of the real time clock chip. Ist das Bit gesetzt hatte der chip die Spannung verloren.
- Bit 6 Configuration Record (Checksum Status Indicator). Ist das Bit gesetzt ist die Checksumme ungültig.
- Bit 5 Incorrect Configuration Information. Ist das Bit gesetzt ist die Konfigurationsinformation ungültig. Die Power-on Tests benötigen:
  - Mindestens eine Diskette (Bit 0 im equipment byte gesetzt).
  - Die Angaben über den Bildschirm müssen mit der Schalterstellung und dem aktuellen Adapter übereinstimmen.
- Bit 4 Memory Size Comparison. Ist das Bit gesetzt ist die vom POST gefundene Größe des Memory nicht identisch mit der konfigurierten.
- Bit 3 Fixed Disk Adapter/Drive C Initialization Status. Ist das Bit gesetzt liegt ein Fehler im Adapter oder im Drive C vor. Es kann nicht gebootet werden.
- Bit 2 Time Status Indicator (POST validity check). Ist das Bit gesetzt so ist die Zeit ungültig.
- Bit 1 Adapters do not match configuration
- Bit 0 Time-out in reading an adapter ID

0F Shutdown Status. Die Bedeutung der Bits ist bei Interrupt 19H erklärt.

10 Diskette Drive Type.

- Bit 7 - 4 Type of first diskette drive installed:
  - 0000 No drive is present.
  - 0001 Double Sided Diskette Drive (48TPI, 360K).
  - 0010 High Capacity Diskette Drive (96TPI, 1.2MB).
  - 0011 720KB

  - 0100 1.44MB
- Bit 3 - 0 Type of second diskette drive installed:
  - 0000 No drive is present.
  - 0001 Double Sided Diskette Drive (48TPI, 360K).
  - 0010 High Capacity Diskette Drive (96TPI, 1.2MB).
  - 0011 720K
  - 0100 1.44MB

11 reserved

12 Fixed Disk Type

- Bit 7 - 4 Defines the type of fixed disk drive installed (drive C):
  - 0000 no fixed disk drive is present.
  - 0001 - 1110 define type 1 bis 14.
  - 1111 Definition der Typen 16 bis 255 befindet sich im extended Byte.
- Bit 3 - 0 Defines the type of fixed disk drive installed (drive D):
  - 0000 no fixed disk drive is present.
  - 0001 - 1110 define type 1 bis 14.
  - 1111 Definition der Typen 16 bis 255 befindet sich im extended Byte.

13 reserved (System flag 1 bei 486).

- Bit7-5 Reserved.
- Bit4 Keyboard Rate/Delay Selection 1 = fast.
- Bit3-2 Reserved.
- Bit1 Network Server Mode. Erlaubt das Booten von der Harddisk, das Keyboard ist aber disabled bis das Passwort eingegeben wurde. 1 = Enable Netword Server Mode.
- Bit0 = 1 Valid Password installed.

14 Equipment

- Bit 7 - 6 Indicates the number of diskette drives installed:
  - 00 1 drive
  - 01 2 drives
  - 10 reserved
  - 11 reserved
- Bit 5 - 4 primary display:
  - 00 das erste display ist an einen Adapter mit eigenem BIOS angeschlossen:
    - Enhanced Graphics Adapter.
    - Professional Graphics Adapter
  - 01 das erste Display ist im 40-Spalten mode und an einen color/graphics adapter angeschlossen.
  - 10 das erste Display ist im 80-Spalten mode und an einen color/graphics adapter angeschlossen.
  - 11 das erste Display ist an den monochrom display und printer adapter angeschlossen.
- Bit 3 - 2 nicht benutzt.
- Bit 1 Math Coprocessor presence bit:
  - 0 Math Coprocessor not installed.

  - 1 Math Coprocessor installed.
- Bit 0 Diskette drive presence bit:
  - 0 Diskette drive not installed.
  - 1 Diskette drive installed.

15-16 Low and High Base Memory. Mit Interrupt 12H wird dieser Wert zurückgegeben.

Gültige Werte:

- 0100H 256k system board RAM
- 0200H 512k system board RAM
- 0280H 640k

17-18 Low and High Expansion Memory. Bei 486 ist in 1B das MSB des 24-Bit Wertes.

Gültige Werte:

- 0200H 512k expansion memory
- 0400H 1024k expansion memory
- 0600H 1536k expansion memory bis
- 3C00H 15360k expansion memory (15M maximum)

19 Drive C Extended. 10H bis FFH geben die Type 16 bis 255 an.

1A Drive D Extended. 10H bis FFH geben die Type 16 bis 255 an.

1B bis 2D reserved bei AT's.

1B High Byte of Extended RAM 24-bit word (see 17-18).

1C System Flag 2

- Bit7 = 1 Extended Memory Limit by INT 15h function 88h for VDISK and OS/2.
- Bit6 = reserved.
- Bit5 = 0 enable caching of split memory.
- Bit4 = 0 enable caching in the range of 12 — 16 MB.
- Bit3 = 0 enable caching in the range of 8 — 12 MB.
- Bit2 = reserved.
- Bit1-0 = Memory test during Power-on Diagnostics.
  - 00 = full test.
  - 01 = moderate test.
  - 10 = minimal test.
  - 11 = memory clear.

1D System Flag 3

- Bit7-6 reserved.
- Bit5 shadow video ROM at C000 0000.
- Bit4 enable system RAM in 512 — 640 KB range.
- Bit3 shadow option ROM at C800 0000.
- Bit2 shadow option ROM at D000 0000.
- Bit1 shadow option ROM at D800 0000.

- Bit0 reserved.

1E Memory Split Boundary. 0 = no split.

1F Control Panel Configuration.

- Bit7-5 reserved.
- Bit4 0 = enable programmable ON/OFF control.
- Bit3-1 Speaker Volume.
- Bit0 0 = enable front panel reset control.

20 Motherboard Memory Bank Configuration.

21 Motherboard Memory Bank Density.

22 User Diskette Language.

23-24 i860 Reserved Memory (23 = LSB, 24 = MSB).

25 HDU SCSI Parameter Translation.

- Bit
- Bit
- Bit

26-2D Reserved.

2E-2F Checksum. High und low byte der Checksumme. Sie wird nur über die Adressen 10 bis 2D gebildet (2E = MSB, 2F = LSB).

30-31 Low and High Expansion Memory. Mit Interrupt 15H Funktion 88H wird dieser Wert zurückgegeben. Er bezieht sich nur auf das Memory oberhalb von 1M.

Gültige Werte:

- 0200H 512k expansion memory
- 0400H 1024k expansion memory
- 0600H 1536k expansion memory bis
- 3C00H 15360k expansion memory (15M maximum)

32 Date Century. BCD value for the century.

33 Information Flag (POD)

- Bit7 When set, this bit indicates that the top 128k of base memory is installed. 512 to 640KB is present.
- Bit6 This bit is set to instruct the Setup utility to put out a first user message after initial setup. Bei DECstation 425 reserved Memory für i860 vorhanden, Größe in Byte 23 (LSB) und 24 (MSB).
- Bit5 - 0 reserved bei AT's.
- Bit5 = 1 wenn Console vorhanden und funktioniert.
- Bit4 = 1 wenn monochrome Display Adapter.
- Bit3 - 2 reserved.
- Bit1 = 1 wenn NVRAM Configuration Data Status is invalid.
- Bit0 = 1 wenn NVRAM Checksum Status is invalid.

34-3F reserved on AT's.

34 POD Information Flag 2.

- Bit7 - 4 Number of serial interface ports detected during POD.
- Bit3 = 1 wenn cache Test Error ergibt.
- Bit2 reserved.
- Bit1 = 1 wenn i860 vorhanden.
- Bit0 = 1 wenn Weitek Prozessor vorhanden.

35 High byte Extended Memory found 24-bit word (see 30-31).

36 CPU Type.

37 CPU Revision Level.

38-3E Password, stored as scan codes.

3F Password Checksum.

Nach diesem definierten Bereich ist bei den 'NEAT' chips Platz für den Eintrag der Erweiterungen. Nur bei absolut kompatiblen sind die Adressen und die Inhalte identisch. Es sollte deshalb immer nur mit den Interrupts gearbeitet werden, weil diese Anpassung im BIOS und IO.SYS vorgenommen wird. æ

Kapitel 7

Interrupt

Es sind zwei Arten von Interrupt zu unterscheiden, Hardware-Interrupt und programmierter Interrupt. Interrupt bedeutet die augenblickliche Arbeit unterbrechen und auf eine Anforderung reagieren. Sowohl die Anforderung als auch die Reaktion kann in einem System auf die unterschiedlichste Weise implementiert sein. Die Anforderung kann z. B. durch setzen eines Bits in einem Flagregister des Gerätes (passiv) oder durch blockieren einer Leitung (aktiv) erfolgen. Die Reaktion kann z. B. das Absuchen der Register (polling) oder das sofortige Bedienen der Anforderung sein. Dabei können sowohl Hardware- als auch Software-Prioritäten eingebaut sein. Diese Prioritäten können sich aus der Abarbeitung einer Tabelle, der physikalischen Nähe zur CPU oder der Rotation eines Mask-Registers ergeben. Auch eine Mischung kann sinnvoll sein. Daraus ergeben sich nicht nur die Reaktionszeiten sondern auch die möglichen Übertragunsgeschwindigkeiten.

7.1 Hardware-Interrupt

Der Prozessor hat zwei Hardware Interrupt-Levels (Leitungen):

- NMI = non maskable interrupt
- INTR = interrupt request

Der Vector der NMI-Leitung ist fest auf 2 vergeben. Es wird auch kein INTA erzeugt.

INTR ist nur erlaubt, wenn im F-Register IF = 1 (INTR enable) gesetzt ist. Ist INTR erlaubt und gültig, so wird INTA erzeugt und damit ein 8-Bit Vector eingelesen. Für diese Verwaltung werden Interrupt Controller, z.B. 8259A, benötigt. Diese müssen bei jedem 'Reset' neu programmiert, d.h. der vorhanden Hardware und Software angepasst werden.

Die IRQ-Leitungen erscheinen im Interrupt Request Register (IRR) des 8259A als den Leitungen entsprechende Bit's. Diese werden in Verbindung mit dem Interrupt Mask Register (IMR) und dem programmierten Priority Resolver an das Interrupt Service Register (ISR) mit dem nächsten INTA übergeben. Damit wird der Vector selectiert, der aus der programmierten 'Basis-Vector-Adresse' und der IRQ-Nr. besteht. Der zweite INTA legt diesen Vector auf den Daten-Bus. Der Processor multipliziert den Vector mit vier. Im real address mode steht an dieser Adresse die Adresse der Interrupt Service Routine (CS:IP 20-Bit). Sie muß die Anforderung des device für den driver aufarbeiten, d.h. für das vorgeschlagene multitasking das Paket packen und in die queue stecken, und sobald als möglich den Interrupt verlassen.

7.2 Programmierter-Interrupt

Mit 8-Bit Vector können 256 Interrupts verwaltet werden. Da keine Anwendung diese für Hardware benutzt, werden einige dem Programmierer als System- oder BIOS-Calls zur Verfügung gestellt. Alle Interrupts können mit dem Befehl 'INT Nr.H' ausgeführt werden. Die Funktion wird dabei im allgemeinen in AH, eine Differenzierung der Funktion wird in AL übergeben. Eine ausführliche Beschreibung für einige Interrupts und deren Funktionen ist in "MS-DOS Programmer's Reference Manual" von Microsoft zu finden.

7.3 Gemeinsamkeiten

Beide Interrupts, INTR von der Hardware mit Vector über den Controller und INT mit angegebener Nr., ergeben den gleichen Zustand des Prozessors.

- Der SP wird um zwei erniedrigt
- Das Flag-Register wird auf den Stack geschrieben
- Die IF und die TF Flag werden nach null gesetzt
- Der SP erneut um zwei erniedriegt
- Das CS Register auf den Stack geschrieben
- CS wird mit dem high-order word des vectors geladen
- Der SP erneut um zwei erniedriegt
- Der IP wird auf den Stack geschrieben
- Der IP wird mit dem low-order word des vectors geladen

Damit ist der Processor bereit, die erste Intruktion des "Interrupt Servers" als nächsten Befehl abzuarbeiten. Um den vorhergehenden Zustand zu erreichen, muß der Server als letztes den Befehl IRET benutzen. Damit wird eben gezeigte Reihenfolge umgekehrt. Dazwischen muß der Server aber alle benutzten Register und Flags selbst verwalten.

7.4 DMA-Interrupt

Es sind sieben DMA Kanäle über zwei 8237A-5 Controller vorhanden. Die Kanäle 0 bis 3 (im ersten Controller) sind für 8-bit Transfers vorgesehen. Im DMA page register stehen die Adressbits A16-A23 und im Controller die Bits A0-A15. Das "byte high enable" (BHE) wird durch Invertieren des Bits A0 erzeugt. Es können also 16MByte angesprochen werden.

Bei manchen (Intel) 80386 Systemen werden den 8-bit DMA-Kanälen 4-bit upper page Register und 8-bit lower page Register zugeordnet. Den 16-bit Kanälen werden ebenfalls 4-bit upper, aber nur 7-bit lower page Register zugeordnet. Damit ergibt sich eine 28-bit Adresse und dadurch mögliche 256 MB physikalischer Speicher.

Der Kanal 4 (Ch0 des zweiten Controller) wird zum kaskadieren benutzt.

Die Kanäle 5 bis 7 sind für 16-bit transfers vorgesehen. Im DMA page register stehen die Adressbits A17-A23. Das Datenbit D0 wird beim programmieren des page registers für die Kanäle 5 bis 7 nicht benutzt. Die Adressbits A1-A16 stehen im Controller. Das bit A0 und BHE werden logisch 'NULL' gesetzt. Deshalb können nur even address transfers gemacht werden. Das base address register muß mit dem halben Wert der realen Adresse belegt werden. Auch der word count bezieht sich auf 16-bit (word, nicht Byte).

NAME	SIZE	NUMBER
Base Address Register	16Bits	4
Base Word Count Register	16Bits	4
Current Address Register	16Bits	4
Current Word Count Register	16Bits	4
Temporary Address Register	16Bits	1
Temporary Word Count Register	16Bits	1
Status Register	8Bits	1
Command Register	8Bits	1
Temporary Register	8Bits	1
Mode Register	6Bits	4
Mask Register	4Bits	1
Request Register	4Bits	1

Tabelle 7.1: Register des 8237

7.4.1 Interrupt Controller 8237A

Die hier beschriebenen Funktionen beziehen sich auf den einzelnen 8237-Chip. Durch den Einbau in einen konkreten Rechner oder die Integration in ein Chip-set können die Funktionen eingeschränkt worden sein. Man müßte deshalb in den meistens nicht vorhandenen Unterlagen des Herstellers nachsehen!

Es sind auch kompatible Nachfolger auf dem Markt mit gesteigerter Leistung. So z. B. von NEC der μ PD71037 mit maximal 10 MHz Taktfrequenz und damit 3,2 MByte Datenübertragungsrate pro Sekunde im Normal Transfer-Modus und 5 MByte/s im Compression-Transfer-Mode.

Die Register des 8237 sind in Tabelle 7.1 auf Seite 159 aufgelistet.

Single Transfer Mode

Im single transfer mode wird nur ein Transfer ausgeführt und dann der Wortzähler erniedrigt und der Adressenzähler entweder erhöht oder erniedrigt. Wenn der Wortzähler auf null geht wird das Signal TC (terminal count) generiert und der Kanal neu initialisiert falls es so vorher programmiert wurde. Zwischen zwei Einzeltransfers kann dem Prozessor erlaubt werden einen Befehl abzuarbeiten.

Block Transfer Mode

Im block transfer mode wird bis zum TC transferiert oder ein externes $\overline{EOP}$ (end of process) auftritt. Der Kanal wird neu initialisiert falls es so vorher programmiert wurde.

Demand Transfer Mode

Wenn der demand mode programmiert wird, wird der Transfer fortgesetzt bis das TC- oder ein externes $\overline{EOP}$-Signal auftritt oder die Anforderung (DREQ) zurückgenommen wird. Die Werte der Adresse und Wortzähler werden in den "Current" Registern gespeichert. Nur $\overline{EOP}$ kann den Kanal neu initialisieren. $\overline{EOP}$ wird erzeugt, wenn TC erreicht ist oder als externes Signal gegeben.

Cascade Mode

Dieser mode wird benutzt um mehrere Controller zusammenzuschalten. Die HRQ und HLDA Signale der zusätzlichen Controller werden mit den Signalen DREQ und DACK eines Kanals der ersten Ebene verbunden. S. Bild 7.1 auf Seite 160

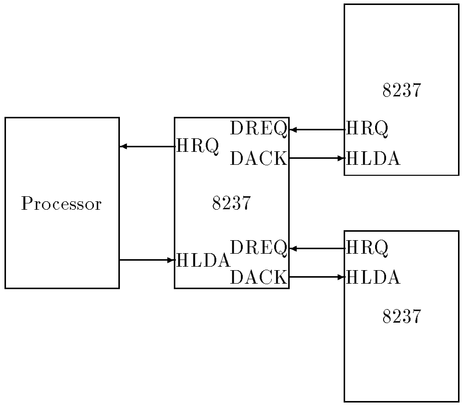


Abbildung 7.1: Cascaded 8237

Memory-to-Memory

Memory-to-Memory DMA-Transfer kann mit den Kanälen 0 und 1 vorgenommen werden. Der Transfer wird angestoßen durch das setzen des DREQ des Kanals 0. Das current address register des Kanals 0 ist die Quelle und wird incrementiert oder decrementiert. Das gelesene Byte wird im temporären Register, es ist nur eins pro chip vorhanden, zwischengespeichert. Der Kanal 1 benutzt dann sein current address register und das temporäre Register und incrementiert oder decrementiert sein current address register. Der current word count des Kanals 1 wird decrementiert. Wird damit die 0 erreicht, wird TC und $\overline{EOP}$ generiert und damit der Transfer abgeschlossen.

Kanal 0 kann so programmiert werden, daß er immer die selbe Adresse behält. Damit kann ein Block mit dem selben Datum gefüllt werden.

Autoinitialize

Durch setzen eines Bits im Mode Register wird der Kanal selbstinitialisierend. Wenn $\overline{EOP}$ erzeugt wird, werden die current address und word count Register mit den Werten aus den Base Registern erneuert. Die Base Register werden nicht verändert. Das mask Bit wird nicht gesetzt. Nach dem selbstinitialisieren kann der selbe Vorgang wiederholt werden.

Priority

Der 8237 hat zwei mögliche Prioritätschemata.

1. Festgelegte Prioritäten der Kanäle, so daß der Kanal mit der höchsten Nummer die niedriegste Priorität hat. Ein angefangener DMA-Transfer kann aber nicht mehr unterbrochen werden.

2. Rotierende Priorität. Der zuletzt benutzte Kanal bekommt die niedriegste Priorität und die anderen rotieren entsprechend, so daß kein Kanal das System auf Dauer belegen kann (64k-Grenze).

Bit Nr.	Wert	Bedeutung
BIT0	0	Memory-toMemory disable
BIT0	1	Memory-toMemory enable
BIT1	0	Channel 0 address hold disable
BIT1	1	Channel 0 address hold enable
BIT1	X	If Bit0 = 0
BIT2	0	Controller enable
BIT2	1	Controller disable
BIT3	0	Normal timing
BIT3	1	Compressed timing
BIT3	X	If Bit0 = 1
BIT4	0	Fixed priority
BIT4	1	Rotating priority
BIT5	0	Late write selection
BIT5	1	Extended write selection
BIT5	X	If bit3 = 1
BIT6	0	DREQ sense active high
BIT6	1	DREQ sense active low
BIT7	0	DACK sense active low
BIT7	1	DACK sense active high

Tabelle 7.2: Command Register 8237

Bit Nr.	Wert	Bedeutung
BIT0+1	00	Select channel 0
	01	Select channel 1
	10	Select channel 2
	11	Select channel 3
BIT2+3	00	Verify transfer
	01	Write transfer
	10	Read transfer
	11	illegal
	XX	If bits 6 and 7 = 1
BIT4	0	Autoinitialize disable
	1	Autoinitialize enable
BIT5	0	Address increment select
	1	Address decrement select
BIT6+7	00	Demand mode select
	01	Single mode select
	10	Block mode select
	11	Cascade mode select

Tabelle 7.3: Mode Register 8237

Address Generation

Um Gehäuseanschlüsse zu sparen, werden die Adressen gemultiplexed. Die oberen Bits müssen also extern zwischengespeichert werden. Ein updaten dieser Bits ist erst nach 256-Transfers nötig, was 255 clock cycles spart.

Current Address Register

Jeder Kanal hat ein 16-Bit current address register. Es wird automatisch incrementiert oder decrementiert. Der Originalwert wird hergestellt, wenn autoinitialize erlaubt ist und $\overline{EOP}$ auftritt.

Current Word Register

Jeder Kanal hat ein 16-Bit current word count register. Es wird nach jedem Transfer decrementiert. Wenn der Wert null wird wird ein TC (Terminal count) Signal generiert. Der originale Wert wird restauriert wenn autoinitialize erlaubt ist. Autoinitialize wird nur gemacht, wenn $\overline{EOP}$ erzeugt wird.

Base Registers

Jeder Kanal hat ein Paar base address und word count register. Diese 16-Bit Register halten die originalen Werte. Diese werden nach Abschluß eines Transfers, wenn autoinitialeze erlaubt ist, mit $\overline{EOP}$ in die current Register geschrieben. Base und current Register werden beim programmieren gleichzeitig beschrieben.

Command Register

Dieses 8-Bit Register kontrolliert den 8237. Es wird durch "Reset" gelöscht. S. Tabelle 7.2 auf Seite 161.

Bit Nr.	Wert	Bedeutung
BIT0+1	00	Select channel 0
	01	Select channel 1
	10	Select channel 2
	11	Select channel 3
BIT2	0	Reset request bit
	1	Set request bit
BIT3-7	X	Don't care

Tabelle 7.4: Request Register 8237

Bit Nr.	Wert	Bedeutung
BIT0+1	00	Select channel 0
	01	Select channel 1
	10	Select channel 2
	11	Select channel 3
BIT2	0	Reset mask bit
	1	Set mask bit
BIT3-7	X	Don't care

Tabelle 7.5: Mask Register 8237

Mode Register

Jeder Kanal hat ein 6-Bit mode Register. Die Bits 0 und 1 des zur Programmierung benutzten Byte werden zur Selektion des Kanals benutzt. S. Tabelle 7.3 auf Seite 161.

Request Register

Der 8237 kann auf Anforderungen sowohl von der Software als auch von der Hardware (DREQ) reagieren. Jeder Kanal ist mit einem Bit in dem 4-Bit request register verbunden. Jedes Bit wird getrennt gesetzt oder zurückgesetzt. Jedes Bit wird zurückgesetzt wenn TC auftritt oder ein externes $\overline{EOP}$ erzeugt wird. Das ganze Register wird durch RESET gelöscht. Eine Softwareanforderung ist nur im Block mode erlaubt. Bei Memory-to-Memory sollte das Bit für Kanal 0 gesetzt werden. S. Tabelle 7.4 auf Seite 162.

Mask Register

Mit dem Maskbit kann für jeden Kanal getrennt die Anforderung durch DREQ verboten werden. Jedes Maskbit wird gesetzt, wenn der dazugehörende Kanal ein $\overline{EOP}$-Signal erzeugt und der Kanal nicht für autoinitialize programmiert wurde. Jedes Bit kann einzeln gesetzt oder zurückgesetzt werden. Auch das ganze Register kann zurückgesetzt werden. Damit wird jede DMA-Anforderung unterbunden.

Status Register

Das Status Register kann nur gelesen werden. Es hält Informationen über jeden Kanal ob ein TC erreicht wurde und ob eine Anforderung ansteht. Die Bits 0-3 werden immer mit einem TC oder einem $\overline{EOP}$ gesetzt. Diese Bits werden mit RESET oder "read status" zurückgesetzt. Die Bits 4-7 werden von dem Gerät gesetzt, das an den Kanal angeschlossen ist.

Temporary Register

Das temporäre Register wird benutzt, um Information während eines Memory-to-Memory Transfers zu halten. Nach dem Ende des Transfers kann das letzte Wort noch gelesen werden. Es wird von RESET gelöscht.

Software Commands

Die Tabelle 7.7 auf Seite 163 gibt die möglichen Befehle wieder. Mit Offset 7 können alle Maskbits bearbeitet werden. Vergleiche Seite 111 und Seite 118.

Bit Nr.	Bedeutung
0	channel 0 has reached TC
1	channel 1 has reached TC
2	channel 2 has reached TC
3	channel 3 has reached TC
4	channel 0 request
5	channel 1 request
6	channel 2 request
7	channel 3 request

Tabelle 7.6: Status Register 8237

Offset	Read	Write
0	Status Register	Command Register
1	Illegal	Request Register
2	Illegal	Single Mask Register Bit
3	Illegal	Mode Register
4	Illegal	Clear Byte Pointer F/F
5	Temporary Register	Master Clear
6	Illegal	Illegal
7	Illegal	All Mask Register Bits

Tabelle 7.7: Software Commands 8237

7.5 IRQ-Interrupt

Das IRQ-Interrupt Verwaltungssystem besteht aus zwei programmierbaren Controllern die jeweils 8 Leitungen bedienen können. Für die Weiterleitung des einen zum anderen geht jeweils ein Interrupt verloren. An dem 8-Bit Steckplatz stehen die IRQ3 bis IRQ7 und IRQ9 zur Verfügung. Der IRQ9 belegt den Platz des IRQ2 des XT. An dem 16-Bit Erweiterungssteckplatz sind die IRQ10 bis IRQ12,IRQ14 und IRQ15 zu belegen.

Die Interruptrequestleitungen enden in dem 'interrupt request register' (IRR) und die Anforderungen werden dort gespeichert. Sie gelangen über den 'priority resolver' unter Berücksichtigung des 'interrupt mask registers' in das 'in service register' (ISR).

7.5.1 Interrupt Controller 8259A

Der Controller 8259A verfügt über 7 Commands, vier Initialisation Command Words (ICW) und drei Operation Control Words (OCW). Zuerst müssen die Controller initialisiert werden. Eine initialisierungs Sequence beginnt mit Adress-Bit $A_0 = 0$ und Data-Bit $D_4 = 1$. Danach folgen, je nach Mode, 2 bis 3 ICW's. Danach können OCW's folgen, die auch im laufenden Betrieb den Mode umschalten können:

1. fully nested mode
2. rotating priority mode
3. special mask mode
4. polled mode

Das BIOS programmiert die Controller vor dem POST folgendermassen:

- Es wird mit OUT DX,AL nach Adresse 20H (DX=20H entsp. Address Contr.1) und nach Adresse A0H (DX=A0H entsp. Address Contr.2) eine 11H geschrieben. Damit wird ein ICW-Ladecyclus eingeleitet und auf Grund des gesetzten Bits D_0 auch ICW4 erforderlich.
- Danach wird die "Vectorbasisadresse" in ICW2 nach T_3 bis T_7 mit 8H bzw. 70H geschrieben. Damit beginnt der Interrupt-Vectorbereich des Controllers 1 mit INT 8 und der des Controllers 2 mit INT 70, d.h. IRQ0=INT 8 und IRQ8=INT 70. Die Interrupt-Service Adressen stehen also bei 0000:0020 für INT 8 und 0000:01C0 für INT 70.

  Achtung:

  Low-Byte first und CS vor IP im Memory.
- Im ICW3 wird der Master und der Slave kenntlich gemacht.

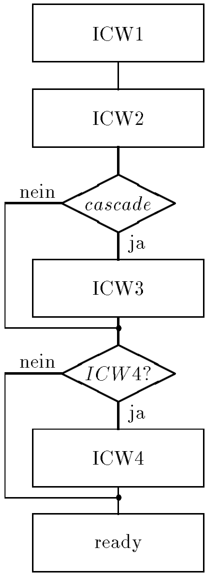


Abbildung 7.2: Ablaufplan der Initialisierungs-Sequenz.

- Im ICW4 wird jeweils nur der μPMode selektiert.
- OCW1 wird in der startup Phase mit 77H belegt; damit werden zur Zeit unerwünschte Interrupts maskiert, in diesem Fall alle.

Die Register können auch gelesen werden. Das gewünschte Register muß aber vorher selektiert worden sein. Diese Auswahl wird bis zur nächsten Änderung beibehalten.

7.5.2 Funktionen mit gerader Adresse

Die Belegung der Register des 8259A Interrupt Kontrollers bei gerader Adresse also Adressbit 0 = 0. Ob gelesen oder geschrieben wird, wenn möglich, wird von dem Befehl IN oder OUT bestimmt.

ICW1

A0	D7	D6	D5	D4	D3	D2	D1	D0

- D0 = IC4
  - 1 = ICW4 needed
  - 0 = no ICW4 needed
- D1 = SNGL
  - 1 = single mode (nur ein 8259A vorhanden; es wird kein ICW3 benötigt)
  - 0 = cascade mode (mehrere 8259A vorhanden)
- D2 = ADI
  - 1 = Address interval of 4 ($A_0 bis A_4$ werden vom 8259A erzeugt)
  - 0 = Address interval of 8 ($A_0 bis A_5$ werden vom 8259A erzeugt)
- D3 = LTIM
  - 1 = Level triggered mode der IRQ Leitungen.
  - 0 = Edge triggered mode der IRQ Leitungen.
- D4 = 1
  - 1 = muß 1 sein, sonst
  - 0 = ist es OCW2 oder OCW3
- D5-D7 = A5-A7 Interrupt Vector Address bits (MCS-80/85 mode only)
- A0 = 0

OCW2

A0	D7	D6	D5	D4	D3	D2	D1	D0

- D0-D2 Interrupt request level to be acted upon (0 - 7)
- D3 = 0 sonst OCW3
- D4 = 0 sonst ICW1
- D5-D7

- 001 non-specific eoi-command
- 011 specific eoi-command
- 101 rotate on non-specific eoi-command
- 100 rotate in automatic eoi mode (set)
- 000 rotate in automatic eoi mode (clear)
- 111 rotate on specific eoi-command
- 110 set priority command
- 010 no operation

- A0 = 0

OCW3

A0	D7	D6	D5	D4	D3	D2	D1	D0

- D0-D1
  - 00 no action
  - 10 no action
  - 01 read IR register on next RD pulse
  - 11 read IS register on next RD pulse
- D2 = 1 = poll command
- D3 = 1 sonst OCW2
- D4 = 0 sonst ICW1
- D5-D6
  - 00 no action
  - 10 no action
  - 01 reset special mask
  - 11 set special mask
- D7 = 0
- A0 = 0

7.5.3 Funktionen mit ungerader Adresse

Die Belegung der Register des 8259A Interrupt Kontrollers bei ungerader Adresse also Adressbit 0 = 1. Ob gelesen oder geschrieben wird, wenn möglich, wird von dem Befehl IN oder OUT bestimmt.

ICW2

A0	D7	D6	D5	D4	D3	D2	D1	D0

- D0-D2 = Adressbit A8-A10 des Interrupt Vectors im MCS80/85 mode
- D3-D7 = Adressbit A11-A15 des Interrupt Vectors im MCS80/85 mode oder T3-T7 im 8086 mode
- A0 = 1

ICW3 (master device)

A0	D7	D6	D5	D4	D3	D2	D1	D0

An die Leitung des gesetzten Bits D0 bis D7 ist ein slave angeschlossen.
A0 = 1.

ICW3 (slave device)

A0	D7	D6	D5	D4	D3	D2	D1	D0

Die Bits D0-D2 enstsprechen der Nummer des Kontrollers (0-7).
A0 = 1.

ICW4

A0	D7	D6	D5	D4	D3	D2	D1	D0

- D0 = 1 = 8086/8088 mode sonst MCS-80/85 mode
- D1 = 1 = auto eoi sonst normal eoi
- D2-D3
  - X0 non bufferd mode
  - 01 bufferd mode/slave
  - 11 bufferd mode/master
- D4 = special fully nested mode
- D5-D7 = 0
- A0 = 1

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0

Die Bits D0 - D7 entsprechen den mask bits im IMR. Ist das Bit nicht gesetzt ist dem Kanal ein Interrupt erlaubt.
A0 = 1.

7.6 Interrupt Level der I/O Device

Es folgt eine Liste der Interrupt-Level in fallender Priorität. Danach eine Liste der DMA-Interrupts.

- NMI- Parity check error
- IRQ 0 System timer
- IRQ 1 Keyboard
- IRQ 2 Rerouting from INT-Controller 2 (INT15H Funktion C0H Byte 4)
- IRQ 3 Serial Port 2 (NI5010-Ethernet Controller)
- IRQ 4 Serial Port 1

- IRQ 5 Parallel Printer Port 2 (IEEE488)
- IRQ 6 Floppy Disk Adapter
- IRQ 7 Parallel Printer Port 1
- IRQ 8 Real time clock
- IRQ 9 Rerouting to IRQ2 der Steckerleiste
- IRQ10 Spare
- IRQ11 Spare
- IRQ12 Spare
- IRQ13 Arithmetic Processor
- IRQ14 Hard Disk Adapter
- IRQ15 Spare (Data Translation 16Bit)

DMA-Interrupts

- DRQ0 8-Bit device
- DRQ1 8-Bit device (SDLC) (NI5010-Ethernet Controller)
- DRQ2 8-Bit device (Diskette) (CAMAC)
- DRQ3 8-Bit device (IEEE488)
- DRQ4 cascade DRQ0 to DRQ3
- DRQ5 16-Bit device (Data Translation 16Bit)
- DRQ6 16-Bit device
- DRQ7 16-Bit device

7.7 Interrupts by number

Die folgenden Interrupts werden von der CPU oder anderer Hardware erzeugt oder werden von Programmen als Software-Interrupts benutzt. Bei den Software-Interrupts sind drei Arten zu unterscheiden: BIOS-Interrupts, die das BIOS zur Verfügung stellt oder selbst benutzt; Betriebssystem-Interrupts, z. B. die Systemcalls des Interrupt 21H; benutzerdefinierte Interrupts.

IBM hat von INTEL reservierte Hardware-Interrupts überschrieben oder als Software-Interrupts benutzt. Sollte bei einem BOUND Error der Drucker den Bildschirminhalt ausdrucken, so kann man sicher sein einen INTEL-Processor und ein IBM-Betriebssystem zu haben!

Zuerst werden die Bedeutungen von INTEL und gegebenenfalls dann die von IBM genannt.

- 00 Divide by zero. Dieser Interrupt wird ausgeführt, wenn der Befehl DIV oder IDIV nicht in einer vernünftigen Zeit beendet werden kann. Er wird auch ausgeführt wenn der Quotient nicht in das Register (AX oder AL) passt.
- 01 Single-step. Wenn die trap flag (Bit 8 im flag register) gesetzt ist, wird nach jedem Befehl der Interrupt 01H ausgeführt. Genutzt wird das beim DEBUG mit Trace.

- 02 Nonmaskable (NMI). Dieser Interrupt wird erzeugt, wenn Memory Parity Error vorhanden ist. MS-DOS besitzt keinen Handler dafür weil der Fehler sehr hardwareabhängig ist. Deshalb ist im BIOS ein "Null"-Handler vorhanden. Die Reaktion auf diesen Interrupt ist also vom Hersteller abhängig.
- 03 Breakpoint Trap. Eine Beschreibung des Interrupts 03 kann bei DEBUG oder SYMDEB unter Breakpoint gefunden werden.
- 04 Overflow Trap. Wenn das Overflow Bit (Bit 11 im flag register) gesetzt ist, wird dieser Interrupt ausgeführt, wenn im Programm der Befehl INTO auszuführen ist.
- 05 BOUND Range Exceeded (manchmal auch Print-Screen). Der Interrupt 05 wird ausgelöst, wenn eine BOUND Anweisung ausgeführt wird um den Wert eines Array Index zu testen und der Test negativ ausfällt, weil die bei der Anweisung angegeben Grenzen überschritten werden. Der Interrupt-Handler sollte den Wert des Index korrigieren, weil nach IRET die CPU die BOUND Anweisung erneut ausführt.

  Manche Systeme oder Programme benutzen den Interrupt 05, z. B. sehr oft für PrtSc (Print screen bei IBM).
- 06 Invalid Opcode. Wenn die augenblickliche Anweisung nicht erlaubt ist, z.B. protected mode Befehle im real address mode, wird der Interrupt 06 ausgeführt. Bei IBM nicht benutzt (IRET), deshalb wird die Benutzung dieser Befehle nicht erkannt.
- 07 Coprocessor not available. Bei IBM nicht benutzt.
- 08 Double-Fault. Bei IBM Hardware IRQ0 (timer tick).
- 09 Coprocessor Segment Overrun. Bei IBM Hardware IRQ1 (keyboard).
- 0A Invalid Task State Segment (TSS). Bei IBM Hardware IRQ2 (I/O channel or slave 8259A).
- 0B Segment not present. Bei IBM Hardware IRQ3 (COM2).
- 0C Stack. Bei IBM Hardware IRQ4 (COM1).
- 0D General Protection. Bei IBM Hardware IRQ5 (fixed disk oder LPT2).
- 0E nur bei 80386 (Page Fault). Bei IBM Hardware IRQ6 (Floppy-disk).
- 0F reserved. Bei IBM Hardware IRQ7 (LPT1).
- 10 Coprocessor Error. Bei IBM Video service.
- 11 reserved. Equipment information. Alignment error bei 80486.
- 12 reserved. Memory size.
- 13 reserved. Disk I/O Service.
- 14 reserved. Serial Port Service.
- 15 reserved. Cassette/Network Service. (Extended Memory (MS) und mehr).
- 16 reserved. Keyboard Service.
- 17 reserved. Printer Service.
- 18 reserved. ROM Basic.
- 19 reserved. Restart System. Oder VDISK !!!
- 1A reserved. Get/Set Time/Date.

- 1B reserved. Control-Break (user defined).
- 1C reserved. Timer tick (user defined).
- 1D reserved. Video Parameter Pointer.
- 1E reserved. Disk Parameter Pointer.
- 1F reserved. Graphics character tabel. Wird vom Befehl "GRAFTABL" ausgefüllt.

Hier hört die Hardwarebelegung der Interrupts durch den Processor von Intel auf.

- 20 Terminate Program (I21F31 oder I21F4C ist besser).
- 21 MS-DOS System Calls.
- 22 Terminate Routine Address.
- 23 Control-C Handler Address.
- 24 Critical Error Handler Address.
- 25 Absolute Disk Read.
- 26 Absolute Disk Write.
- 27 Terminate and Stay Resident.
- 28 Idle Interrupt.
- 2A DECnet-DOS to NETBIOS.
- 2E change master environment (undocumented !).
- 2F Multiplex Interrupt.
- 40 Reserved für BIOS. (revector diskette pointer wenn fixed disk und diskette adapter benutzt wird.)
- 41 Fixed disk parameter für erste disk.
- 46 Fixed disk parameter für zweite disk.
- 4A Interval Interrupt.
- 5C DECnet-DOS to NETBIOS.
- 60 used by debugg example COMMSCOP.
- 67 EMM (expanded memory).
- 69 DECnet-DOS CTERM services.
- 6A DECnet-DOS LAT services.
- 6E DECnet-DOS Network I/O Control Block.
- 70 IRQ08 Real-time clock.
- 71 IRQ09 software redirected to IRQ2 der Steckerleiste.
- 72 IRQ10 spare.
- 73 IRQ11 spare.

- 74 IRQ12 spare.
- 75 IRQ13 Numeric coprocessor.
- 76 IRQ14 fixed-disk controller.
- 77 IRQ15 spare. (Data Translation card 16Bit).

7.8 Interrupt sharing

IBM hat einen Hardware Standard festgelegt, der es erlaubt einen Interrupt-Level von mehreren Device zu benutzen. Er wird im Folgenden erläutert.

Da der DMA-Controller auf "high oder low level sensitiv" programmierbar ist, würde ich behaupten, daß das auch für DMA gilt!

Da Interruptroutinen im ROM nicht für protected mode existieren, war es die Absicht diesen Standard nur im real address mode zu benutzen.

7.8.1 Überblick

Die meisten Adapter, die Interrupt-Request benutzen, halten die IRQ-Leitung auf dem low-level und ziehen sie bei Anforderung auf den high-level. Dagegen wird bei den Adaptern, die den shared-interrupt Standard benutzen, die IRQ-Leitung mit einem pull-up Wiederstand auf high-level gezogen. Jeder an diese IRQ-Leitung angeschlossene Adapter kann einen Interrupt durch einen low-level Puls erzeugen. Die zweite Flanke erzeugt den Interrupt. Der Puls soll zwischen 125 und 1000 Nanosekunden lang sein.

Der Adapter muß ein Interrupt Status Bit (INT) und ein Interrupt Enable Bit (ENA) haben, die von der Software gelesen und geschrieben werden können.

Jeder Adapter muß mit einer belegten IRQ-Leitung eine eigene Anforderung verriegeln bis eine Aufhebung der Verriegelung über die I/O-Adressen 2F2-2F7 oder 6F2-6F7 erfolgt.

Wenn die interrupt sharing logic entriegelt wird und das Status Bit auf einem high-level ist, so muß der Adapter die Interrupt-Anforderung weiter geben. Dadurch geht ein zweiter, während der Abarbeitung des ersten, ankommender Interrupt nicht verloren.

7.8.2 Programm Unterstützung

In einer multitasking Umgebung werden die unterschiedlichen Aufgaben nicht in einer bestimmten Reihenfolge aktiviert und deaktiviert. Im Folgenden werden Möglichkeiten der Benutzung aufgezeigt für:

- Link a task's interrupt handler to a chain of interrupt handlers.
- Share the interrupt level while the task is active.
- Unlink the interrupt handler from the chain when the task is deactivated.

Linking to a chain

Jede neu aktivierte Task ersetzt den Interrupt Vector mit einem Pointer zu sich selbst. Der alte Interrupt Vector wird als Forward Pointer (FPTR) benutzt und wird an einer festgelegten Stelle des neuen Interrupt Handlers abgelegt.

Sharing the Interrupt level

Wenn der neue Handler durch einen Interrupt die Kontrolle erhält, liest er das Interrupt Status Register seines Boards um festzustellen, ob der Interrupt hier erzeugt wurde. Wenn dem so ist, wird er bedient, der Interrupt disabled (CLI) und ein non-specific End of Interrupt (EOI) Kommando zu den Interrupt Kontrollern geschickt. Danach wird eine der Adressen 02F2-02F7 für IRQ9, IRQ3-IRQ7 oder 06F2-06F7 für IRQ10-IRQ15 beschrieben, um den Interrupt zu entriegeln. Zur Dekodierung dieses Befehls reichen die unteren 11 Bit aus. Danach benutzt der Handler den Befehl IRET.

Unlinking from the chain

Um sich aus der chain auszutragen, muß zuerst die Position festgestellt werden. Dazu wird der Vector benutzt um den FPTR zu finden. Ist es der eigene, so wird er durch den abgelegten ersetzt, andernfalls wird weiter gesucht.

Error recovery

Wird beim austragen einer Task aus der Liste der Interrupt Handler ein Fehler festgestellt, z.B. ein Handler hat eine ungültige SIGNATURE, muß eine Error Recovery Procedur benutzt werden. Das Austragen eines fehlerhaften Handlers beseitigt gegebenenfalls keine Störung, sondern blockiert nur den ganzen Interrupt Level. æ

Kapitel 8

Devices

Hier werden die Bemerkungen zu den Devices gesammelt.

Devices sind nach meiner Definition alle über spezielle Codes ansprechbaren Teile eines Rechners, in diesem Fall also alle, die mit den Befehlen "IN" oder "OUT" zu bearbeiten sind.

Devices belegen mindestens eine Adresse im I/O-Bereich, also zwischen 0 und 64k. Meistens werden 8 oder 16 Adressen reserviert. Die Register, die sich hinter den Adressen verbergen, können 8-Bit oder 16-Bit sein. Es sind Kommando-, Status- und Daten-Register gebräuchlich. Manchmal sind über eine Adresse mehrere Register zu erreichen, auf die meistens nicht wahlfrei sondern nur sequential zugegriffen werden kann.

Devices können auch einen IRQ-Interrupt belegen. Meistens ist er mit einer Brücke auf eine IRQ-Leitung zu legen, so daß eine gewisse Auswahl möglich ist. Oft sind nicht alle Möglichkeiten ausgeschöpft worden. Deshalb drängeln sich alle um die wenigen des XT-Steckplatzes. Auch die Software müsste die Änderung mitmachen können.

Manche Devices können auch einen DMA-Interrupt belegen. Auch diese sind meistens mit einer Brücke wählbar.

Kein Interrupt, und auch keine Adresse, darf in einem System mehrfach vergeben werden. Manchmal sind die Adressen an die Vergabe des Interrupts gekoppelt!

8.1 Massenspeicher

Als Massenspeicher sind anzusehen:

- Flexible Platten (Floppy)
- Feste Platten (Winchester)
- Magnetbänder (Kassetten)
- Optische Platten (CD)
- Bubble's

Massenspeicher unterscheiden sich nicht nur in dem Medium, Band oder Platte, sondern auch in der Übertragungsrate (Byte/sec.) beim schreiben und lesen, in der Geschwindigkeit des Mediums z. B. 75 ips (inch pro sec.) oder 300 RPM (Umdrehungen pro Minute), Aufzeichnungsdichte z. B. 6250 bpi (Bit pro inch) und Abstand z. B. 135 tpi (traks pro inch).

8.1.1 Aufzeichnungsverfahren

Um Daten auf magnetisierbaren Medien zu speichern wird ein Takt benötigt zu dessen Änderung der Zustand des Datums betrachtet wird. Es werden also zwei Signale verglichen. Der Takt kann von einer separaten Spur genommen werden oder stammt aus einem Oszillator, dessen Präzision der Aufgabe angepasst sein muß. Ein vollständiger Takt wird oft auch Bit-Zelle oder Fenster genannt. Der Flußwechsel sollte in der Mitte des Fensters erfolgen, dort wo auch der Takt bei dem zu verwendenden 1:1 Tastverhältniss wechselt. Die Abweichung wird als "read window margin" bezeichent. Je geringer die Zahl ist, um so besser ist die Qualität des Laufwerks.

8.1.2 NRZ Non Return to Zero

Die NRZ-Daten ändern sich an der Taktgrenze nur wenn sich das Datum ändert. Mehrere gleiche Daten erzeugen also nur einen Flußwechsel. Während gesamtem "mark"-Intervall HIGH, während gesamten "space"-Intervall LOW.

NRZI (NRZ invertable (eigengetacktet)). Flanke für space, keine Flanke für mark.

RZ (Return to Zero). Während gesamtem space-Intervall LOW, während mark kurzzeitig HIGH.

Manchester (FM)(Eigengetacktet). Positive Flanke für space, negative Flanke für mark.

Biphase mark (Manchester II (eigengetacktet)). Flanke zu jeder Bitperiode. Bei space **keine** Flanke während der Bitperiode, bei mark eine Flanke.

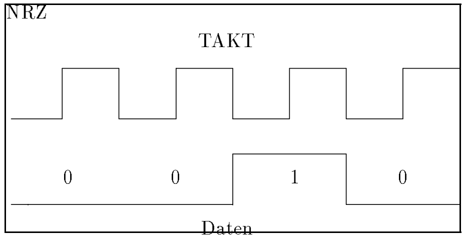


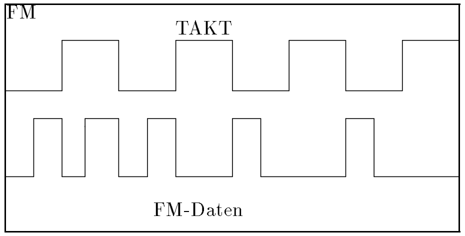
Abbildung 8.1: NRZ-Datenformat

FM Frequenz Modulation

Bei dem FM-Aufzeichnungsverfahren wird in der ersten Takthälfte immer ein Impuls aufgezeichnet, in der zweiten nur dann, wenn das betreffende Bit ein 1-Signal sein soll.

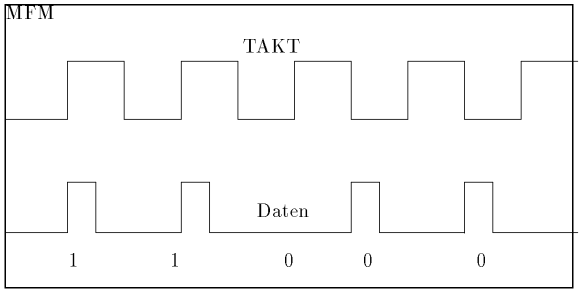
MFM Modifizierte Frequenz Modulation

Bei dem modifizierten FM-Verfahren wird der erste Impuls weggelassen und nur für ein 1-Signal ein Impuls in der Mitte des Takts gegeben. Ein 0-Signal wird am Anfang des Takts nur dann ein Signal erzeugen, wenn das vorhergehende Bit ein 0-Signal war. Jede Bit-Zelle enthält also höchstens einen Flußwechsel und damit kann auf dem gleichen Medium die doppelte Dichte als mit FM aufgezeichnet werden.



æ

Abbildung 8.2: MF-Datenformat



æ

Abbildung 8.3: MFM-Datenformat

RLL Run-length-limited

MFM wird auch als RLL 1.3 bezeichnet. Mit RLL 2.7, wie sie die RLL-Kontroller benutzen, können ca. 50% mehr Daten aufgezeichnet werden. 2,7 heißt, daß das Bitmuster des binären Datenstromes mindestens 2 und höchstens 7 Nullen zwischen eingrenzenden Einsen hat, das heißt in einer vorgegebenen minimalen Zeit (oder minimalen Länge auf der Spur der Disk) können jetzt 3 Bit aufgezeichnet werden, gegenüber 1 Bit bei MFM. Da die Codierung aber die Anzahl der Bits verdoppelt, wird nur der Faktor 1,5 erreicht.

ZBR Zone Bit Recording

VTCR (Variable Track Capacity Recording) wird heute in der Form des ZBR verstärkt zur Kapazitätssteigerung eingesetzt. Da die Verlängerung einer Spur zwischen einer inneren und der nächsten äußeren nur wenige zusätzliche Byte erlauben würde, wird die Anzahl der Sektoren über einen Bereich konstant gehalten, bis die Verlängerung einen zusätzlichen Sektor erlaubt. Weil die Drehzahl der Platte aber gleich bleibt, ergibt sich daraus eine sich verändernde Datenübertragungsrate. Dieses wird von dem Platteninterface mit Zwischenspeichern abgefangen und auch die Sektoren und Köpfe verwaltet, so daß die Platte vom Betriebssystem "normal" angewendet werden kann. Läßt der Kontroller variable Transferraten zu, können oft benötigte Daten in den äußeren Bereich, mit der höchsten Übertragungsrate, gelegt werden.

8.1.3 Fehlerbehandlung

Fehler müssen erkannt werden und nach Möglichkeit behoben werden. Stellvertretend für viele Möglichkeiten werden im Folgenden die verbreitetsten vorgestellt.

CRC Cyclic Redundancy Check

Über einen Block von Daten wird mit einem Polynom ein Prüfzeichen ermittelt. Oft wird das Polynom $X_{16}+X_{12}+X_5+1$ verwendet. Die Binärzeichen eines Datenblocks werden als Koeffizienten des Polynoms verwendet.

Nimmt man bei einer Codewortlänge von **n** an, daß **i** Informationsstellen bekannt sind, dann kann man die redundanten Binärsymbolstellen im Codewort dadurch berechnen, daß man zunächst alle n-i=k redundanten Binärstellen auf 0 setzt und auf das entstandene Codewort:

$$C^* = c_0 x^{n-1} + c_1 x^{n-2} + \ldots + C_{i-1} x^{n-1} + 0 + 0 + \ldots + 0$$

folgende Gleichung anwendet:

$$\frac{C^*(x)}{G(x) = Q(X) + \frac{R(x)}{G(x)}}$$

Addiert man zu $C^*(x)$ das Restpolynom R(x) und dividiert das Ergebnis durch das Generatorpolynom G(x), ergibt sich:

$$\frac{C^*(x) + R(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)} + \frac{R(x)}{G(x)} = \frac{C(x)}{G(x)}$$

Das bedeutet, daß das zulässige Codewort über die Prozedur $C(x) = C^*(x) + R(x)$ gebildet wird.

ECC Error Checking and Correction

Wenn das Polynom auf die gelesenen Daten angewendet wird, so darf kein Rest mit dem Prüfzeichen auftreten. Wird ein Rest festgestellt, so kann ein einfacher Fehler mit einer simplen Korrektur beseitigt werden. Bei mehrfachen Fehlern kann nur eine aufwendige Korrektur erfolgreich sein. Diese ist nur bei sehr teuren Geräten eingebaut.

8.1.4 Organisation

Alle Massenspeicher benötigen eine Organisationsstruktur, um den Inhalt verwalten zu können (Filesystem).

Ein langes Stück Band kann in kürzere Teile aufgeteilt und dann mit einer Tabelle, die am Anfang steht, verwaltet werden.

Die Oberfläche von rotierenden Speichern dagegen muß anders organisiert werden. Ein Punkt ist vorgegeben, nämlich der Weg den der Kopf zwischen zwei Schritten des Kopfantriebs zurücklegt. Damit ist die Entfernung zwischen zwei Kreisen, die Tracks, oder über mehrere Köpfe gesehen Zylinder, genannt werden, gegeben. Sie können aber mehr Daten aufnehmen, als für die Organisation sinnvoll ist. Deshalb werden die Tracks in Sektoren unterteilt. Die Unterteilung kann durch die Hardware z. B. mit Löchern und Sensoren erfolgen, oder durch in den Plattenkontroller eingebaute Software. Diese Software ist oft steuerbar, so daß die Anzahl der Sektoren von dem Programm bestimmt werden kann.

Um die einzelnen Sektoren identifiezieren zu können, wird an den Anfang jedes Sektors nicht nur ein bestimmtes Bitmuster, das den Anfang eines Sektors kennzeichet, geschrieben, sondern auch eine Kennung für den Sektor selbst. Diese Kennung besteht z. B. aus der Zylindernummer, der Oberflächennummer und der Sektornummer.

8.1.5 Rotierende Speicher

In einem Gehäuse oder einer Hülle befindet sich eine runde Scheibe aus Trägermaterial auf die eine dünne magnetisierbare Schicht aufgetragen wurde. Darüber befindet sich ein Schreib-/Lese-Kopf der radial oder tangential mit einem Schrittmotor oder einer "voice-coil" bewegt werden kann. Die Scheibe wird von einem Motor angetrieben, der von einer Steuerung und einem Sensor auf einer konstanten Geschwindigkeit gehalten wird [1]. Dadurch und bei gleichbleibender Aufzeichnungsfrequenz ergibt sich eine unterschiedlich grosse Menge auftragbarer Daten auf den einzelnen Spuren, da die Spurlänge nach innen immer kürzer wird. Auch dadurch ergibt sich eine maximale Länge der Bewegung der Köpfe bei entsprechend geringem Aufwand.

Es sind auch Laufwerke auf dem Markt bei denen die Geschwindigkeit der Platte unter den Köpfen konstant gehalten wird [2]. Bei manchen Laufwerken wird die Schreibdichte beim Überschreiten einer bestimmten Zylindergrenze umgeschaltet.

Floppy

Bei der Floppy besteht die Scheibe aus flexiblem Material und die Köpfe werden auf die Oberfläche angedrückt. Dadurch entsteht Abrieb, der, verstärkt durch Fremdkörper, über kurz oder lang zur Zerstörung des Mediums führen wird.

Floppies werden mit 300 oder 360 RPM (UpM) angetrieben.

Winchester

Winchesterlaufwerke sind in sich geschlossene Systeme, die nur einen Luftdruckausgleich zulassen. Damit ist eine Verschmutzung minimiert. Sie lassen aber einen Mediumtausch wie bei den Front- oder Toploadern nicht mehr zu. Es sind neuerdings auch Laufwerke auf dem Markt, die als Wechselkasseten ausgeführt sind. In den Rechner wird ein Rahmen eingebaut, in den die Winchester eingeführt wird.

[1] constant linear velocity CLV
[2] constant angular velocity CAV

Bernoulli

Bernoullilaufwerke benutzen Platten aus flexiblem Medium, das durch ca. 2000 RPM Stabilität bekommt, genügend Luft für das Polster auf dem die Köpfe schweben und damit keinen Abrieb erzeugt. In Verbindung mit besserem Material werden Kapazitäten von 20MB und 44MB erreicht. Da sie wechselbar sind, könnten sie eine Alternative zu kleinen Winchestern sein. Sie können eingebaut oder extern benutzt werden. Meistens sind sie mit einer SCSI-Schnittstelle ausgerüstet.

8.1.6 Bänder

Bänder sind wie Tonbänder nur in anderen Verpackungen.

Tapes

Tapes werden in Rechenzentren in unterschiedlicher Breite und Länge benutzt.

Kassetten

Kassetten sind in vielfältiger Form vorhanden.

8.1.7 Optische Speicher

Optische Speicher werden als "nur lesbare" und neuerdings auch als beschreibbare angeboten. Zur Zeit sind sie noch recht langsam. Durch die große Speicherdichte könnten sie die Magnetbänder ablösen.

8.2 Bus-Systeme für Platten und Geräte

Ein Bussystem fängt immer bei dem Kontroller an, der die Schnittstelle zu dem Rechnerbus darstellt. Im allgemienen gliedert sich ein Disk-Controller in die folgenden vier Funktionsabschnitte (Bild 8.4):

1. den Seriell-/Parallel-Umsetzer,
2. den Daten-Formatierer,
3. die Fehlererkennungs- und -Korrekturlogik (EDAC) und
4. die Laufwerkssteuerlogik.

Der Seriell-Paralell-Umsetzer hat die Aufgabe, die von der Host-CPU kommenden parallelen Daten in ein serielles Format umzuwandeln, das vom Laufwerk verarbeitet werden kann. Darüber hinaus ist in ST506- und Floppy-Controllern eine Umwandlung zwischen NRZ- und MFM-Codierung erforderlich, die in ESDI- und SMD-Laufwerken direkt im Laufwerk vorgenommen wird.

Der Datenformatierer gliedert die Daten in Sektoren und fügt jedem Datenblock die erforderlichen Steuerinformationen hinzu, damit die einzelnen Sektoren später wieder gefunden und gelesen werden können.

Der Fehlererkennungs- und -korrekturteil erzeugt bestimmte Redundanz-Bytes, die mit auf die Disk geschrieben werden und beim Lesen Aufschluß über die Korrektheit der Informationen geben.

Als vierte Funktion eines Disklaufwerks kommt die Selektion und Positionierung der Köpfe hinzu.

Probleme bei der Implementierung dieser einzelnen Funktionsabschnitte hängen in erster Linie mit deren Komplexität und den bestehenden erheblichen Geschwindigkeitsunterschieden zusammen. Während etwa die Positionierung der Schreib-/Leseköpfe eher langsam abläuft, kann der Durchsatz der Datenschnittstelle eines Laufwerks 20 MBit/s oder gar mehr betragen.

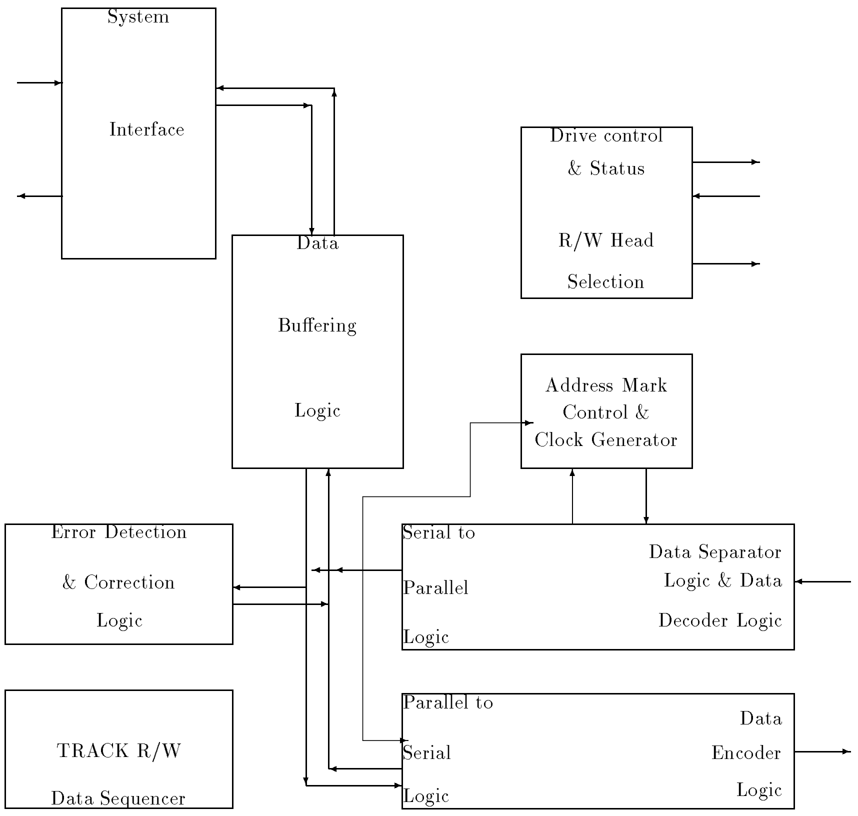


Abbildung 8.4: Disk-Kontroller

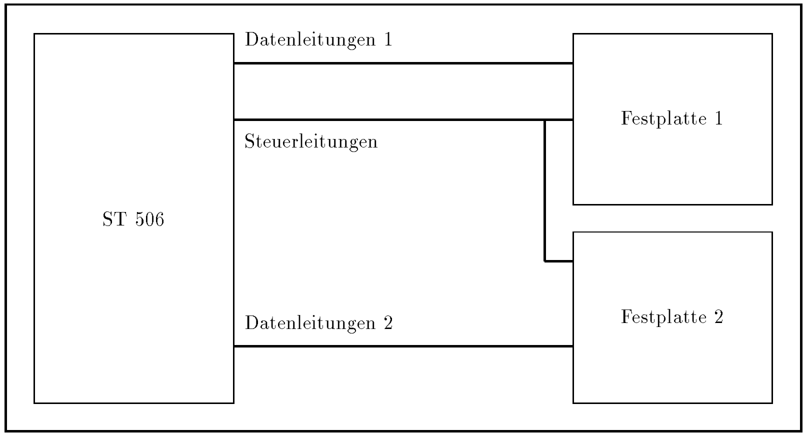


Abbildung 8.5: ST506 Verkabelung

8.2.1 ST506/412

In Bild 8.5 auf Seite 180 werden die Verbindungen zwischen dem ST506-Kontroller und den Laufwerken gezeigt. Kontroller haben bei MFM-Aufzeichnung eine Übertragunsfrequenz von 5 MBit pro Sekunde. Bei RLL-Aufzeichnung sind es 7,5 MBits pro Sekunde. Bei Advanced RLL knapp 10 MBit pro Sekunde. Daraus ergeben sich Datentransferraten von ca. 650 KBytes/s bei MFM und 960 KBytes/s bei RLL.

æ

Bei manchen Disketten-Laufwerken ist der PIN 34 mit dem Signal "READY" belegt.

8.2.2 AT-BUS (IDE)

IBM Task File Interface wird auch AT-File-Interface oder kurz AT-Interface genannt. Wird auch mit IDE, für integrierte Device Elektronik, bezeichnet. Western Digital und Compaq entwickelten es für Laptops und Conner entwickelte Festplatten dafür.

Die Controller sind sehr klein und meistens für 16 Bit Datentransfer und 16MHz Bus-Takt ausgelegt. Die erreichbaren Übertragungsraten hängen vom Laufwerk ab und können zwischen 1,25 MByte für Kato CP2024 (Conner) und 12,8 MByte für MK1034FC (Toshiba) liegen.

Ein/Ausgaberegister des AT-Interface

1F0 Ein- und Ausgabe der Schreib-/Lesedaten. Alle Daten außer den ECC-Bytes bei den Befehlen 22H, 23H, 32H und 33H werden mit 16 Bit übertragen. Die Daten sind gültig, wenn im Statusregister das DRQ-Bit gesetzt ist.

1F1 Hier sollten die error-bits sein

Pin	Signal	Pin	Signal
1	GND	2	Speed Select
3	Res	4	not used
5	GND	6	not used
7	GND	8	Index
9	GND	10	Drive Motor 0 oder 1
11	GND	12	Drive Select 1 oder 0
13	GND	14	Drive Select 0oder 1
15	GND	16	Drive Motor 1 oder 0
17	GND	18	Direction Select
19	GND	20	Step
21	GND	22	Write Data
23	GND	24	Write Gate
25	GND	26	Track 00
27	GND	28	Write Protect
29	GND	30	Read Data
31	GND	32	Side 1 Select
33	GND	34	Diskette Change

Tabelle 8.1: Disketten Steckerbelegung

1F1 Die Schreibvorkompensation wird intern vorgegeben. In dem Register Parameter übergeben, z. B. Cache on/off.

1F2 Enthält die Anzahl der mit dem nächsten Zugriff zu bearbeitenden Sektoren. Beim Kommando *Set Drive Parameter* steht in diesem Register die Anzahl der Sektoren pro Spur.

1F3 Startsektor für den nächsten Zugriff

1F5 Die Bits 0 und 1 sind die Bits 8 und 9 für die Zylinderadresse.

1F6
- Bits 0 bis 3 : Binär kodierte Kopfnummer
- Bit 4 : Laufwerksnummer 0 = Master, 1 = Slave
- Bits 5 bis 7 : liegt fest mit 101 = 512 Bytes pro Sektor

1F7 Statusregister

Bit	Name	Beschreibung
0	ERR	Fehler aufgetreten.
1	IDX	Indeximpuls
2	CORR	mit ECC korrigierte Daten übertragen
3	DRQ	fertig für Datentransfer
4	SKC	Suchvorgang ausgeführt
5	WFT	Schreibfehler festgestellt
6	RDY	Laufwerk ist fertig
7	BSY	Laufwerk führt Befehl aus

Lesen des Statusregisters löscht den IRQ14.

1F7 nach dem schreiben des Registers wird der Befehl ausgeführt

3F6 Lesen des Statusregisters 1F7 ohne IRQ14 zu löschen.

3F6 Reset, Interruptmaske

Pin	Signal	Pin	Signal
1	GND	2	reserved
3	GND	4	head select 2^2
5	GND	6	wire gate
7	GND	8	seek complete
9	GND	10	track 0
11	GND	12	write fault
13	GND	14	head select 2^0
15	GND	16	reserved (to pin7 j2)
17	GND	18	head select 2^1
19	GND	20	index
21	GND	22	ready
23	GND	24	step
25	GND	26	drive select 1
27	GND	28	drive select 2
29	GND	30	drive select 3
31	GND	32	drive select 4
33	GND	34	direction in

Tabelle 8.2: Disk Steckerbelegung A

Pin	Signal	Pin	Signal
1	drive selected	2	GND
3	reserved	4	GND
5	reserved	6	GND
7	reserved (to j1 pin 16)	8	GND
9	reserved	10	reserved
11	GND	12	GND
13	+MFM data	14	-MFM data
15	GDN	16	GND
17	+MFM read data	18	-MFM read data
19	GND	20	GND

Tabelle 8.3: Disk Steckerbelegung B

Pin	Signal	E/A	Beschreibung
2,19, 22,24, 26,30,40	GND		Masse
1	RES	E	Reset-Signal vom Rechner
3,4	D7,D8	E/A	Bidirektionaler Datenbus (8/16 Bit)
5,6	D6,D9	E/A	zwischen Rechner und Festplatte
7,8	D5,D10	E/A	zur Übertragung von Daten, Status-
9,10	D4,D11	E/A	und Steuerinformayionen. Die
11,12	D3,D12	E/A	Leitungen werden hochomig
13,14	D2,D13	E/A	geschaltet, wenn das Laufwerk
15,16	D1,D14	E/A	nicht selektiert ist.
17,18	D0,D15	E/A	"
20			abgekniffener Pin
21	/IOCHRDY (reserved!!)	A	Rechner muß bei Laufwerkszugriffen warten, bis dieses Signal inaktiv wird. Bei manchen wird auch GND angegeben!
23	/IOWR	E	Schreibsignal für E/A-Portadressen
25	/IORD	E	Lesesignal für E/A-Portadressen
27	/IOCHRDY		wie Pin 21 ??
28	ALE	E	Address Latch Enable
29	frei		reserviert
31	IRQ14	A	Interrupt request
32	IO16	A	Anzeige für 16-Bit Datentransfer
34	/PDIAG	E/A	Passed Diagnostics Bei manchen nicht benutzt!
33	A1	E	Adreßsignale vom Rechner zur
35	A0	E	Auswahl von Registeradressen
36	A2	E	"
37	/CS0	E	Card select
38	/CS1	E	
39	/ACT	E/A	Kann LED treiben, um Aktivität anzuzeigen.

Tabelle 8.4: Pinbelegung AT-Interface

Address	Lesen	Schreiben
1F0	Daten 8/16	Daten 8/16
1F1	Fehlerbits	Kompensation
1F2	Sektorzähler	Sektorzähler
1F3	Startsektor	Startsektor
1F4	Zylinder LSB	Zylinder LSB
1F5	Zylinder MSB	Zylinder MSB
1F6	Laufwerks- , Kopfnummer	Laufwerks- , Kopfnummer
1F7	Status	Kommando
3F6	2. Status	Reset, IRQ-Maske
3F7	aktuelle Adresse	nicht benutzt

Tabelle 8.5: Register des AT-Interface

Bit	Name	Beschreibung
0		nicht benutzt
1	/IRQEN	0 = IRQ14 wird nach Ende aktiv
2	Reset	1 = Software reset
3-7		nicht benutzt

3F7 Aktive Adresse

Bit 0	0 = Master aktiv
Bit 1	1 = Slave aktiv
Bits 2 - 5	Einerkomlement des eingeschalteten Kopfes
Bit 6	0 = Laufwerk schreibt gerade
Bit 7	1 = nur für Floppy: Diskette gewechselt

AT-Festplattenkommandos

Einige Befehle sind verbindlich. Werden herstellerabhängige Befehle verwendet, muß dafür ein Driver vorhanden sein oder die Anpassung ist im BIOS oder in einem ROM.

Code	Kommando	Beschreibung
1X	Recalibrate	Kopf auf Spur 0 fahren
2X	Read Sector	1 - 256 Sektoren ab angegebenen Sektor lesen
20	" with retry	" mit Lesewiederholungen bei Fehlern
21	" no retry	" ohne Lesewiederholung mit Fehlermeldung
22	" long with retry	nach den Date werden auch die Fehlerkeorrekturbytes gelesen
23	" long no retry	mit Korrekturbytes ohne Wiederholung
3X	Write Sector	1 - 256 Sektoren ab angegebenen schreiben
4X	Read Verify Sectors	Lesbarkeit der angegebenen Sektoren prüfen
50	Format Track	angegebene Spur formatieren
7X	Seek	Kopf zur angegebenen Spur fahren
90	Exec Drive Diag.	Laufwerksselbsttest Starten
91	Set Drive Para.	Laufwerksparameter an eingebauten Controller übergeben

Für Laufwerke von Quantum sind herstellerabhängige Spezialbefehle bekannt. Die Auswahl geht durch die Register 1F2 bis 1F6.

Code	Kommando	Beschreibung
C4	Read multiple	mehrere Sektoren werden als Block übertragen
C5	Write multiple	mehrere Sektoren werden als Block übertragen
C6	set multiple	Anzahl der Sektoren pro Block in 1F2
E4	Read sect buff	512 Bytes aus dem Sektorbuffer lesen
E8	Write sect buff	512 Bytes in den Sektorbuffer schreiben
EC	identify drive	Laufwerksdaten lesen
EF	Cache	controllercache mit Daten in 1F1 schalten
		Datum 55 = cache aus
		Datum AA = cache ein
EX	Power commands	Kommandos zur Leistungsreduzierung
E0	standby mode	Spindelmotor aus, schaltet bei Zugriff wieder ein
E1	idle mode	Spindelmotor ein
E2	auto power down	Spindelmotor aus, nach Zugriff wieder aus
		Wartezeit in 5-Sekunden-Schritten über 1F2
E3	"	Spindelmotor ein, sonst wie E2
E5	read power mode	Zustand des Spindelmotors in 1F2 lesen
E6	sleep mode	Laufwerk aus, nur zurück über reset
FX	power save	F8, F9, FA, FB und FD wie E0 - E5
		Zeit in 0,1-Sekunden-Schritten
F0	set config	Cache-Strategie und Fehlerbehandlung
	read config	aktuelle Konfiguration lesen
	defect list	Defektenliste lesen

æ

8.2.3 ESDI-BUS

Bei dem **E**nhanced **S**mall **D**evice **I**nterface befindet sich der Datenseparator zur Trennung von Daten und Synchronisationssignalen in der Festplattenelektronik. Transfer-Raten von 10 bis 15 MBits/s, Daten im NRZ-Format (16-Bit Datenstrom) oder im RRL-Code, paralleler Transfer für Befehle und Status. Keine Steppingpulse vom Kontroller sondern Befehl. Der μpd7262 kann z. B.:

- Kontrolle der seriellen ESDI-Kommando-Schnitstelle;
- Kontrolle von bis zu sieben Laufwerken mit
  - bis zu 80 Köpfen/Laufwerk,
  - bis zu 4096 Zylindern/Laufwerk oder
  - bis zu 256 Sektoren/Spur;
- Bearbeitung von bis zu 65536 Sektoren mit einem Kommando,
- maximale Datenübertragungsrate 18 MBit/s
- sowohl soft- als auch hardsektorierte Laufwerke anschließbar,
- programmierbare Sektorlänge von 128 bis 65536 Byte/Sektor,
- Multi-Sektor-, Multi-Track- und Multi-Zylinder-Verarbeitung ist möglich;
- Implied-Seek und Overlapped-Seek (Parallel-Seek) programmierbar;
- Möglichkeit des Lesens eines Sektors trotz fehlerhaften ID-Feldes;
- Fehlererkennung und Korrektur wahlweise CRC; 32-Bit-ECC oder 56-Bit-ECC;
- System-Clock maximal 18 MHz.

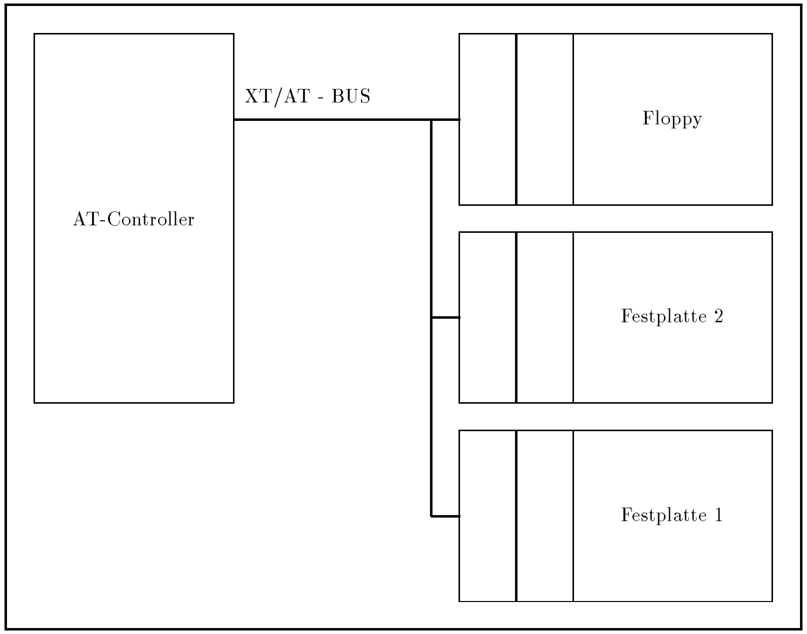


Abbildung 8.6: XTAT Verkabelung

Pin	Signal	Pin	Signal
1	GND	2	Head select 2^3
3	GND	4	Head select 2^2
5	GND	6	write gate
7	GND	8	config/status data
9	GND	10	transfer ack
11	GND	12	attention
13	GND	14	head select 2^0
15	GND	16	sector/byte ck/fnd
17	GND	18	head select 2^1
19	GND	20	index
21	GND	22	ready
23	GND	24	transfer req.
25	GND	26	drive select 1
27	GND	28	drive select 2
29	GND	30	drive select 3
31	GND	32	read gate
33	GND	34	command data

Tabelle 8.6: ESDI-Pinbelegung A

Pin	Signal	Pin	Signal
1	drive selected	2	IBG found
3	command complete	4	write ref clock
5	write ref. clock	6	GND
7	write clock	8	write clock
9	write ref. clock	10	read clock
11	read clock	12	GND
13	NRZ write data	14	NRZ write data
15	GND	16	GND
17	NRZ read data	18	NRZ read data
19	GND	20	load pt/file,m early warning

Tabelle 8.7: ESDI-Pinbelegung B

8.2.4 SCSI-BUS

Ein Teil der Aufgaben eines Geräte Controllers ist in die Geräte selbst verlagert worden. Der Controller wurde zum Host-Adapter. Nicht mehr nur Pegel auf Leitungen sind das Protokoll, sondern es werden komplexe Befehle über den 8-Bit breiten Datenpfad an die Geräte geschickt. Dabei kann jedes Gerät Initiator oder Target sein. Bis zu acht Teilnehmer können an einem SCSI-Bus angeschlossen sein. Jeder Teilnehmer kann bis zu acht Geräte verwalten, so daß an einem Host-Adapter maximal 56 Geräte angeschlossen sein können. Diese können auch ohne den Rechner zu belasten untereinander arbeiten, z. B. macht die Festplatte ein Backup auf ein Magnetband.

Die Geräte werden mit einem 50-poligen Kabel, s. Tabelle 8.8 auf Seite 189, angeschlossen. Nur in dem letzten Gerät darf der Bus-Abluss verbleiben.

Es sind Kontroller mit "single ended" Leitungen vorhanden, bei denen die Leitungslänge 6 Meter nicht überschreiten darf. Bei Kontrollern mit differentiellen Leitungen kann die Gesamtlänge 25 Meter erreichen.

Kabellänge und Geschwindigkeit

Aus einem Benchmarktest mit Chips der angeführten Firmen als Target und als Initiator sind bei den Längen die Übertragungsgeschwindigkeiten (MByte/s):

50 cm		
	National	NCR
National	3.5 4.9	2.6 2.7
NCR	1.8 2.3	1.5 2.0
3 m		
National	3.1 3.9	2.3 2.5
NCR	1.6 2.2	1.4 1.9
6 m		
National	2.5 3.0	2.0 2.2
NCR	1.4 2.0	1.3 1.8

Übertragungsgeschwindigkeiten der Platteneinheiten reichen von 1,25 MByte für KATO CP2020 (Conner) bis 12,5 MByte für SD340-H-04 (TEAC) falls der Druckfehlerteufel nicht dabei mitgemischt hat. Die Leistungsfähigkeit der Geräte, des Kontrollers und des Bus-Systems sollten aufeinander abgestimmt sein. Die konventionelle DMA-IS kann z. B. 3,7 MB/s im normalen Modus. Dazu Bemerkungen auf Seite 159.

Die SCSI-Host-Adapter können in drei Kategorien eingeteilt werden:

1. CPU unterstützte
2. DMA slave
3. Bus master

Jede Kategorie zeigt die Entwicklung bei der Fertigung von integrierten Schaltungen auf. Die billigen, langsamen CPU-unterstützten (100 DM) sollten wirklich nur dort benutzt werden, wo es auf Perfomance nicht ankommt sondern nur der Preis von Interesse ist. DMA-slaves werden zur Zeit mit ca. 500 DM und DMA-master mit ca. 800 DM gehandelt. Von dem Gesamtpreis also nur ca. 10%.

Signal	Pin	Pin	Signal
Masse	1	2	DB0(Datenbit 0)
Masse	3	4	DB1(Datenbit 1)
Masse	5	6	DB2(Datenbit 2)
Masse	7	8	DB3(Datenbit 3)
Masse	9	10	DB4(Datenbit 4)
Masse	11	12	DB5(Datenbit 5)
Masse	13	14	DB6(Datenbit 6)
Masse	15	16	DB7(Datenbit 7)
Masse	17	18	DBP(Parity-Bit)
Masse	19	20	Masse
Masse	21	22	Masse
Masse	23	24	Masse
frei	25	26	Terminator Power
Masse	27	28	Masse
Masse	29	30	Masse
Masse	31	32	ATN (Attention)
Masse	33	34	Masse
Masse	35	36	BSY (Busy)
Masse	37	38	ACK (Acknowledge)
Masse	39	40	RST (Reset)
Masse	41	42	MSG (Message)
Masse	43	44	SEL (Select)
Masse	45	46	C/D (Control/Data)
Masse	47	48	REQ (Request)
Masse	49	50	I/O (Input/Output)

Tabelle 8.8: SCSI-Busbelegung (intern)

Pin	Signal	Pin	Signal
1	GND	14	SC0
2	SC1	15	SC2
3	SC3	16	SC4
4	SC5	17	SC6
5	SC7	18	SPAR
6	GND	19	GDN
7	SSEL	20	SATN
8	GDN	21	SMSG
9	spare	22	SACK
10	SRST	23	SBSY
11	SCD	24	SREQ
12	SIO	25	GDN
13	GDN		

Tabelle 8.9: SCSI-Busbelegung (extern)

CPU-unterstützte Host-Adapter können mit maximal 2 Buszyklen pro Byte rechnen wenn mit REP MOVS im Gerätetreiber gearbeitet wird. Der Adress- und/oder Bufferbereich ist oft auf 128 beschränkt. Mit diesen Bursts kann die maximale Burstrate des ISA-Bus von ca. 1,5 MB/s erreicht werden. Sie wird aber gegebenenfalls von dem Overhead im ROM-BIOS gedrückt, oft auf 50% was immer noch 50% schneller ist als die meisten MFM-Kontroller. CPU-unterstützte Kontroller eignen sich nur für singletasking Anwendungen, da die CPU während eines Transfers nur mit dem Kontroller beschäftigt ist.

DMA-slave Kontroller transferieren nur ein Byte pro DREQ und währen damit langsamer als CPU-unterstützte Kontroller. Deshalb werden die Daten in Bursts zu 8 bis 128 und mehr Byte übertragen. Da bei DMA-Transfer nur ein Buszyklus benötigt wird, ist solch ein Kontroller fast doppelt so schnell wie ein CPU-unterstützter. Die 3MHz Standardversion des 8237A DMA Interruptkontrollers erlaubt ca. 1,2 MB/s Datenrate, aber der Overhead, der mit jedem 4 Wort Burst verbunden ist, drück die Datenrate auf ca. 0,87 MB/s. Die meisten AT-Clones benutzen aber den schnelleren 5MHz 8237A, mit dem dann Datenraten von 1,7 MB/s erreicht werden können.

Ein DMA-Bus-master "besitzt" für eine gewisse Zeit den Bus und generiert die Adressen selber. Deshalb können die Wait-states entfallen, die normalerweise bei einem DMA-Transfer für unbekannte Karten eingebaut sind. Der Transfer wird in mehreren Schritten ausgeführt. Zuerst wird ein DMA-Kanal in den Kaskaden-Mode gesetzt. Dann wird mit DREQ ein DMA-Zyklus angefordert. Wird durch DACK das erlaubt, wird mit MASTER der Bus in "Besitz" genommen. Er darf aber nur eine gewisse Zeit in diesem Zustand bleiben (ca. 15μs), da sonst das Memory nicht mehr erneuert (refreshed) wurde. Danach wird DREQ zurückgenommen. Mit der Rücknahme von DACK verschwindet auch "MASTER". Die Datenrate der SCSI-chips reicht z. Z. bis 50 MB/s bei 32Bit Breite. Mit 80 ns DRAM kann eine Datenrate von 10 Mbytes/s erreicht werden. Die Devices sollten dann aber auch in diese Größenordnung vordringen.

Die Signale sind nicht gerichtet sondern bidirektional. Zu den acht Datenleitungen kommt noch eine mit dem Parity-Bit.

RST Hard-Reset ist ein Puls von mindestens 25 Millisekunden.

SEL Der Initiator erzeugt das Select-Signal und legt gleichzeitig die ID-Nummer des Target auf den Bus. Ist das Target bereit, antwortet es mit einem Puls auf der Busy-Leitung.

BSY Puls vom Target zum Initiator wenn bereit.

C/D Control/Data. Der Pegel ist high für Daten, Status oder Meldungen. Bei Datenübertragungen ist der Pegel low.

I/O Input/Output. Ist der Pegel low, erfolgt der Datentransfer zum Target. Ist der Pegel high, erfolgt der Datentransfer zum Initiator.

REQ Request. Wird das Signal erzeugt, ist ein Byte auf dem Bus bereit. Das Signal muß zurückgenommen werden wenn das ACK-Signal kommt.

ACK Acknowledge.

ATN Attention.

MSG Message.

Das Protokoll läßt sich am besten mit Phasen beschreiben.

Frei Bus free phase. Targets leiten die Frei-Phase durch Aktivieren des Busy-Signals gefolgt von einem Reset ein.

Belegen Arbitration phase.

Auswahl Selection/Reselection phase.

Übertragung Information transfer phase.

- Bus free phase.
- Arbitation phase.
- Selection phase.
- Reselection phase.
- Command phase.
- Data phase.
- Status phase.
- Message phase.

Bus free

Die *Bus free* Phase zeigt an, das der Bus nicht von einem Gerät benutzt wird, aber vorhanden ist. Dieser Zustand tritt ein, wenn an **SEL** und **BSY** mindestens 400 ns lang ein negatives Signal anliegt. Die SCSI-Geräte müssen dann alle Signale innerhalb von 800 ns freigeben.

Arbitration phase

Hier ist es einem Gerät möglich, die Kontrolle über den Bus entweder als Initiator oder als Target zu erlangen. In Systemen, die die Reselection Phase unterstützen, ist die Arbitration zwingend vorgeschrieben. Dies gilt auch für SCSI-2.

Selection phase

In dieser Phase wird es einem Initiator freigestellt, ein Zielgerät auszuwählen, um Schreib- oder Lesebefehle auszuführen.

Reselection phase

Diese Phase ist optional und ermöglicht es einem Target, eine Verbindung mit einem Initiator aufzubauen, um eine Operation weiterzubearbeiten, die bereits vom Initiator gestartet und kurzfristig vom Target unterbrochen wurde.

Information transfer phase

Die Command-, Data-, Status- und Message-Phasen werden als Information-Transfer-Phasen bezeichnet, weil sie weder Daten noch Informationen über den Bus trasnportieren. Die **MSG-**, **C/D-** und **I/O-**Signale steuern die verschiedenen Transferzustände sowie die Richtung des Transfers während der Phase. Das Target sendet diese Signale aus und kontrolliert folglich den Wechsel von einem Zustand in einen anderen. Mit einem **ATN-**Signal kann der Initiator eine Message-Out-Phase erreichen.

Command phase

Diese Phase erlaubt es einem Target, Befehle von einem Initiator zu empfangen.

Data phase

In der Data-In-Phase erwartet das Zielgerät, daß Daten zum Initiator übertragen werden, in der Data-Out-Phase, daß der Host Daten zum Target sendet.

Status phase

In diesem Zustand überträgt das Target Statusinformationen zum Initiator.

Message phase

Auch hier gibt es In- und Out-Phasen, die beide gleichzeitig mehrere Nachrichten während einer Phase übermittel können. Das erste Byte, das übergeben wird, zeigt an, ob es sich um eine Single-Byte- oder eine Multiple-Byte-Message handelt. In der Message-In-Phase sendet das Zielgerät Nachrichten zum Initiator, in der Message-Out-Phase werden Informationen vom Host ans Target übergeben.

æ

Code	Beschreibung	Richtung	
00H	Command complet	In	
01H	Extended Message	In	Out
02H	Save Data Pointer	In	
03H	Restore Pointers	In	
04H	Disconnect	In	
05H	Initiator detecded Error		Out
06H	Abort		Out
07H	Message reject	In	Out
08H	No Operation		Out
09H	Message parity error		Out
0AH	Linked command complet	In	
0BH	wie 0BH mit flag	In	
0CH	Bus Device Reset		Out
80H-FFH	Identify	In	Out
In	Target to Initiator		
Out	Initiator to Target		

Tabelle 8.10: SCSI-Message Codes

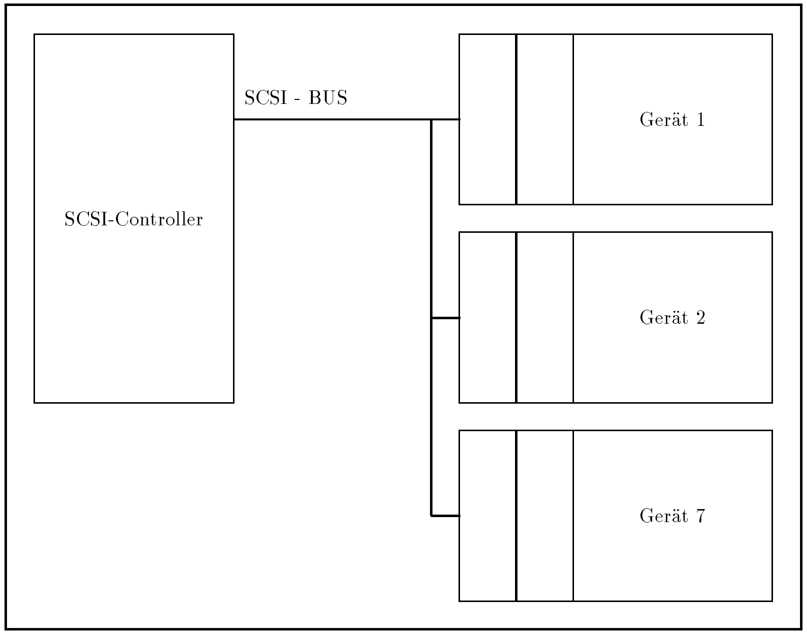


Abbildung 8.7: SCSI Verkabelung

SCSI-2

Die Spezifikation des weiterentwickelten SCSI — SCSI-2 genannt — ist kompatibel zur Spezifikation der Grundversion. Sie enthält jedoch einige neue Leistungsmerkmale und Funktionseigenschaften. Durch drei wichtige Zusätze ergibt sich eine wesentliche Verbesserung der Leistungen der SCSI-2-Schnittstelle gegenüber SCSI. Diese Zusätze sind:

1. eine schnelle, synchrone Datenübertragung,

2. eine breite Datenübertragung und

3. "tagged"- (Identifizierbare)Befehle.

Die Erweiterung der SCSI-2-Funktionen beruht hauptsächlich auf der Standardisierung bestehender Befehlssätze (d. h. CCS) und der Hinzufügung von Befehlssätzen für Gerätetypen, die bisher nicht in der SCSI-Spezifikation berücksichtigt waren (z. B. Scanner, Mediumwechsler usw.).

Durch schnelle, synchrone und breite Datenübertragung bietet die SCSI-2-Schnittstelle eine maximale Übertragungsrate von 40 MByte/s. In der SCSI-2-Schnittstelle wurde die maximale Übertragungsbreite von 8 auf 16 oder 32 Bit erhöht. Genau wie bei der synchronen Datenübertragung kann die breite Datenübertragung als Option angewandt werden und benötigt den Initiator und das Ziel zur Einstellung der richtigen Breite. Ein zweites Kabel, das "B"-Kabel, vereinfacht den Betrieb, da dieses 68adrige Kabel bereits die zusätzlichen Leitungen für eine 16- oder 32-Bit-Datenübertragung enthält. Dabei sind jeweils ein Satz B-Kabel-Anforderungssignale und B-Kabel-Quittungssignale definiert. Das in SCSI spezifizierte Kabel "A" hat sich für SCSI-2 nicht geändert. Dieses Kabel hat acht Datenleitungen und die für die SCSI-Schnittstelle benötigten Steuersignale. Das 50adrige A-Kabel wird für die unteren 8 Bit eines jeden Datentransfers verwendet. Darüber hinaus finden alle Übertragungen von Befehls- und Statusmitteilungen ebenfalls auf dem A-Kabel statt.

Eine weitere, neue Eigenschaft des SCSI-2 verbessert ebenfalls die Systemleistung: "tagged"-(Identifizierbare)Befehle. Diese Eigenschaft erlaubt den SCSI-2-Zielsystemen mehrere Befehle, die von den einzelnen Initiatorsystemen auf den SCSI-2-Bus geschickt werden, in Warteschlangen für ein Logiksystem einzureihen. Wie viele der SCSI/SCSI-2-Funktionen, sind auch "tagged"-Befehle eine optionale Leistung. Diese "Tag"-Werte (Zahlen von 0 bis 255) erlauben eine eindeutige Identifizierung der Befehle. Um die Leistungsfähigkeit zu erhöhen, kann die Reihenfolge der in der Warteschlange stehenden Befehle durch die Zielsysteme manipuliert werden. Eine Optimierung erfolgt durch allgemein bei Warteschlangen angeandte Methoden wie Befehlssortierung und Befehlskämmung.

Die Befehlssortierung ordnet die Ausführung einer Befehlsschlange neu. Bei absteigender Sortierfolge (nach Plattenzellen) würde zum Beispiel der Schreib-/Lesekopf niemals über einen Zylinder, für den eine Anforderung ansteht, hinweggehen. Auf diese Weise werden die Suchzeiten, die normalerweise die Schnelligkeit erheblich beschränken, minimiert.

Das Befehlskämmen verringert die andere große Beschränkung der Leistung, die Umdrehungswartezeiten. Diese Methode kombiniert eine Reihe von wartenden Befehlen, die auf sequentielle Plattenzellen zugreifen, in eine einzige Mehrfachsektor-Operation. Der komplette Satz der Befehle wird dann in einer einzigen Umdrehung ausgeführt, anstatt wie sonst üblich, in einer Umdrehung pro Befehl.

Hitachi veröffentlicht für die 3.5 Inch drives mit SCSI-2 fast folgenden commans set:

Code	Name
00	test unit ready
01	rezero unit
03	request sense
04	format unit
07	reassign blocks
08	read
0A	write
0B	seek
12	inquiry
15	mode select
16	reserve
17	release
1A	mode sense
1B	start/stop unit
1C	receive diagnostic results
1D	send diagnostic
25	read capacity
28	read(10)
2A	write(10)
2B	seek(10)
2E	write and verify
2F	verify
37	read defect data
3B	write buffer
3C	read buffer
3E	read long
3F	write long
41	write same
4C	log select
4D	log sense
55	mode select(10)
5A	mode sense(10)

8.2.5 IPI

Intelligent Peripheril Interface. Übertragunsraten größer 6 MByte/s z. B. für Disk-Arrays. 5 MHz Takt ergeben mit 16 Bit breitem Bus 10 MByte Transferrate.

8.2.6 SMD/ESMD

Storage Module Device (enhanced) Interface mit 24 MHz Takt ergibt maximal 3 MB/s Übertragungsrate. Daten im NRZ-Format. Keine Eigenintelligenz und keine Diagnosefunktionen per Definition.

8.2.7 QIC

Quarter **I**nch **C**ompatibility Tape drive interfaces.

QIC-02	Intelligent hardware tape interface
QIC-36	Simple jardware tape interface
QIC-104/111	SCSI-1 hardware tape interface
QIC-121	SCSI-2 hardware tape interface

8.2.8 Disk-Array

8.2.9 QIC-Aufzeichnungsformate

Bis zu QIC-1000 wird mit GCR (Group Coded Recording) aufgezeichnet. Bei QIC-1350 wird RLL-1.7 verwendet.

QIC-24 9-track; 60 MByte

QIC-120 15-track; 125 MByte

QIC-150 18-track; 150 MByte

QIC-320

QIC-525 26-track; 525 MByte

QIC-1000 1000 MByte

QIC-1350 1350 MByte

	QIC-1000	QIC-6000	12 GB
Spuren	30	96 Daten 16 Servo	144 Daten 24 Servo
Aufzeichnungsdichte	45 000	50 800	101 600
Datendichte	3600	67 773	135 466
Codierung	GCR	RLL 1.7	RLL 1.7
Fehlerkorrektur	Reed Solomo Level 2	Reed Solomo Level 6	Reed Solomo Level 6
Medium	900 oe	900 oe	1200 oe

8.2.10 DAT

Magnetband-Kassettenlaufwerke in DAT-Technik (**D**igital **A**udio **T**ape) mit Aufzeichnungsverfahren DDS (**D**igital **D**ata **S**torage) sind im Normungsverfahren (August 1991). Wie bei Video-8-Laufwerken wird das Schrägspuraufzeichnungsverfahren (Helical-Scan) benutzt. Auf einem Band von nur 3,8 mm Breite und einer Länge von ca. 63 m lassen sich, bei einer Laufzeit von 2 Stunden, rund 1,3 Gigabyte Daten unterbringen. Die Date-Transferrate von 183 kBit/s liegt etwa um den Faktor zwei höher als bei den meisten 1/4-Zoll-Datenspeichern, befindet sich aber etwas unter den Werten heutiger 1/2-Zoll-Laufwerken. Die Aufzeichnungstrommel (Kopf) rotiert mit 2000 Upm, das Band bewegt sich mit 8,15 mm/s. Damit wird eine Aufzeichnungsdichte von ca. 61 Kbpi erreicht (1/2-Zoll-Bänder 6250 bpi).

8.3 CD-ROM

Die runde Scheibe einer CD-ROM hat einen Durchmesser von 12 cm und ist etwa 1,2 mm dick. Sie hat eine Kapazität von rd. 550 MB, das entspricht dem Inhalt von etwa 270 000 Schreibmaschinenseiten, oder 1 000 Büchern bzw. 15 000 Bildern. Die Daten werden auf der Platte von innen nach außen laufend spiralförmig in Form winziger Vertiefungen (Pits) festgehalten. Eine CD-ROM enthält etwa 5 Milliarden dieser Pits, von denen jeder nur 0,6 μm breit und 0,12 μm tief ist. Die einzelnen Daten-Spuren sind auf der Platte so eng angeordnet, daß auf einen Millimeter 630 dieser Spuren nebeneinander Platz haben. Zum Vergleich: auf einer Diskette liegen "nur" 37 Spuren pro Millimeter nebeneinander.

Um auf der rotierenden Platte von innen nach außen eine gleichbleibende Aufzeichnungsdichte zu erreichen, dreht sich die Platte, wenn der Lesekopf ganz innen ist, mit 500 Umdrehungen pro Minute; ist der Kopf ganz außen, reduziert sich die Drehzahl auf 200 Umdrehungen pro Minute[3].

Das Abtasten geschieht mit Laserstrahlen über hochpräzise Optiken und Steuer-Elektroniken. Das Licht des lesenden Laserstrahls wird von der Platten-Oberfläche (Land) und den Vertiefungen (Pits) unterschiedlich reflektiert. Aus der Interpretation des Laufzeit-Unterschieds erkennt das System, ob eine 0 oder eine 1 als Information an den Computer weiterzugeben ist.

Die Platten selbst sind durch eine Lackschicht vor äußeren Einflüssen geschützt und unterliegen auch im Gebrauch keinem Verschleiß.

8.4 Tastaturen

Das ROM BIOS beinhaltet zwei Services für das Keyboard, eine ISR (interrupt service routine) und eine DSR (device service routine). Die ISR wird mit dem INT 09H von der Hardware benutzt und stellt jeweils zwei Byte in den Buffer, der bei 40:1E beginnt. Die DSR wird als INT 16H von dem Betriebssystem oder dem Benutzerprogramm zum Zugriff auf den Buffer benutzt. Der Vektor des INT 09H wird mit F000:E987 initialisiert und der des INT 16H mit F000:E82EH.

Wird in das Port 64H das Kommando 20H geschrieben, bedeuten die Bits im Port 60H dann:

[3] CAV constant angular velocity

I/O Address	Read/Write	Description
60H	W	keyboard data input buffer
60H	R	keyboard data output buffer
62H	W	8042 keyboard input buffer
61H	R/W	8255 output register (XT-only!!!)
62H	R/W	8255 input register (XT-only!!!)
63H	R/W	8255 command mode register (XT-only!!!)
64H	W	8042 keyboard input buffer
64H	W	8042 commands
64H	R	8042 status

Tabelle 8.11: Keyboard I/O ports

Location	size	Description
17H	1 Byte	Keyboard shift flags
18H	1 Byte	Extended keyboard shift flags
19H	1 Byte	Work area for ALT key and numeric keypad input
1AH	1 Byte	Offset to next character in keyboard buffer
1CH	1 Byte	Offset to first character in keyboard buffer
1EH	16 Words	Keyboard buffer of 16 word entries
71H	1 Byte	Bit 7 = 1 if CTRL ALT DEL is pressed
72H	1 Byte	Set to 1234H if CTRL ALT DEL is pressed
80H	1 Byte	Address to start of keyboard buffer
82H	1 Byte	Address to end of keyboard buffer
96H	1 Byte	Status Byte
97H	1 Byte	Status Byte

Tabelle 8.12: Keyboard System RAM Data

- Bit 0 EKI enable KBD interrupt
- Bit 1 reserved
- Bit 2 SYS System flag (sets status reg(2))
- Bit 3 INH Key lock inhibit override (1=disable)
- Bit 4 DKB disable KBD
- Bit 5 KBD Keyboard type (0=AT, 1=PC)
- Bit 6 KCC Keycode conversion (1=enabled)
- Bit 7 reserved

Wird das Port 64H nur gelesen, so ist es das Status Register.

- Bit 0 OBF output buffer full
- Bit 1 IBF input buffer full
- Bit 2 SYS System flag (0=cold reset, 1=hot reset)
- Bit 3 C/D Command/Data (0= Data oder idle, 1=Command oder busy)
- Bit 4 KBEN Keyboard enable (1=enable)
- Bit 5 TTIM transmit time-out
- Bit 6 RTIM receive time-out
- Bit 7 PERR receive parity error

8.4.1 Befehle für den Tastaturkontroller

- 20 read mode register
- 21-3F read keyboard controller RAM (Byte 1 - 31)
- 60 write mode register
- 61-7F write keyboard controller RAM (Byte 1 - 31)
- AA self test. Wenn kein Fehler gefunden wird steht im output buffer eine 55H, das OBF Bit wird gesetzt und KIRQ gemacht, wenn EKI gesetzt ist.
- AB KBD Interface test. Es werden die Clock- und Datenleitungen getestet. Das Ergebniss wird in den output buffer geschrieben, das OBF Bit gesetzt und KIRQ gemacht, wenn EKI gesetzt ist.
  - 00 no error
  - 01 keyboard clock line stuck low
  - 02 keyboard clock line stuck high
  - 03 keyboard data line stuck low
  - 04 keyboard data line stuck high
- AC diagnostic dump schickt 16 Bytes des Kontroller-RAM, den augenblicklichen Zustand des Input- und Output-Ports, zum System.

- AD disable keyboard setzt Bit 4 im Mode-Register.
- AE enable keyboard cleared das Bit 4 im Mode-Register.
- C0 read input port (P10 - P17). Dieser Befehl liest das Input-Port und schreibt den Inhalt in den Output-Buffer.
- D0 read output port (P20 - P27). Dieser Befehl liest das Output-Port und schreibt in den Output-Buffer.
  - Bit 0 -RC
  - Bit 1 A20 Gate
  - Bit 2 Speed select. Wird nur vom Award BIOS benutzt!
  - Bit 3 shadow enable
  - Bit 4 output buffer full
  - Bit 5 —
  - Bit 6 -KCKOUT
  - Bit 7 KDOUT
- D1 write output port. Die nächsten Daten, die zum Keyboard Data Port (PORT 60H) geschrieben werden, werden zum Output-Port des Kontrollers geleitet. Die Definition der Bits ist wie oben.
- E0 read test inputs (T0, T1). Bit 0 ist Data und Bit 1 ist Clock.
- F0-FF pulse output port (P20 - P27). Bits 0 - 3 des Kontroller Output-Port werden für ca. 6μs gepulst.

8.5 Bildschirme

Um zu sehen was man schreibt, braucht man nicht nur einen Monitor sondern auch einen Display-Adapter. Zu der Vielfallt an Monitoren kommt noch eine große Zahl an Adaptern. Nicht jede Kombination ist sinnvoll. Adapter mit Display-Prozessor haben oft ein Control-Port das nur auf 200H oder 300H gelegt werden kann!

8.5.1 Vertikale Auflösung

Die vertikale Auflösung kann durch teilen der horizontalen Ablenkfrequenz durch die vertikale Ablenkfrequenz errechnet werden. Ein monochrome Display hat z. B. bei einer horizontalen Ablenkfrequenz von 18 432 Hz und einer vertikalen Ablenkfrequenz von 50 Hz eine vertikale Auflösung von 368 Zeilen.

$$\frac{18432KHz(lines-per-second)}{50Hz(frames-per-second)} = 368 lines-per-frame$$

Davon müssen ca. 5 Prozent für die Justierung abgezogen werden, so daß 350 Zeilen verbleiben.

8.5.2 Horizontale Auflösung

Die Punkte oder pixels (picture elements) jeder Zeile werden durch eine Dot-Clock auf dem Controller erzeugt. Die horizontale Auflösung kann durch teilen der Dot-Frequenz durch die horizontale Ablenkfrequenz errechnet werden. Ein monochrome Display hat z. B. bei einer Dot-Frequenz von 16 257 KHz eine Auflösung von 882 pixels per line. Davon müssen 10 bis 15 Prozent für die Justierung abgezogen werden, so daß 720 pixels verbleiben.

8.5.3 Input Signale

Composite

Es ist ein Signal wie z. B. in einem Fernseher. Die Signale werden in dem Monitor getrennt.

Digital

Die Signale werden getrennt auf separaten Leitungen übertragen. Die Anzahl der Leitungen z. B. für jede Farbe bestimmt die Zahl der darstellbaren Farben. Bei je zwei Leitungen sind vier Kombinationen möglich, zusammen also 64 Farben (4 X 4 X 4).

Analog

Bei analog werden für jede Farbe nur eine Leitung gebraucht. Der Kontroller muß die digitalen Signale also in analoge wandeln. Mit einem 6 Bit DAC (digital analog converter) können also $2^{3*6} = 262144$ Farben dargestellt werden.

8.5.4 Adapter Typen

In der Tabelle 8.13 auf Seite 201 werden die Monitor-Adapter Typen mit den wesentlichen Merkmalen aufgelistet. Darüberhinaus werden für spezielle Aufgaben noch Sondertypen hergestellt.

Adapter Type	Year Intro.	Dot Clock	Monitor Support	Horizontal Freq.	Vertical Freq.	Resolution Color
MDA	1981	16 257 KHz	Digital Mono	18 430 Hz	50 Hz	720x350x1
HGC	1982	16 257 KHz	Digital Mono	18 430 Hz	50 Hz	720x350x1
CGA	1981	14 318 KHz	Composite color Digital RGB	15 750 Hz	60 Hz	640x200x16
MGA	1982		Digital Mono	18 100 Hz	50 Hz	720x350x1
EGA	1985	16 257 KHz	Digital RGB	21 860 Hz	60 Hz	640x350x16
		14 318 KHz	Digital RGB	15 750 Hz	60 Hz	640x200x16
		16 257 KHz	Digital Mono	18 430 Hz	50 Hz	720x350x4
PGA	1984			30 500 Hz	60 Hz	640x480x256
VGA	1987	25 175 KHz	Analog	31 500 Hz	70 Hz	640x400x16
		28 175 KHz	Analog	31 500 Hz	70 Hz	720x400x16
		25 175 KHz	Analog	31 500 Hz	60 Hz	640x480x16
		25 175 KHz	Analog	31 500 Hz	70 Hz	640x350x16
MCGA	1987			31 500 Hz	70 Hz	320x400x4
				31 500 Hz	70 Hz	640x400x2
				31 500 Hz	60 Hz	640x480x2
				31 500 Hz	70 Hz	320x200x256
VESA	1990				70 Hz	1024x768x16
8514/A Mode 0	1990	44,9 MHz		35 520 Hz	43,48 Hz	1024x768x16
8514/A Mode 1	1990	25,17 MHz		31 470 Hz	59,94 Hz	640x480x256
8514/A		65 MHz		48 500	60	1024x768x256
8514/A		80 MHz		50 000	43	1280x1024
8514/A		110 MHz		65 000	60	1280x1024
8514/A		125 MHz		65 000	43	1600x1200
8514/A		160 MHz		80 000	60	1600x1200
XGA	1990			35 520 Hz	43,48 Hz	640x480x256
				35 520 Hz	43,48 Hz	1024x768x256
				35 520 Hz	43,48 Hz	640x480x65 536
				35 520 Hz	43,48 Hz	1056x400x16

Tabelle 8.13: Tabelle der Monitor-Adapter

8.5.5 Video Datenformate

Jeder Platz in dem Video-Adapter Memory entspricht einem Punkt auf dem Bildschirm. Das erste Byte entspricht der linken oberen Ecke. Wird die Adresse erhöht, bewegt sich der Punkt von links nach rechts und von oben nach unten. Die Daten in dem Buffer sind entweder alphanumerisch oder graphisch.

Alphanumerisches Format

Im alphanumerischen Format werden für jedes Symbol zwei Bytes benutzt. Das erste Byte ist die ASCII Darstellung des Symbols. Das zweite sind die Attribute.

Alphanumerische Zeichen werden als Matrix von Punkten, Zeichenzellen, abgebildet. Der Kontroller liest eine Zeile im ASCII Format und wandelt sie in eine entsprechende Anzahl von Punkzeilen um. Gleichzeitig werden die Attribute in entsprechende Signale für die Farben und Intensität gewandelt.

Graphisches Format

Für jedes pixel wird der Wert in den buffer geschrieben. Bei z. B. 16 Farben besteht jedes pixel aus 4 Bit. MDA und CGA Adapter werden vom Video-Service des BIOS ROM unterstützt. EGA und VGA haben ein eigenes ROM BIOS.

8.5.6 Adapter Hardware Komponenten

In allen Video-Adaptern sind die gleichen Funktionseinheiten vorhanden.

1. Video Buffer

   Der Videobuffer ist RAM auf dem Adapter das in den Adressbereich des Prozessors hineinreicht. Jeder Platz in dem Memory entspricht einem Platz auf dem Bildschirm. Die Adapter haben unterschiedliche Adressbereiche.

2. Kathodenstrahlröhren Kontroller (CRTC)

   Der CRTC (Cathode Ray Tube Controller) generiert die horizontalen und vertikalen Zeitsignale die den Elektronenstrahl ablenken. Er beeinflußt auch den Adressenzähler um mit den Zeitsignalen synchron zu sein. Er kontrolliert auch die Größe und den Platz des Cursor im Textmodus und selektiert den Teil des Buffers, der dargestellt wird.

3. Zeichengenerator

   Im Textmodus werden die Zeichen als Bytepaare in den Buffer geschrieben. Das erste ist der ASCII Wert des Zeichens. Das zweite sind die Attribute, z. B. Fordergrund-/Hintergrund-Farbe, Blinken, Intensität. Der Zeichengenerator erzeugt aus dem ASCII Wert des Zeichens eine Punktmatrix, die auf dem Bildschirm dargestellt werden kann. Dazu benötigt er eine Zeichentabelle, die je nach Adapter im ROM oder im RAM an einer festen Stelle sein muß.

4. Attribute Dekoder

   Der Attributekontroller benutzt die Attributebits um die Signale des Signalgenerators den Attributen entsprechend zu beinflussen.

5. Video Signal Generator

   Der Signalgenerator erzeugt die Signale, die über die Verbindungsleitung zu dem Monitor gelangen, z. B. composite, digital oder analog.

6. Mode Kontroll Registers

   Die Mode Kontroll Register stellen den Adapter in den entsprechenden Mode. Der Zeichengenerator muß im graphischen Mode umgangen werden. Die Zeichen können 9 oder 8 Punkte breit sein.

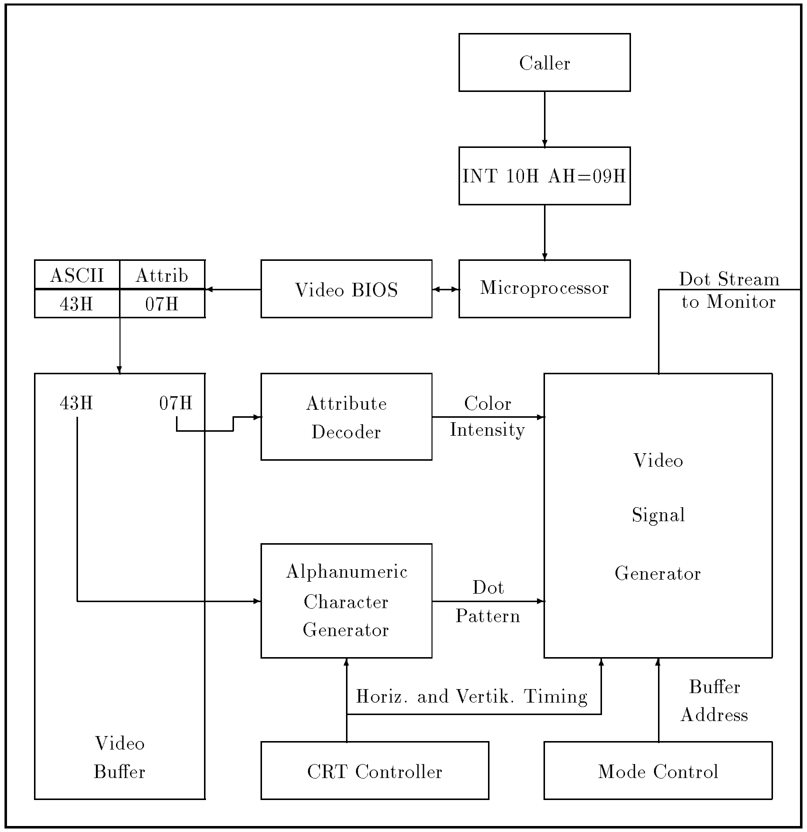


Abbildung 8.8: Darstellung eines Zeichens auf dem Bildschirm

Location	Länge	Description
10H	1	Number of devices installed Bits 15-14 = number of printer adapters Bits 13-12 = reserved Bits 11-09 = number of asynchrone adapters Bits 07-06 = number of disk drives Bits 05-04 = initial video mode Bit 02 = pointing device Bit 01 = 1 if math coprocessor Bit 00 = Diskette available for boot
49H	1	video mode setting
4AH	2	number of columns on screen (40 or 80)
4CH	2	current page size (in bytes)
4EH	2	offset of current page address
50H	16	cursor position on each page. Two bytes/page. First byte (low order) of each pair is column, second is row. 0,0 is the upper left corner.
60H	2	cursor type defined as 6845-compatible starting and ending scan lines. High-order byte holds starting scan line; low-order byte ending scan line.
62H	1	current page number
63H	2	6845-compatible I/O port number for current mode 03B4H = monochrome 03D4H = color
65H	1	current mode select register
66H	1	current palette value
84H	1	number of rows on screen minus one (24 or 25)
85H	2	character height (bytes/character)
87H	1	video control bits Bit 7 = clear RAM Bit 6-5 = Memory on adapter Bit 4 = not used Bit 3 = 0 if EGA-compatible adapter is active Bit 3 = 1 if EGA-compatible adapter is inactive Bit 2 = wait for display enable Bit 1 = 0 if color or ECD monitor is attached to EGA Bit 1 = 1 if monochrome monitor is attached to EGA Bit 0 = 0 if translate cursor video modes 0-3 when using ECD monitor in 350 line mode Bit 0 = 1 if inhibit cursor translation
88H	1	EGA switch data Bits 7-4 = Feature connector bits 3-0 Bits 3-0 = option switches 3-0
A8H	2	pointer to video parameter table and overrides (in segment:offset format)
100H	1	print flag: Bit 7 = 1 print screen in progress

Tabelle 8.14: Video Data Definition

Datendefinitionen die vom INT 10H, video service, benutzt werden sind im RAM-Segment 40 abgespeichert und in Tabelle 8.14 numerisch gelistet.

I/O Address	Read/Write	Index	Desciption
03B4H	R/W	—	CRTC index register
03B5H	W	01H	horizontal total
”	”	02H	horizontal display enable end
”	”	03H	start horizontal blanking
”	”	04H	end horizontal blanking
”	”	05H	start horizontal retrace pulse
”	”	06H	end horizontal retrace
”	”	07H	vertical total
”	”	08H	overflow
”	”	09H	preset row scan
”	”	0AH	maximum scan line/cursor start
”	”	0BH	cursor end
”	”	0CH	start address high
”	”	0DH	start address low
”	R/W	0EH	cursor location high
”	”	0FH	cursor location low
”	”	10H	light pen high
”	”	11H	light pen low
03B8H	”	—	mode control register

Tabelle 8.15: MDA I/O port

8.5.7 MDA Monochrome display adapter

Das 6845 CRTC index register ist mapped zu dem I/O port 03B4H. Der Wert der in das index register geschrieben wird, bestimmt das Register, das am Port 03B5H erscheint. Das 6845 mode control register liegt direkt am I/O Port 03B8H.

CRT controller register: Tabelle 8.15

I/O Address	Read/Write	Index	Desciption
03D4H	R/W	—	CRTC index register
03D5H	W	01H	horizontal total
"	"	02H	horizontal display enable end
"	"	03H	start horizontal blanking
"	"	04H	end horizontal blanking
"	"	05H	start horizontal retrace pulse
"	"	06H	end horizontal retrace
"	"	07H	vertical total
"	"	08H	overflow
"	"	09H	preset row scan
"	"	0AH	maximum scan line/cursor start
"	"	0BH	cursor end
"	"	0CH	start address high
"	"	0DH	start address low
"	R/W	0EH	cursor location high
"	"	0FH	cursor location low
"	"	10H	light pen high
"	"	11H	light pen low
03D8H	"	—	mode control register
03D9H	"	—	palette register

Tabelle 8.16: CGA I/O port

8.5.8 CGA Color graphics adapter

Das 6845 CRTC index register ist mapped zu dem I/O port 03D4H. Der Wert der in das index register geschrieben wird, bestimmt das Register, das am Port 03D5H erscheint. Das 6845 mode control register liegt direkt am I/O Port 03D8H; das palette register am I/O Port 03D9H.

CRT controller register: Tabelle 8.16

I/O address	Read/Write	Index	Description
03C2H	W	—	miscellaneous output
"	R	—	input status 0
3B2H/3D2H	R	—	input status 1
3BAH/3DAH	W	—	feature control

Tabelle 8.17: EGA General Register

8.5.9 EGA Enhanced graphics adapter

General Register: Tabelle 8.17

CRT sequencer register: Tabelle 8.18 auf Seite 208.

CRT cotroller gegisters: Tabelle 8.19 auf Seite 208.

Attribute Controller Registers: Tabelle 8.20 auf Seite 208.

Graphics controller registers: Tabelle 8.21 auf Seite 209.

I/O address	Read/Write	Index	Description
03C4H	R/W	—	Sequencer index register
03C5H	W	00h	Reset
"	"	01H	Clocking mode
"	"	02H	Map mask
"	"	03H	Character map select
03CCH	"	04H	Memory mode

Tabelle 8.18: EGA Sequencer Register

I/O Address	Read/Write	Index	Desciption
03B4H/03D4H	R/W	—	CRTC index register
03B5H/03D5H	W	01H	horizontal total
"	"	02H	horizontal display enable end
"	"	03H	start horizontal blanking
"	"	04H	end horizontal blanking
"	"	05H	start horizontal retrace pulse
"	"	06H	end horizontal retrace
"	"	07H	vertical total
"	"	08H	overflow
"	"	09H	preset row scan
"	"	0AH	maximum scan line/cursor start
"	"	0BH	cursor end
"	R/W	0CH	start address high
"	"	0DH	start address low
"	"	0EH	cursor location high
"	"	0FH	cursor location low
"	W	10H	vertical retrace start
"	R	10H	light pen high
"	"	11H	light pen low
"	W	11H	vertical retrace end
"	"	12H	vertical display enable end
"	"	13H	offset
"	"	14H	underline location
"	"	15H	start vertical blank
"	"	16H	end vertical blank
"	"	17H	CRTC mode control
"	"	18H	line compare

Tabelle 8.19: EGA I/O port

I/O address	Read/Write	Index	Description
03C0H	W	—	Attribute controller index register
"	"	00-0FH	Palette register 00H — 0FH
"	"	10H	Attribute mode control register
"	"	11H	Overscan color register
"	"	12H	Color plane enable register
"	"	13H	Horizontal PEL panning register

Tabelle 8.20: EGA Attribute controller register

I/O address	Read/Write	Index	Description
03CAH	R/W	—	Graphics 2 position register
03CCH	"	—	Graphics 1 position register
03CEH	"	—	Graphics controller index register
03CFH	"	00H	Set/Reset
"	"	01H	Enable Set/Reset
"	"	02H	Color compare
"	"	03H	Data rotate
"	"	04H	Read map select
"	"	05H	Graphics mode register
"	"	06H	Miscellaneous
"	"	07H	Color don't care
"	"	08H	Bit mask

Tabelle 8.21: EGA Graphics controller register

8.5.10 VGA Video graphics adpter

Reneral registers: Tabelle 8.22

Sequencer registers: Tabelle 8.23 auf Seite 210.

CRT controller registers: Tabelle 8.24 auf Seite 210.
Attribute controller registers: Tabelle 8.25 auf Seite 211.

Graphics controller registers: Tabelle 8.26 auf Seite 211.

Digital to Analog Converter (DAC) registers: Tabelle 8.27 auf Seite 211.

I/O address	Read/Write	Index	Description
03CCH	R	—	Miscellaneous Output
03C2H	W	—	Miscellaneous Output
03C2H	R	—	Input status 0
03BAH/03DAH	R	—	Input status 1
03CAH	R	—	Feature control
03BAH/03DAH	W	—	Feature control
03C3H	R/W	—	VGA enable
03C7H	R	—	DAC state

Tabelle 8.22: VGA General register

I/O address	Read/Write	Index	Description
03C4H	R/W	—	Sequencer index register
03C5H	”	00H	Reset
”	”	01H	Clocking mode
”	”	02H	Map mask
”	”	03H	Character map select
”	”	04H	Memory mode

Tabelle 8.23: VGA Sequencer register

I/O Address	Read/Write	Index	Desciption
03B4H/03D4H	R/W	—	CRTC index register
03B5H/03D5	”	01H	horizontal total
”	”	02H	horizontal display enable end
”	”	03H	start horizontal blanking
”	”	04H	end horizontal blanking
”	”	05H	start horizontal retrace pulse
”	”	06H	end horizontal retrace
”	”	07H	vertical total
”	”	08H	overflow
”	”	09H	preset row scan
”	”	0AH	maximum scan line/cursor start
”	”	0BH	cursor end
”	”	0CH	start address high
”	”	0DH	start address low
”	”	0EH	cursor location high
”	”	0FH	cursor location low
”	”	10H	vertical retrace start
”	”	11H	vertical retrace end
”	”	12H	vertical display enable end
”	”	13H	offset
”	”	14H	underline location
”	”	15H	start vertical blank
”	”	16H	end vertical blank
”	”	17H	CRTC mode control
”	”	18H	line compare

Tabelle 8.24: VGA I/O port

I/O address	Read/Write	Index	Description
03C0H	R/W	—	Attribute controller register
03C1H	”	00H-0FH	Palette register 00H – 0FH
”	”	10H	Attribute mode control register
”	”	11H	Overscan color register
”	”	12H	Color plane enable register
”	”	13H	Horizontal PEL panning register

Tabelle 8.25: VGA Attribute controller register

I/O address	Read/Write	Index	Description
03CEH	R/W	—	Graphics controller index register
03CFH	”	00H	Set/Reset
”	”	01H	Enable set/reset
”	”	02H	Color compare
”	”	03H	Data rotate
”	”	04H	Read map select
”	”	05H	Graphics mode register
”	”	06H	Miscellaneous
”	”	07H	Color don't care
”	”	08H	Bit mask
AFFFH	”	00H	Plane 0 System latch
AFFFH	”	01H	Plane 1 System latch
AFFFH	”	02H	Plane 2 System latch
AFFFH	”	03H	Plane 3 System latch

Tabelle 8.26: VGA Graphics controller register

I/O address	Read/Write	Index	Description
03C8H	R/W	—	Pixel address (write mode)
”	W”	—	Pixel address (read mode)
03C6H	R	—	DAC state register
”	R/W	—	Pixel mask (USER don't write)
03C9H	R/W	00H	Red value color 0
”	”	”	Green value color 0
”	”	”	Blue vlue color 0
”	”	01H	Red value color 1
...	...	...	...
”	”	FFH	Read value color FFH
”	”	FFH	Green vlue color FFH
”	”	FFH	Blue value color FFH
03C1H	”	14H	Color select register

Tabelle 8.27: VGA DAC register

Pin	Abkürzung EIA	gemäß V.24		Signal
1	AA	101		Ground
2	BA	103	zum Modem	Data send
3	BB	104	vom Modem	Data receive
4	CA	105	zum Modem	Request to send (RTS)
5	CB	106	vom Modem	Clear to send (CTS)
6	CC	107	zum Modem	Data set ready (DSR)
7	AB	102		Ground
8	CF	109	vom Modem	Data Carrier Detect (CD)
15	DB	114	vom Modem	Transmitter Clock
17	DD	115	vom Modem	Receiver Clock
20	CD	108.2	zum Modem	Data Terminal ready (CTR)
22	CE	125	vom Modem	Ring Indicator
23	CI	112	vom Modem	Data Signal Rate
24	DA	113	zum Modem	Transmitter Clock

Tabelle 8.28: Pinbelegung der seriellen Schnittstelle

8.6 Serielle Schnittstelle

Neben einfachen und doppelten Karten für serielle Schnittstellen sind Vielfach I/O Karten inclusive IDE Plattenkontroller vorhanden. Für besondere Anwendungen sind 4, 8, 16 und mehrfach Karten mit Software für DOS zu bekommen. Die Anschlüß sind in 9, 15, 24 und 25 poligen D-sub ausgeführt. Die Pinbelegung ist genormt.

8.7 Pointer Devices

8.7.1 Mouse

8.7.2 Griffel

8.8 Messinterfaces

Bei den Interfaces zur Steuerung der Apparaturen und zum Messen der Signale muß auf drei wichtige Punkte geachtet werden:

1. Adressen dürfen nicht mehrfach vorhanden sein. Die Anzahl der Register ist oft nicht angegeben. Sind Definitions-Files vorhanden, kann dort nachgesehen werden. Sonst mit dem Debugger feststellen (Unsicherheit bei 3-state Outputregistern).

2. Interrupts dürfen nicht mehrfach belegt sein. Shared-Interrupt Devices sind selten.

3. DMA-Kanäle dürfen nicht mehrfach belegt sein.

4. Der Adressbereich von Memmory-mapped Geräten muß gegebenenfalls in den Registern des Chipsets bearbeitet werden. Z. B. read-only, cachable u.s.w..

8.8.1 IEEE488

IEEE488 oder GPIB ist ein 8-Bit breiter Bus mit Handshake Leitungen. Die Maximale Länge aller Kabel ist 20m. Pro Kontroller können maximal 15 Geräte angeschlossen werden. Bus-Expander und -Extender sind von NI erhältlich.

Ein GPIB-Gerät kann als Talker, Listener und/oder Controller funktionieren. Als Talker werden Data messages zu einem oder mehreren Listener geschickt. Ein Kontroller managed den Informationfluß durch Versenden von Befehlen an die Geräte.

GPIB-Lines

Über die 8 Datenleitungen werden Daten und Befehle übertragen. Alle Befehle und die meisten Daten benutzen den 7-Bit ASCII oder ISO Code. Das 8te Bit wird enweder nicht oder als Parity-Bit benutzt. Die drei Handshakeleitungen sind jeweils mit einer Erdleitung verdrillt (twisted pair).

NFRD (not ready for data) Diese Leitung wird von allen Geräten benutzt, wenn sie Befehle empfangen. Listener benutzen diese Leitung, wenn sie Daten empfangen.

NDAC (not data accepted) Diese Leitung wird von allen Geräten benutzt, wenn sie Befehle empfangen. Listener benutzen diese Leitung, wenn sie Daten empfangen.

DAV (data valid) Der Kontroller benutzt diese Leitung, wenn er Befehle schickt, der Talker, wenn er Daten schickt.

Von den 5 Interface Management Leitungen sind nicht gepaart die Leitungen EOI und REN.

ATN (attention) Der Kontroller treibt die Leitung true, wenn er Befehle schickt. Ist die Leitung false, können Talker Daten schicken.

IFC (interface clear)

REN (remote enable)

SRQ (service request)

EOI (end or identify)

GPIB-Pinbelegung

Es wird ein 24-adriges abgeschirmtes Kabel benutzt. An beiden Enden sind sowohl Stecker als auch Buchse, so daß sie gestapelt werden können. Es werden negative-logic TTL-Pegel (low = true) benutzt.

Name	Pin Nr.	Pin Nr.	Name
DIO1	1	13	DIO5
DIO2	2	14	DIO6
DIO3	3	15	DIO7
DIO4	4	16	DIO8
EOI	5	17	REN
DAV	6	18	GND
NRFD	7	19	GND
NDAC	8	20	GND
IFC	9	21	GND
SRQ	10	22	GND
ATN	11	23	GND
SHIELD	12	24	SIGNAL GND

Address	read	write
02E1	DATA IN	BYTE OUT
06E1	INTERRUPT STATUS 1	INTERRUPT MASK 1
0AE1	INTERRUPT STATUS 2	INTERRUPT MASK 2
0EE1	SERIAL POLL STATUS	SERIAL POLL MODE
12E1	ADDRESS STATUS	ADDRESS MODE
16E1	CMD PASS THROUGH	AUXILIARY MODE
1AE1	ADDRESS 0	ADDRESS 0/1
1EE1	ADDRESS 1	END OF STRING

Tabelle 8.29: IOtech Register

GP488B IOtech

Die GPIB-Karte GP488B von IOtech benutzt den Kontroller 7210 von NEC. Dafür gelten die Adressen und Register in der Tabelle 8.29 auf Seite 214. Auf der Karte können die DMA-Kanäle 1, 2 oder 3 selektiert werden. Als IRQ-Ebene kann zwischen 2, 3, 4, 5, 6 oder 7 gewählt werden. Beide Unterbrechungsmöglichkeiten sind für "shared interrupt" (s. Seite 171) ausgelegt. Es können 0, 1, 3 oder 5 "wait states" selektiert werden.

IOtech stellt eine *multi language library* zur verfügung.

IEEE488 NI

8.8.2 CAMAC

DSP Technology

KSC

KSC benutzt scheinbar weder IRQ noch DREQ!

æ

Offset	Register
0	data low
1	data middle
2	data high
3	lower camac commands
4	upper camac commands
5	camac control/status
6	lam low
7	lam middle
8	lam high
9-F	reserved

Tabelle 8.30: Kinetic Systems 2925 Registers

Kapitel 9

Bestandteile von MS-DOS

MS-DOS ist ein modulares Betriebssystem, das aus mehreren Teilen mit speziellen Funktionen besteht. Wenn MS-DOS in das Memory geladen wird, wird es den Gegebenheiten angepasst. Dazu dienen unter anderem die Dateien CONFIG.SYS und AUTOEXEC.BAT. Im CONFIG.SYS können zusätzliche Gerätetreiber angegeben werden, die dann das Betriebssystem vergrößern.

9.1 Die drei wichtigsten Teile

Die drei wichtigsten Teile von MS-DOS sind:

Modul	MS-DOS Filename	PC-DOS Filename
MS-DOS BIOS	IO.SYS	IBMBIO.COM
MS-DOS kernel	MSDOS.SYS	IBMDOS.COM
MS-DOS shell	COMMAND.COM	COMMAND.COM

Tabelle 9.1: Die drei wichtigsten Teile

Diese Teile werden in dieser Reihenfolge direkt oberhalb der Vektortabelle geladen. Die Initialisierungsmodule werden nach Ende ihrer Arbeit freigegeben.

Das MS-DOS BIOS wird von dem OEM geliefert, der MS-DOS vertreibt, üblicherweise für einen bestimmten Rechner.
Der MS-DOS Kern wird von Microsoft geliefert und ist zur gleichen Zeit bei allen OEM's die selbe Version. Die shell ist ein ersetzbares Modul und wird üblicherweise von Microsoft als COMMAND.COM geliefert.

9.1.1 MS-DOS BIOS

In dem File IO.SYS befindet sich das MS-DOS BIOS und das Initialisierungsmodul SYSINIT. Das MS-DOS BIOS wird von einem OEM für einen bestimmten Rechner zugeschnitten. SYSINIT wird von Microsoft hergestellt und von dem OEM in IO.SYS eingebunden.
Das MS-DOS BIOS besteht aus einer Reihe von residenten device drivern und einem zusätzlichen Initialisierungsmodul des OEM. Die device driver erscheinen zuerst im MS-DOS BIOS weil sie resident bleiben nach dem Initialisieren. Die Initialisierungsroutine und SYSINIT werden nach dem Initialisieren entfernt. Es sind mindestens fünf driver resident: CON, PRN, AUX, CLOCK$ und ein driver für ein block device (Floppy). Die residenten character device driver befinden sich in der Liste vor den block device drivern. Installierbare character device driver werden vorher, block device driver hinterher eingetragen. Damit können vorhandene driver überschrieben werden. Der NUL device driver, der der erste in der Liste sein muß, ist im MS-DOS Kern vorhanden.

Device driver code kann zwischen IO.SYS und ROM aufgeteilt sein. Viele MS-DOS Systeme und alle PC-DOS kompatiblen haben ein ROM BIOS in dem sich primitive Routinen für die device befinden. Damit können viele unterschiedliche Hardware-Kombinationen von dem gleichen Betriebssystem genutzt werden.
Das IO.SYS ist absolut adressiert und bezieht sich auf das im Register CS angegebene Segment. Deshalb gilt wie bei den .COM files die Beschränkung auf 64kB. Größere IO.SYS sind möglich, wenn die Verwaltung der Segmente selbst vorgenommen wird und die device header in den ersten 64kB vorhanden sind.

9.1.2 MS-DOS Kern

Der Kern ist das Herz von MS-DOS. In ihm befinden sich alle Funktionen eines Betriebssystems in einer hardwareunabhängigen Form. MS-DOS sellt die folgenden Dienste, die über device driver gehen, zur Verfügung:

- File und Directory Management
- Character device input und output
- Zeit und Datum Unterstützung

Dienste die nicht über device driver gehen sind:

- Memory Management
- Task und Umgebungs Management
- Landesabhängige Konfiguration

Programme können Systemfunktionen mit der INT Instruktion erreichen. MS-DOS stellt die Interrupts 20H bis 3FH für diese Zwecke zur Verfügung.

- 20H Terminate Programm
- 21H MS-DOS Function Call
- 22H Terminate Routine Address
- 23H CTRL-C Handler Address
- 24H Critical Error Handler Address
- 25H Absolute Disk Read
- 26H Absolute Disk Write
- 27H Terminate and Stay Resident
- 28H-2EH Reserved
- 2FH Multiplex
- 30H-3FH Reserved

Der Interrupt 21H stellt die Systemfunktionen zur Verfügung. Dafür muß das Register AH mit der Funktionsnummer, andere Register der Funktion entsprechend, geladen und eine INT 21H Instruktion benutzt werden.
MS-DOS kann mit dem CONFIG.SYS file beeinflusst werden. Es ist ein Textfile in dem Kommandos zur Änderung der Größe oder Konfiguration von Tabellen oder zum Laden von zusätzlichen device drivern angegeben sind. Der file wird gelesen, wenn MS-DOS das erste mal nach "Reset" geladen wird.

9.1.3 SYSINIT

Nachdem der Betriebsystem-Lader die Files IO.SYS und MSDOS.SYS in das Memory geladen hat, wird die Kontrolle an den Anfang des IO.SYS abgegeben und damit beginnt die Systeminitialisierung. Hier schreibt MS-DOS nichts vor, weil die IBM-PC-Prozedur der de facto Standard geworden ist. Trotzdem geht MS-DOS von seine Standards aus.
IO.SYS kann in drei Module geteilt werden:

- Die residenten Device Drivers
- Die grundsätzlichen MS-DOS BIOS Initialisierungs-Module
- Das MS-DOS System-Initialisierungs-Modul SYSINIT

Die Initialisierungs-Module werden entfernt, sobald die Kontrolle an die shell abgegeben wird. Die residenten device driver bleiben im Memory und stehen deshalb im IO.SYS vor den Initialisierungs-Modulen.

9.1.4 MS-DOS Shell

Der Kommando-Interpreter ist das erste Programm das geladen wird nachdem MS-DOS geladen und initialisiert wurde. Es ist das Interface zwischen Benutzer und MS-DOS Kern. Der übliche shell bei MS-DOS ist der COMMAND.COM, ein kommando-orientiertes Interface. Andere können menü-gesteuert oder bildschirm-orientiert sein. COMMAND.COM kann ersetzt werden durch einen anderen Interpreter dadurch das der neue file in COMMAND.COM umbenannt wird oder durch den SHELL Befehl im CONFIG.SYS. Wird der SHELL Befehl benutzt können Parameter übergeben werden.
COMMAND.COM kann interne Befehle ausführen, Programme laden und ausführen oder batch files interpretieren. Die Programme die von COMMAND.COM ausgeführt werden sind .EXE oder .COM files, die von einem block device geladen wurden. Für batch files steht eine eingeschränkte Sprache zur Verfügung. Damit können Befehlsfolgen in Abhängigkeit von Parametern abgearbeitet werden. Wenn MS-DOS den COMMAND.COM startet wird nach dem file AUTOEXEC.BAT gesucht und der Inhalt interpretiert. COMMAND.COM hat auch terminate, CTRL-C und critical error handler deren Adressen in die Vektoren 22H, 23H und 24H geschrieben werden.
.COM files können per Definition nur in ein Segment geladen werden. Der COMMAND.COM schreibt seinen, über 64kB hinausgehenden, nichtresidenten Teil in das höchste Memory und entlässt dann das nichtbenötigte Memory. Das fällt aber nicht auf bis ein Programm beendet wird. Dann erhält der residente Teil des COMMAND.COM die Kontrolle zurück und prüft eine Checksumme in einem Bereich des höchsten Memories. Stimmt diese mit der abgespeicherten überein, wird davon ausgegangen das der nichtresidente Teil nicht überschrieben wurde. Anderenfalls wird der nichtresidente Teil von der Platte nachgeladen. In dem nichtresidenten Teil sind die internen Kommandos und der Batchprozessor enthalten. Der Kontrollprozessor und der Reloader sind im residenten Teil untergerbracht. Dadurch kann ein Anwendungsprogramm sich um die Größe des nichtresidenten Teils ausdehnen.
Die Kommandozeile wird zuerst auf einen internen Befehl hin untersucht. Danach wird in der augenblicklichen Directory nach einem file mit dem Kommandonamen und in der Reihenfolge mit der Namenserweiterung .COM, .EXE und .BAT gesucht. Danach wird die Suche in der im path angegebenen Reihenfolge fortgesetzt. Wird ein .COM oder .EXE file gefunden wird vom COMMAND.COM die EXEC Funktion (Interrupt 21H Funktion 4BH) benutzt. Batchfiles werden vom nichtresidenten Teil des Kommandoprzessors selbst interpretiert. Ist die Suche erfolglos, wird die Mitteilung "*Bad command or file name*" ausgegeben.

9.2 MS-DOS Umgebung

Der Pointer beim Offset 2CH im program segment prefix (PSP) zeigt auf einen Memorybereich dessen Länge mit dem Befehl SHELL und dem Parameter /E: gesetzt werden kann. Die Längenangabe wird auf eine durch 16 teilbare Zahl (paragraph) gerundet und kann maximal 32KByte sein. In diesen Bereich werden die Angaben von COMSPEC, PATH und PROMPT kopiert. Zusätzlich kann mit dem Befehl SET dieser Bereich verändert werden. Die danach gültigen Angaben befinden sich hintereinander, durch ein Null-Zeichen getrennt, in dem Memorybereich. Am Ende steht noch ein Null-Zeichen.

Benutzt ein Program die EXEC-Funktion um ein anderes Program zu laden und auszuführen, kann ein Pointer zu einer neuen Umgebung mitgegeben werden, sonst wird die alte Umgebung benutzt. Die neue Umgebung ist dem ersten Program nicht zugänglich. Änderungen darin gehen nach Ende des zweiten Programs verloren. Die Umgebung ist auch nicht für Datenaustausch zwischen Programmen gedacht. Dafür ist die intra application area in der ROM communication area vorhanden.

9.3 Das ROM-BIOS

Das ROM-BIOS sollte nicht mit dem MS-DOS-BIOS verwechselt werden. Der Anfang ist durch den Prozessor-Reset auf CS:IP = F000:FFF0 festgelegt. Es ist Schnittstelle zwischen der Hardware und, in Verbindung mit Teilen des IO.SYS, dem Betriebssystem MS-DOS. In dem ROM-BIOS befinden sich nicht nur die POST-Routinen (power on self test), die den Processor, die DMA- und Interrupt-Kontroler, die clock und das Memory testen, sondern auch die hardware-abhängigen Interrupt-Routinen. Das BIOS wird z. B. von 'AWARD', 'AMI' oder 'Phoenix' hergestellt. Bei IBM sind folgende Interrupts im ROM-BIOS vorhanden:

IRQ 09H Redirect IRQ2

IRQ 13H Redirect 80287

INT 02H NMI

INT 05H Print Screen

INT 09H Keyboard Hardware

INT 0EH Diskette Hardware

INT 10H Video BIOS

INT 11H Equipment

INT 12H Real mode memory size

INT 13H Disk entry

INT 14H Communications BIOS for RS232

INT 15H Cassette und andere (versionsabhängig)

INT 16H Keyboard BIOS

INT 17H Printer adapter BIOS

INT 19H Boot strap loader

INT 1AH Time of day

INT 1CH Timer 1

INT 70H IRQ8 Alarm

INT 76H Disk Hardware

0000	Cluster ist frei und benutzbar
0002 — FFEF	Cluster ist belegt. Die eingetragene Zahl ist die Nummer des Clusters mit der Fortsetzung des Files.
FFF0 — FFF6	reserviert, nicht benutzt.
FFF7	Bad cluster, nicht benutzbar!
FFF8 — FFFF	Dies ist der letzte Cluster in einem File.

Tabelle 9.2: FAT-Einträge

9.4 Boot Block

Die Vielzahl der Sektoren eines Massenspeichers müssen verwaltet werden. Dazu benutzt jedes System eine andere Strategie.

Bei älteren, noch nicht so intelligenten, Systemen wurde nach dem Reset der Block 0 von Spur 0 und Kopf 0 gelesen und als Urlader (Root of Boot) in dem Memory abgelegt. In diesem Block standen dann die Informationen über die Platte, so daß weitere Blöcke eingelesen werden konnten. So zog mann sich mit den Haaren aus dem Sumpf.

Unter DOS ist nur noch bei den Floppies der erste Block (Sektor 1, Spur 0, Seite 0) der Boot Block. Bei den Festplatten ist dort die Patition Tabelle, danach erst folgt der Boot Block. Das setzt also Entscheidungsmöglickeit, also laufendes Programm, voraus. Das befindet sich im BIOS.

Zusätzlich zu dem kleinen Boot-Programm befinden sich in dem Boot Block Angaben über das Medium, also z. B. Anzahl der Bytes pro Sektor, Anzahl der Sektoren pro Spur, Gesamtzahl der Sektoren, Anzahl der Köpfe usw.

9.5 Clusters

Nachdem die Platte formatiert wurde, ist die kleinste Einheit, die der Kontroller ansprechen kann, ein Block mit, bei DOS, 512 Byte. Diese Einheiten müssen verwaltet werden können, das heißt, das System muß den Zustand (frei, belegt, defekt) erkennen können. Auch hierfür sind unterschiedliche Strategien vorhanden. Eine Verwaltung auf dieser Basis ist sehr aufwendig, vor allem, wenn die Kapazitäten steigen. Deshalb wird die Sektorverwaltung nur bei Floppies vorgenommen. Bei den Festplatten werden mehrere Sektoren, z. B. 4, zu sogenannten Clustern zusammengefaßt und diese verwaltet. Damit sinkt der Verwaltungsaufwand aber bei kleinen Files steigt der verschwendete Plattenplatz, da nur jeweils ein ganzer Cluster belegt werden kann.

9.6 FAT File Allocation Table

Die FAT beginnt nach dem Boot Block und muß Platz haben für alle Clustereinträge. Die Länge ist also abhängig von der Plattenkapazität. Die Nummerierung beginnt mit 2 und wird bis zum Ende des Datenbereichs durchgezählt. Für jeden Cluster werden zwei Byte benutzt:

9.7 Root Directory

Die Root Directory befindet sich zwischen der FAT und dem Daten Bereich. Die Länge hängt von der Kapazität der Platte ab und wird von dem Programm FORMAT festgelegt. Eine Root Directory auf einer 1.2 MB Floppy wird z. B. mit 14 Clustern zu 512 Byte angelegt. Da jeder Eintrag aus 32 Byte besteht, sind also 224 Einträge möglich. Werden mehr benötigt, muß ein Unterverzeichnis angelegt werden. Bei

Bezeichnung	Größe	Format
Filename	8	ASCII Zeichen
Extension	3	ASCII Zeichen
Attributes	1	Jedes Bit bezeichnet ein Attribut
reserved	10	unbenutzt, bis jetzt
Zeit	2	kodiertes Wort
Datum	2	kodiertes Wort
erster FAT Eintrag	2	Wort
File Größe	4	Long integer

Tabelle 9.3: Direktory Einträge

einer 32 MB Festplatte werden 8 Cluster, entsprechend 512 Einträge, eingerichtet. Die 32 Byte werden aufgeteilt in:
Das Volume-label Bit darf auf einer Platte nur einmal vergeben werden und auch nur in der Root-Direktory. Dann wird das Namens- und Extension-Feld des Direktoryeintrags als zusammngehörig betrachtet und als "label" bezeichnet.

9.8 Program Segment Prefix

Das Präfix wird beim laden eines .EXE oder .COM Programms automatisch vor den ausführbaren Code gelegt. Es besteht aus 256 Byte (16 paragraph) memory.

- PSP:0000H Terminate Vektor. In den ersten zwei Byte steht eine Interrupt 20H Instruktion. Es ist der von CP/M benutzte 'warm boot'. Er sollte von neuen Programmen nicht benutzt werden.
- PSP:0002H In diesem Wort steht die Adresse des Segments das dem letzten vom Programm benutzten Paragraph folgt. Es soll nur für Programmlängenberechnungen benutzt werden und nicht um freies Memory zu erkunden.
- PSP:0004H Dieses Byte ist reserviert.
- PSP:0005H Im fünften Byte steht eine far call instruction gefolgt von Offset und Segment Adresse des Interrupt 21H. Soll nicht benutzt werden. Bedient nur die CP/M Funktionen 00H bis 24H.
- PSP:000AH - 0015H Hier werden die Vectoren der Interrupts 22H, 23H und 24H des Vorgängers abgelegt. Das neue Programm, zu dem dieser PSP gehört, muß diese Vektoren abspeichern, da es selbst neue Vektoren eintragen kann. Wird dieses Programm beendet, werden die alten Vektoren, des Vorgängers, restauriert.
- PSP:0016H - 002B reserviert.
- PSP:002CH Hier steht die Segmentadresse des ersten Paragraphen der Umgebung. Dort stehen auch die Informationen, die mit dem Befehl 'set' angegeben wurden.
- PSP:002EH - 004F reserviert.
- PSP:0050H Hier steht eine INT21H Instruktion gefolgt von RETF. Ein call zu dieser Adresse erreicht alle Funktionen. Ein INT21H im Programm ist aber schneller.
- PSP:0053H - 005BH reserved.
- PSP:005CH - 006BH Erster File Control Block (FCB)
- PSP:006CH - 007BH Zweiter FCB
- PSP:007CH - 007FH reserved.
- PSP:0080H - 00FFH Rest des Kommandos und disk transfer area (DTA)

Wert	Bedeutung
01	Read only
02	Hidden
04	System
08	Volume label
10	Sub-Directory
20	Archive
40	nicht benutzt, muß 0 sein
80	nicht benutzt, muß 0 sein

Tabelle 9.4: Tabelle der Attribut Bits

9.9 Struktur der .EXE files

Anders als .COM files beinhaltet der .EXE file Informationen die es dem .EXE Programm und MS-DOS erlauben, alle Fähigkeiten der 8086 Familie zu benutzen. Der Linker legt diese Informationen in einem header am Anfang des .EXE files an. Der header ist mindestens 512 Byte lang. Dieser header wird beim laden des .EXE files von MS-DOS in eine temporäre area geholt.

- 00 - 01H In den ersten zwei Byte steht die .EXE Signatur 4DH und 5AH (ASCII M und Z). Nur wenn beide stimmen und der file als ausführbarer angesprochen wurde, wird er als .EXE file behandelt. Sonst als .COM file unabhängig von der Extension.
- 02 - 03H Hier steht die Zahl der gültigen Bytes in der letzten 512 Byte großen Seite des files.
- 04 - 05H Hier steht die Zahl der verwendeten 512 Byte großen Seiten. Die Zahl der Bytes ist (N-1)*512+vorhergehendes Wort.
- 06 - 07H Hier steht die Zahl der Einträge in die relocation table.
- 08 - 09H Hier steht die Zahl der benutzten paragraphs des headers. Normal 32, denn 32*16=512.
- 0A - 0BH Hier steht die Minimum Zahl der paragraphs die das Programm zum Ausführen braucht (Parameter MINALLOC).
- 0C - 0DH Hier steht die Maximum Zahl der paragraphs die das Programm zum Ausführen braucht (Parameter MAXALLOC).
- 0E - 0FH Hier steht die Paragraph-Adresse des Stack-Segment relativ zum Start des Lademoduls. MS-DOS addiert zu diesem Wert die Start-Segment-Adresse und legt das Ergebnis im SS Register ab bevor das Programm ausgeführt wird.
- 10 - 11H Hier steht der anfängliche Wert des Stack Pointer der in das SP Register geladen wird.
- 12 - 13H Hier steht die one's complement checksum. MS-DOS benutzt dieses Wort zur Zeit nicht. (Wird sie nicht ermittelt, dann ist nur ein Wort umsonst. Wird sie aber nur nicht benutzt, so wird viel Zeit vergeudet!)
- 14 - 15H Hier steht der anfängliche Wert des Intruktions Pointers der in das IP Register geladen wird.
- 16 - 17H Hier steht der anfängliche Wert des Code Segments, der wie beim Stack Segment behandelt dann in das CS Register geladen wird.
- 18 - 19H Hier steht der Offset der relocation pointer table zum Anfang des files.
- 1A - 1BH Hier steht die Overlay Nummer des files. Normalerweise 0H, wenn nicht eine Hochsprache benutzt wird die Overlays zulässt.

Der vorhergehende Teil des headers wird 'formatted area' genannt. Danach folgt die relocation pointer table, wenn nicht Overlay-Informationen dazwischen geschoben werden. æ

Kapitel 10

MS-DOS Systemaufrufe

10.1 Liste der Systemaufrufe nach Funktionsgruppen

10.1.1 Character input

- 01 Character Input with Echo
- 03 Auxiliary Input
- 06 Direct Console I/O
- 07 Unfiltered Character Input without Echo
- 08 Character Input without Echo
- 0A Buffered Keyboard Input
- 0B Check Keyboard Status
- 0C Flush Buffer, Read Keyboard

10.1.2 Character output

- 02 Character output
- 04 Auxiliary output
- 05 Print character
- 06 Direct console I/O
- 09 Display string

10.1.3 Disk management

- 0D Disk reset
- 0E Select disk
- 19 Get current disk
- 1B Get default drive data

- 1C Get drive data
- 2E Set/reset verify flag
- 36 Get disk free space
- 54 Get verify flag

10.1.4 File management

- 0F Open file with FCB
- 10 Close file with FCB
- 11 Find first file
- 12 Find next file
- 13 Delete file
- 16 Create file with FCB
- 17 Rename file
- 1A Set DTA address
- 23 Get file size
- 2F Get DTA address
- 3C Create file with handle
- 3D Open file with handle
- 3E Close file
- 41 Delete file
- 43 Get/set file attributes
- 45 Duplicate file handle
- 46 Force duplicate file handle
- 4E Find first file
- 4F Find next file
- 56 Rename file
- 57 Get/set date/time of file
- 5A Create temporary file
- 5B Create new file
- 5C Lock/unlock file region

10.1.5 Information management

- 14 Sequential read
- 15 Sequential write
- 21 Random read
- 22 Random write
- 24 Set relative record
- 27 Random block read
- 28 Random block write
- 3F Read file or device
- 40 Write file or device
- 42 Move file pointer
- Interrupt 25 Absolute disk read
- Interrupt 26 Absolute disk write

10.1.6 Directory management

- 39 Create directory
- 3A Remove directory
- 3B Change current directory
- 47 Get current directory

10.1.7 Process management

- 00 Terminate process
- 31 Terminate and stay resident (TSR)
- 4B Load and execute program (EXEC)
- 4C Terminate process with return code
- 4D Get return code of child process
- 59 Get extended error information
- Interrupt 20 Terminate program
- Interrupt 27 Terminate and stay resident (TSR)

10.1.8 Memory management

- 48 Allocate memory block
- 49 Free memory block
- 4A Resize memory block
- 58 Get/set allocation strategy

10.1.9 Sonstiges system management

- 25 Set interrupt vector
- 26 Create new program segment prefix (PSP)
- 29 Parse filename
- 2A Get date
- 2B Set date
- 2C Get time
- 2D Set time
- 30 Get MS-DOS version number
- 33 Get/set CTRL-C check flag
- 34 Return address of InDOS flag
- 35 Get interrupt vector
- 38 Get/set current country
- 44 IOCTL (I/O controll character)
- 5E Network machine name/printer setup
- 5F Get/make assign list entry
- 62 Get program segment prefix address
- 63 Get lead byte table (version 2.25 only)
- Interrupt 22 Terminate routine address
- Interrupt 23 CTRL-C handler address
- Interrupt 24 Critical error handler address
- Interrupt 2F Multiplex interrupt

10.2 Interrupt Nr. 00H Divide by Zero

Der ROM BIOS handler ist eine einfache IRET Instruktion. MS-DOS lässt den Vektor aber auf eine Routine zeigen die eine Warnung auf den Schirm schreibt und das Programm abbricht. Dieser Abbruch kann den Rechner und das Betriebssystem in einen instabilen Zustand versetzen. Der Benutzer sollte das System neu starten und den Fehler beseitigen oder einen handler benutzen der den Interrupt 00H wie MS-DOS den Interrupt 24H behandelt.

10.3 Interrupt Nr. 01H Single-Step

Wenn die TF-Flag (bit 8 im Flag Register) gesetzt ist wird nach jeder Instruktion der Interrupt 01H ausgeführt. Da er auf eine IRET Instruktion zeigt, sieht es so aus, als würde nichts geschehen. Nur die Debugger, die die einzigen Anwender für diesen Interrupt sind, überschreiben den Vektor mit der Adresse ihres handlers.

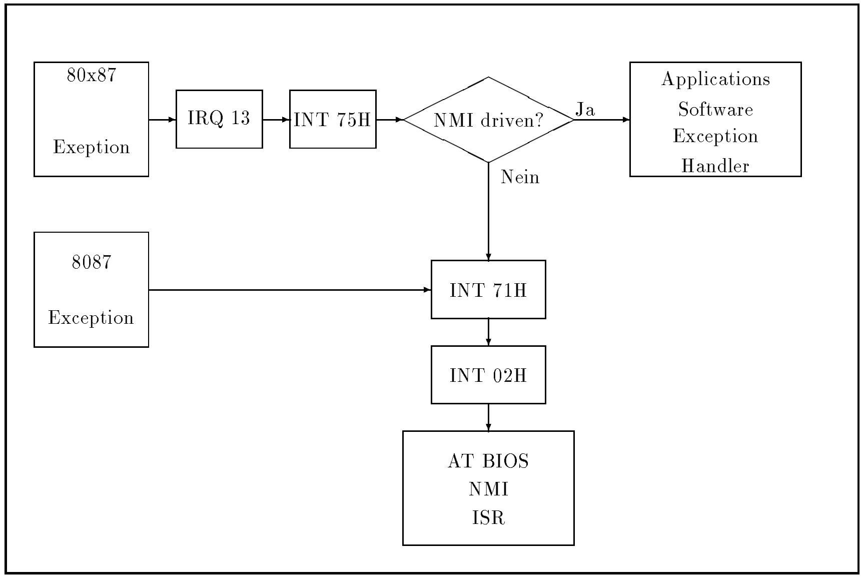


Abbildung 10.1: Coprocessor exeption handling

10.4 Interrupt Nr. 02H NMI Handler

Der NMI Handler zeigt die Parity-Error Message 'Check 1' für einen Fehler im Memory auf dem Board oder 'Check 2' für einen Fehler im Memory auf dem I/O Kanal. Er wird nur vom BIOS benutzt. Ein eigener handler scheint nicht sinnvoll zu sein, da auch die Speicherzellen für den code defekt sein können und deshalb falsch abgearbeitet würden. Das gilt für IBM.

Das Phoenix-BIOS für AT's unterscheidet:

1. **Memory Parity Error.** Das Phoenix-BIOS schreibt "Memory Parity Error" oder "I/O Card Parity Error" und dahinter die Adresse falls der BIOS-NMI-Handler sie finden kann. Wenn man dann der Aufforderung ein "S" für shut off NMI einzutippen folgt, wird in dem CMOS-RAM Port 70H das Bit 7 gesetzt und damit der NMI abgeschaltet und die restlichen Tests ausgeführt.
2. **DMA bus time out.** Wenn ein DMA-Device den bus länger als 7.8 μs erfolgt die NMI-Meldung "*DMA BUS Time-out*".
3. **Coprocessor exeptions.** Die IRQ-13-Leitung wird vom Coprocessor belegt, die über den INT 75H die AT-BIOS Coprocessor handling routine anspringt. NMI-Routinen für den Coprocessor könen sich in die in dem Bild 10.1 auf Seite 229 dargestellte Folge einreihen.

æ

10.5 Interrupt Nr. 03H Breakpoint Trap

Der Interrupt 03H wird in Verbindung mit dem Interrupt 01H von den Debuggern benutzt. Er wird von dem speziellen 1-Byte opcode 0CCH erzeugt. Dieser opcode wird von den Debuggern an die Adresse eines Breakpoints geschrieben. Wird die Adresse im Programmablauf erreicht wird ein Interrupt 03H erzeugt und der Debugger schreibt den originalen code in die Adresse. Z. B. mit trace kann dann der weitere Verlauf ermittelt werden.

10.6 Interrupt Nr. 04H Overflow Trap

Da die meisten Programme mit den Instruktionen JO (jump if overflow) und JNO (jump if no overflow) auskommen wird es nicht nötig sein den Vektor für den Interrupt 04H zu überschreiben um den Befehl INTO benutzen zu können. Will man es dennoch, muß das overflow-bit (bit11) in dem Flag-Register gesetzt werden.

10.7 Interrupt Nr. 05H Print Screen

Mit der Taste Print Screen wird der Interrupt 05H ausgeführt. Ein mehrfaches Betätigen der Taste wird nicht beachtet. Die Interrupts bleiben erlaubt. Der Printerstatus wird auf 'busy' und 'paper' out untersucht. Wenn beim initialisieren ein Error vorhanden ist, wird eine 77H in das Statusbyte (50:00) geschrieben und der Interrupt verlassen. Es wird bei IBM nur der Printer 0 benutzt.

Will man den original Interrupt 05H (Bound range exeeded) benutzen, muß der handler die Bedingungen ändern, da nach dem IRET der Prozessor die BOUND Instruktion erneut ausführt (LOOP).

10.8 Interrupt Nr. 06H Invalid Opcode

Der Interrupt 06H wird von IBM nicht benutzt und der Vektor zeigt auf eine IRET Intruktion. Deshalb werden protected mode Befehle im real mode nicht erkannt.

10.9 Interrupt Nr. 07H Coprocessor not available

Der Interrupt 07H wird von IBM nicht benutzt und der Vektor zeigt auf eine IRET Instruktion.

10.10 Interrupt Nr. 08H Timer

Der Processor meint mit diesem Interrupt 'double-fault'. Das BIOS macht aber keinen Unterschied.

Diese BIOS-Routine bedient den Timer der mit 1.19318 MHz getaktet wird und mit dem Teilerfaktor 65536 ca. 18.2 Interrupts pro Sekunde ergibt.

Dieser Handler benutzt den Timercounter (40:6C - 40:6F) in der ROM-Communication area von der auch die Tageszeit abgeleitet wird. Er zählt auch den Motor time out count (40:40) der Disketten herunter und schaltet gegebenenfalls den(ie) Motor(en) aus und setzt die flags zurück.

Er führt auch jedesmal den Interrupt 1CH durch, wenn der Benutzer einen geschrieben hat. Sonst steht dort nur IRET.

10.11 Interrupt Nr. 09H Keyboard Hardware Interrupt

Der Processor meint mit diesem Interrupt 'coprocessor segment overrun'. Das BIOS macht aber keinen Unterschied.

Der Keyboard Handler wartet auf ein Zeichen. Wurde eine Taste betätigt wird ein Interrupt 15H mit der Funktion 4FH ausgeführt.

10.12 Interrupt Nr. 0AH Invalid Task State Segment oder Network-Adapter

Mit diesem Interrupt meint der Processor 'Invalid Task State Segment'. Bei IBM wird der Vektor mit dem IRQ2-Vektor belegt oder es ist der zweite Interrupt-Kontroller angeschlossen oder der IRQ9.

Ist eine DEPCA-Karte vorhanden, so ist das der Interrupt des Network-Adapters mit Mouse.

10.13 Interrupt Nr. 0BH Segment not present oder secondary serial

Mit diesem Interrupt meint der Processor 'segment not present'. Bei IBM wird der Vektor mit dem IRQ3-Vektor belegt.

Dieser Interrupt wird auch vom scheduler des DECnet-DOS benutzt, wenn der data link layer DLL den Interrupt 69H zu sich umlenkt.

10.14 Interrupt Nr. 0CH Stack oder primary seriall

Mit diesem Interrupt meint der Prozessor 'stack'. Bei IBM wird der Vektor mit dem IRQ4-Vektor belegt.

10.15 Interrupt Nr. 0DH General Protection oder XT hard Disk

Mit diesem Interrupt meint der Prozessor 'general protection'. Bei IBM wird der Vektor mit dem IRQ5-Vektor belegt für die XT-fixed-disk oder den LPT2.

10.16 Interrupt Nr. 0EH Disketten Hardware Interrupt

Mit diesem Interrupt meint der 80386 Prozessor 'page fault'.

Der BIOS-Interrupt 0EH behandelt den Disketten Interrupt, nachdem der letzte Disketten-Befehl abgearbeitet wurde. Die Flag wird in seek_status gesetzt. Bei dem letzten Disketten-Befehl wurde der Interrupt 15H mit AH = 90H (Device Busy Loop) gerufen, um anderen Tasks die Arbeit zu ermöglichen. Jetzt wird ein Interrupt 15H mit der Funktion 91H und der Unterfunktion 01H ausgeführt, um das Ende anzuzeigen. Der Interrupt 0EH benutzt:

1. Location 3EH in System data area und
2. I/O Adresse 3F0H, Disketten controller statusregister A.

In Intels Technical Reference Manual für das Board 303 findet man:

This interrupt occurs when the floppy disk drive requires attention. During multi-sector transfers, the controller interrupts for each sector transferred. A completed operation also generates an interrupt. The BIOS handler sets bit 7 of the byte at 40:30h to indicate that an interrupt occurred.

10.17 Interrupt Nr. 0FH Printer control

In Peter Norton's Programmers Guide findet man "used in printer control". Wird in MS-DOS benutzt.

10.18 Interrupt Nr. 10H Video Services

Der Interrupt 10H des IBM PC ROM BIOS ist für die video services zuständig. Die Funktion wird im AH Register und die Unterfunktion im AL Register übergeben. Graphics pixel Koordinaten und Zeiger Spalten und Zeilen Koordinate sind immer auf Null bezogen. Nicht alle Funktionen werden von allen Adaptern ausgeführt.

10.18.1 Funktion 00H Set Video Mode

Diese Unterfunktionen benutzen die in der letzten Spalte angegebenen Bildschirmadressen.

00H	16 Graustufen Text	EGA:64-Farben	40X25	B000:8000H
01H	16/8 Farben Text	EGA:64-Farben	40X25	B000:8000H
02H	16 Graustufen Text	EGA:64-Farben	80X25	B000:8000H
03H	16/8 Farben Text	EGA:64-Farben	80X25	B000:8000H
04H	4 Farben graphics		320X200	B000:8000H
05H	4 Graustufen graphics		320X200	B000:8000H
06H	2 Graustufen graphics		640X200	B000:8000H
07H	monochrome text		80X25	B000:0000H
08H	16 Farben graphics		160X200	B000:0000H
09H	16 Farben graphics		320X200	B000:0000H
0AH	4 Farben graphics		640X200	B000:0000H
0DH	16 Farben graphics		320X200	A000:0000H
0EH	16 Farben graphics		640X200	A000:0000H
0FH	monochrome graphics		640X350	A000:0000H
10H	16/64 Farben graphics		640X350	A000:0000H
11H	2 Farben graphics		640X480	A000:0000H
12H	16 Farben graphics		640X480	A000:0000H
13H	256 Farben graphics		320X200	A000:0000H

Der Videomode wird im Register AL übergeben. Zurück kommt in AL

20h Mode ist größer 7

30h Mode ist zwischen 0 und 5 oder 7

3Fh Mode ist 6

Wird der gleiche Mode erneut gesetzt, wird der Schirm gelöscht. Um ein Zurücksetzen der Farbpaletteninformation zu vermeiden sollte der Schirm nur mit der Funktion 06h gelöscht werden. Der Cursor wird im graphischen Mode nicht gezeigt. Ist Bit 7 gesetzt, also Videomode größer 80h, Wird der Bildschirm auf EGA- und VGA-Karten nicht gelöscht.

10.18.2 Funktion 01H Set Cursor Size and Shape

Diese Parameter werden in der Adresse 40:60h abgespeichert.

- CH =

  Bit0 - 4 starting scan line
  Bit5 to shut cursor off
  Bit6 - 7 muß 00h sein, sonst wird das Blinken des Cursor unberechenbar.

- CL =

  Bit0 - 4 ending scan line
  Bit5 - 6 show cursor
  Bit7 undefined

$CH < CL$ normaler einteiliger cursor; $CH > CL$ zweiteiliger cursor; CH=20H kein cursor.

10.18.3 Funktion 02H Set Cursor Position

Die Parameter werden in der Adresse 40:50h abgespeichert.

- BH = display page (0 in graphics)
- DH = row number (0 ist oben)
- DL = line number (0 ist links)

10.18.4 Funktion 03H Read Cursor Positon, Size and Shape

Im BH Register muß die gewünschte Bildschirmseite angegeben werden. Die Informationen für die Koordinaten werden von der Adresse 40:50h gelesen; die für den Cursor-Type von der Adresse 40:60h.

- CH = starting scan line
- CL = ending scan line
- DH = row number
- DL = column number

10.18.5 Funktion 04H Read Light-Pen Position

Diese Funktion liest den Status und die Position des Lichtgriffels. Sie sind nicht auf VGA-Adaptern vorhanden und nicht nützlich bei lang nachleuchtenden Schirmen. Die Position kann für hochauflösende Geräte nicht gut genug bestimmt werden.

- AH = status
  - 01H pen triggered
  - 00H not triggered
- BX = pixel column number (0 - 319, 639).
- CH = pixel line number (0 - 199).
- CL = pixel line number (0 - nnn) für Modes über 200 Linien Auflösung.
- CX = pixel line number for some EGA modes
- DH = character row number
- DL = character column number

10.18.6 Funktion 05H Select Active Page

Diese Funktion setzt die aktive Seite. Wird die vorgehende erneut gesetzt, so wird der Schirm gelöscht. Die Seitenzahlen beginnen bei 0 und gehen im Textmode bis 7. Die Seite 0 fängt am Anfang des Bildschirmspeichers an. Die Anzahl von Seiten für EGA und VGA hängt nur von der Größe des Memories ab. Ein Umschalten der Seiten veränder deren Inhalt nicht. Unabhängig von der aktiven Seite kann Text mit den Funktionen 02h, 09h und 10h auf jede Seite geschrieben werden.

- AL = page number
  - 00-07H 40-column text modes
  - 00-03H 80-column text modes
  - varies EGA graphics modes

  Jede Seite = 2 KB in 40-column text mode, 4KB in 80-column text mode.

10.18.7 Funktion 06H Scroll Window Up

Es gelten die gleichen Unterfunktionen wie bei Funktion 07H.

10.18.8 Funktion 07H Scroll Window down

Mit den Registern CH, DH, CL und DL wird ein Fenster definiert das mit den Funktionen 06h und 07h auf- oder abwärts gerollt werden kann um die in dem Register AL angegebene Zahl von Zeilen. Wird in AL 00h angegeben, so wird das Fenster gelöscht. Informationen ausserhalb des Fensters bleiben bestehen. Nur ein Fenster kann eröffnet werden. Diese Funktionen gelten für graphischen und Text Mode.

- AL = number of lines to scroll (00H blanks screen)
- BH = display attributes for blank lines
- CH = row number of upper left corner (des Fensters).
- CL = column of upper left corner (des Fensters).
- DH = row number of lower right corner (des Fensters).
- DL = column number of lower right corner (des Fensters).

10.18.9 Funktion 08H Read Character and Attribute at Cursor

Nur im Textmode muß im Register BH die gewünschte Seite angegeben werden.

- Im text mode
  - AH = color attributes of character
  - AL = ASCII character from current location
- Im graphics mode AL = ASCII character

10.18.10 Funktion 09H Write Character and Attribute

Die Funktionen 09h und 0Ah sind vergleichbar. Die Funktion 09h wird im graphischen Mode gebraucht und im Text Mode, wenn die Kontrolle über Vordergrund und Hintergrund Attribute für jedes Zeichen benötigt wird.

Für CGA-Graphics zeigt der Interrupt 1Fh auf eine Tabelle mit der Bitmap der Zeichen 80 — FFh. Für EGA- und VGA-Graphics wird der Interrupt 43h benutzt.

Wenn das Zeichen geschrieben ist, muß der Cursor mit der Funktion 02h bewegt werden.

Soll ein Zeichen ohne Änderung des Attributs geschrieben werden sollte die Funktion 0Ah benutzt werden.

Ist das Bit 7 im Register BL gesetzt, wird das neue Zeichen mit dem alten XORed.

- AL = ASCII character to write
- BH = display page, im graphischen Mode 13h Hintergrundfarbe.
- BL = text attribute or graphics foreground color.
- CX = number of times to write character (repeat count).

10.18.11 Funktion 0AH Write Character only

Mit dieser Funktion werden Zeichen auf den Schirm geschrieben ohne die Attributes zu ändern. Ist Bit 7 im Register BL gesetzt, so wird das alte Zeichen mit dem neuen XORed. Alle Zeichen werden dargestellt.

- AL = ASCII character to write
- BH = display page
- BL = graphics foreground color, nicht benutzt im text mode.
- CX = number of times to write character (repeat count).

10.18.12 Funktion 0BH Select Color Palette

Diese Funktion setzt die Farben in "medium resolution graphics mode". Im CGA-graphics-Mode selektiert Bit 4 im Register BL zwischen normaler und hoher Intensität. EGA und VGA emulieren dies durch Auswahl einer Palette mit hochintensiven Farben.

BH	=	00h
	=	Mode = 4 oder 5 setzt Hintergrundfarbe mit dem Wert in BL.
	=	Mode = 0, 1, 2 oder 3 setzt Randfarbe mit dem Wert in BL.
	=	Mode = 6 setzt Vordergrundfarbe mit dem Wert in BL.
BL	=	(0 — 31), when colors (16 — 31 are high intensity background set).
	oder	
BH	=	01h select Palette für Modes 4 oder 5.
BL	=	00h Palette = grün (1), rot (2), braun (3).
	=	01h Palette = cyan (1), magenta (2), weiß (3).

10.18.13 Funktion 0CH Write Pixel Dot

Bis auf den Mode 13H wirkt Bit 7 im Register AL als Inverterflag. Ist Bit 7 im Register AH (Funtion-Register) gesetzt, wird der alte Farbwert mit dem neuen XORed.

- AL = color attribute of pixel
- CX = pixel column number
- DX = pixel raster line number

10.18.14 Funktion 0DH Read Pixel Dot

- Input
  - BH = page number
  - CX = pixel column number
  - DX = pixel raster line number
- Output AL = pixel color attribute

10.18.15 Funktion 0EH Write Character as TTY

Das Zeichen im Register AL in die aktuelle Cursorposition geschrieben und der Cursor um ein Zeichen weiter bewegt. Die Werte im CMOS und BIOS gelten.

- AL = ASCII character
- BH = display page
- BL = foreground color of character, nicht benutzt in text mode

Die Cursor Position wird verändert. Beep, Backspace, Linefeed und carriage return werden ausgeführt. Alle anderen Zeichen werden displayed.

10.18.16 Funktion 0FH Get Current Video Mode

Mit dieser Funktion kann man die Modeinformationen erhalten und für spätere Restaurierung abspeichern.

- AH = characters per line (20, 40 oder 80) (40:4Ah).
- AL = current video mode (s. Funktion 00H) (40:49h).
- BH = active display page (40:62h).

10.18.17 Funktion 10H set palette/color register

Diese Funktion hat 14 Unterfunktionen um die Farbpaletten der EGA/VGA video controller zu kontrollieren. Diese Unterfunktionen kontrollieren Farbe, Blinken und den Video-DAC des VGA-Kontrollers.

Unterfunktion 00H set single palette

Diese Unterfunktion setzt einen einzelnen Farbwert in der internen 16-Werte Palette des Attribut Kontrollers.

- BH = neuer Farbwert.
- BL = Palettenregister.

Unterfunktion 01H set overscan register

Diese Unterfunktion setzt die "overscan" Farbe, Randfarbe, des augenblicklich gültigen Video-Modes. Im Register BH befindet sich der Wert der Farbe die gesetzt werden soll.

Unterfunktion 02H set all palette register

Diese Unterfunktion setzt alle 16 internen Paltte-Register des Attribute Kontrollers und das Overscan-Register (11h) mit den Werten aus einer Tabelle mit 17 Werten auf die ES:BX zeigt. Byte 16 ist die Randfarbe.

Unterfunktion 03H togle intensity/blinking bit

Diese Unterfunktion setzt die Hintergrundintensität und das Vordergrundblinken.

- BL = 00 enable background intensity.
- BL = 01 enable foreground blinking.

Unterfunktion 04H ist reserviert

Unterfunktion 05H ist reserviert

Unterfunktion 06H ist reserviert

Unterfunktion 07H read individual palette register

Mit dieser Unterfunktion kann der Farbwert in das Register BH für das Paletten-Register, das in BL in dem Bereich von 00 — 0Fh angegeben wurde, gelesen werden.

Unterfunktion 08H read overscan register

Mit dieser Unterfunktion wird der Farbwert des Overscan-Registers der gültigen Farbpalette in das Register BH gelesen.

Unterfunktion 09H read all palette register

Mit dieser Unterfunktion wird der Inhalt des gültigen Paletten-Registers und das Overscan-Register in eine Tabelle gelesen, auf die die Register ES:DX zeigen. Die Tabelle muß 17 Byte lang sein. Die Bytes 0 — 15 enthalten die Farbwerte und das Byte 16 den Wert der Randfarbe.

Unterfunktion 10H set individual color register

Diese Unterfunktion liest eine RGB-Farbtabelle, auf die die Register ES:DX zeigen. Das erste zu setzende DAC-Farbregister muß im Register BX angegeben werden. Im Register CX wird die Anzahl zu setzender DAC-Farbregister angegeben. Für jede Grundfarbe muß ein Byte angegeben werden, wobei die Bits 0 — 5 die Intensität, 00h = off und 3Fh höchste, angeben.

- BX = color register to set.
- CH = green value to set.
- CL = blue value to set.
- DH = red value to set.

Unterfunktion 11H ist reserviert

Unterfunktion 12H set block of color registers

Diese Unterfunktion setzt eine aufeinander folgende Reihe von DAC-Farbregistern.

- BX = first color register to set.
- CX = number of color register to set.
- ES:DX = pointer to table of color values. Table Format is red, green, blue, red, green, blue, ...

Unterfunktion 13H select color paging mode

Nicht gültig im Mode 13H. Im Register BL steht 00H für *select paging mode* oder 01H für *select page*. Die 256 Farbregister des DAC-Chip können entweder in 4 Blöcke zu 64 Registern oder in 16 Blöcke zu 16 Registern aufgeteilt werden.

BL = 00 select block mode. BH = 00 setzt 4 Blocks zu 64 Registern. BH = 01 setzt 16 Blocks zu 16 Registern.

BL = 01 select individual block. BH = 00 — 03 für 4-Block Mode. BH = 00 — 0Fh für 16-Block Mode.

Unterfunktion 14H ist reserviert

Unterfunktion 15H read single DAC color register

Im Register BX wird das Farbregister angegeben, das gelesen werden soll.

- CH = grün Wert
- CL = blau Wert
- DH = rot Wert

Unterfunktion 16H ist reserviert

Unterfunktion 17H read block of color register

Mit dieser Unterfunktion kann ein Bereich von DAC-Farbregistern in einen Memory-Bereich eingelesen werden der mit den Registern ES:DX angegeben wurde.

BX = erstes DAC-Farbregister.

CX = Anzahl der Farbregister, die gelesen werden sollen.

ES:DX = Pointer zu einem Buffer mit jeweils 3 Byte für jedes Farbregister in def Folge Rot, Grün, Blau ...

Unterfunktion 18H ist reserviert

Unterfunktion 19H ist reserviert

Unterfunktion 1AH read color paging status

BH = current page.

BL =current pagin mode
= 00h 4 pages of 64 registers
= 01h 16 pages of 16 registers

Unterfunktion 1BH sum color values to gray shades

Diese Unterfunktion liest die Werte für die angegebenen Farbregister und berechnet den Grauwert mit 30% rot, 59% grün und 11% blau.

- BX = color register to start with
- CX = count of how many to sum.

10.18.18 Funktion 11H load character generator

Diese Funktion hat 15 Unterfunktionen.

Unterfunktion 00H load user text mode font

Diese Unterfunktion lädt einen benutzerdefinierten Font in die Memory-Map 2, die im Register BL angegeben wurde.

BH = number of bytes per character.

BL = block to load (valid values are 00h — 07h).

CX = number of characters to store.

DX = character ID of first character in ES:BP table.

ES:BP = pointer to the user table.

Unterfunktion 01H load ROM 8x14 text mode font

Diese Unterfunktion lädt einen ROM-residenten 8x14 Font in die Memory-Map 2, die im Register BL angegeben wurde.

Unterfunktion 02H load ROM 8x8 double dot text mode font

Diese Unterfunktion lädt einen ROM-residenten 8x8 Font in die Memory-Map 2, die im Register BL angegeben wurde.

Unterfunktion 03H set block specifier

EGA- und VGA-compatible Hardware erlaubt die gleichzeitige Darstellung von 512 Zeichen im Textmode. Alle Zeichen werden in der video memory map 2 in 8K Blöcken abgelegt. Jeder 8K Block kann 256 Zeichen speichern.

- Der Wert der Bits 4, 1 und 0 des Registers BL zeigt auf den ersten Zeichenblock, der Wert der Bits 5, 3 und 2 des Registers BL zeigt auf den zweiten Zeichenblock.
- Der Wert des Bits 3 in jedem Zeichen-Attribute gibt an von welchem Block das Zeichen genommen werden soll.
  - Bit3 = 0, das Zeichen wird von dem Block genommen der durch die Bits 4, 1 und 0 im Register BL bezeichnet wird.
  - Bit3 = 1, das Zeichen wird von dem Block genommen der durch die Bits 5, 3 und 2 im Register BL bezeichnet wird.
- Sind die Bits 4, 1 und 0 gleich den Bits 5, 3 und zwei wird die Vordergrundintensität mit dem Bit3 des Zeichenattributs geschaltet.

Unterfunktion 04H load 8x16 ROM text mode font

Diese Unterfunktion lädt den ROM-residenten Font 8x16 in den Zeichenblock der in dem Register BL angegeben wird.

Unterfunktion 10H load user text mode font

Diese Unterfunktion lädt den Zeichensatz aus der Tabelle, die von den Registern ES:BP angegeben wird.

BH = number of bytes per character.

BL = block to load (0 — 7).

CX = number of characters to store.

ES:BP = pointer to table.

Unterfunktion 11H load ROM 8x14 text mode font

Diese Unterfunktion lädt den ROM-residenten Font 8x14 in den Zeichenblock der in dem Register BL angegeben wird.

Unterfunktion 12H load ROM 8x8 double dot text mode font

Diese Unterfunktion lädt den ROM-residenten Font 8x8 in den Zeichenblock der in dem Register BL angegeben wird.

Unterfunktion 14H load 8x16 ROM text mode font

Diese Unterfunktion lädt den ROM-residenten Font 8x16 in den Zeichenblock der in dem Register BL angegeben wird.

Unterfunktion 20H set user graphics characters pointer at INT 1FH

Diese Unterfunktion setzt den Vector des Interrupt 1Fh auf den Wert, der in den Registern ES:BP gegeben wird. ES:BP zeigt auf eine Tabelle mit den Zeichen 80h bis FFh.

Unterfunktion 21H set user graphics font pointer at INT 43H

Diese Unterfunktion setzt den Vector des Interrupt 43h auf den Wert, der in den Registern ES:BP gegeben wird. ES:BP zeigt auf eine Tabelle mit den Zeichen 80h bis FFh.

- BL = rows on screen specifier.
  - 00h if user-supplied
  - 01h = 14 (0Eh) rows
  - 02h = 25 (19h) rows
  - 03h = 43 (2Bh) rows
- CX = points (bytes/character)
- DL = rows per screen if BL = 00h
- ES:BP = pointer to user table.

Unterfunktion 22H use ROM 8x14 font for graphics

- BL = rows on screen specifier.
  - 00h if user-supplied
  - 01h = 14 (0Eh) rows
  - 02h = 25 (19h) rows
  - 03h = 43 (2Bh) rows
- DL = rows per screen if BL = 00h

Unterfunktion 23H use ROM 8x8 double dot font for graphics

- BL = rows on screen specifier.
  - 00h if user-supplied
  - 01h = 14 (0Eh) rows
  - 02h = 25 (19h) rows
  - 03h = 43 (2Bh) rows
- DL = rows per screen if BL = 00h

Unterfunktion 24H use ROM 8x16 font for graphics

- BL = rows on screen specifier.
  - 00h if user-supplied
  - 01h = 14 (0Eh) rows
  - 02h = 25 (19h) rows
  - 03h = 43 (2Bh) rows
- DL = rows per screen if BL = 00h

Unterfunktion 30H get font pointer information

Input BH = font pointer
= 00h return current INT 1Fh pointer
= 01h return current INT 43h pointer
= 02h return current ROM font 8x14 font pointer.
= 03h return current ROM font 8x8 font pointer.
= 04h return current ROM font 8x8 font pointer (top).
= 05h return current ROM font 9x14 font pointer.
= 06h return current ROM font 8x16 font pointer.
= 07h return current ROM font 9x16 font pointer.
= >07h returns all registers (including AX) preserved.

Output CX = bytes per character.
DL = maximum number of rows on screen.
ES:BP = pointer to character table.

10.18.19 Funktion 12H alternate select

Diese Unterfunktionen müssen im Register BL angegeben werden!

Unterfunktion 10H return configuration information

Mit dieser Unterfunktion wird der Status der EGA- oder VGA-Hardware ermittelt.

BH = 00h = EGA/VGA color mode; 01h = EGA/VGA monochrome mode.

BL = memory available

- 00h = 64k
- 01h = 128k
- 02h = 192k
- 03h = 256k

CH = adapter bits

CL = switch settings

Unterfunktion 20H switch to alternate print screen routine

Diese Unterfunktion setzt den INT 05h Vektor auf eine alternative Routine. Die BIOS print screen routine geht von 25 Zeilen aus.
WO WIRD DIE NEUE ANGEGEBEN????????????
Im Grafikkarten-BIOS!

Unterfunktion 30H select scan lines for text modes

Das VGA-BIOS setzt alle Textmodes beim POST mit 400 lines. Eine Änderung wird erst mit dem nächsten Modewechsel übernommen. Im 200 Linemode wird jede line zweimal geschrieben.

Input: AL = number of scan lines

- 00h = 200 scan lines.
- 01h = 350 scan lines.
- 02h = 400 scan lines.

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

Unterfunktion 31H enable/disable default palette loading during set mode

Input: AL =

- 00h enable
- 01h disable

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

Unterfunktion 32H enable/disable video

Input: AL =

- 00h enable
- 01h disable

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

Unterfunktion 33H enable/disable summing to gray scale

Input: AL =

- 00h enable
- 01h disable

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

Unterfunktion 34H enable/disable cursor scaling

Input: AL =

- 00h enable
- 01h disable

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

Unterfunktion 35H switch display

Input: AL =

- 00h turn off initial video adapter. The user must establish a 128-Byte save area pointed to by ES:DX.
- 01h
- 02h
- 03h

ES:DX = pointer to 128-Byte save buffer area.

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

Unterfunktion 36H video screen off/on

Input: AL =

- 01h screen off
- 00h screen on

Output: AL =

- 12h funktion is supported.
- 00h VGA not active.

10.18.20 Funktion 13H Write Character String

- AL = subfunction number
  - 00H string shares attribute in BL, cursor unchanged
  - 01H string shares attribute in BL, cursor advanced
  - 02H each character has attribute, cursor unchanged
  - 03H each character has attribute, cursor advanced
- BH = active display page
- BL = string attribute (nur wenn AL = 00H oder 01H)
- CX = length of character string
- DH = starting row number
- DL = starting column number
- ES:BP = address of string to be displayed

Für AL = 00H oder 01H string = (char,char,char,char,...)
Für AL = 02H oder 03H string = (char,attr,char,attr,...)
Für AL = 01H oder 03H ist die neue cursor Position hinter dem letzten Character.

AL bit0 = 0 the cursor is restored to where it was before the write string operation begun.

AL bit0 = 1 the cursor remains at the last character written.

AL bit1 = 0 carriage return, line feed, backspace and bell are interpreted as commands rather than a printable characters. The string contains only display characters, and the attribute is taken from the BL register.

AL bit1 = 1 each string character to be displayed is followed by iys attribute.

10.18.21 Funktion 1AH read/write display combination code

Der Code für den aktiven Monitor wird im Register BL, der für den inaktiven im Register BH angegeben.

00h kein Display angeschlossen.

01h monochrome Display Adapter (MDA) mit monochrome Monitor.

02h color graphics Adapter (CGA) mit color Monitor.

04h enhanced graphics Adapter (EGA) mit color Monitor.

05h EGA mit monochrome Monitor.

06h professional graphics Adapter (PGA) mit color Monitor.

07h video graphics Adapter (VGA) mit monochrome analog Monitor.

08h VGA mit color analog Monitor.

0Ah MCGA mit digitalem color Monitor.

0Bh MCGA mit analogen monochrome Monitor.

0Ch MCGA mit analogem color Monitor.

FFFFh Display type unknown.

Unterfunktion 00H read display combination code

Input: AH = 1Ah; AL = 00h

Output:
- AL = 1Ah Funktion is supported.
- BH = inactive display code.
- BL = active display code.

Unterfunktion 01H write display combination code

Input:
- AH = 1Ah
- AL = 01h
- BH = alternate display code (if any).
- BL = active display code.

Output: AL = 1Ah Funktion is supported.

10.18.22 Funktion 1BH return functionality/state information

Diese Funktion schreibt den Status der Video-Hardware in ein 40h-Byte Buffer auf den die Register ES:DI zeigen.

Input:
- AH = 1Bh
- BX = 0000h implementation type
- ES:DI = pointer to 40h byte buffer

Output: AL = 1Bh function succesful

Offset	Size	Description
DI+00h	DWORD	Segment/Offset of static functionality fixed address
DI+04h	1 Byte	video mode
DI+05h	1 Word	column/row
DI+07h	1 Word	display buffer length
DI+09h	1 Word	starting address of buffer
DI+0Bh	1 Word	array of 8 cursor positions
DI+1Bh	1 Word	cursor type
DI+1Dh	1 Byte	active page
DI+1Eh	1 Word	CRT controller base address
DI+20h	1 Byte	current value of 3x8 register
DI+21h	1 Byte	current value of 3x9 register
DI+22h	1 Byte	rows on screen
DI+23h	1 Word	character height (bytes/character)
DI+25h	1 Byte	active display code
DI+26h	1 Byte	inactive display code
DI+27h	1 Byte	number of colors available in current mode
DI+29h	1 Byte	display pages supported for current video mode
DI+2Ah	1 Byte	number of scan lines in current video mode 00b = 200 01b = 350 10b = 400 11b = 480
DI+2Bh	1 Byte	primary character block 00b = block 0 01b = block 1 10b = block 2 11b - 11111111b = blocks 3 - 255
DI+2Ch	1 Byte	secondary character block 00b = block 0 01b = block 1 10b = block 2 11b - 11111111b = blocks 3 - 255
DI+2Dh	1 Byte	miscellaneous state information Bit5 = 0 Background intensity; = 1 blinking Bit4 = 1 cursor emulation is active Bit3 = 1 mode set auto palette loading disabled Bit2 = 1 mono Display is atteched Bit1 = 1 gray scale summing is active Bit0 = 1 all modes on all displays active
DI+31h	1 Byte	video memory available 00b = 64k 01b = 128k 10b = 192k 11b = 256k
DI+32h	1 Byte	save pointer state information Bit5 = 1 display combination code (DCC) extension active Bit4 = 1 palette override active Bit3 = 1 graphics font override active Bit2 = 1 text mode font override active Bit1 = 1 dynamic save area active Bit0 = 1 512-character set active

Der erste Eintrag in der Functionality/State Tabelle ist ein Doppelwort-Pointer zu einer 16-Byte ROM-Tabelle. Die Tabelle ist im BIOS an der Adresse E000:305Fh.

Offset	Value	Description
00h	FFh	video modes supported Bit7 = 1 mode 7 supporterd Bit6 = 1 mode 6 supporterd Bit5 = 1 mode 5 supporterd Bit4 = 1 mode 4 supporterd Bit3 = 1 mode 3 supporterd Bit2 = 1 mode 2 supporterd Bit1 = 1 mode 1 supporterd Bit0 = 1 mode 0 supporterd
01h	F0h	video modes supported Bit7 = 1 mode F supporterd Bit6 = 1 mode E supporterd Bit5 = 1 mode D supporterd Bit4 = 1 mode C supporterd Bit3 = 1 mode B supporterd Bit2 = 1 mode A supporterd Bit1 = 1 mode 9 supporterd Bit0 = 1 mode 8 supporterd
02h	0Fh	video modes supported Bit3 = 1 mode 13 supporterd Bit2 = 1 mode 12 supporterd Bit1 = 1 mode 11 supporterd Bit0 = 1 mode 10 supporterd
07h	07h	scan line modes available for text modes Bit2 = 1 400 scan lines supporterd Bit1 = 1 350 scan lines supporterd Bit0 = 1 200 scan lines supporterd
08h	02h	number of character blocks available in text mode
09h		maximum number of active character blocks available in text modes
0Ah	FFh	miscellaneous Bit7 = 1 color paging supporterd Bit6 = 1 color palette supporterd Bit5 = 1 EGA palette supporterd Bit4 = 1 cursor emulation supporterd Bit3 = 1 default palette loading supporterd Bit2 = 1 character font loading supporterd Bit1 = 1 gray scale summing supporterd Bit0 = 1 all modes on all displays supporterd
0Bh	0Eh	miscellaneous Bit3 = 1 display combination codes (DCC) supporterd Bit2 = 1 background intensity/blinking control supporterd Bit1 = 1 save/restore supporterd Bit0 = 0 light pen not supporterd
0Eh	3Fh	save pointer functions Bit5 = 1 DCC extension Bit4 = 1 palette override Bit3 = 1 graphics font override Bit2 = 1 alpha font override Bit1 = 1 dynamic save area Bit0 = 1 512 character set supporterd

10.18.23 Funktion 1CH save/restore video state

Der sicherbare Teil des Video-Status besteht aus drei Teilen:

1. Hardware State: Der Inhalt der CRT-, Attribute- und Controller-Register.
2. Video BIOS Status: Die Bereiche der BIOS Data Area, die sich im System-Memory befinden (40:xx).
3. DAC Status: Der Inhalt der DAC-Kontroll- und -Farb-Register.

Sie könen einzeln oder in Kombinationen gesichert oder restauriert werden. Die Statusinformationen sind in einem RAM-Buffer, auf den die Register ES:BX zeigen. Der Buffer besteht aus einem Bereich von 20 Byte und ein, zwei oder drei Bereichen mit 64 Byte je nach Anforderung.

Offset	Description
00h	word offset of hardware save area,if saved
02h	word offset of BIOS RAM save area, if saved
04h	word offset of DAC save area, if saved
06h - 1Fh	reserved

Unterfunktion 00H return save/restore buffer size needed

Diese Unterfunktion gibt die Buffergröße in 64-Byte Blöcken zurück je nach Anforderung in CX.

Input: CX = Video state to store

- Bit0 = 1 save video hardware state
- Bit1 = 1 save video BIOS data area
- Bit2 = 1 save video DAC state and color registers

Output: AL = 1Ch Function is supported
BX = buffer size block count (1 block = 64 Byte)

Unterfunktion 01H save current video state

Diese Unterfunktion speichert die in CX angegebenen video states in dem RAM auf das ES:BX zeigt.

Dabei werden die Register verändert! Nach dem sichern sofort restaurieren!!

Input: CX = Video state to store

- Bit0 = 1 save video hardware state
- Bit1 = 1 save video BIOS data area
- Bit2 = 1 save video DAC state and color registers

ES:BX = pointer to system RAM buffer

Output: AL = 1Ch Function is supported

Unterfunktion 02H restore current video state

Input: CX = Video state to restore

- Bit0 = 1 save video hardware state
- Bit1 = 1 save video BIOS data area
- Bit2 = 1 save video DAC state and color registers

ES:BX = pointer to system RAM buffer

Output: AL = 1Ch Function is supported

hardware state save area

Offset	I/O Address	Index	Description
00h	03C4h		Sequencer Index
01h	3B4		CRT Controller Index
02h	03CEh		Graphics Controller Index
03h	03C0h		Attribute Controller Index
04h	03CAh		Feature Control

Sequencer Register

Offset	I/O Address	Index	Description
05h	03C5	01h	clocking mode
06h	03C5	02h	map mask
07h	03C5	03h	character map select
08h	03C5	04h	memory mode
09h	03CC	00h	miscellaneous output

CRT Controller Registers

Offset	I/O Address	Index	Description
0Ah	3B5h	01h	horizontal total
0Bh	3B5h	01h	horizontal display enable end
0Ch	3B5h	01h	start horizontal blanking
0Dh	3B5h	01h	end horizontal blanking
0Eh	3B5h	01h	start horizontal retrace pulse
0Fh	3B5h	01h	end horizontal retrace
10h	3B5h	01h	vertical total
11h	3B5h	01h	overflow
12h	3B5h	01h	preset row scan
13h	3B5h	01h	maximum scan line
14h	3B5h	01h	cursor start
15h	3B5h	01h	cursor end
16h	3B5h	01h	start address high
17h	3B5h	01h	start address low
18h	3B5h	01h	cursor location high
19h	3B5h	01h	cursor location low
1Ah	3B5h	01h	vertical retrace start
1Bh	3B5h	01h	vertical retrace end
1Ch	3B5h	01h	vertical display enable end
1Dh	3B5h	01h	offset
1Eh	3B5h	01h	underline location
1Fh	3B5h	01h	start vertical blank
20h	3B5h	01h	end vertical blank
21h	3B5h	01h	CRTC mode control
22h	3B5h	01h	line compare

Attribute controller registers

Offset	I/O Address	Index	Description
23h - 32h	03C1h	00h - 0Fh	Palette registers 00h - 0Fh
33h	03C1h	10h	attribute mode control register
34h	03C1h	11h	overscan color register
35h	03C1h	12h	color plane enable register
36h	03C1h	13h	horizontal PEL panning register

Graphics controller registers

Offset	I/O Address	Index	Description
37h	03CFh	00h	set/reset
38h	03CFh	00h	enable set/reset
39h	03CFh	00h	color compare
3Ah	03CFh	00h	data rotate
3Bh	03CFh	00h	read map select
3Ch	03CFh	00h	graphics mode register
3Dh	03CFh	00h	miscellaneous
3Eh	03CFh	00h	color don't care
3Fh	03CFh	00h	bit mask
40h	40:63h		CRTC base address low
41h	40:64h		CRTC base address high
42h	AFFFFh	00h	plane 0 system latch
43h	AFFFFh	01h	plane 1 system latch
44h	AFFFFh	02h	plane 2 system latch
45h	AFFFFh	03h	plane 3 system latch

BIOS state save area

Location	Size	Description
40:49h	1 Byte	video mode setting
40:4Ah	1 Word	number of columns on screen
40:4Ch	1 Word	current page size
40:4Eh	1 Word	current page address
40:50h	8 Words	cursor position on each page. Two bytes/page. First Byte (low order) of each pair is column, second is row. 0,0 is upper left corner of screen.
40:60h	1 Word	cursor type defined as 6845 video chip compatible starting and ending scan lines. High0order byte holds starting scan line; low-order byte holds ending scan line.
40:62h	1 Byte	current page number.
40:63h	1 Word	6845-compatible I/O port number for current mode. port 03D4h or 03B4h).
40:65h	1 Byte	current mode select register.
40:66h	1 Byte	current palette value.
40:84h	1 Byte	number of rows on screen (24 or 25).
40:85h	1 Word	character height (bytes/character).
40:87h	1 Byte	video control bits Bits6 - 5 = memory on video hardware 00b=64k, 01b=128k, 10b=192k, 11b=256k Bit3 = 0 EGA-compatible mode is active Bit2 = 0 wait for display enable Bit1 = 0 color or ECD monitor is attached Bit1 = 1 monochrome monitor is attached Bit0 = 0 translate cursor video modes 0 - 3 when using ECD monitor in 350 line mode
40:88h	1 Byte	EGA/VGA switch data Bits7 - 4 = feature connector bits 3 - 0, respectively Bits3 - 0 = option switches 3 - 0, respectively
40:89h	1 Word	VGA control bits Bit7 = 200 line Bit4 = 400 lines Bit3 = no palette load Bit2 = mono monitor Bit1 = gray scaling
40:8Ah	1 Byte	index to the display combination code table
40:A8h	1 Word	Pointer to video parameter table and overrides

DAC state save area

Offset	I/O Address	Index	Description
00h	03C7h		read/write mode DAC
01h	03C8h		Pixel address
02h	03C6h		Pixel mask
03h	03C9h	00h	red value color 0
04h	03C9h	00h	green value color 0
05h	03C9h	00h	blue value color 0
06h	03C9h	01h	red value color 1
07h	03C9h	01h	green value color 1
08h	03C9h	01h	blue value color 1
300h	03C9h	FFh	red value color FF
301h	03C9h	FFh	green value color FF
302h	03C9h	FFh	blue value color FF
303h	03C1h	14h	color select register

10.18.24 Funktion 4FH Super VGA Mode

Das folgende ist kein verbindlicher Standard, sonder ein Vorschlag der VESA.

Das VESA-BIOS liefert in AX den Status 004FH zurück, falls die Funktion unterstützt wird. Ein Wert <> 0 deutet auf einen Fehler hin.

Super VGA mode numbers

Alle echten VESA mode numbers sind 15 Bit breit (Bit 0 bis 14). Bit 15 steuert das Löschen (=0) des Videospeichers beim Aufruf. Da Bit 8 immer gesetzt ist, beginnen die VESA mode numbers mit 100H.

Mode	Auflösung	Farben
100h	640x400	256
101h	640x480	256
102h	800x600	16
103h	800x600	256
104h	1024x768	16
105h	1024x768	256
106h	1280x1024	16
107h	1280x1024	256

Unterfunktion 00H Return Super VGA Information

Diese Unterfunktion liefert dem aufrufenden Progamm Informationen über die grundsätzlichen Fähigkeiten der Super-VGA-Hardware. Die Unterfunktion füllt einen Informationsblock von 256 Byte Länge an einer durch den Aufrufer vorgegebenen Speicherstelle.

Input: AX = 4F00H diese Unterfunktion ES:DI = Zeiger auf einen Buffer von 256 Bytes

Output: AX = Status

	VESA_Signatur	db	'VESA'
	VESA_Version	dw	?
Aufbau des Buffers:	OEM_String_Ptr	dd	?
	Capabilities	db	4dup(?)
	Video_Mode_Ptr	dd	?

VESA_Signatur enhält die vier Zeichen 'VESA' als Kennzeichnung eines gültigen Blockes.

VESA_Version besteht aus zwei Bytes. Das höherwertige enthält die Hauptversions-, das niederwertige die Unterversionsnummer.

OEM_String_Ptr enthält einen Far-Pointer, der auf eine mit 0 abgeschlossene Zeichenkette zeigt.

Capabilities ist zur Zeit (Anfang 1990) noch reserviert.

Video_Mode_Ptr zeigt auf eine Liste der unterstützten Modes. Jeder Eintrag ist 16 Bit breit. Die Liste wird mit FFFFH abgeschlossen.

Unterfunktion 01H Return Super VGA Information

Diese Unterfunktion liefert Informationen zu einem ganz bestimmten Modus.

Input: AX = 4F01H diese Unterfunktion CX = Super VGA video node ES:DI = Zeiger auf einen Baffer von maximal 256 Bytes

Output: AX = Status

Der erste Teil des Informationsblock ist immer vorhanden:

Mode_Attributes	Word
Win_A_Attributes	Byte
Win_B_Attributes	Byte
Win_Granularity	Word
Win_Size	Word
Win_A_Segment	Word
Win_B_Segment	Word
Win_Func_Ptr	Dword
Byte_Per_Scanline	Word

Der zeite Teil ist nur vorhanden, wenn Bit 1 von Mode_Attributes gesetzt ist.

X_Resolution	Word
Y_Resolution	Word
X_Char_Size	Byte
Y_Char_Size	Byte
Number_Of_Planes	Byte
Bits_Per_Pixel	Byte
Number_Of_Banks	Byte
Memory_Model	Byte
Bank_Size	Byte

Mode_Attributes beschreibt mit den Bits 0 bis 4 einige wichtige Eigenschaften des Video-Modus:

Bit0 zeigt durch 1 an, daß der angeforderte Modus mit der momentan vorhandenen Hardware eingeschaltet werden kann. Dabei wird auch der angeschlosse Monitor berücksichtigt.

Bit1 ist gleich 1, falls die zusätzlichen Informationen (siehe später) verfügbar sind.

Bit2 zeigt durch 1 an, daß das BIOS solche Funktionen wie TTY-Ausgabe, Bildschirm-Rollen oder Ausgabe eines Pixels in dem angeforderten Modus unterstützt.

Bit3 unterscheidet zwischen Monoschrom- (0) und Farbmodus (1). Mono-Adresse ist 3B4H, Farb-Adresse ist 3D4H.

Bit4 unterscheidet zwischen Text- (0) und Grafikmodus (1).

Bit5 - 15 sind reserviert.

Win_A_Attributes und Win_B_Attributes:

Bit0 gibt durch 1 Auskunft darüber, daß ein Window unterstützt wird.

Bit1 teilt durch 1 mit, daß das Window lesbar ist.

Bit2 teilt durch 1 mit, daß das Window beschreibbar ist.

Bit3 - 7 reserviert.

Win_Granularity bestimmt den kleinsten Abstand in KByte, mit dem das Window über den Bildschirmspeicher verschoben werden kann.

Win_Size gibt die Größe des Windows in KByte an.

Win_A_Segment und Win_B_Segment geben die Segmentadressen der Windows im Adreßraum der CPU wieder.

Win_Func_Ptr enhält einen Far-Pointer, über den Funktion 5 (CPU video memory control) direkt aufrufbar ist.

Bytes_Per_Scanline spezifiziert, aus wie vielen Bytes jede logische Scanline besteht.

X_Resolution und Y_Resolution geben die Auflösung in x- und y-Richtung an. In Grafikmodi handelt es sich um die Anzahl Pixel, in Textmodi um die Anzahl Zeichen.

X_Char_Size und Y_Cgar_Size bestimmen die Größe einer Zeichenzelle.

Number_Of_Planes gibt die Anzahl der Speicherebenen wieder.

Bits_Per_Pixel gibt an, wie viele Bits zur Definition der Farbe eines Pixels notwendif sind.

Memory_Model spezifiziert die grundsätzliche Art der Speicherorganisation dieses Modus. Folgende Möglichkeiten sind definiert:

00	Textmodus
01	CGA-Grafik
02	Hercules-Grafik
03	4 Ebenen planar
04	gepackte Pixel
05	nicht verkettet 4, 256 Farben
06 - FF	noch nicht definiert

Number_Of_Banks gibt die Anzahl der Speicherbänke an, in denen die Scanlines enthalten sind. Der Rest der Division $\frac{Scanlinenumber}{Number\_Of\_Banks}$ ergibt die Bank, in der die Scanline enthalten ist.

Bank_Size spezifiziert die Größe einer Bank in Vielfachen von KByte.

Unterfunktion 02H Set Super VGA Video Mode

Initialisiert einen Super-VGA-Modus.

Input: AX = 4F02H diese Unterfunktion BX = video mode

Output: AX = Status

Unterfunktion 03H Return current video mode

Input: AX = 4F03H diese Unterfunktion

Output: AX = Status BX = momentan eingeschalteter video modus. (Bit 15 gelöscht).

Unterfunktion 04H Save/Restore supe VGA state

Input: AX = 4F04H diese Unterfunktion

- DL = 00 fordert Rückgabe der Puffergröße
  - CX = angeforderte Zustände
  - Bit 0 = Zustand der Hardware
  - Bit 1 = Zustand der BIOS-Daten
  - Bit 2 = Zustand des DAC
  - Bit 3 = Zustand der Super-VGA

  Im Register BX steht dann die Anzahl der 64-Byte Blöcke zur Aufnahme des Zustandes.
- DL = 01 save super VGA video state
  - CX = angeforderte Zustände (vgl. 00)
  - ES:BX = Zeiger auf den Puffer
- DL = 02 restore super VGA video state
  - CX = angeforderte Zustände (vgl. 00)
  - ES:BX = Zeiger auf den Puffer

Output: AX = Status

Unterfunktion 05H CPU video memory window control

Input: BH = 0 select super VGA video memory window BL = window number; 0 = window A, 1 = window B DX = Position des windows im video memory (in Vielfachen der Granularität)

Output: AX = Status

Input: BH = 1 return super VGA video memory window BL = window number; 0 = window A, 1 = window B

Output: AX = Status DX = Position des windows im video memory (in Vielfachen der Granularität)

10.19 Interrupt Nr. 11H Get Peripheral Equipment List

Im AX Register erscheint eine Liste von Bits in der Form: PPMURRRUFFVVUUCI

- PP number of printers installed
- M = 1 if internal modem installed
- RRR number of RS-232 ports installed
- U nicht benutzt (keine Bedeutung)
- FF number of floppy-disk drives minus 1 (0 = ein drive)
- VV initial video mode
  - 00 = reserved
  - 01 = 40X25 color
  - 10 = 80X25 color
  - 11 = 80X25 monochrome
- C = 1 if math coprocessor installed
- I = 1 if IPL diskette installed (Initial Program Load)

Bei Systemen mit 80386 Prozessor wird das EAX Register benutzt.
Bit 24 zeigt den Weitek Coprozessor an. Bit 2 zeigt ein pointing device an.

10.20 Interrupt Nr. 12H Get Usable Memory Size (KB)

Im AX Register wird die Zahl der vorhandenen 1KByte Blöcke zurückgegeben.

10.21 Interrupt Nr. 13H Disk Services

Wenn eine Festplatte installiert ist, wird der INT 13h Vektor von dem BIOS auf den INT 40h verbogen. Dort wird die Anforderung im DL Register auf das höchste Bit (80h) untersucht. Ist es gesetzt, ist die Anforderung für eine Festplatte, andernfalls für eine Floppy. Die Laufwerks-Nummer wird immer auf die letzte Stelle begrenzt, also 0 oder 1 (nicht Phoenix-BIOS).

Das BIOS setzt den DMA-Kontroller mit den benötigten Daten und schreibt dann in den Device-Kontroller den Kommando-Block. Der DMA-Kontroller benutzt den DMA-Kanal 2. Ist der Befehl abgearbeitet, wird ein INT 0Eh (Disketten ISR) ausgeführt. Dieser behandelt den Interrupt und setzt Bit7 in der Adresse 40:3Eh. Der Int 40H, Disketten DSR, monitored diese Adresse für eine Sekunde. Ein Interrupt zeigt an, daß ein Byte oder bis zu 7 Byte von dem Interrupt 40h gelesen werden kann. Erfolgt kein Interrupt in der Zeit, wird vom INT 40h das Time-out-Bit in der Adresse 40:41h gesetzt.

Bei den schnellen Festplatten erzeugt das BIOS einen INT 15h, AH = 90h Device Busy (AL = 00h fixed disk). Wenn der Festplatten interrupt erfolgt, erzeugt das BIOS einen INT 15h, AH = 91h Interrupt complete (AL = 00h fixed disk).

Warum wird das bei den langsamen Floppies nicht gemacht? Lochkarten!!!

Phoenix-BIOS erlaubt mit dem EXDSKBIO.DRV zwei zusätzliche externe Floppy-Laufwerke.

Die folgenden Funktionen haben alle den gleichen input:

AL	function number
AL	number of sectors
CH	track number (0:1023)
CL	sector number (1:17)
DH	head number (0:15)
DL	drive number(0:1 or 80:81)
ES:BX	transfer address

CF = 1 if error.

10.21.1 Funktion 00H Reset Disk System

Mit 00 -07H (also 8 Laufwerke) für floppy disk oder 80-FFH (also 128 Laufwerke) für fixed disk im DL Register (IBM-BIOS), AL Register (Microsoft), wird die drive nummer angegeben. Der error code steht dann im AH Register (Microsoft).

10.21.2 Funktion 01H Get Disk Status

Im Register AL (IBM-BIOS), AH (Microsoft) wird der Statuscode der letzten Operation zurückgegeben und der code nach null gesetzt (warum?).

- 00H no error
- 01H invalid command
- 02H address mark not found
- 03H write attempt on write-protected disk (floppy only)

- 04H sector not found
- 05H reset failed (fixed disk only)
- 06H floppy disk removed (floppy only)
- 07H bad parameter table (fixed disk only)
- 08H DMA overflow (floppy only)
- 09H DMA crossed 64KB boundary
- 0AH bad sector flag (fixed disk only)
- 0BH bad cylinder
- 0CH media type not found
- 0DH bad number of sectors
- 0EH control data address mark found
- 0FH DMA arbitration level out of range
- 10H data error
- 11H ECC data error (fixed disk only)
- 20H controller failed
- 40H seek failed
- 80H time out
- AAH drive not ready (fixed disk only)
- BBH undefined error (fixed disk only)
- CCH write fault (fixed disk only)
- E0H status error (fixed disk only)

10.21.3 Funktion 02H Read Disk Sectors

- AL = number of sectors(not value checked)
- CH = cylinder number(not value checked)
- CL = sector number(not value checked)
- DH = head number(0-1 diskette, 0-15 fixed disk)(not value checked)
- DL = drive number(0-1 diskette, 80H-81H fixed disk)(value checked)
- ES:BX = buffer address

10.21.4 Funktion 03H Write Disk Sectors

- AL = number of sectors(not value checked)
- CH = cylinder number(not value checked)
- CL = sector number(not value checked)
- DH = head number(0-1 diskette, 0-15 fixed disk)(not value checked)
- DL = drive number(0-1 diskette, 80H-81H fixed disk)(value checked)
- ES:BX = buffer address

10.21.5 Funktion 04H Verify Disk Sectors

- AL = number of sectors
- CH = cylinder number
- CL = sector number
- DH = head number
- DL = drive number

10.21.6 Funktion 05H Format Disk Tracks

- AL = number of sectors
- CH = cylinder number
- DH = head number
- DL = drive number
- ES:BX = on return 4-byte address field entries, 1 per sector:
  - byte 0 cylinder number
  - byte 1 head number
  - byte 2 sector number
  - byte 3 sector-size code:
    - 00H 128 bytes per sector
    - 01H 256 bytes per sector
    - 02H 512 bytes per sector (standard)
    - 03H 1024 bytes per sector

10.21.7 Funktion 06H Format bad sectors

Für Winchester vorgesehen. (XT)

10.21.8 Funktion 07H Format drive

Für Winchester vorgesehen. (XT)

10.21.9 Funktion 08H Get Current Drive Parameters

Im DL Register muß die drive nummer angegeben werden, dann kommt zurück:

- AX = 00H
- BH = 00H
- BL = drive type
- CH = low-order 8 bits of 10-bit maximum number of cylinders
- CL = bits 7 and 6 high-order 2 bits of 10-bit maximum number of cylinders; bits 5-0 maximum number of sectors/track
- DH = maximum head number
- DL = number of drives installed
- ES:DI = address of floppy-disk-drive parameter table

Format of drive parameter table:

Offset	Function
00h	1st specify byte: step-rate time (SRT); head unload time
01h	2nd specify byte: head load time; DMA mode
02h	wait time after operation until drive motor turned off
03h	Bytes per sector: 0=128, 1=256, 2=512, 3=1024
04h	number of last sector
05h	length of gap between sectors
06h	data transfer length (DTL). Data length if sector length not specified.
07h	length of gap between sectors
08h	fill value for formatted sectors
09h	head settle time
0Ah	motor start up time

10.21.10 Funktion 09H Initialize Hard-Disk Parameter Table

Im Register DL muß die Nummer der fixed disk angegeben werden (80H=first, 81H=second). Interrupt Nr. 41 zeigt auf die Tabelle für Laufwerk C und 46 für das zweite Laufwerk. Ist die carry flag gesetzt steht im Register AH das status byte.

10.21.11 Funktion 0AH Read Long

Nur für Winchester gültig.
Liest 512-byte sector und 4-byte ECC code. Sonst wie bei Funktion 02H.

10.21.12 Funktion 0BH Write Long

Nur für Winchester gültig.
Schreibt 512-byte sector und 4-byte ECC code. Sonst wie bei Funktion 03H.

10.21.13 Funktion 0CH Seek to Cylinder

Nur für Winchester gültig.
Positioniert den Kopf. Sonst wie bei den Funktionen 02H und 03H.

10.21.14 Funktion 0DH Alternate Disk Reset

Nur für Winchester gültig.
Im DL Register muß die drive nummer angegeben werden.

10.21.15 Funktion 0EH Diagnostic 1: Read test buffer

Für Winchester vorgesehen. (XT)

10.21.16 Funktion 0FH Diagnostic 2: Write test buffer

Für Winchester vorgesehen. (XT)

10.21.17 Funktion 10H Test for Drive Ready

Nur für Winchester gültig.
Im DL Register muß die drive nummer angegeben werden. Im AH Register steht dann der Status.

10.21.18 Funktion 11H Recalibrate Drive

Nur für Winchester gültig.
Im DL Register muß die drive nummer angegeben werden. Im AH Register steht dann der Status.

10.21.19 Funktion 12H Memory Diagnostic

Für Winchester vorgesehen.

10.21.20 Funktion 13H Drive Diagnostic

Für Winchester vorgesehen.

10.21.21 Funktion 14H Controller Diagnostic

Nur für Winchester gültig.
Im AH Register steht dann der Status.

10.21.22 Funktion 15H Get Disk Type

Im DL Register muß die drive nummer angegeben werden.

- AH = drive type code:
  - 00H no drive present
  - 01H cannot sense when floppy disk is changed
  - 02H can sense when floppy disk is changed
  - 03H fixed disk ; dann ist CX:DX = number of sectors

10.21.23 Funktion 16H Check for Change of Floppy Disk Status

Nur für Floppy gültig.
Im DL Register muß die drive nummer angegeben werden.

- AH = 00H no change
- AH = 01H invalid floppy disk parameter
- AH = 06H floppy-disk changed
- AH = 80H drive not ready

10.21.24 Funktion 17H Set Disk Type

Nur für Floppy gültig.

- DL = drive number
- AL = floppy-disk type code
  - 00 = invalid type
  - 01 = 360 KB floppy in 360 KB drive
  - 02 = 360 KB floppy in 1.2 MB drive
  - 03 = 1.2 MB floppy in 1.2 MB drive
  - 04 = 720 KB floppy in 720 KB drive or 720 KB or 1.44 MB drive in 1.44 MB drive

10.21.25 Funktion 18H Set Media Type for Format

Nur für Floppy gültig.
Input Register:

- CH = low order 8 of 10 bits maximum number of tracks
- CL =
  - Bits 7 und 6 high order two bits of maximum track
  - Bits 5 bis 0 maximum sectors per track
- DL = drive number (0 - 1 allowed, value checked)

Output Register:

- ES:DI = Pointer to drive parameters table for this media type, unchanged if (AH) is non-zero.
- AH =
  - 00H, CY = 0, Track and Sectors/Track combination is supported
  - 01H, CY = 1, Funktion is not available
  - 0CH, CY = 1, Track and Sectors/Track combination is not supported

10.22 Interrupt Nr. 14H Serial port Services

Die Adressen der vorhandenen Communication Ports stehen in den Memory-Zellen 400H und folgenden. Die Time-out Konstanten stehen in den Memory-Zellen 47CH und folgenden. Diese Zahl bezieht sich nur auf die Anzahl der äusseren loops des wait_for_status-BIOS-Programms. Sie wird intern mit vier multipliziert. Die Gültigkeit der Anforderung wird über die Länge der Befehlstabelle geprüft. Das Unterprogramm wird über diese Befehlstabelle (lookup table) angesprungen.

10.22.1 Funktion 00H Initialize port Parameters

- AL = serial port parameters (bit settings BBBPPSCC):
  - BBB baud rate:
    - 000 110 baud
    - 001 150 baud
    - 010 300 baud
    - 011 600 baud
    - 100 1200 baud
    - 101 2400 baud
    - 110 4800 baud
    - 111 9600 baud
  - PP parity code:
    - 00 none
    - 01 odd
    - 10 none
    - 11 even
  - S number of stop bits code:
    - 0 one stop bit
    - 1 two stop bits
  - CC character size:
    - 00 unused
    - 01 unused
    - 10 7-bit character size
    - 11 8-bit character size
- DX = serial port number (0 = first port)

10.22.2 Funktion 01H Send One Character

- AL = character to send
- DX = serial port number (0 oder 1, value checked)(BIOS Rev. abhängig)
- AH = error status received (s. funktion 03H)

10.22.3 Funktion 02H Receive One Character

- DX = serial port number
- AL = character received
- AH = error status received (s. funktion 03H)

10.22.4 Funktion 03H Get Port Status

- DX = serial port number
- AX = serial port status:
  - 8000H time out (Bit 7 of AH)
  - 4000H transfer shift register empty (Bit 6 of AH)
  - 2000H transfer holding register empty (Bit 5 of AH)
  - 1000H break detect (Bit 4 of AH)
  - 0800H framing error (Bit 3 of AH)
  - 0400H rarity error (Bit 2 of AH)
  - 0200H overrun error (Bit 1 of AH)
  - 0100H data ready (Bit 0 of AH)
  - 0080H received line signal detect (Bit 7 of AL)
  - 0040H ring indicator (Bit 6 of AL)
  - 0020H data set ready (Bit 5 of AL)
  - 0010H clear to send (Bit 4 of AL)
  - 0008H data receive line signal detect (Bit 3 of AL)
  - 0004H trailing edge ring detector (Bit 2 of AL)
  - 0002H delta data set ready (Bit 1 of AL)
  - 0001H delta clear to send (Bit 0 of AL)

10.23 Interrupt Nr. 15H Miscellaneous System Services

Nicht alle Funktionen sind in jedem BIOS vorhanden.

Die Funktionen 80h, 81h, 82h, 85h, 90h und 91h sind Ösen für Multitasking- Aufsätze, die unbedingt das BIOS benutzen wollen.

10.23.1 Funktion 00H Turn On Cassette Motor

Eigentlich nur für die alten PC's gültig.

10.23.2 Funktion 01H Turn Off Cassette Motor

Eigentlich nur für die alten PC's gültig.

10.23.3 Funktion 02H Read Data from Cassette

Eigentlich nur für die alten PC's gültig.

- CX = number of bytes to read
- ES:BX = buffer address
- CF = 1 error
- AH = error status:

  - 01H CRC error
  - 02H bit signals scrambled
  - 03H no data found
- DX = number of bytes read
- ES:BX = location following last byte read

10.23.4 Funktion 03H Write Data to Cassette

Eigentlich nur für die alten PC's gültig.

- CX = number of bytes to write
- ES:BX = buffer address

on return:

- CX = 00H
- ES:BX = location following last byte written

10.23.5 Funktion 20h PRINT.COM

Unterfunktion 00h Disable critical region flag

Gibt an, daß PRINT nicht die userflag setzen soll, wenn es Dosfunktionen benutzt.

Unterfunktion 01h Set critical region flag

ES:BX ist ein Pointer zu einem Byte, das incrementiert wird, wenn eine Dosfunktion benutzt wird.

10.23.6 Funktion 4FH Scan Code

Jeder Tastendruck erzeugt einen Interrupt 09H. Das BIOS liest daraufhin den scan code ein. Das BIOS führt dann einen Interrupt 15H mit der Funktion 4FH aus. Deshalb kann der scan code der Tastatur der Interruptroutine gegebenenfalls bearbeitet übergeben werden, wenn der Benutzer eine Interrupt 15H Funktion 4FH Routine zur Verfügung stellt. Ob das BIOS diese Funktion vorgesehen hat kann mit Interrupt 15H und Funktion C0H nachgesehen werden.

10.23.7 Funktion 80H Device Open

Diese Funktion kann von multitasking Betriebssystemen benutzt werden.

Das BIOS kommt normal mit AH = 00h und zurückgesetzter Carryflag zurück. Der Benutzer muß vorher die Carryflag zurücksetzen.

Ein multitasking Betriebssystem kann mit dieser Funktion die Kontrolle über ein logisches Device übernehmen und die Verteilung auf die unterschiedlichen Prozesse vornehmen.

- BX = device ID
- CX = process ID

10.23.8 Funktion 81H Device Close

Diese Funktion kann von multitasking Betriebssystemen benutzt werden.

Das BIOS kommt normal mit AH = 00h und zurückgesetzter Carryflag zurück. Der Benutzer muß vorher die Carryflag zurücksetzen.

Ein multitasking Betriebssystem kann mit dieser Funktion die Kontrolle über ein logisches Device für einen Prozess entlassen.

- BX = device ID
- CX = process ID

10.23.9 Funktion 82H Program Termination

Diese Funktion kann von multitasking Betriebssystemen benutzt werden.

Das BIOS kommt normal mit AH = 00h und zurückgesetzter Carryflag zurück. Der Benutzer muß vorher die Carryflag zurücksetzen.

Ein multitasking Betriebssystem kann mit dieser Funktion die Kontrolle über alle logischen Device für einen Prozess entlassen weil der Prozess gleich aufgegeben wird.

BX = program ID

10.23.10 Funktion 83H Event Wait

Der Interrupt 15h hat zwei Wait-Funktionen. Der Unterschied ist, daß der rufende Prozess bei der Funktion 83h fortgeführt wird, bei der Funktion 86h aber die Zeit abwartet.

Die Unterfunktion 00h setzt in einem angegebenen Byte das Bit7 nach Ablauf der Zeit. Die Unterfunktion 01h bricht den Aufruf der Unterfunktion 00h ab.

Diese Funktion benutzt den INT 70h, die Real Time Clock ISR. INT 70h hat zwei Funktionen: den Alarm interrupt service und den periodic interrupt service.

Da üblicherweise die RTC mit 1024 ticks pro Sekunde programmiert ist, können bessere Auflösungen erziehlt werden als mit INT 21h Funktion 2Ch (hundertstel Sekunden) oder INT 08h (18,2 mal pro Sekunde).

Set Interval

Mit dieser Unterfunktion wird der Alarm-Prozess gestartet. Die Kontrolle wird mit gesetzter Carryflag zurückgegeben, wenn ein Prozess bereits gestartet war.

- AL = 00H set interval

  Input:
  - ES:BX pointer to a byte in callers memory that will have the high order bit set (80H) as soon as possible after the interval expires.
  - CX,DX = number of microseconds to elapse before posting (CX = RTC_HIGH, DX = RTC_LOW).

  Output:
  - AH = 83h
  - AL = Bit6 of CMOS RAM 0Bh is set, if successful.
  - AL = 00 function busy
  - CF = 0 no error
  - CF = 1 error, function busy

Cancel Interval

Macht den laufenden Prozess rückgängig.

- AL = 01H cancel

Returns: carry if AL not = 00H or 01H or if function AL=0 already busy.

10.23.11 Funktion 84H Joystick Support

Diese Funktion liest den Joystick Port 201H.

- DX = 0 read the current switches. Returns in AL the switch settings in Bits 7 - 4.
- DX = 1 read the resistive inputs. Returns:
  - AX = A(x) value
  - BX = A(y) value
  - CX = B(x) value
  - DX = B(y) value

10.23.12 Funktion 85H System Request

Wenn die Taste SysRq gedrückt wird, so wird 8500H in das Register AX geschrieben und ein Interrupt 15H ausgeführt. Wird sie losgelassen, wird 8501H in das Register AX geschrieben und ein neuer Interrupt 15H ausgeführt. Ein Benutzerprogramm kann sich in den Interrupt 15H einklinken und die Unterfunktionen 00H (Taste gedrückt) und 01H (Taste losgelassen) in der Reihenfolge der Funktion 85H benutzen.

10.23.13 Funktion 86H Wait

Diese Funktion unterbricht das rufende Programm für die in den Registern CX und DX angegebene Zeit. Ist die Zeit abgelaufen, wird Bit7 in der Adresse 40:A0h gesetzt. Ist die RTC mit 1024 Ticks pro Sekunde programmiert, wird der Wert in den Registern CX und DX auf ein Vielfaches von 976 μs gerundet.

Input:
- CX = high byte of wait interval in microseconds
- DX = low byte of wait interval in microseconds

Output:
- AL = mask written to interrupt controller 2 (if succesful)
- AL = unmodified if function busy
- CF = 0 no error
- CF = 1 function busy, wait not performed

10.23.14 Funktion 87H Move Extended Memory Block

Mit dieser Funktion wird ein Block (max 65k Byte) von oder zu dem Memorybereich oberhalb 1MB übertragen. Dazu wird in den protected mode umgeschaltet. Diese Funktion hat keinen ERROR! Ein überschreiben von Daten einer RAMdisk wird nicht erkannt. Weil die Interrupts abgeschaltet sind können bei größeren Blöcken und entsprechender Baudrate gegebenenfalls Interrupts verloren gehen.

on call:

- CX = length in words
- ES:SI = address of block move descriptor table (GDT)
  - 00-0FH Zero
  - 10-11H Segment length in bytes (2*CX-1 or greater)
  - 12-14H 24-bit source address
  - 15H Access rights byte (93H)
  - 16-17H Zero
  - 18-19H Segment length in bytes (2*CX-1 or greater)
  - 1A-1CH 24-bit destination address
  - 1DH Access rights byte (93H)
  - 1E-1FH Zero
  - 20-2FH Zero
- Note: Diese Datenstruktur setzt eine global descriptor table (GDT) ein, die von der CPU im protected mode benutzt wird. Die Bytes bei den Offsets 00-0FH und 20-2FH werden von dem ROM BIOS ausgefüllt bevor der mode umgeschaltet wird. Die 24-bit Adressen sind lineare Adressen und nicht Segment und Offset. Das niedrigere Byte steht vorn.

Dieser Aufruf ist nicht mit der Compatibility Box des OS/2 zu benutzen.

Um in den real address mode zu kommen, muß ein special value in das CMOS RAM geschrieben und dann ein reset signal erzeugt (INT19) werden. Die power-up sequence erkennt den Wert im CMOS RAM und kehrt zu dem code zurück, der den shutdown angestossen hat. Dabei sind eine Reihe von Randbedingungen zu beachten. Der Inhalt der Register ist verloren. Das Stack-Segment-Register (SS) wird zu Null initialisiert und der Stack-Pointer (SP) ist undefiniert. Wurde ein shutdown erkannt, wird SS=30H und SP=100H gesetzt. Das aber überlappt das Ende der Interrupt-Vektor-Tabelle. Vor dem Restaurieren der Benutzer SS und SP werden noch 6 Byte auf den Stack geschrieben und damit die Vektoren der Interrupts FEH und FFH zerstört.

Soweit die Aussagen in Microsofts Encyclopedia von 1988. In Intels technical reference manual für 303 board liest es sich so:

input:
- CX = number of words to move, maximum count = 8000H (32768) words
- ES:SI GDT pointer

output:
- AH = 00H successful
- AH = 01H RAM parity (parity error cleared on return)
- AH = 02H exeption interrupt occurred
- AH = 03H gate A20 failed
- CF = 0 successful
- CF = 1 error

Global Descriptor Table for Function 87H

Offset	Contents
00H	dummy. initialisiert to 0
08H	Descriptor that points to this GDT as a data segment. Initialisiert to 0
10H	Descriptor that points to the source to be moved.
18H	Descriptor that points to the target for the move.
20H	Descriptor that BIOS uses to make the protected mode code segment used in the move.
28H	Descriptor that BIOS uses to make the protected mode stack segment used in the move.

Tabelle 10.1: GDT for Function 87H

10.23.15 Funktion 88H Obtain Size of Extended Memory

Im Register AX steht nach dem Aufruf die Zahl der Kilobytes die oberhalb von 1MB installiert sind.

10.23.16 Funktion 89H Processor to protected mode

Diese Funktion schaltet den Prozessor in den Protected Mode und gibt die Kontrolle an das Segment ab, auf das die GDT (global descriptor table) zeigt.

- ES:SI zeigt auf eine GDT die zum initialisieren der IDTR, GDTR und Stack Segment Selector benutzt wird.
- BH Offset into the interrupt descriptor table stating where the first eight hardware Interrupts will begin. (Interrupt level 1).
- BL Offset into the interrupt descriptor table stating where the second eight hardware Interrupts will begin. (Interrupt level 2).

Es werden acht descriptors benötigt:

- first is dummy, user initialized to 0
- points to the GDT as a Data Segment
- points to the user defined interrupt descriptor table (IDT)
- points to the user's Data Segment (DS)
- points to the user's Extra Segment (ES)
- points to the user's Stack Segment (SS)
- points to the Code Segment that this function will return to.
- is used by this function to establish a code segment for itself.

Table Entry	Offset from ES:DI	Size	Description
0	00h	8 Bytes	Dummy. Must be initialized to 0
1	08h	8 Bytes	Descriptor referencing this GDT
2	10h	8 Bytes	Descriptor referencing caller-defined IDT
3	18h	8 Bytes	Descriptor referencing caller's data segment
4	20h	8 Bytes	Descriptor referencing caller's extra segment
5	28h	8 Bytes	Descriptor referencing caller's stack segment
6	30h	8 Bytes	Descriptor referencing code segment for return
7	38h	8 Bytes	Descriptor used to map BIOS code segment

Offset	Contents
00H	Segment limit (1 to 64 KB)
02H	Bits 0:23 of physical address for segment.
05H	Access rights byte. Must be 93H
06H	Reseved. Must be 0.
07H	Bits 24:31 of physical address for segment.

Tabelle 10.2: Segment Descriptor for Function 87H

Alle Descriptoren müssen von dem Benutzer initialisiert werden bis auf den BIOS code segment Descriptor. Das zugriffberechtigungs Byte wird mit 09Bh, die Adresse mit E0000h und das Datensegment Limit mit FFFFh initialisiert.

Da kein BIOS mit linearer Adressierung vorhanden sein wird, muß der Benutzer alle I/O Kommandos selbst bearbeiten. Die Interrupt Kontroler müssen natürlich neu initialisiert werden. Es ist kein Weg zurück in den 'real mode' vorhanden. Es muß gebootet werden.

10.23.17 Funktion 90H Wait

Device busy loop

Diese Funktion kann von multitasking Betriebssystemen benutzt werden.

Das BIOS kommt normal mit AH = 00h und zurückgesetzter Carryflag zurück. Der Benutzer muß vorher die Carryflag zurücksetzen.

Ein multitasking Betriebssystem kann mit dieser Funktion die Kontrolle an eine andere Task abgeben wenn diese Funktion von dem BIOS für Festplatte, Diskette, paralelem Printer, serieller Schnittstelle oder Keyboard Service benutzt wurde.

Input:
- AL = Device Type:
  - 00h = fixed disk time-out
  - 01h = diskette time-out
  - 02h = keyboard (no time out)
  - 03h = pointing device
  - 21h = waiting for keyboard input
  - 80h = network (no time out)
  - FCh = fixed disk reset time-out
  - FDh = diskette drive motor start
  - FEh = printer time-out
- CF = 0
- ES:BX = points to a network control block

Output: AH = 00h wait time not satisfied CF =

- 0 = wait time not satisfied (the service must use its own wait routine).
- 1 = minimum wait time satisfied (wait performed).

10.23.18 Funktion 91H Post

Interrupt complete
Diese Funktion kann von multitasking Betriebssystemen benutzt werden.
Das BIOS kommt normal mit AH = 00h und zurückgesetzter Carryflag zurück. Der Benutzer muß vorher die Carryflag zurücksetzen.
In einem multitasking Betriebssystem kann diese Funktion von dem BIOS für Festplatte, Diskette, serieller Schnittstelle oder Keyboard Service benutzt werden. Es wird damit das Ende einer I/O-Operation angezeigt. Es kann dann die Task, die den Service zuletzt benutzt hat, erneut aktiviert werden. Da der Printer-Service nicht mit Interrupt betrieben wird, kann diese Funktion damit nicht benutzt werden.

Input:
- AL = Device Type:
  - 00h = fixed disk time-out
  - 01h = diskette time-out
  - 02h = keyboard
  - 03h = pointing device time-out
  - 80h = network
  - FCh = fixed disk reset time-out
  - FDh = diskette drive motor start
- CF = 0
- ES:BX = points to a network control block

Output: none

10.23.19 Funktion C0H Configuration Parameter Pointer

Mit dieser Funktion kann der BIOS-Revisionlevel festgestellt werden. Man kann damit z.B. sehen ob der Interrupt 09H den Interrupt 1AH aufruft um den scan-code der Tastatur interpretieren zu können bevor er im Tastatustreiber verschwindet. Diese Funktion gibt zurück:

- AH = 00H und CY = 0 wenn im BIOS vorhanden, andernfalls AH=86 und CY=1.
- ES:BX = parameter address pointer zu einem Bereich von z. Z. 5 Worten im BIOS-ROM.
  - first word = 8 = length of following table
  - Byte 1 system model byte
  - Byte 2 system model type byte
  - Byte 3 BIOS revision level
  - Byte 4
    - 80H DMA channel 3 used by BIOS (z. Z. noch nicht)
    - 40H cascaded interrupt level 2
    - 20H real time clock available
    - 10H keyboard scan code hook 1AH
    - 08H reserved
    - 04H extended BIOS data area allocated
    - 02H reserved
    - 01H reserved
  - Byte 5 reserved (feature bytes)
  - Byte 6 reserved
  - Byte 7 reserved
  - Byte 8 reserved

10.23.20 Funktion C1H Get extended BIOS data segment

This function returns in ES the segment address of the BIOS extended data area.

10.23.21 Funktion C2H PS/2 mouse support

10.23.22 Funktion D8H EISA-BIOS-Calls

Bit7 = 1 im Register AL zeigt einen 32-Bit Aufruf an.

Unterfunktion 00/80H Read Slot Configuration Information

Input: CL = slot number (including embedde and virtual devices) 0=motherboard, 1= slot 1, ...

Output: AH =

00 success; carry = 0
80 invalid slot number; carry = 1
82 non-volatile memory corrupt; carry = 1
83 empty slot; carry = 1
86 invalid BIOS routine call; carry = 1
87 invalid system configuration; carry = 1
AL = ID information:

- Bit7 = 1 duplicate ID - validates ID in bits 3:0.
- Bit6 = 0 Product ID is readable.
- Bit5-4 = slot type
  - 00 expansion slot
  - 01 imbedded device
  - 10 virtual reserved
  - 11 reserved
- Bit3-0 duplicate ID number

BH = major revision level of configuration utility
BL = minor revision level of configuration utility
CH = checksum (MSByte) of configuration file
CL = checksum (LSByte) of configuration file
DH = number of device functions
DL = combined function information byte

- Bit5 = slot has one or more port initialisation entries.
- Bit4 = slot has one or more port range entries.
- Bit3 = slot has one or more DMA entries.
- Bit2 = slot has one or more interrupt (IRQ) entries.
- Bit1 = slot has one or more memory entries.
- Bit0 = slot has one or more function type definition

SI = Bytes 3 and 2 of four byte compressed ID.
DI = Bytes 1 and 0 of four byte compressed ID.

Unterfunktion 01/81H Read Configuration Information

Input: CH = Function number to read

CL = slot number

DS = Segment for return data buffer

SI = Offset to return data buffer (ESI for 32-Bit calls)

Output: AH =

00 success; carry = 0

80 invalid slot number; carry = 1

81 invalid function number; carry = 1

82 non-volatile memory corrupt; carry = 1

83 empty slot; carry = 1

86 invalid BIOS routine call; carry = 1

87 invalid system configuration; carry = 1

Die Adresse in DS:(E)SI zeigt auf einen 320 Byte Block mit den Konfigurationsinformationen für eine "Expansion Board Fucntion". Die Feldgrößen in dem Datenbereich sind festgelegt. Der Konfigurationsfile muß sich an diese Datenstruktur halten.

Offset	Bedeutung
0000	four byte compressed ID
0004	ID and slot information
0006	code revision level
0008	selections
0022	function information
0023	TYPE, SUBTYPE ASCII string
0073	memory configuration
00B2	interrupt configuration byte
00C0	DMA channel description
00C8	port I/O information
0104	initialization Data

Offset 0000 4 Bytes compressed ID

- Offset 00
  - Bit7 = 0
  - Bit6 - 2 = ID character 1
  - Bit1 - 0 = höhere Bits des ID character 2
- Offset 01
  - Bit7 - 5 = ID character 2
  - Bit4 - 0 = ID character 3
- Offset 02
  - Bit7 - 4 digit 1 product number
  - Bit3 - 0 digit 2 product number
- Offset 03
  - Bit7 - 4 digit 3 product number
  - Bit3 - 0 single character board revision level

Offset 0004 2 Bytes ID and slot information

- Offset 04
  - Bit7 = 1 duplicate ID is present
  - Bit6 = 1 ID is not readable
  - Bit5 - 4 slot type
    - 00 = expansion slot
    - 01 = imbedded device
    - 10 = virtual reserved
    - 11 = reserved
  - Bit3 - 0 numeric identifier for duplicate CFG filenames (IDs) z. B. 0001 = 1st duplicate (CFG file 1ACE0105), 1111 = 15th (CFG file FACE0105).
- Offset 05
  - Bit7 = 0 configuration complete
  - Bit6 - 2 reserved
  - Bit1 = 0 IOCHKERR is supported by this board
  - Bit0 = 0 ENABLE is supported

Offset 0006 2 Bytes vendor specific code revision level

- Offset 06 minor revision level
- Offset 07 major revision level

Offset 0008 26 Bytes selections

Offset 0022 1 Byte function information

- Bit7 = 1 function is disabled
- Bit6 = CFG extension free-form data
- Bit5 = port initialisation entry(s) follows
- Bit4 = port range entry(s) follows
- Bit3 = DMA entry(s) follows
- Bit2 = interrupt (IRQ) entry(s) follows
- Bit1 = memory entry(s) follows
- Bit0 = Type/subtype ASCII string entry follows

Offset 0023 80 Bytes TYPE and SUBTYPE ASCII string

Offset 0073 63 Bytes memory configuration

- Offset 73 memory configuration byte
  - Bit7 = entries to follow. 0 = last entry; 1 = more entrie follow.
  - Bit5 = 1 shared memory
  - Bit4-3 memory type
    - 00 = system (base or extd)
    - 01 = expanded
    - 10 = virtual
    - 11 = other
  - Bit1 = 1 chacheable
  - Bit0 = 0 ROM; = 1 RAM

- Offset 74 memory data size
  - Bit3-2 decode size
    * 00 = 20
    * 01 = 24
    * 10 = 32
    * 11 = reserved (64 für i860?)
  - Bit1-0 data size
    * 00 = BYTE
    * 01 = WORD
    * 10 = DWORD
    * 11 = reserved (QWORD?)
- Offset 75 (lsb) 76 77 (msb) memory start address divided by 100h.
- Offset 78 (lsb) 79 (msb) memory size - bytes divided by 400h. 0 means 64M.

Offset 00B2 14 Bytes interrupt configuration

- Bit7 = 0 last entry; = 1 more entries follow.
- Bit6 = 1 shareable
- Bit5 = 0 edge triggered; = 1 level triggered.
- Bit3-0 = Interrupt (0 - F).

Offset 00C0 DMA channel description

- Bit7 = 0 last entry; = 1 more entries follow.
- Bit6 = 1 shareable
- Bit2-0 = DMA channel number (0-7).

Offset 00C1 DMA channel description

- Bit5-4 DMA timing
  - 00 = def. ISA comp.
  - 01 = TYPE A
  - 10 = Type B
  - 11 = Type C, Burst
- Bit3-2 Transfer size
  - 00 = 8-bit
  - 01 = 16-bit
  - 10 = 32-bit
  - 11 = reserved

Offset 00C8 60 Bytes port I/O information

- Offset C8
  - Bit7 = 0 last entry; = 1 more entries follow.
  - Bit6 = 1 shareable
  - Bit4-0 number of ports (minus 1)
- Offset C9 (lsb) CA (msb) - I/O port address

Offset 0104 60 Bytes initialisation Data

- Offset 104
  - Bit7 = 0 last entry; = 1 more entries follow.
  - Bit2 = 0 write value to port; = 1 use mask and value.
  - Bit1-0 type of access
    * 00 = 8-bit
    * 01 = 16-bit
    * 10 = 32-bit
    * 11 = reserved
- Offset 105 (lsb) 106 (msb) port I/O address
- Offset 107 - 10E

  If Byte 0. Bit2 = 0 (no mask), then the port value is written to bytes 3-6 according to the settings of bits 1:0, as follows:

	Offset 10A	Offset 109	Offset 108	Offset 107
00				port value
01			MSB of port value	LSB of port value
10	MSB of port value	3rd byte	2nd byte	LSB of port value

  If Byte 0, Bit2 = 1 (use mask), then bits 1:0 = number of bytes/port value/mask:

	Off 10E	Off 10D	Off 10C	Off 10B	Off 10A	Off 109	Off 108	Off 107
00							Mask	Value
01					Mask 6	=MSB	Value 4	=MSB
10	Mask dou	ble word	ByteA=	MSB	Value dou	ble word	Byte6=	MSB

Unterfunktion 02/82H Clear non-volatile Memory

Input: BH = major revision level; BL = minor revision level.

Output: AH =

00 success; carry = 0

84 error; carry = 1

86 invalid BIOS routine call; carry = 1

Unterfunktion 03/83H Write non-volatile Memory

Diese BIOS-Routine schreibt in das nichtflüchtige Memory die Konfigurationsinformationen für jeden Steckplatz und erzeugt eine CRC checksum.

Input: CX = length of data structure (CX = 0 empty slot) includes two bytes for configuration file checksum.

DS = Segment for return data buffer

SI = Offset to return data buffer (ESI for 32-Bit calls)

Output: AH =

00 success; carry = 0

84 error; carry = 1

85 memory full; carry = 1

86 invalid BIOS routine call; carry = 1

10.23.23 Funktion DDH Gate A20 off

Diese Funktion muß ausgeführt werden nachdem vom protected mode aus der real mode erreicht wird. Damit setzt der 8042 slave processor die Leitung A_20 auf zero. Der error code steht im Register AL.

- AL = 00H operation succesfull - 8042 has accepted command.
- AL = 02H failure - 8042 unable to accept command.

10.23.24 Funktion DFH Gate A20 on

Diese Funktion muß vor dem umschalten in den protected mode ausgeführt werden. Sie veranlasst den 8042 slave processor die Leitung A_20 von dem 80286 kontrollieren zu lassen. Der error code steht im Register AL.

- AL = 00H operation succesfull - 8042 has accepted command.
- AL = 02H failure - 8042 unable to accept command.

10.24 Interrupt Nr. 16H Keyboard Services

Ist der Keyboard Buffer leer, wird der INT 15h mit der Funktion AH=90h (Device Busy) und AL=02h (Keyboard) gerufen. Damit hat das Betriebssystem die Möglichkeit eine andere Aufgabe zu erledigen, während es auf Keyboardinput wartet.

Wurde eine Taste betätigt, stellt der INT 09h (Keyboard ISR) das Zeichen in den Keyboard Buffer und erzeugt einen INT 15h mit der Funktion AH=91h (interrupt complete) und AL=02h (Keyboard). Die read keyboard Funktion liest das Zeichen, entfernt es aus dem Buffer und übergibt es im Register AX dem Rufer. Zusätzlich werden die Keyboard LED's überprüft und gegebenenfalls angepasst.

10.24.1 Funktion 00H Read Next Character

- If ASCII Characters:
  - AH = standard PC keyboard scan code
  - AL = ASCII character
- If extended ASCII codes:
  - AH = extended ASCII code
  - AL = 00H

Diese Funktion wartet auf ein Zeichen und entfernt es danach aus dem keyboard buffer.

10.24.2 Funktion 01H Report if Character Ready

- ZF = 1 = character not ready
- AH = s. Funktion 00H
- AL = s. Funktion 00H

Diese Funktion kehrt sofort zurück. Das Zeichen wird nicht aus dem buffer entfernt.

10.24.3 Funktion 02H Get Shift Status

AL = shift status:

- 01H right shift active
- 02H left shift active
- 04H CTRL active
- 08H ALT active
- 10H Scroll Lock active
- 20H Num Lock active
- 40H Caps Lock active
- 80H Insert state active

Mehrere bits können gleichzeitig gesetzt sein.

10.24.4 Funktion 03H Set typematic rate and delay

Unterfunktion 05H Parameter for typematic

Wenn die Parameter gültig sind, schickt die Funktion das Komando Set Rate (F3H) zu dem Keyboard Kontroller.

BH = Delay value (in milliseconds)

- 00 = 250
- 01 = 500
- 02 = 750
- 03 = 1000

BL = Typematic Rate (in characters per second)

00H = 30.0	0BH = 10.9	16H = 4.3
01H = 26.7	0CH = 10.0	17H = 4.0
02H = 24.0	0DH = 9.2	18H = 3.7
03H = 21.8	0EH = 8.5	19H = 3.3
04H = 20.0	0FH = 8.0	1AH = 3.0
05H = 18.5	10H = 7.5	1BH = 2.7
06H = 17.1	11H = 6.7	1CH = 2.5
07H = 16.0	12H = 6.0	1DH = 2.3
08H = 15.0	13H = 5.5	1EH = 2.1
09H = 13.3	14H = 5.0	1FH = 2.0
0AH = 12.0	15H = 4.6	20H - FFH

10.24.5 Funktion 05H Write Keyboard

Place ASCII Character/Scan code combination in Keyboard buffer as if struck from keyboard. The buffer pointer (40:1Ch) is adjusted to point to the next available location in the keyboard buffer.
Input Register:

- CL = ASCII Character
- CH = Scan code

Output Register:

- AL:
  - 00H = successfull operation
  - 01H = unsuccessfull - buffer full

10.24.6 Funktion 10H Extended Read

Extended read interface for the enhanced keyboard.

Output:
- AH = Scan code or character ID for special character
- AL = ASCII code or other translation of character

10.24.7 Funktion 11H Extended ASCII Status

Extended ASCII Status for the enhanced keyboard, otherwise same as Funktion 02H.

Output:
- AH = scan code or character ID for special character (if ZF is set).
- AL = ASCII code or other translation of character (ZF is set).
- ZF = 0 no key in buffer
- ZF = 1 key waiting

10.24.8 Funktion 12H Extended Shift Status

Return the extended shift status in register AX. (KB_FLAG_n in KYBD_BIOS)

AH =
- Bit 7 = 1 Sys Req pressed (40:18H Bit 2)
- Bit 6 = 1 Caps lock active (40:18H Bit 6)
- Bit 5 = 1 Num lock active (40:18H Bit 5)
- Bit 4 = 1 Scroll lock active (40:18H Bit 4)
- Bit 3 = 1 Right Alt active (40:96H Bit 3)
- Bit 2 = 1 Right Ctrl active (40:96H Bit 2)
- Bit 1 = 1 Left Alt active (40:18H Bit 1)
- Bit 0 = 1 Left Ctrl active (40:18H Bit 0)

AL = Copy of keyboard shift flags byte (40:17H)
- Bit 7 = 1 insert active
- Bit 6 = 1 caps lock active
- Bit 5 = 1 num lock active
- Bit 4 = 1 scroll lock active
- Bit 3 = 1 Alt pressed
- Bit 2 = 1 Ctrl pressed
- Bit 1 = 1 left shift pressed
- Bit 0 = 1 right shift pressed

10.25 Interrupt Nr. 17H Printer Services

Die Adressen der vorhandenen Printer stehen in den Memory-Zellen 408H und folgenden. Die Time-out Konstanten stehen in den Memory-Zellen 478H und folgenden. Ist der ausgewählte Printer busy (80H im Status-Register) wird ein Interrupt 15H mit der Funktion 90H und der Unterfunktion FEH erzeugt.

10.25.1 Funktion 00H Send Byte to Printer

- AL = character to be printed
- DX = printer number
- AH = status (s. Funktion 02H)

10.25.2 Funktion 01H Initialize Printer

- DX = printer number
- AH = status (s. Funktion 02H)

10.25.3 Funktion 02H Get Printer Status

- DX = printer number
- AH = status
  - 01H time out
  - 02H unused
  - 04H unused
  - 08H I/O error
  - 10H printer selected
  - 20H out of paper
  - 40H printer acknowledgment
  - 80H printer not busy (bit off = 0 = busy)

Mehrere bits können gleichzeitig gesetzt sein.

10.26 Interrupt Nr. 18H Transfer Control to ROM-BASIC

Dieser Interrupt gibt die Kontrolle an das eingebaute BASIC ab. Der Vektor kann aber auch, wenn das ROM-BASIC fehlt, zu einer Routine zeigen, die z. B. über ein Netzt bootet. Meistens zeigt er auf die message *no boot device available*, die benutzt wird, wenn der INT 19H weder auf der Floppy noch auf der Festplatte einen Bootblock findet.

10.27 Interrupt Nr. 19H Reboot Computer

Dieser Interrupt macht einen Start. Er liest die 512 Byte des Boot-Sektors in das Memory beginnend mit der Adresse 0:7C00H. Danach wird dir Kontrolle an diese Adresse abgegeben. Der Vector steht an der Adresse 0:64H und wird mit F000:E6F2H initialisiert.

Wird kein Boot-Sektor gefunden, wird der Interrupt 18H aufgerufen, dessen Vector z. B. von einer Netzwerkarte initialisiert worden ist. Dann kann über ein angeschlossenes Netz gebootet werden. Nur in einem PC-BIOS zeigt der Vector des Interrupt 18H auf das ROM-BASIC!

Wurde ein Boot-Sektor gefunden und geladen, wird das Programm ab der Adresse 0:7C00H abgearbeitet. Dabei werden manche Tests und Konfigurationen nicht erneut durchlaufen. Dafür sind aber einige Voraussetzungen zu erfüllen. Ist die Sys-Flag (Bit 2) im Status Port (I/O address 64H=keyboard control) gesetzt, so ist ein shutdown unterwegs. Dann ist das CMOS-shutdown-byte (wort 0FH) gültig mit:

- 0 = soft reset or unexpected shutdown
- 1 = shutdown after memory size
- 2 = shutdown after memory test
- 3 = shutdown with memory error
- 4 = shutdown with boot loader request
- 5 = jmp dword request - interrupt chips and 287 are initialized
- 6 = protected mode test3 passed
- 7 = protected mode test3 failed
- 8 = protected mode test1 failed
- 9 = block move shutdown request
- A = jmp dword request - w/o interrupt chips initialized

Schreibt man in das status port shut_cmd (FE) und ist der vorhergehende startup normal gewesen, wird ein RESET ausgeführt da im Wort 0FH des CMOS-RAM eine Null steht.

Dieser Interrupt Vektor wird von VDISK, dem virtuellen Disk Device Driver, überschrieben!

Eine andere Möglichkeit neu zu Starten (Softboot):

Man schreibt in die Adresse 472H das Wort 1234 und macht einen JUMP zum BIOS-Anfang (FFFF:0000). Der letzte Befehl ist dann noch INT 20H.

10.28 Interrupt Nr. 1AH Get/Set Time/Data

10.28.1 Funktion 00H Read Current Clock Count

- AL = midnight signal
- CX = high-order word of tick count
- DX = low-order word of tick count

10.28.2 Funktion 01H Set Current Clock Count

- CX = high-order word of tick count
- DX = low-order word of tick count

10.28.3 Funktion 02H Read Real-Time Clock

- CF = 0 clock running; =1 clock stopped
- CH = hours in BCD
- CL = minutes in BCD
- DH = seconds in BCD

10.28.4 Funktion 03H Set Real-Time Clock

- CH = hours in BCD
- CL = minutes in BCD
- DH = seconds in BCD
- DL =
  - 00H standard time
  - 01H daylight saving time

10.28.5 Funktion 04H Read Date from Real-Time Clock

- CF = 0 clock running; 1 clock stopped
- CH = century in BCD (19 oder 20)
- CL = year in BCD
- DH = month in BCD
- DL = day in BCD

10.28.6 Funktion 05H Set Date in Real-Time Clock

- CH = century in BCD (19 oder 20)
- CL = year in BCD
- DH = month in BCD
- DL = day in BCD

10.28.7 Funktion 06H Set Alarm

- CH = hours in BCD
- CL = minutes in BCD
- DH = seconds in BCD
- CF = status:
  - 0 opeartion succesful
  - 1 alarm already set or clock stopped

10.28.8 Funktion 07H Reset Alarm

Diese Funktion schaltet den Alarm aus.

10.29 Interrupt Nr. 1BH Keyboard Break Address

Dieser Vektor zeigt auf einen Code der abgearbeitet wird, wenn die Tasten CTRL und BREAK gleichzeitig betätigt werden. Der POST lässt den Vektor auf eine IRET-Instruktion zeigen. Der Code, der abgearbeitet wird, muß besonders sorgfältig die 8259 Interrupt Kontroller behandeln, da ein oder mehrere Interrupts anstehen können.

10.30 Interrupt Nr. 1CH Timer Tick

Dieser Vektor zeigt auf einen Code der abgearbeitet wird, wenn die Systemuhr unterbricht. Der POST lässt den Vektor auf eine IRET-Instruktion zeigen. Der Code, der abgearbeitet wird, muß alle Register retten. Die Hardware Interrupts der 8259 sind disabled. Der Code sollte recht kurz sein um nicht mit dem Memory-refresh in Konflikt zu kommen. Er wird jedesmal nach dem Update der Tageszeit aufgerufen.

10.31 Interrupt Nr. 1DH Video Parameters

Dieser Vektor zeigt auf einen Datenbereich mit Parametern, die zum initialisieren des 6845 auf dem Video-Adapter benötigt werden. Es können mehrere Tabellen vorhanden sein. Der Post lässt den Vektor auf eine Tabelle im ROM zeigen.

10.32 Interrupt Nr. 1EH Diskette Parameters

Dieser Vektor zeigt auf einen Datenbereich mit Parametern, die für die Diskettenlaufwerke benötigt werden. Der Post lässt den Vektor auf eine 11-Byte lange Tabelle im ROM an der Adresse F000:EFC7H zeigen.

00H Erstes Data-Byte des Disketten "specify command".

- Bits 7 - 4 Drive type / transfer rate
- Bits 3 - 0 Head unload time. Default is 240 milliseconds (0FH).

01H Zweites Data-Byte des Disketten "specify command".

- Bits 7 - 1 Head load time. Default is 01H (4 milliseconds).
- Bit 0 non-DMA mode flag.

Da die Köpfe mit dem Motorstart geladen werden und der Motor länger braucht wird die Head-load-Zeit eigentlich garnicht benötigt.

Die non-DMA flag ist immer zero, um anzuzeigen, daß DMA benutzt wird.

02H Motor turn-off delay.

03H Bytes per sector.

04H Sectors per track.

05H Gap length.

06H Data length.

07H Gap length for format.

08H Fill Byte for format. The default is F6H.

09H Head settle time.

0AH Motor start time.

10.33 Interrupt Nr. 1FH Graphics Character Extension

Dieser Interrupt wird von dem Befehl GRAFTABL benutzt. Damit wird die Darstellung der Zeichen 80H - FFH (128-255) festgelegt. Es ist ein Pointer zu einer Tabelle.

10.34 Interrupt Nr. 20H Terminate Program

Der Interrupt 20H ist eine der Möglichkeiten ein Programm zu beenden. Er informiert das System, daß das Programm vollständig abgearbeitet ist und daß das bisher benötigte Memory nicht mehr gebraucht wird. Im CS Register muß die Segment Adresse des PSP stehen. CP/M Dinosaurier!

- Als Antwort auf einen Interrupt 20 Aufruf führt MS-DOS folgende Arbeiten aus:
  - Von der Adresse PSP:000AH wird der termination handler vector (Interrupt 22H) restauriert.
  - Von der Adresse PSP:000EH wird der CTRL-C vector (Interrupt 23H) restauriert.
  - Von der Adresse PSP:0012H wird der critical error handler vector (Interrupt 24H) restauriert.
  - Die file buffers werden auf die Platte geschrieben.
  - Es wird bei der Adresse des termination handlers weiter gemacht.

  Der termination handler entlässt alle vom Programm belegten Memoryblöcke einschlieslich dem Umgebungsblock und alle dynamisch belegten, die nicht vorher entlassen worden sind. Alle files die mit handles geöffnet wurden werden geschlossen, wenn sie nicht schon vorher geschlossen wurden. Die Kontrolle wird dann dem Ursprungsprozess übergeben, meistens wird das der COMMAND.COM sein.
- Wenn die Kontrolle zum COMMAND.COM geht, wird zuerst der residente Teil angesprochen, der, wenn nötig, den transienten Teil lädt. Wenn ein batch file abgearbeitet wird, wird die nächste Zeile interpretiert oder, wenn nicht vorhanden, der Benutzer mit dem Prompt aufgefordert das nächste Kommando einzugeben.

- Alle files, die mit FCBs benutzt wurden, müssen vor der Benutzung von Interrupt 20H geschlossen werden, sonst gehen die Daten verloren.
- Der Interrupt 20H ist veraltet. Neue Programme sollten Interrupt 21 Funktion 31H (terminate and stay resident) oder Funktion 4CH (terminate process with return code) benutzen.

10.35 Interrupt Nr. 21H System Calls

Mit dem Interrupt 21H werden fast alle Systemanforderungen (function request oder system calls) erfüllt. Die Funktionsnummer wird im Register AH, eine erforderliche Differenzierung der Funktion wird im Register AL übergeben.

10.35.1 Funktion 00H terminate program

Die Funktion 00H wird auch vom Interrupt 20H gerufen; es gelten die Bemerkungen zu Interrupt 20H.

Vorher muß das CS Register mit der Segment Adresse des PSP (Program Segment Prefix) geladen werden. Von den folgenden PSP offsets werden die Interrupt-Vektoren restauriert:

- 0AH Interrupt 22H
- 0EH Interrupt 23H
- 12H Interrupt 24H

Alle file buffers werden auf die Platte geschrieben und alle handles geschlossen. Die Kontrolle wird dann dem Interrupt 22H übergeben.

Jeder file der mit einem FCB geöffnet und in der Länge geändert wurde muß vorher geschlossen werden, da sonst die Einträge in das Verzeichniss nicht richtig sind. Alle flags, die das Verhalten des Systems beinflussen (Memory allocation strategy, break, verify), müssen zurückgesetzt werden.

Die Funktion 00H sollte durch die Funktion 31H oder 4CH ersetzt werden denn beide brauchen die Adresse des PSP nicht.

10.35.2 Funktion 01H read keyboard and echo

Mit der Funktion 01H wird ein 8-bit Zeichen von dem standard input in das AL Register gelesen. Eigentlich ist der standard input das Keyboard, er kann aber auch zu einem anderen device oder einem file umgeleitet werden. Das Echo wird zum standard output geleitet.

Die Funktion 01H wartet auf ein Zeichen, wenn keins ansteht. Mit der Funktion 0BH kann der status abgefragt werden.

Extended characters müssen mit zwei calls gelesen werden. Das erste Byte ist dann 00H. Es kann sich dabei aber auch um ein echtes Zeichen handeln, wenn z. B. von einem File gelesen wird. Ob der Input umgeleitet wurde kann mit der Funktion 44 (IOCTL) Unterfunktion 00H (Get Device Data) festgestellt werden.

Carriage-return und linefeed werden nicht interpretiert sondern als einzelne Zeichen geechoed.

Ein File-Ende-Zeichen (EOF) wird nicht erkannt wenn von einem File gelesen wird.

Ein Control-C-Zeichen wird nicht erkannt wenn von einem File gelesen wird. Nur wenn Break erlaubt wurde wird bei Control-C der Interrupt 23H ausgeführt.

Es sollte besser die Funktion 3FH (read file or device) benutzt werden.

10.35.3 Funktion 02H display character

Mit der Funktion 02H wird das 8-bit Zeichen aus dem DL Register zum standard output geschrieben. Der standard output ist das aktive display oder ein anderes device oder file. Ein backspace löscht das Zeichen nicht. Wenn ein Control-C erkannt wird nach dem Aussenden des Zeichens wird der Interrupt 23H ausgeführt.

Es sollte besser die Funktion 40H benutzt werden.

10.35.4 Funktion 03H auxiliary input

Die Funktion 03H wartet auf ein 8-bit Zeichen vom auxiliary input (COM1) und liest es in das AL Register. Die Funktion 03H ist nicht interruptgetrieben oder gebuffert, deshalb für schnelle Datenübertragung nicht geeignet.

Fehlererkennung ist nicht möglich. Die Parameter, die die Hardware zulässt, können mit MODE gesetzt werden.

Es kann mit CTRL-C von der console unterbrochen werden.

Es sollte besser die Funktion 3FH (read file or device) benutzt werden.

10.35.5 Funktion 04H auxiliary output

Die Funktion 04H schreibt ein 8-bit Zeichen aus dem DL Register zum standard auxiliary output (COM1). Die Funktion 04H wartet wenn das device busy ist.

Fehlererkennung ist nicht möglich. Die Parameter, die die Hardware zulässt, können mit MODE gesetzt werden.

Es kann mit CTRL-C von der console unterbrochen werden.

Es sollte besser die Funktion 40H (write file or device) benutzt werden.

10.35.6 Funktion 05H print character

Die Funktion 05H schreibt ein 8-bit Zeichen aus dem DL Register zu dem standard printer (PRN). Die Funktion 05H wartet wenn das device busy ist.

Fehlererkennung ist nicht möglich. Die Parameter, die die Hardware zulässt, können mit MODE gesetzt werden.

Es kann mit CTRL-C von der console unterbrochen werden.

Es sollte besser die Funktion 40H (write file or device) benutzt werden.

10.35.7 Funktion 06H direct console i/o

Mit der Funktion 06H wird ein 8-bit Zeichen vom standard input in das AL Register gelesen wenn im Register DL FFH steht oder das Zeichen in DL auf den standard output gegeben wenn es zwischen 00H und FEH ist. Die Funktion 06H kann kein rubout Zeichen senden. Die extended characters (2-Byte) müssen als 2 Byte gelesen werden. Ob der standard input umgeleitet und damit keine extended characters möglich sind, kann man mit der Funktion 44H (IOCTL) Unterfunktion 00H (Get Device Data) feststellen. Wenn ein Control-C Zeichen gelesen wird, wird es nicht interpretiert.

Es sollten besser die Funktionen 3FH (read file or device) oder 40H (write file or device) benutzt werden.

10.35.8 Funktion 07H unfiltered character input

Mit der Funktion 07H wird auf ein 8-bit Zeichen vom standard input (console) gewartet und in das AL Register gelesen. Es wird nicht geechoed und CTRL-C wird nicht interpretiert.
Die extended characters (2-Byte) müssen als 2 Byte gelesen werden. Ob der standard input umgeleitet und damit keine extended characters möglich sind, kann man mit der Funktion 44H (IOCTL) Unterfunktion 00H (Get Device Data) feststellen.
Es sollte besser die Funktion 3FH (read file or device) benutzt werden.

10.35.9 Funktion 08H character input

Mit der Funktion 08H wird auf ein 8-bit Zeichen vom standard input gewartet und in das AL Register gelesen. Es wird nicht geechoed.
Die extended characters (2-Byte) müssen als 2 Byte gelesen werden. Ob der standard input umgeleitet und damit keine extended characters möglich sind, kann man mit der Funktion 44H (IOCTL) Unterfunktion 00H (Get Device Data) feststellen.
Control-C wird nur interpretiert, wenn der input nicht umgeleitet wurde, also nicht von einem file gelesen wird oder BREAK erlaubt wurde.
Es sollte besser die Funktion 3FH (read file or device) benutzt werden.

10.35.10 Funktion 09H display string

Die Funktion 09H sendet eine Zeichenkette zu dem standard output. Die Kette muß mit dem Dollar-Zeichen ($), das nicht angezeigt wird, abgeschlossen sein. Die Anfangs-Adresse der Kette befindet sich in den Registern DS:DX.
Es werden alle ASCII codes bis zum ersten Dollar-Zeichen angezeigt. Die Darstellung der nichtdruckbaren Zeichen hängt von der angeschlossenen Hardware ab.
Carriage return und linefeed werden gegebenenfalls eingefügt (jeweils nach Bildschirmmode).
Es wird ab der augenblicklichen cursor Position angezeigt. Die neue Position ist direkt hinter dem angezeigten string.
Control-C wird erkannt und ausgeführt.
Es sollte besser die Funktion 40H (write file or device) benutzt werden.

10.35.11 Funktion 0AH bufferd keyboard input

Die Funktion 0AH sammelt character vom standard input in einen in den Registern DS:DX angegebenen Memory Buffer. Die Aufgabe ist beendet wenn entweder ein CR (0DH) erkannt wird oder der vorletzte Speicherplatz belegt wurde. Die Zeichen werden auf dem standard output geechoed.
Die MS-DOS editing keys sind aktiv.
Der Buffer, der mit DS:DX angegeben wird, muß folgendes Format haben:

- BYTE0 maximum numbers of character to read (1-255); this value must be set by the process before function 0AH is called.
- BYTE1 count of characters read (does not include the carriage return);this value is set by the function 0AH before returning to the process.
- BYTE2-(n+2) actual string of characters read, including the carriage return; n= number of bytes read.

Es werden also maximal 254 Zeichen gelesen.
Control-C wird ausgeführt wenn der input nicht zu einem file umgeleitet wurde, es sei denn das BREAK erlaubt wurde.
Wenn der input umgeleitet wurde, wird kein EOF erkannt.

10.35.12 Funktion 0BH check keyboard status

Mit der Funktion 0BH erhält man im AL Register einen Wert zurück, der angibt ob character vohanden sind.

- AL = 00H keine Zeichen vorhanden.
- AL = FFH ein oder mehrere Zeichen vorhanden.

Ist der Input umgeleitet worden, wird der standard input auf Vorhandensein von Zeichen untersucht. Sonst der type-ahead buffer.

Die Zahl der Zeichen wird nicht erkannt.

Control-C wird ausgeführt.

Es wird kein Zeichen aus dem buffer entfernt.

10.35.13 Funktion 0CH flush buffer, read keyboard

Die Funktion 0CH cleared den standard input buffer und führt dann die zweite, im AL Register angegebene Funktion (01H, 06H, 07H, 08H, 0AH0, aus. Andere als die angegebenen Funktionen, z.B. AL=00H, führen nur zu einem Löschen des Buffers. Die Funktion 0CH kann benutzt werden um in besonderen Situationen den type-ahead Buffer zu umgehen. Die Bedingungen der angegebenen Funktion bleiben erhalten. Soll die Funktion 06H, direct I/O, benutzt werden, so muß im Register DL eine FFH stehen, da die Funktion 0CH eine Lesefunktion ist. Wird die Funktion 0AH verlangt, muß in den Registern DS:DX Segment:Offset des Receive-Buffers stehen. Bei den Funktionen 01H, 06H, 07H oder 08H steht der 8-Bit ASCII character von dem standard input device in dem Register AL. Control-C wird nur erkannt, wenn der Input nicht redirekted ist. Ist der Input redirekted, wird Control-C nur erkannt, wenn Control-Break erlaubt ist.

10.35.14 Funktion 0DH reset disk

Die Funktion 0DH schreibt alle internen MS-DOS file buffers, die seit dem letzten write geändert wurden, auf die Platte. Alle Buffer werden dann als frei markiert.

Die Funktion 0DH erneuert nicht die Einträge in das Verzeichniss. Dazu muß Die Funktion 10H oder 3EH benutzt werden.

Die Funktion 0DH sollte Bestandteil einer eigenen Control-C Interrupthandlingroutine sein, um das System in einem bekannten Zustand zu hinterlassen, wenn die Anwendungssoftware abgebrochen wird.

10.35.15 Funktion 0EH select disk

Die Funktion 0EH setzt den default disk drive auf den im DL Register angegebenen Wert. Wenn die Funktion erfolgreich zurück kommt steht im AL Register die Anzahl der im System vorhandenen logischen drives. Bei MS-DOS Ver. 3.0 und höher (wie lange??) wird ein Minimum von 5 (warum denn auch die richtige Zahl?) zurückgegeben.

Der Wert für die drive Nummer ist der ASCII-Wert für den Buchstaben des drive abzüglich des Werts für den Buchstaben A, so daß 0 = drive A, 1 = drive B, u.s.w. wird.

10.35.16 Funktion 0FH open file with FCB

Die Funktion 0FH öffnet den file, der im file control block (FCB) benannt wird, auf den die Register DS:DX zeigen.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

Es können nur vorhandene Files in diesem Verzeichnis benutzt werden.

Wird der default drive später geändert, so wird die drive number im FCB erneuert.

Die Recordlänge kann nur vor dem ersten I/O von den anfänglichen 128 Byte geändet werden.

Es sollte lieber die Funktion 3DH (Open File with Handle) verwendet werden.

10.35.17 Funktion 10H close file with FCB

Die Funktion 0FH schliesst den file, der im file control block (FCB) benannt wird, auf den die Register DS:DX zeigen.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

Es sollte lieber die Funktion 3EH (Close File) verwendet werden.

10.35.18 Funktion 11H search for first entry

Die Funktion 11H sucht in dem augenblicklichen Verzeichnis nach dem ersten file das der Angabe in den Registern DS:DX entspricht.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.19 Funktion 12H search for next entry

Die Funktion 12H sucht in dem augenblicklichen Verzeichnis nach dem nächsten file das der Angabe in den Registern DS:DX entspricht. Diese Funktion geht davon aus, daß zuerst die Funktion 11H erfolgreich war.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.20 Funktion 13H delete file

Die Funktion 13H löscht alle Einträge die dem in den Registern DS:DX angegebenen FCB in dem augenblicklichen Verzeichnis entsprechen.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.21 Funktion 14H sequential read

Die Funktion 14H liest den nächst folgenden Daten-Block des files und schreibt die Daten in die augenblickliche disk transfer area (DTA).

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.22 Funktion 15H sequential write

Die Funktion 15H schreibt den nächst folgenden Daten-Block von der augenblicklichen disk transfer area (DTA) in das in den Registern DS:DX angegebene file.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.23 Funktion 16H create file with FCB

Die Funktion 16H legt einen Verzeichniseintrag in dem augenblicklichen Verzeichnis für den in den Registern DS:DX angegebenen file an und öffnet ihn. Besteht der Eintrag schon, so wird der file geöffnet und die Länge mit null angegeben.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.24 Funktion 17H rename file

Die Funktion 17H benennt einen oder mehrere files in dem augenblicklichen Verzeichnis um. In den Registern DS:DX steht die Adresse des modifizierten FCB der folgenden nichtstandard Format hat:

- 00H drive number
- 01-08H old filename (padded with blanks)
- 09-0BH old file extension (padded with blanks)
- 0CH-10H zeroed out
- 11H-18H new filename (padded with blanks)
- 19H-1BH new file extension (padded with blanks)
- 1CH-24H zeroed out

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.25 Funktion 18h nicht benutzt

10.35.26 Funktion 19H get current disk

Die Funktion 19H gibt den code des augenblicklichen drives im Register AL zurück. Dabei entspricht 0 dem drive A u. s. w.. **Achtung:** In anderen Aufrufen entspricht 0 dem default drive!

10.35.27 Funktion 1AH set disk transfer address DTA

Die Funktion 1AH legt den Platz fest der für FCB disk I/O benutzt wird. Die Adresse der DTA wird in den Registern DS:DX angegeben.

Wird keine Adresse angegeben, so benutzt MS-DOS die Adresse beim Offset 80H des PSP.

Der angegebene Platz muß groß genug sein für einen einfachen block transfer. Der vorgegebene Wert für Arbeiten mit FCB ist 128. Dieser Wert kann nach erfolgreichem Öffnen oder Erstellen eines Files im record size field des FCB geändert werden. Wenn der Wert zu klein ist kann anderer code überschrieben werden.

Die angegebene Adresse muß weit genug von der Segmentgrenze entfernt sein weil bei deren überschreiten der transfer abgebrochen wird.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.28 Funktion 1BH get default drive data

Die Funktion 1BH gibt Informationen über die disk in dem default drive.

- AL = number of sectors per cluster
- CX = number of bytes per sector
- DX = number of clusters
- DS:BX segment:offset of the file allocation table (FAT) identification byte.

FAT ID Byte:

FFH double-sided, 8 sectors/track, 40 tracks/side

FEH single-sided, 8 sectors/track, 40 tracks/side

FDH double-sided, 9 sectors/track, 40 tracks/side

FCH single-sided, 9 sectors/track, 40 tracks/side

F9H double-sided, 15 sectors/track, 40 tracks/side or double-sided, 9 sectors/track, 80 tracks/side

F8H fixed disk

F0H others

Der Pointer in DS:BX zeigt nur auf das FAT ID Byte. Mehr Informationen können mit Interrupt 25H, Interrupt 21H Funktion 36H oder Funktion 44H Unterfunktion 0DH minor code 60H erhalten werden.

10.35.29 Funktion 1CH get drive data

Die Funktion 1CH gibt Informationen über die disk in dem im DL Register angegebenen drive, wobei 0 = default drive, 1 = drive A u. s. w. ist.

- AL = number of sectors per cluster
- CX = number of bytes per sector
- DX = number of clusters
- DS:DX segment:offset of the file allocation table (FAT) identification byte.

FAT ID Byte Bedeutung s. Funktion 1BH.

Der Pointer in DS:BX zeigt nur auf das FAT ID Byte. Mehr Informationen können mit Interrupt 25H, Interrupt 21H Funktion 36H oder Funktion 44H Unterfunktion 0DH minor code 60H erhalten werden.

10.35.30 Funktion 1Dh nicht benutzt

10.35.31 Funktion 1Eh nicht benutzt

10.35.32 Funktion 1FH get default drive parameter block

In DS:BX steht ein Zeiger auf den DPB. Ist die Funktion 32h des Interrupt 21h.

10.35.33 Funktion 20h nicht benutzt

10.35.34 Funktion 21H random read

Die Funktion 21H liest aus dem in den Registern DS:DX angegebenen file den im relative-record field des FCB angezeigten record in das Memory.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.35 Funktion 22H random write

Die Funktion 22H schreibt aus der DTA in den angegeben record des files.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.36 Funktion 23H get file size

Die Funktion 23H sucht in dem augenblicklichen Verzeichniss den angegebenen file und gibt die Größe des files in records zurück.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.37 Funktion 24H set relative record

Die Funktion 24H setzt das relative-record Feld eines FCB auf die Position, die von den current-block und current-record Feldern angegeben werden.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.38 Funktion 25H set interrupt vector

Die Funktion 25H schreibt die Adresse die sich in den Registern DS:DX befindet in die Interrupt-Vektor-Tabelle an den Platz der sich aus der in dem Register AL angegebenen Interrupt ergibt.

Der Interrupt Vektor sollte niemals direkt gesetzt werden.

Die vorhergehende Adresse kann mit der Funktion 35H erhalten werden, und sollte für eine spätere Erneuerung abgespeichert werden.

10.35.39 Funktion 26H create new PSP

Die Funktion 26H eröffnet ein neues program segment prefix (PSP) an der im DX Register angegebenen Segment Adresse und kopiert den augenblicklichen PSP dahin. Die Information im Offset 06H des neuen PSP über das zur Verfügung stehende Memory wird angepasst. Die augenblicklichen Interrupt Vektoren für 22H, 23H und 24H werden ab Offset 0AH abgespeichert.

Ein .COM file kann direkt dahinter geladen und gestartet werden. Ein .EXE file kann nicht geladen oder gestartet werden. Mit der Funktion 4BH können .COM, .EXE und overlays geladen werden.

Offset	Size	Description
00	2 Bytes	INT20 instruction
02	word	memory size in paragraphs
04	Byte	nicht benutzt
05	5 Bytes	CP/M entry point (FAR jump to 000C0h)
06	word	size of first segment for .COM files
0A	dword	terminate address (old INT22)
0E	dword	break address (old INT23)
12	dword	critical error handler (old INT24)
16	word	parent PSP segment
18	20 Bytes	DOS2+ open file table, FFh=unused
2C	word	DOS2+ environment segment
2E	dword	DOS2+ process's SS:SP on entry to last INT21 call
32	word	DOS3+max open files
34	dword	DOS3+open file table address
38	dword	DOS3+ pointer to previous PSP
3C	20 bytes	unused by DOS versions $\leq$ 4.01
50	3 bytes	DOS function dispatcher
53	9 bytes	unused
5C	16 bytes	FCB#1, first commandline argument
6C	20 bytes	FCB#2, second commandline argument
80	128 bytes	command tail, default DTA buffer

BUG: the jump address (Offset 5) is 2 bytes too low for DOS2+ on PSP's created by the EXEC function.

10.35.40 Funktion 27H random block read

Die Funktion 27H liest die Anzahl der im Register CX angegebenen records in die disk transfer area (DTA). Segment:Offset des geöffneten FCB stehen in den Registern DS:DX.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.41 Funktion 28H random block write

Die Funktion 28H schreibt die im Register CX angegebene Anzahl records von der DTA in den file.

CP/M Dinosaurier sollten nicht mehr verwendet werden!

10.35.42 Funktion 29H parse file name

Die Funktion 29H untersucht einen String auf einen erlaubten Filenamen in der Form drive:filename.ext. Wird er gefunden, so wird ein nichtgeöffneter FCB für den file erzeugt.

Aufruf:

- AL = code to control parsing, as follows (bits 0-3 only):

Bit0 = 0 stop parsing if file separator is found.

Bit0 = 1 ignore leading separators(parse off white space).

Bit1 = 0 set drive number field in FCB to 0(current drive) if string does not include a drive identifier.

Bit1 = 1 set drive as specified in the string; leave unaltered if string does not include a drive identifier.

Bit2 = 0 set filename field in the FCB to blanks if string does not include a filename.

Bit2 = 1 leave filename field unaltered if string does not include a filename.

Bit3 = 0 set extension field in FCB to blanks if string does not include a filename extension.

Bit3 = 1 leave extension field unaltered if string does not include a filename extension.

- DS:DI segment:offset of string to parse.
- ES:DI segment:offset of buffer for unopened FCB

Ergebnis:

- AL =
  - 00H string does not contain wildcard characters.
  - 01H string contains wildcard characters.
  - FFH drive specifier invalid.
- DS:SI segment:offset of first byte following the parsed string.
- ES:DI segment:offset of unopened FCB.

Die Bits 4-7 müssen 0 sein.
Die Funktion 29H kann keine pathnames analysieren.

10.35.43 Funktion 2AH get date

Die Funktion 2A gibt das System Datum in binärer Form zurück.

- AL = day of week (0= Sonntag, 1= Montag, ...)
- CX = year (1980—2099)
- DH = month (1—12)
- DL = day (1—31)

Jahre ausserhalb des angegebenen Bereichs können mit der Funktion 2A nicht erhalten werden.

10.35.44 Funktion 2BH set date

Die Funktion 2B schreibt die binären Werte in die system date counter als Anzahl der Tage seit dem 1. Januar 1980.

- CX = year (1980—2099)
- DH = month (1—12)
- DL = day (1—31)

Das Jahr muß einen 16-bit Wert zwischen 1980 und 2099 haben. Die Angabe nur der letzten beiden Ziffern erzeugt einen Fehler (FFH im Register AL).

10.35.45 Funktion 2CH get time

Die Funktion 2CH gibt die System Zeit in binärer Form (24 Stunden) zurück.

- CH = hours (0—24)
- CL = minutes (0—59)
- DH = seconds (0—59)
- DL = hundredths of seconds (0—99)

10.35.46 Funktion 2DH set time

Die Funktion 2D setzt die System Zeit in binärer Form (24 Stunden).

- CH = hours (0—24)
- CL = minutes (0—59)
- DH = seconds (0—59)
- DL = hundredths of seconds (0—99)

10.35.47 Funktion 2EH set/reset verify flag

Die Funktion 2E setzt die MS-DOS interne verify flag an oder aus um MS-DOS zu veranlassen Schreiboperationen auf die Platte zu bestätigen.

- AL = 00H turn verify off
- AL = 01H turn verify on

Weil Plattenfehler selten sind ist der Standard für verify off. Er kann von dem Benutzer mit dem Befehl VERIFY geändert werden. Mit der Funktion 54H kann der Zustand der Flag erhalten werden.

10.35.48 Funktion 2FH get disk transfer address

Die Funktion 2F gibt in den Registern ES:BX die Adresse der DTA zurück. MS-DOS hat keine Möglichkeit festzustellen wie groß der Buffer ist, deshalb muß das Programm sicherstellen, daß alle Records hinein passen. Wenn die DTA nicht mit der Funktion 1AH gesetzt wurde, so benutzt MS-DOS einen Buffer von 128 Byte dessen Adresse bei Offset 80H des PSP steht.

10.35.49 Funktion 30H get MS-DOS version number

Die Funktion 30H gibt die major und minor Versions Nummer des Betriebssystems zurück.

- AL = major version number
- AH = minor version number
- BH = OEM's serial number (00H = PC-DOS, FFH oder anders für MS-DOS)
- BL:CX = 24-bit OEM serial number

Ist die Versions Nummer unterhalb einer gewünschten ist die Funktionsfähigkeit aller verwendeten Systemaufrufe nicht gegeben.

10.35.50 Funktion 31H Terminate and Stay Resident (TSR)

Die Funktion 31H beendet ein Programm und gibt die Kontrolle an das vorhergehende Programm zurück ohne das erstere aus dem Memory zu entfernen. Im DX Register muß die Zahl der Paragraphen (16Byte, minimum 6 Paragraphen = 60H Byte) für das beendete Programm stehen.

Der Platz für den Umgebungsblock kann entlassen werden wenn man das Register ES mit dem Wert des Segments im Offset 2CH des PSP lädt und die Funktion 49H benutzt.

Der Prozess, der die Funktion 31H benutzt, sollte einen Fehlercode (00H = kein Fehler) im Register AL übergeben. Der Elternprozess kann dann mit der Funktion 4DH den code benutzen. Wenn die Kontrolle an den COMMAND.COM geht, so kann der code mit ERRORLEVEL in einem batchfile benutzt werden.

Gegenüber dem Interrupt 27H kann mit der Funktion 31H ein Programm größer als 64KByte resident gemacht werden.

Geöffnete Dateien können von DOS nicht geschlossen werden. Das Kann zu mysteriösen Systemabstürzen führen.

10.35.51 Funktion 32H Get Disk Parameter Block (DPB)

Input: DL = Laufwerksnummer (0=standard, 1 = A ...)

Output: DS:BX = far pointer zu DPB.

00 Logische Gerätebezeichnung des Laufwerks (0=A, 1=B, *ldots*).

01 Physikalische Gerätebezeichnung.

02 Anzahl der Bytes pro Sektor.

04 Interleave-Faktor.

05 Sektoren pro Cluster minus 1.

06 Reservierte Sektoren für den Bootsektor.

08 Anzahl der Dateizuordnungstabellen (FAT).

09 Maximale Anzahl der Hauptverzeichniseinträge.

0B Erster Datensektor.

0D Nummer des letzten belegten (belegbaren) Clusters.

0F Anzahl der Sektoren für eine FAT (bis Version 3.3 nur 1 Byte).

11 Erster Sektor im Stammverzeichnis.

13 Zeiger auf den zugehörigen Gerätretreiber.

17 Media Descriptor.

18 Bei Inhalt FF wurde das Gerät noch nicht angesprochen.

19 Zeiger auf den nächsten DPB. Am Ende der Reihe steht hier FFFF.

1D Startcluster für die Suche nach freien Clustern.

1F Anzahl freier Cluster.

Das Media descriptor Byte wird in Tabelle 10.3 auf Seite 298 gezeigt.

Wert	Beschreibung	Seiten	Spuren	Sekt/Spur	Kapazität
F0	3 1/2	2	80	18	1,44 MByte
F8	Festplatte				
F9	5 1/4	2	80	15	1,2 MByte
	3 1/2	2	80	9	720 KByte
FC	5 1/4	1	40	9	180 KByte
FD	5 1/4	2	40	9	360 KByte
FE	5 1/4	1	40	8	160 KByte
FF	5 1/4	2	40	8	320 KByte

Tabelle 10.3: Media descriptor Byte

10.35.52 Funktion 33H Control-C check

Die Funktion 33H zeigt den Status der CTRL-C flag oder setzt ihn. Die Funktion 33H wurde um die Unterfunktion 5 erweitert.

- AL = 00H get current CTRL-C check flag
- AL = 01H set CTRL-C check flag to status in DL
- AL = 02h get and set extended Control-Break checking state. DL = 00 for OFF and 01 for ON.
- AL = 05H get boot device
- AL = 06H get DOS-version

If AL = 01H:

- DL = 00H set CTRL-C check flag to off
- DL = 01H set CTRL-C check flag to on

Die Funktion 33H gibt zurück:

- AL = 00H flag set successfully
- AL = FFH code in AL on call not 00H or 01H

If AL was 00H or 05H on call:

- DL = 00H CTRL-C check flag off
- DL = 01H CTRL-C check flag on
- DX = Laufwerksnummer (1 = A)

Da die control-c check flag für alle Programme gültig ist, muß das Programm den vorhergehenden Zustand sich merken und anschliessend restaurieren.

10.35.53 Funktion 34H InDOS Flag

Die Funktion 34H gibt die Adresse der InDOS flag (critical section flag) in den Registern ES:BX zurück. Microsoft gibt keine Garantie für diese Funktion in zukünftigen Versionen von MS-DOS. Die InDOS flag ist ein Byte innerhalb des MS-DOS Kerns. Der Wert wird bei jeder Benutzung von Interrupt 21H incremented und nach Ende einer Interrupt 21H Funktion decremented. Wird ein critical error, Interrupt 24H ausgeführt, so wird die InDOS flag auf null gesetzt.

DOS-Calls sollten nur benutzt werden, wenn die critical section flag **und** die critical error flag zero sind. Für DOS2.x ist das nächste Byte die critical error flag. Für DOS3.x ist es das vorhergehende, bis auf COMPAQ DOS3.0, dort ist sie 1AAh vor der critical section flag).

10.35.54 Funktion 35H get interrupt vector

Die Funktion 35H gibt in den Registern ES:BX die Adresse für den in dem Register AL angegebenen Interrupt zurück.

Programme sollten niemals die Vektoren direkt bearbeiten.

10.35.55 Funktion 36H get disk free space

Die Funktion 36H gibt Informationen über den im Register DL angegebenen drive zurück (0 = default, 1 = drive A, u.s.w.).

- AX = number of sectors per cluster
- BX = number of clusters available
- CX = number of bytes per sector
- DX = number of clusters on drive

Zuerst sollte das Register AX auf den Fehlercode FFFFH untersucht werden.

Die Zahl der freien Bytes ergibt sich aus $BX \times AX \times CX$.

Die Funktion 36H erkennt keine "verlorenen" cluster, d.h. sie werden als belegt gezählt.

10.35.56 Funktion 37h

Unterfunktion 00h Get Switchar

Return the character which is used to introduce command switches. Wenn danach im Register AL eine FFh steht, ist diese Funktion nicht vorhanden. Sonst steht im Register DL der augenblickliche Switchcharacter.

Unterfunktion 01h Set Switchchar

Im Register DL wird ein Zeichen übergeben mit dem die Zeichen für die Pfad-Trennung (\) und Parameter-Kennung (/) in (/) bzw. (-) geändert wird. MS-DOS ↔ UNIX. Ab DOS 4.0 nicht mehr möglich.

Unterfunktion 02h Get DEV availability

Get the state of the flag which makes a \DEV\ prefix to character device names mandantory. Returns in register DL the device avalability (always FFh under DOS4.0).

Unterfunktion 03h Set DEV availability

Set the state of the flag which makes a \DEV\ prefix to character device names mandantory. Call with:

DL = 00 means \DEV\ must preceed device names.

DL = 00 means \DEV\ need not preceed device names.

Die zweite Zeile soll sicherlich 01 sein. Wird wohl ein Druckfehler sein!

10.35.57 Funktion 38H get/set country date

Die Funktion 38H hat zwei Unterfunktionen die die landesabhängigen Daten setzen oder lesen. Steht im Register DX FFFFH wird der angegebene country code gesetzt.

- AL = 00H current country
- AL = 01 -FEH country code between 1 and 254
- AL = FFH country code of 255 or greater, specified in BX
- BX = country code if AL=FFH
- DS:DX segment:offset of 34-byte buffer

Der country code entspricht der internationalen Telefonvorwahlnummer.

Der Buffer auf den DS:DX zeigt (lesen/schreiben):

- 00H word date format:
  - 0 = United states (m/d/y)
  - 1 = Europe (d/m/y)
  - 2 = Japan (y/m/d)
- 02H ASCIIZ currency symbol (5 bytes)
- 07H ASCIIZ character used as thousands separator
- 09H ASCIIZ character used as decimal separator
- 0BH ASCIIZ character used as date separator
- 0DH ASCIIZ character used as time separator
- 0FH Byte position of currency symbol:
  - 00H currency symbol preceds value with no space
  - 01H currency symbol follows value with no space
  - 02H currency symbol preceds value with one space
  - 03H currency symbol follows value with one space
- 10H Byte number of decimal places in currency
- 11H Byte time format (00H=12-hour clock,01H=24-hour clock)
- 12H Dword case-mapping call address
- 16H ASCIIZ character used as separator in data lists
- 18H 10 bytes reserved

The case-mapping call address is the segment:offset of a FAR procedure that performs country-specific, lowercase-to-uppercase mapping on ASCII characters in the range 80H through 0FFH. The character to be mapped must be placed in the AL register before the call is made. If the (lower-case) character has an uppercase value, that value is returned in AL. If the character has no such value, AL is unchanged.

Der country code wird von dem keyboard driver (KEYB) abgeleitet. Ist keiner angegeben, so wird der vom OEM eingebaute benutzt.

10.35.58 Funktion 39H create directory

Die Funktion 39H eröffnet ein neues Verzeichniss mit dem in den Registern DS:DX angegebenen ASCIIZ-path. Im Register AX erscheint der error code.

MS-DOS erzeugt auch die current (·) und parent (··) Verzeichniseinträge.

10.35.59 Funktion 3AH remove directory

Die Funktion 3AH löscht ein Verzeichniss mit dem in den Registern DS:DX angegebenen ASCIIZ-path. Im Register AX erscheint der error code.

10.35.60 Funktion 3BH change current directory

Die Funktion 3BH setzt den Zeiger auf den in den Registern DS:DX angegebenen ASCIIZ-path. Der String ist auf 64 Zeichen begrenzt. Es ist die einzige Möglichkeit die augenblickliche directory unter MS-DOS zu ändern.

10.35.61 Funktion 3CH create file with handle

Die Funktion 3CH eröffnet einen file mit dem in den Registern DS:DX angegebenen ASCIIZ-path, teilt die im Register CX angegebenen Attribute zu und gibt im Register AX die handle Nummer zurück, wenn die cary flag clear ist. Sonst steht in AX der Errorcode. Ist der File vorhanden, so wird er eröffnet und die Länge nach null gesetzt. Damit sind vorhandene Informationen nicht mehr erreichbar! Um das zu vermeiden, können die Funktionen 3DH, 4EH, 5AH oder 5BH benutzt werden.

Attribute Bits im CX Register:

- Bit0 = read-only
- Bit1 = hidden file
- Bit2 = system file
- Bit3 – Bit5 sind mit Funktion 3CH nicht erlaubt, müssen 0 sein.
- Bit6 – Bit15 müssen 0 sein, für spätere Erweiterung.

Mögliche Fehlermeldungen im AX Register:

- 03H path not found
- 04H too many open files; ein einzelner Process kann nicht mehr als 20 Files zur gleichen Zeit geöffnet halten, 5 davon werden normalerweise von den standard device belegt.
- 05H acces denied; wenn der File in der Rootdirectory eröffnet werden soll und sie schon voll ist oder wenn ein File mit gleichen Namen und read-only Attribut vorhanden ist.

10.35.62 Funktion 3DH open handle

Die Funktion 3DH öffnet den in den Registern DS:DX mit ASCIIZ angegebenen file und liefert ein 16-bit handle in dem AX Register für die nächsten Zugriffe. Im AL Register stehen die folgenden Zugriffsrechte:

- AL-bit 7
  - 0 child process inherits file
  - 1 child process does not inherit file
- AL-bits 4-6
  - 000 compatibility mode
  - 001 deny read/write access
  - 010 deny write access
  - 011 deny read access
  - 100 deny none
- AL-bit 3 reserved
- AL-bits 0-2
  - 000 read-only access
  - 001 write-only access
  - 010 read/write access

Die Funktion 3DH eröffnet jeden File, auch hidden und system.

Mögliche Fehlermeldungen im AX Register:

- 02H file not found
- 03H path not found
- 04H too many open files; ein einzelner Process kann nicht mehr als 20 Files zur gleichen Zeit geöffnet halten, 5 davon werden normalerweise von den standard device belegt.
- 05H acces denied; wenn der in DS:DX angegebene path eine directory oder ein volume label ist oder ein read/write Zugriff auf einen read-only File erfolgen soll.
- 0CH invalid access code; wenn in den Bits 0 - 2 im Register AL etwas anderes als 000, 001 oder 010 steht.

Wenn der augenblickliche Prozess einen neuen Prozess mit der Funktion 4BH erzeugt, so kann mit Bit7 angegeben werden, ob der File vererbt werden soll oder nicht. Wird er vererbt, so werden auch die Zugriffs- und Teilhaberechte vererbt (access and sharing).

Die Bits 4 - 6 geben die Zugriffsrechte anderer Benutzer auf dem Netzwerk für diesen File an. Es sind zur Zeit folgende definiert:

000 Compatibility. Erlaubt anderen Prozessen auf der gleichen Maschine unbegrenzten Zugriff auf den File. Prozesse auf anderen Maschinen können über das Netz nur mit read-only zugreifen. Wenn ein File schon mit einem anderen "sharing mode" geöffnet wurde, kann er nicht mehr im compatibility mode geöffnet werden.

001 Deny read and write. Erlaubt ausschließlichen Zugriff auf das File. Jeder weitere Versuch, auch des gleichen Prozesses, den File zu öffnen wird abgewiesen. Ein Öffnen des Files mit diesem Attribut ist nicht erlaubt, wenn der File schon mit anderen Attributen geöffnet wurde, auch nicht wenn es der selbe Prozess ist.

010 Deny write access. Erlaubt anderen Prozessen den File mit read-only zu eröffnen. Ein Öffnen des Files mit diesem Attribut ist nicht erlaubt, wenn der File schon mit anderen Attributen von einem anderen Prozess geöffnet wurde.

011 Deny read access. Erlaubt anderen Prozessen den File mit write-only zu eröffnen. Ein Öffnen des Files mit diesem Attribut ist nicht erlaubt, wenn der File schon mit anderen Attributen von einem anderen Prozess geöffnet wurde.

100 Deny none. Gleich dem compatibility mode, nur erlaubt es anderen Prozessen nicht den File im compatibility mode zu öffnen. Ein Öffnen des Files mit diesem Attribut ist nicht erlaubt, wenn er schon im compatibility mode von einem anderen Prozess eröffner wurde.

Bei einem Fehler wegen der Attributes wird der Interrupt 24H benutzt mit dem Errorcode 02H (drive not ready). Das falsche Attributbit kann mit der Funktion 59H (get extended error information) ermittelt werden.

10.35.63 Funktion 3EH close handle

Die Funktion 3EH schließt den file dessen handle in dem Register BX angegeben wurde. Vorher müssen alle "file locks" aufgehoben werden, sonst ist das Ergebnis nicht vorherzusagen.

10.35.64 Funktion 3FH read handle

Die Funktion 3FH liest von einem file oder device, das mit einer handle geöffnet wurde.

Input:

- BX = handle number
- CX = number of bytes to read
- DS:DX = segment:offset of data buffer

Output:

- AX = number of bytes read from file
- DS:DX segment:offset of data read from file
- If function is not successful AX = errorcode

Vom standard input wird nur bis zum ersten CR gelesen.

Ist im Register AX eine 0, so steht der Filepointer am Ende des Files.

Ist AX kleiner al CX, so wurde nur ein Teil (der Rest) gelesen.

Im Netzwerk muß der Lesende Zugriff auf die Directory **und** den File haben.

10.35.65 Funktion 40H write handle

Die Funktion 40H schreibt eine im Register CX angegebene Zahl von Bytes von einem in den Registern DS:DX angegebenen Datenbuffer zu einem file oder device, dessen handle in dem Register BX angegeben wurde. Danach steht in AX die Zahl der geschriebenen Byte. Ist die carry flag gesetzt, so ist in AX der errorcode.
Geschrieben wird ab der Position des Pointers. Danach steht der Pointer hinter dem letzten Byte.
Ist die Zahl in CX = 0, so wird der Rest des Files abgeschnitten oder der File wird bis zum Pointer verlängert. In der FAT werden die cluster freigegeben oder belegt.
Die handle number muß eine der vordefinierten (0 bis 4) oder mit einer der Funktionen 3CH, 3DH, 5AH oder 5BH ermittelten sein.
Der Zugriff wird verweigert wenn der File mit read-only geöffnet wurde. Im Netz wird der Zugriff auch verweigert wenn der File oder der record von einem anderen Prozess gesperrt worden ist.
Wenn die Zahl in AX kleiner ist als die in CX angeforderte, so ist die disk voll. In diesem Fall ist die carry flag nicht gesetzt.

10.35.66 Funktion 41H delete directory entry

Die Funktion 41H löscht in dem Verzeichniss den Eintrag des in den Registern DS:DX angegebenen file. Die Angabe kann aus einem vollen Weg (path) bestehen, muß aber mit einem Null-Zeichen (ASCIIZ) enden. Es sind keine Ersatzzeichen (wildcards) erlaubt. Files mit directory- oder volume- attribute oder read-only Files können mit der Funktion 41H nicht gelöscht werden. Netzbenutzer müssen "create access" zu der Directory haben, in der der File gelöscht werden soll.
When invoked via INT21/AX=5D00h, this function has the undocumented behavior of allowing wildcards in the filename, deleting all matching files. Registers DS:DX has pointer to ASCIZ pathname of file to delete. Carry flag (CF) ist set on error. Errorcode 02h or 05h in register AX.

10.35.67 Funktion 42H move file pointer

Die Funktion 42H setzt die Position des Filepointers für die nächste read oder write Operation.

- AL = method code:
  - 00H byte offset from beginning of file
  - 01H byte offset from current location of file pointer
  - 02H byte offset from end of file
- BX = handle number
- CX:DX = offset value to move pointer:
  - CX most significant half of a doubleword value
  - DX least significant half of a doubleword value

Wenn die carry flag clear ist, so ist in DX:AX die neue Positon des Filepointers, absolut byte offset vom Anfang des Files.
Ist die carry flag nicht clear, so steht in AX der errorcode.
Der Wert in CX:DX gibt an, wie weit der Pointer bewegt werden soll. Bei der Methode 00H wird CX:DX als positve 32-Bit Integer verstanden, vom Anfang des Files.
Bei den Methoden 01H und 02H wird der Wert in CX:DX als positver oder negativer 32-Bit Wert verstanden. Mit der Methode 01H kann man den Filepointer also um 2 GByte nach vorne oder nach hinten bewegen. Mit der Methode 02H vom Ende des Files aus.
Abhängig von dem Wert in CX:DX kann mit den Methoden 01H und 02H eine Position vor dem Anfang des Files erreicht werden. Die Funktion 42H erzeugt dann keinen errorcode, Zugriffe auf das File können aber zu unerwarteten Fehlern führen. Mit der Funktion 59H kann man mehr Informationen über den Fehler bekommen.

10.35.68 Funktion 43H get/set file attributes

Die Funktion 43H setzt die in dem Register CX angegebenen Attributes oder erhält sie dahin zurück. Die Unterfunktionen werden mit dem Wert im AL Register (00H = get, 01H = set) angegeben. In den Registern DS:DX steht die Adresse zu dem ASCIIZ string des pathname.

- CX-bit0 read-only file
- CX-bit1 hidden file
- CX-bit2 system file
- CX-bit5 archive

Die Funktion 43H kann die volume oder directory attribute Bits (Bits 3 und 4) nicht setzen.

Netzbenutzer müssen create access zu der Directory haben, in der für den File die Attribute read-only, hidden oder system geändert werden sollen. Das archive Bit kann immer geändert werden.

10.35.69 Funktion 44H IOCTL

Die Funktion 44H ist eine Sammlung von Unterfunktionen, die es einem Prozess erlauben mit einem device driver direkt zu kommunizieren. Die Unterfunktionen werden im AL Register angegeben. Eine Differenzierung der Unterfunktionen wird in dem Register CL angegeben. Die Funktion 44H kann nur bei Gerätetreibern verwendet werden die sich an die Norm halten und mit Geräten die IOCTL unterstützen.

- Subfunktion 00H get device data
- Subfunktion 01H set device data
- Subfunktion 02H receive control data from character device
- Subfunktion 03H send control data to character device
- Subfunktion 04H receive control data from block device
- Subfunktion 05H send control data to block device
- Subfunktion 06H check input status
- Subfunktion 07H check output status
- Subfunktion 08H check if block device is removable
- Subfunktion 09H check if block device is remote
- Subfunktion 0AH check if handle is remote
- Subfunktion 0BH change sharing retry count
- Subfunktion 0CH generic I/O control for handles
  - minor code 45H: set iteration counter
  - minor code 65H: get iteration counter
- Subfunktion 0DH generic I/O control for block device
  - minor code 40H get device parameters
  - minor code 60H set device parameters

- minor code 41H write track on logical drive
- minor code 61H read track on logical drive
- minor code 42H format and verify track on logical drive
- minor code 62H verify track on logical drive

- Subfunktion 0EH get logical drive map
- Subfunktion 0FH set logical drive map

Funktion 44H Subfunktion 00H Get Device Data

Mit der Subfunktion 00H in Register AL wird Information über ein character device oder einen file erhalten, das mit einer handle angesprochen wird. In dem Register BX steht die handle Nummer. Wenn die Carry-Flag clear ist bedeuten die Bits in dem Register DX:

- for a file (bit7 = 0):
  - Bit15 = file is remote (DOS3+)
  - Bit14 = don't set file date/time on closing (DOS4+)
  - Bit12 = network device (DOS3+)
  - Bit11 = media not removable
  - Bit7 = 0 Handle refers to a file
  - Bit6 = 0 File has been written
  - Bit0-5 Drive number (0 = A, 1 = B, ...)
- for a device (bit7 = 1):
  - Bit15 = 0 Reserved
  - Bit14 = 1 Processes control strings transfered by IOCTL Subfunction 02H and 03H, set by MS-DOS.
  - Bit8-13 = 0 Reserved
  - Bit7 = 1 Handle refers to a device
  - Bit6 = End of file on input
  - Bit5 = 0 Checks for control characters (cooked mode)
  - Bit5 = 1 Does not check for control characters (raw mode).
  - Bit4 = 0 Reserved
  - Bit3 = 1 Clock device
  - Bit2 = 1 Null device
  - Bit1 = 1 Standard output device
  - Bit0 = 1 Standard input device
- Wenn das Carry-Bit gesetzt ist steht der Errorcode in dem Register AX:
  - 01H invalid IOCTL subfunction
  - 05H access denied
  - 06H invalid handle

Die Bits 8 - 15 im Register DX entsprechen den oberen 8 Bit des device-driver attribute word.
Die handle in BX muß auf ein geöffnetes File oder device zeigen.
Bit 5 im Register DX gibt an, ob die handle im "raw" oder im "cooked" mode ist. Im cooked mode wird CTRL-C, CTRL-P, CTRL-S und CTRL-Z erkannt und die Kontrolle an den Interrupt 23H abgegeben. In raw mode werden diese Zeichen nicht beachtet, nur CTRL-C von der Konsole wird erkannt, wenn BREAK erlaubt wurde.

Funktion 44H Subfunktion 01H Set Device Data

Die Unterfunktion 01H ist das Komplement der Unterfunktion 00H. Die Register haben die gleiche Bedeutung.

Die handle in BX muß auf ein geöffnetes File oder device zeigen.

Bit 5 im Register DX gibt an, ob die handle im "raw" oder im "cooked" mode ist. Im cooked mode wird CTRL-C, CTRL-P, CTRL-S und CTRL-Z erkannt und die Kontrolle an den Interrupt 23H abgegeben. In raw mode werden diese Zeichen nicht beachtet, nur CTRL-C von der Konsole wird erkannt, wenn BREAK erlaubt wurde.

Funktion 44H Subfunktion 02H Receive Control Data

Receive control data from character-oriented device driver.

- BX = handle number
- CX = number of bytes to transfer
- DS:DX = segment:offset of data buffer

Im Register AX steht dann die Zahl der gelesenen Bytes.

Diese Funktion erlaubt den Austausch von Informationen zwischen dem Programm und dem device driver. Das Gerät muß daran nicht beteiligt sein.

Funktion 44H Subfunktion 03H Send Control Data

Send control data to character-oriented device driver.

- BX = handle number
- CX = number of bytes to transfer
- DS:DX = segment:offset of data buffer

Im Register AX steht dann die Zahl der geschriebenen Bytes.

Diese Funktion erlaubt den Austausch von Informationen zwischen dem Programm und dem device driver. Das Gerät muß daran nicht beteiligt sein.

Funktion 44H Subfunktion 04H Receive Control Data

Receive control data from block-oriented device driver.

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- CX = number of bytes to transfer
- DS:DX = segment:offset of data buffer

Diese Funktion erlaubt den Austausch von Informationen zwischen dem Programm und dem device driver. Das Gerät muß daran nicht beteiligt sein.

Funktion 44H Subfunktion 05H Send Control Data

Send control data to block-oriented device driver.

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- CX = number of bytes to transfer
- DS:DX = segment:offset of data buffer

Diese Funktion erlaubt den Austausch von Informationen zwischen dem Programm und dem device driver. Das Gerät muß daran nicht beteiligt sein.

Funktion 44H Subfunktion 06H Check Input Status

- BX = handle number
- If function is successful: Carry flag is clear.
  - AL = 00H not ready
  - AL = FFH ready
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 05H acces denied
  - AX = 06H invalid handle
  - AX = 0DH invalid data (bad control string).

Status	Device	Input File
00H	Not ready	Pointer at EOF
0FFH	Ready	Ready

Funktion 44H Subfunktion 07H Check Output Status

- BX = handle number
- If function is successful: Carry flag is clear.
  - AL = 00H not ready
  - AL = FFH ready
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 05H acces denied
  - AX = 06H invalid handle
  - AX = 0DH invalid data (bad control string).

Status	Device	Output File
00H	Ready	Ready
0FFH	Ready	Ready

Output Files zeigen immer den Status Ready, auch wenn die Disk voll ist oder keine Disk im Drive vorhanden ist!

Funktion 44H Subfunktion 08H Check if Block-Device is removable

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- If function is successful: Carry flag is clear.
  - AX = 00H storage medium removable
  - AX = 01H storage medium not removable
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 0FH invalid drive
- War die Unterfunktion erfolgreich, kann der Benutzer aufgefordert werden, das Medium zu wechseln, falls ein File nicht gefunden wurde.
- Error-Code 01H meint, daß MS-DOS die Unterfunktion nicht erkennt. Dieser Fehler meint aber auch, meistens, daß der Device-Driver nicht über diese Unterfunktion verfügt. MS-DOS nimmt dann an, das das Medium nicht zu wechseln ist.

Funktion 44H Subfunktion 09H Check if Block-Device is remote

Mit dieser Unterfunktion kann festgestellt werden, ob ein angefordertes Block-Device local, am eigenen Rechner angebaut, oder remote, umgelenkt (redirected) zu einem anderen Rechner (server), ist.

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- If function is successful: Carry flag is clear.
  - DX = bit 12 = 1 drive is remote
  - DX = bit 12 = 0 drive is local
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 0FH invalid drive
- Anwendungsprogramme sollten diese Unterfunktion nicht verwenden. Für den Anwender sollte es gleich sein, ob das device local oder remote ist!
- Error-Code 01H kann meinen das der device-driver diese Unterfunktion nicht unterstützt oder das das Netzwerk nicht gestartet ist.

Funktion 44H Subfunktion 0AH Check if Handle is remote

Mit der Unterfunktion kann festgestellt werden, ob ein handle zu einem file oder device gehört, das local (eigener Rechner) oder remote (server) ist.

- BX = handle
- If function is successful: Carry flag is clear.
  - DX = attribute word for file or device
    - bit15 = 1 remote
    - bit15 = 0 local

- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 06H invalid handle
- Anwendungsprogramme sollten diese Unterfunktion nicht verwenden. Für den Anwender sollte es gleich sein, ob das file local oder remote ist!
- Error-Code 01H kann meinen das der device-driver diese Unterfunktion nicht unterstützt oder das das Netzwerk nicht gestartet ist.

Funktion 44H Subfunktion 0BH Change Sharing Retry Count

Diese Unterfunktion setzt den Wiederholungszähler, der nach einem file-sharing Fehler abgearbeitet werden muß, bevor ein Fehler zu dem rufenden Prozess zurückgegeben wird.

- CX = pause between retries
- DX = number of retries
- If function is successful: Carry flag is clear.
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
- Die Pause zwischen den Versuchen ist von der CPU und der CPU-Clock abhängig. MS-DOS führt eine Verzögerungsschleife aus, die aus 65 536 Instruktionen für jede Wiederholung besteht. Der aktuelle (V. 3.30) code ist:

```
xor     cx,cx

loop    $
```

- Wenn die *carry flag* gesetzt ist, kann das bedeuten das diese Funktion nicht unterstützt wird oder *file sharing* (SHARE.EXE) nicht geladen wurde.
- Mit der Unterfunktion 0BH kann das System auf die Anforderungen einer Anwendung mit *shared files* abgestimmt werden.
- Die alten Werte für *pause* und *retries* sollten vor verlassen der Anwendung wiederhergestellt werden um Überraschungen bei folgenden Anwendungen zu vermeiden!

Funktion 44H Subfunktion 0CH Generic I/O Control for Handles

Diese Unterfunktion setzt oder liest den iteration count eines character-oriented device.

- BX = handle
- CH = category code:
  - 05H printer
- CL = minor function code:
  - 45H set iteration count
  - 65H get iteration count
- DS:DX = segment:offset of 2-byte buffer receing or containing iteration-count word

Funktion 44H Subfunktion 0DH Generic I/O Control for Block Device

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- CH = category code:
  - 08H disk drive
- CL = minor function code:
  - 40H set parameters for block device
  - 41H write track on logical drive
  - 42H format and verify track on logical drive
  - 60H get parameters for block device
  - 61H read track on logical drive
  - 62H verify track on logical drive
- DS:DX = segment:offset of parameter block

Funktion 44H Subfunktion 0EH Get Logical Drive Map

Mit dieser Unterfunktion kann festgestellt werden, ob mehr als ein logisches Laufwerk einem block device zugeordnet wurde.

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- If function is successful: Carry flag is clear.
  - AL = 00H only one letter assigned to the block device
  - AL = 01-1AH logical drive letter (A - Z) mapped to block device
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 05H invalid drive

Funktion 44H Subfunktion 0FH Set Logical Drive Map

- BL = drive number (0 = default, 1 = A, 2 = B ...)
- If function is successful: Carry flag is clear.
  - AL = 00H only one letter assigned to the block device
  - AL = 01-1AH logical drive letter (A - Z) mapped to block device
- If function is not successful: Carry flag is set.
  - AX = 01H invalid function
  - AX = 05H invalid drive

10.35.70 Funktion 45H duplicate file handle

Die Funktion 45H stellt eine zusätzliche handle für einen geöffneten file zur Verfügung. Im Register BX steht die vorhandene handle und im Register AX erscheint die neue handle.

10.35.71 Funktion 46H force duplicate file handle

Die Funktion 46H erzwingt das Öffnen einer neuen handle. Im Register BX steht die vorhandene handle und im Register CX die zum öffnen gezwungen werden soll.

10.35.72 Funktion 47H get current directory

Die Funktion 47H gibt die Adresse eines 64-byte buffers in den Registern DS:SI für den in dem Register DL angegebenen drive zurück. Der Buffer enthält den path zu dem augenblicklichen Verzeichniss ohne die Angabe des drive und den ersten backslash.

10.35.73 Funktion 48H allocate memory

Die Funktion 48H teilt dem fordernden Prozess einen Block des memory zu. Die Zahl der paragraphs (16 Byte) steht in dem Register BX. Im Register AX wird die segment address des zugeteilten Blocks zurückgegeben.

10.35.74 Funktion 49H free allocated memory

Die Funktion 49H gibt den mit der Funktion 48H erhaltenen Memoryblock frei. In dem Register ES steht die segment address.

10.35.75 Funktion 4AH set block

Die Funktion 4AH passt die Größe des vorher zugeteilten Memories an. In dem Register BX steht die neue Größe des blocks in paragraphs und in dem Register ES die segment address des vorher zugeteilten blocks. Die carry-flag wird bei einem Feler gesetzt. Mögliche Fehler sind 07h, 08h und 09h. Wenn der Fehler 08h auftritt, ist im Register BX die maximale Größe. DOS3.x vergrößert auf dieses Maximum.

10.35.76 Funktion 4BH load overlay or program

Die Funktion 4BH läd einen file in das Memory und führt es gegebenenfalls aus.

- AL = subfunktion code:
  - 00H load and execute program
  - 01H load only
  - 03H load overlay
  - 04h reportedly called by MSC spawn(P_NOWAIT,...), when running under DOS4.x.
- DS:DX = segment:offset of ASCIIZ pathname for an executable program file
- ES:BX = segment:offset of parameter block

Parameter Block Format:

- wenn AL = 00H
  - 00H Segment pointer of the environment to be passed.
  - 02H Offset of command-line tail for the new PSP.
  - 04H Segment of command-line tail for the new PSP.
  - 06H Offset of the first file control block, to be copied into new PSP at offset 5CH.
  - 08H Segment of first file control block.
  - 0AH Offset of the second file control block, to be copied into new PSP at offset 6CH.
  - 0CH Segment of second file control block.
  - 0Eh will hold subprogram's initial SS:SP on return.
  - 12h will hold entry point (CS:IP) on return.
- wenn AL = 03H
  - 00H Segment address where overlay is to be loaded.
  - 02H Relocation factor to apply to loaded image.

10.35.77 Funktion 4CH end process

Die Funktion 4CH beendet den Prozess und kehrt zu dem rufenden mit einem code im AL Register zurück.

10.35.78 Funktion 4DH get return code child process

Die Funktion 4DH stellt den return code eines mit der Funktion 4BH geladenen und entweder mit der Funktion 31H oder 4CH beendeten Programms wieder her.

- AH = Termination type:
  - 00H Child terminated normally.
  - 01H Child was terminated by user's entry of a CTRL-C.
  - 02H Child was terminated by critical error handler.
  - 03H Child terminated normally and stayed resident.
- AL = return code:
  - Value passed by the child program in register AL when it terminated with Interrupt 21H Function 4CH or 31H.
  - 00H if the child terminated using Interrupt 20H, Interrupt 27H or Interrupt 21H Function 00H.

10.35.79 Funktion 4EH find first file

Die Funktion 4EH sucht in dem augenblicklichen Verzeichniss nach dem ersten Eintrag für den in den Registern DS:DX angegebenen pathname (ASCIZ string mit drive, path und wildcards) mit dem in dem Register CX angegebenen Attribute. Die Attribute sind:

Bit0 read only

Bit1 hidden file

Bit2 system file

Bit3 volume label

Bit4 subdirectory

Bit5 written since backup (archive)

Bit8 shareble (Novell NetWare).

Die Caryflag ist gesetzt, wenn ein Fehler auftrat. Mögliche Fehler sind 02h und 12h. In der DTA steht folgender Datablock:

PCDOS3.10/4.01, MSDOS3.2/3.3

00 Byte drive letter

01 11 Bytes searche template

0C Byte search attribute

DOS2.x and some DOS3.x

00 Byte search attributes

01 Byte drive letter

02 11 Bytes search template

DOS2.x and most 3.x

0D Word entry count within directory

0F DWord pointer to DTA

13 Word cluster number of start of parent directory

PCDOS4.01,MSDOS3.2/3.3

0D Word entry count within directory

0F Word cluster number of start of parent directory

11 4 Bytes reserved

all versions

15 Byte attribute of file found

16 Word file time, bits 11-15 hour, bits 5-10 minute, bits 0-4 seconds/2.

18 Word file date, bits 9-15 year-1980, bits 5-8 month, bits 0-4 day.

1A DWord file size

1E 13 Bytes ASCIZ filename+extension

10.35.80 Funktion 4FH find next file

Die Funktion 4FH sucht nach erfolgreicher Funktion 4E weiter.

10.35.81 Funktion 50H set program segment prefix address

Mit der Funktion 50H kann ein TSR-Programm eine interne MS-DOS Variable mit einem PSP beschreiben. Register BX = PSP Segment. Da keine internen Stacks benutzt werden, ist diese Funktion reentrant (DOS3+).

10.35.82 Funktion 51H get program segment prefix address

Mit der Funktion 51H kann ein TSR-Programm eine interne MS-DOS Variable lesen. Register BX = PSP Segment des aktuellen DOS-SDA. Da keine internen Stacks benutzt werden, ist diese Funktion reentrant (DOS3+).

10.35.83 Funktion 52H get 'DOS-Info-Block'

Die Funktion 52H liefert in den Registern ES:BX den Pointer zu einem "DOS-Info-Block" (List-of-Lists (LoL)). Der Offset -2H liefert den Offset des ersten MCB (Memory Control Block). Der Offset -4 liefert das Segment, beim ersten null.

Im Offset 0H des MCB steht eine Kennung. M steht für ein TSR-Programm das noch einen Nachfolger hat. Z steht für das letzte residente Programm.

Offset 0 hat Zeiger auf den ersten Drive Parameter Block (DPB).

Offset 4 hat Zeiger auf den letzten MS-DOS-Puffer.

Offset 8 hat Zeiger auf den Uhr-Treiber.

Offset C hat Zeiger auf den Konsolen-Treiber.

Offset 10 ist ein Wort mit der maximalen Länge eines Sektors.

Offset 12 hat Zeiger auf den MS-DOS-Puffer.

Offset 16 hat Zeiger auf die Pfadtabelle.

Offset 1A hat Zeiger auf die System File Tabelle.

Offset 2E ist ein Wort mit "Einträge in der System file table".

Offset 30 ist ein Byte mit der Anzahl der Laufwerke.

Offset 31 ist ein Byte mit der Zahl des letzten Laufwerks.

10.35.84 Funktion 53H Translate BIOS Parameter Block

Versionsabhängige Unterschiede in der Struktur der Blöcke.

Input: DS:SI zeigt auf einen BIOS Parameter Block (BPB).

Output: ES:BP zeigt auf einen DPB.

Format of BIOS Parameter Block:

00 Word Number of byte/sector

02 Byte Number of sectors/cluster. Corresponds to (DPB byte 04)+1.

03 Word Number of reserved sectors

05 Byte Number of FATs

06 Word Number of root directory entries

08 Word Total number of sectors

0A Byte Media descriptor

0B Word Number of sectors per FAT

DOS3+

0D Word Number of sectors per track

0F Word Number of heads

11 DWord Number of hidden sectors

15 11 Bytes reserved

DOS4.0

15 DWord Total number of sectors if word at 8h contains zero

19 6 Bytes reserved

1F Word Number of cylinders

21 Byte Device type

22 Word Device attributes (removable media, etc)

10.35.85 Funktion 54H get verify state

Die Funktion 54H gibt den Wert der MS-DOS verify flag zurück. Im Register AL steht 00H für verify off oder 01H für verify on.

10.35.86 Funktion 55h Create New PSP

Erzeugt einen neuen PSP für einen Kind-Prozeß.

Register	Bedeutung
DX	segment number at which to set up PSP
SI	(DOS3+) value to place in memory size field at DX:[0002h]
	the current PSP segment is set to the segment of the new PSP under DOS2+

10.35.87 Funktion 56H change directory entry

Die Funktion 56H benennt einen file um. In den Registern DS:DX steht der alte ASCIZ pathname und in den Registern ES:DI der neue. Stimmen die Verzeichnisse der beiden Angaben nicht überein, so wird der file in das neue Verzeichniss übertragen. Wenn diese Funktion über die Funktion 5D00h des Interrupt 21h benutzt wird, sind wildcards erlaubt. Die Fehler 02h, 03h, 05h und 11h sind möglich. Wenn diese Funktion über die Funktion 5D00h des Interrupt 21h benutzt wird, tritt der Fehler 12h auf. Dann war die Funktion erfolgreich.

10.35.88 Funktion 57H get/set date/time of file

Die Funktion 57H zeigt das Datum und die Zeit der letzten Änderung des files an oder setzt sie neu.

Unterfunktion 02h Get unknown information

BX unknown (0000h through 0004h)

DS:SI unknown pointer

ES:DI pointer to result buffer

CX size of result buffer

CX size of returned data

Unterfunktion 03h Get unknown information

BX unknown (0000h through 0004h)

DS:SI unknown pointer

ES:DI pointer to result buffer

CX size of result buffer

CX size of returned data

Unterfunktion 04h Unknown

BX file handle

DS:SI unknown pointer

ES:DI pointer to result buffer

CX size of result buffer

10.35.89 Funktion 58H get/set allocation strategy

Die Funktion 58H zeigt die Strategie der Memory-Zuweisung an oder setzt sie neu.

- AL = 00H get allocation strategy
  - AX = 00H first fit is set
  - AX = 01H best fit is set
  - AX = 02H last fit is set
- AL = 01H set allocation strategy
  - BX = 00H use first (lowest available) block that fits
  - BX = 00H use block that fits best
  - BX = 00H use last (highest available) block that fits

10.35.90 Funktion 59H get extended error

Die Funktion 59H zeigt Informationen über den extended error code an.

- AX = extended error code:
  - 00H no error encountered
  - 01H invalid finction number
  - 02H file not found
  - 03H path not found
  - 04H too many file open; no handles available
  - 05H access denied (verweigert)
  - 06H invalid handle
  - 07H memory cotrol blocks destroyed
  - 08H insufficient memory

- 09H invalid memory block address
- 0AH invalid environment
- 0BH invalid format
- 0CH invalid access code
- 0DH invalid data
- 0EH reserved
- 0FH invalid disk drive
- 10H attempt to remove current directory
- 11H device not the same
- 12H no more files
- 13H write protected disk
- 14H unknown unit
- 15H drive not ready
- 16H invalid command
- 17H data error based on cyclic redundancy check (CRC)
- 18H length of request structure invalid
- 19H seek error
- 1AH non MS-DOS disk
- 1BH sector not found
- 1CH printer out of paper
- 1DH write fault
- 1EH read fault
- 1FH general failure
- 20H sharing violation
- 21H lock violation
- 22H invalid disk change
- 23H FCB unavailable
- 24H sharing buffer exceeded
- 25H-31H reserved
- 32H unsupported network request
- 33H remote machine not listening
- 34H duplicate name on network
- 35H network name not found
- 36H network busy
- 37H device no longer exists on network
- 38H netBIOS command limit exceeded
- 39H error in network adapter hardware
- 3AH incorrect response from network
- 3BH unexpected network error
- 3CH remote adapt incompatible
- 3DH print queue full

- 3EH queue not full
- 3FH not enough room for print file
- 40H network name deleted
- 41H access denied
- 42H incorrect network device type
- 43H network name not found
- 44H network name limit exceeded
- 45H netBIOS session limit exceeded
- 46H temporary pause
- 47H network request not accepted
- 48H print or disk redirection paused
- 49H-4FH reserved
- 50H file already exists
- 51H reserved
- 52H cannot make directory
- 53H failure on Interrupt 24H
- 54H out of structures
- 55H already assigned
- 56H invalid password
- 57H invalid parameter
- 58H net file fault

- BH = error class:
  - 01H out of resource (such as storage)
  - 02H temporary situation, expected to end; not an error
  - 03H authorization problem
  - 04H internal error in system software
  - 05H hardware failure
  - 06H system software failure, such as missing or incorrect configuration files; not the fault of the active process
  - 07H application programm error
  - 08H file or item not found
  - 09H file or item of invalid format or type or otherwise unsuitable
  - 0AH file or item interlock
  - 0BH drive contains wrong disk, disk has bad spot or other problem with storage medium
  - 0CH already exist
  - 0DH unknown
- BL = suggested action:
  - 01H perform a reasonable number of retries before prompting user to choose Abort or Ignore in response to error message
  - 02H perform a reasonable number of retries, with pauses between, before prompting user to choose Abort or Ignore in response to error message

- 03H prompt user to enter correct information, such as drive letter or filename
- 04H clean up and exit application
- 05H exit immediatly without cleanup
- 06H ignore; informational error
- 07H prompt user to remove cause of error (change disk) and then retry

- CH = location of error:
  - 01H unknown
  - 02H block device
  - 03H network
  - 04H serial device
  - 05H memory related

10.35.91 Funktion 5AH create temporary file

Die Funktion 5AH benutzt die Systemuhr um einen verwechselungsfreien filename zu erzeugen. Der filename wird an den angegebenen path angehängt und der temporäre file geöffnet. Die zurückgegebene handle kann für weiter Operationen benutzt werden. Der Aufruf erfolgt mit:

- CX = file attribute:
  - 00H normal file
  - 01H read-only file
  - 02H hidden file
  - 04H system file
- DS:DX = segment:offset des ASCII path der mit einem "backslash"endet und 13 Byte Platz für den erzeugten filename.

Ist die Funktion erfolgreich, so steht in dem Register AX die handle und in den Registern DS:DX das Segment und der Offset zu dem ganzen pathname.

10.35.92 Funktion 5BH create new file

Die Funktion 5BH erzeugt einen neuen file mit dem angegebenen pathname. Ein Überschreiben eines vorhandenen files wie bei der Funktion 3C wird als Fehler erkannt.

10.35.93 Funktion 5CH lock/unlock

Die Funktion 5CH erlaubt einem Process, der sich in einer Netzwerk oder multitasking Umgebung befindet, einen Bereich von Bytes in einem geöffneten file vor fremder Benutzung zu schützen oder den Schutz aufzuheben.

- AL = 00H lock region; 01H unlock region
- BX = handle
- CX:DX = 4-byte integer specifying beginning of region to be locked or unlocked (offset in bytes from beginning of file)
- SI:DI = 4-byte integer specifying length of region (measured in bytes)

10.35.94 Funktion 5DH

Unterfunktion 00h Server Function Call

Execute a specified INT21h call using the sharing rules for the specified network machine number and process.

DS:DX pointer to DOS parameter list

DPL contains all register values for a call to INT21h.

Offset	Size	Description
00h	Word	AX
02h	Word	BX
04h	Word	CX
06h	Word	DX
08h	Word	SI
0Ah	Word	DI
0Ch	Word	DS
0Eh	Word	ES
10h	Word	reserved (0)
12h	Word	computer ID (0000h=current system)
14h	Word	process ID (PSP segment on specified computer)

Unterfunktion 01h Commit all files

Flush all disk buffers and update the directory entry for each file which has been written to since opening or the last commit.

DS:DX pointer to DOS parameter list, only computer ID and process ID fields used.

The computer and process IDs are stored but ignored under DOS3.3.

Unterfunktion 02h Share.exe - Close file by name

DS:DX pointer to DOS parameter list, only fields DX, DS, computer ID, and process ID used.

DPL's DS:DX pointer to ASCIZ name of file to close.

Unterfunktion 03h Share.exe - Close all files for given computer

DS:DX pointer to DOS parameter list, only conputer ID used.

Unterfunktion 04h Share.exe - Close all files for given process

DS:DX pointer to DOS parameter list, only computer ID and process ID fileds used.

Unterfunktion 05h Share.exe - Get open file list entry

DS:DX pointer to DOS parameter list.

DPL's BX index of sharing record.

DPL's CX index of SFT in sharing record's SFT list.

On Return

ES:DI pointer to ASCIZ filename.

BX network machine number of SFT's owner.

CX number of locks held by SFT's owner.

Unterfunktion 06H Get DOS SDA

In den Registern DS:SI steht ein far Pointer zu der swappable Data Area (SDA). Enthält Adressen für aktuelle PSP, DTA und stacks. Im Register CX wird die Anzahl der Bytes zurückgegeben, die geswapped werden sollen. In DX steht die Zahl der Bytes, die immer geswapped werden müssen.

DS:DI pointer to nonreentrant data area (includes all three DOS stacks)(critical error flag is first byte).

CX size in bytes of area which must be swapped while in DOS.

DX dize in bytes of area which must always be swapped.

Unter DOS 4.0 sollte die Funktion 5D0Bh des Interrupts 21h benutzt werden.

Unterfunktion 07h Get redirected printer mode

Determine whether redirected printer output is treated as a single print job or as multiple print jobs.

DL=00 redirected output is combined.

DL=01 redirected output in separate print jobs.

Unterfunktion 08h Set redirected printer mode

Specify whether redirected printer output should be treated as a single print job or as multiple print jobs.

DL=00 redirected output is combined.

DL=01 redirected output placed in separate jobs, start new print job now.

Unterfunktion 09h Flush redirected printer output

Force all redirected printer output to be sent to the printer, and start a new print job.

Unterfunktion 0AH set extended error information

Diese Unterfunktion kopiert eine vom rufenden Programm definierte Datenstruktur nach MS-DOS. (DPL)

- DS:DX = address of 11-word data structure:
  - word 0: Register AX as returned by Function 59H.
  - word 1: Register BX
  - word 2: Register CX
  - word 3: Register DX
  - word 4: Register SI
  - word 5: Register DI
  - word 6: Register DS
  - word 7: Register ES
  - word 8 - 0AH: reserved; should be 0.

Unterfunktion 0BH Get List of SDA

Return the address of a list of regions which must be swapped out and restored to allow DOS to be reentered.

DS:SI pointer to swappable data area list.

Copying and restoring the swappable areas allows DOS to be reentrant unless it is in a critical section delimited by calls to INT2A function 80h and INT2A function 81h or 82h.

Format of swappable data area list:

Offset	Size	Description
00h	Word	count of data areas
02h	N Bytes	count copies of data area record
	00h	DWORD address
	04h	WORD length and type
		bit 15 set if swap always, clear if swap in DOS
		bits 14-0: length in bytes

Format of PCDOS 4.01 swappable data area:

Offset	Size	Description
00h	Byte	critical error flag
01h	Byte	InDOS flag (count of active INT21h calls)
02h	Byte	drive on which current critical error occurred or FFh
03h	Byte	locus of last error
04h	Word	extended error code of last error
06h	Byte	suggested action for last error
07h	Byte	class of last error
08h	DWord	ES:DI pointer for last error
0Ch	DWord	current DTA
10h	Word	current PSP
12h	Word	stores SP across an INT23
14h	Word	return code from last process termination
		cleared after reading with AH=4Dh
16h	Byte	current drive
17h	Byte	extended break flag
18h	2 Bytes	unknown

Remainder need only be swapped if in DOS:

Offset	Size	Description
1Ah	Word	value of AX on call to INT21h
1Ch	Word	PSP segment for sharing/network
1Eh	Word	network machine number for sharing/network (0000h=current)
20h	Word	first usable memory block found when allocating memory
22h	Word	best usable memory block found when allocating memory
24h	Word	last usable memory block found when allocating memory
26h	2 Bytes	apparently not referenced by kernel
		unknown
2Fh	2 Bytes	apparently not referenced by kernel
30h	Byte	day of month
31h	Byte	month
32h	Word	year - 1980
34h	Word	number of days since 1-1-1980
36h	Byte	day of week (0=Sunday)
		unknown
38h	30 Bytes	device driver request header
58h	DWord	pointer to device driver entry point
5Ch	22 Bytes	device driver request header
72h	30 Bytes	device driver request header
90h	6 Bytes	unknown
96h	6 Bytes	CLOCK$ transfer record
9Ch	2 Bytes	unknown
9Eh	128 Bytes	buffer for filename
11Eh	128 Bytes	buffer for filename
19Eh	21 Bytes	findfirst/findnext search data block
1B3h	32 Bytes	directory entry for found file
1D3h	88 Bytes	copy of current directory structure for drive being accessed
22Bh	11 Bytes	FCB-format filename, use unknown
236h	Byte	unknown
237h	11 Bytes	wildcard destination specification for rename
		unknown
24Dh	Byte	attribute mask for directory search
		unknown
252h	Byte	flag indicating how DOS function was invoked
		unknown
25Ch	Byte	type of process termination (00h - 03h)
		unknown
260h	DWord	pointer to Drive Parameter Block for critical error invocation
264h	DWord	pointer of stack frame containing user registers on INT21h
268h	Word	stores SP
26Ah	DWord	pointer to DOS Drive Parameter Block for unknown use
26Eh	Word	segment of disk buffer
		unknown
278h	Byte	Media ID byte returned by INT21 function 1Bh and 1Ch
279h	Byte	apparently not referenced by kernel
27Ah	DWord	unknown pointer
27Eh	DWord	pointer to current SFT
282h	DWord	pointer to current directory structure for drive being accessed
286h	DWord	pointer to caller's FCB
		unknown
28Eh	DWord	pointer to a JFT entry in process handle table
292h	Word	offset in DOS CS of first filename argument
294h	Word	offset in DOS CS of second filename argument
		unknown

Offset	Size	Description
2A2h	Word	appears to be directory cluster number
		unknown
2AEh	DWord	offset in file
		unknown
2B4h	Word	bytes in partial sector
2B6h	Word	number of sectors
		unknown
2BEh	DWord	number of bytes appended to file
2C2h	DWord	pointer to a disk buffer for unknown use
2C6h	DWord	pointer to a System File Table for unknown use
2CAh	Word	used by INT21h dispatcher to store caller's BX
2CCh	Word	used by INT21h dispatcher to store caller's DS
2CEh	Word	temporary storage while saving/restoring caller's registers
2D0h	DWord	pointer to prev call frame (offset 264h) if INT21h reentered
		also switched to for duration of INT24h
		unknown
2ECh	Word	stores DS during call to [List-of-Lists+37h]
		unknown
2F3h	DWord	pointer to user-supplied filename
		unknown
2FBh	Word	stores SS during call to [List-of-Lists+37h]
2FDh	Word	stores SP during call to [List-of-Lists+37h]
2FFh	Byte	flag, nonzero if stack switched in calling [List-ofLists+37h]
300h	21 Bytes	FindFirst search data for source file(s) of a rename operation
315h	32 Bytes	directory entry for file being renamed
335h	331 Bytes	critical error stack
480h	384 Bytes	disk stack
600h	384 Bytes	character I/O stack
780h	Byte	flag affecting AH=08h
781h	Byte	appears to be a drive number
bis 78ah	Word	unknown

10.35.95 Funktion 5EH get machine name/printer setup

Die Funktion 5EH mit ihren Unterfunktionen ist für das "Netzwerk" zuständig.

Unterfunktion 00H get machine name

Das Register DX zeigt auf ein 16 Byte Array für den Namen. Wenn im Register CH danach eine null ist, ist der Name nicht gesetzt.

Unterfunktion 01H set machine name

Specify the system's network machine name and number.

CH 00h undefine name, other define name.

CL name number.

DS:DX pointer to 15-character blank-padded ASCIZ name.

Unterfunktion 04H set printer mode

Specify whether the printer should be operated in text or binary mode.

BX redirection list index

DX mode; bit0: set if binary, clear if text (tabs expanded to blanks)

Unterfunktion 05H get printer mode

Determine whether the printer is being operated in text or binary mode.

BX redirection list index

Returns

DX printer mode

10.35.96 Funktion 5FH get/make/cancel assign list entry

Die Funktion 5FH mit ihren Unterfunktionen ist für das "Netzwerk" zuständig.

Unterfunktion 00h Get redirection mode

Determine whether disk or printer redirection is current enabled.

BL redirection type; 03h printer, 04h disk drive

Returns

CF clear if successful

BH redirection state; 00h off, 01h on

Unterfunktion 01h Set redirection mode

Specify whether disk or printer redirection is to be enabled or disabled.

BL redirection type; 03h printer, 04h disk drive

BH redirection state; 00h off, 01h on

Unterfunktion 05h Get redirection list extended entry

Return the source and target of a given redirection, as well as its status and type.

BX redirection list index

DS:SI pointer to buffer for ASCIZ source device name

ES:DI pointer to buffer for destination ASCIZ network path

Returns if CF is clear:

BH device status flag (bit0 clear if valid)

BL device type (03h if printer, 04h if drive)

CX stored parameter value (user data)

BP NETBIOS local session number

DS:SI buffer filled

ES:DI buffer filled

Unterfunktion 06h Get redirection list

This function appears to be similar to INT21h function 5F02h (get redirection list) and INT21h function 5F05h.

10.35.97 Funktion 60h Canonicalize Path String

Gibt die wahre Form eines Dateinamens zurück; SUBST, JOIN, ASSIGN und network redirections werden aufgelöst.

DS:SI pointer to ASCIZ relative path string or directory name.

ES:DI pointer to 128-byte buffer for ASCIz canonical fully qualified name.

Returns with error:

AX 02h invalid source name, 03h invalid drive or malformed path.

Returns:

AH 00h

AL destroyed

buffer filled with qualified name

10.35.98 Funktion 61H

Unused, returns immediatly with AL=00h.

10.35.99 Funktion 62H get PSP

Mit der Funktion 62H wird die Segment Adresse des PSP des laufenden Prozesses in dem Register BX zurückgegeben.

10.35.100 Funktion 63H Get Lead Byte Table

Die Funtion 63H ist nur in MS-DOS Version 2.25 vorhanden.

10.35.101 Funktion 64H Get Lead unknown flag

The purpose of this function is not known.

10.35.102 Funktion 65H Get Extended Country Information

Im Register BX muß die Zahl der code page stehen (-1 = current). Im Register CX steht die gewünschte Zahl der Daten. Ist die Tabelle länger, wird nur die angegebene Zahl berücksichtigt. Im Register DX steht die country ID (-1 = current). In den Registern ES:DI steht der pointer zu dem Datenbuffer.

Unterfunktion 01H extended country information buffer

- 00H byte Info ID = 01
- 01H word size (38 or less)
- 03H word country ID
- 05H word code page

Unterfunktion 02H extended country uppercase table

- 00H byte info ID = 02
- 01H dword pointer to uppercase table

Unterfunktion 04H extended country file uppercase table

- 00H byte info ID = 04
- 01H dword pointer to file name uppercase table

Unterfunktion 05H Get pointer to filename terminator table

Return information about the characters which terminate a filename.

BX code page (-1= global code page)

DX country ID (-1= current country)

ES:DI pointer to country information buffer.

CX size of buffer ($\geq$5).

Returns: CF clear if succesful

CX size of country information returned

ES:DI pointer to country information

Format of country information:

Offset	Size	Description
00h	Byte	info ID
01h	DWord	pointer to filename character table

Format of filename terminator table:

Offset	Size	Description
00h	Word	table size
02h	7 Bytes	unknown
09h	Byte	length of following data
0Ah	N Bytes	characters which terminate a filename

Unterfunktion 06H extended country collating table

- 00H byte Info ID = 06
- 01H dword pointer to collating table

Unterfunktion 20H capitalize character

Capitalize a character using the capitalization rules for the current country.

DL character to capitalize

Returns:

DL capitalized character

Unterfunktion 21H capitalize string

Capitalize a string using the capitalization rules for the current country.

DS:DX pointer to string to capitalize

CX length of string

Unterfunktion 22H capitalize string

Capitalize a string using the capitalization rules for the current country.

DS:DX pointer to ASCIZ string to capitalize

Unterfunktion 23H Determine if character represents yes/no response

Compare the specified character against the YES and NO response for the current country.

DL character

DH second character of double-byte character (if applicable).

Returns:

AX type: 00h no, 01h yes, 02h neither yes nor no.

Unterfunktion A0 Country-dependent filename capitalization

Capitalize a filename character using the filename capitalization rules for the current country. Nicht in DOS 4.x.

DL character to capitalize

Returns:

DL capitalized character

Unterfunktion A1 Country-dependent filename capitalization

Capitalize a filename string using the filename capitalization rules for the current country. Nicht in DOS 4.x.

DS:DX pointer to filename string

CX length of string

Unterfunktion A2 Country-dependent filename capitalization

Capitalize a filename string using the filename capitalization rules for the current country. Nicht in DOS 4.x.

DS:DX pointer to ASCIZ filename to capitalize

10.35.103 Funktion 66H Get/Set Global Code Page

Unterfunktion 01H get global code page

Im Register BX ist die active code page und im Register DX ist die system code page.

Unterfunktion 02H set global code page

Im Register BX ist die active code page und im Register DX ist die system code page.

10.35.104 Funktion 67H Set Handle Count

Im Register BX muß die Zahl der erlaubten geöffneten handles stehen. Es wird das Memory benutzt, das mit Interrupt 21H Funktion 4A bereitgestellt wurde. Ist die Zahl kleiner als die bisher erlaubte wird die Änderung erst übernommen, wenn die Zahl der geöffneten kleiner als die neue Zahl geworden ist.

10.35.105 Funktion 68H Flush Buffer

Flushes DOS file buffers to disk, ensuring that all information has been securely written to the file.

10.35.106 Funktion 69h Get/set disk serial number

Determine or specify a disk's serial number and volume label.

AL subfunction: 00h get, 01h set

BL drive (0=default, 1=A, ...)

DS:DX pointer to disk info

Returns:

AL =00h buffer filled with appropriate values from extended BPB.

AL =01h extended BPB on disk set to values from buffer.

Format of disk info:

Offset	Size	Description
00h	Word	info level (zero)
02h	DWord	disk serial number (binary)
06h	11 Bytes	volume label or "NO NAME"
11h	8 Bytes	(AL=00)filesystem type ("FAT12" or "FAT16")

10.35.107 Funktion 6Ah unknown

10.35.108 Funktion 6Bh unknown

10.35.109 Funktion 6CH Extended Open/Create

In DOS V4 zusätzlich zu den anderen handle open und create Funktionen.

- BX open mode:
  - Bit00 - 02
    - 000 read access
    - 001 write access
    - 010 read/write access
  - Bit03 reserved (must be 0)
  - Bit04 - 06 sharing mode
    - 000 compatibility mode
    - 001 read/write access
    - 010 write access denied
    - 011 read access denied
    - 100 full access
  - Bit07
    - 0 inherited by child process
    - 1 private to current process
  - Bit08 - 12 reserved
  - Bit13
    - 0 INT24 functions normally for this file.
    - 1 INT24 is not generated during any file access involving this file.
  - Bit14
    - 0 write file to buffer only when internal DOS file buffer is full.
    - 1 flush buffer to file after every file write (as with INT21/F68).
  - Bit15 reserved
- CX file attribute (only if creating)
- DX function control
  - Bit00 = 0 if file exists, generate error and take no action.
  - Bit00 = 1 if file exists, open the file.
  - Bit01 = 1 if file exists, truncate and open.
  - Bit04 = 0 if file does not exist, generate error and take no action.
  - Bit04 = 1 if file does not exist, create the file.
  - Bit08 - 15 reserved
- DS:SI ASCIIZ file specification

10.35.110 Funktion 89H Delay Process Execution

10.36 Interrupt Nr. 22H Terminate Routine Address

Der Vektor des Interrupts 22H, Memory-Plätze 0000:0088H - 0000:008BH, beinhaltet die Adresse der Routine die die Kontrolle erhält, wenn das laufende Programm mit Interrupt 20H, Interrupt 27H oder Interrupt21H Funktionen 00H, 31H oder 4CH beendet wird.

Dieser Interrupt sollte niemals direkt benutzt werden.

Die Adresse in dem Vektor wird nach dem Laden aber vor dem Laufen in die Offsetes 0AH bis 0DH des PSP geschrieben. Als Teil der MS-DOS Abschlussbehandlung eines Programms wird die Adresse erneuert falls sie geändert wurde.

10.37 Interrupt Nr. 23H CTRL-C Handler Address

Der Vektor des Interrupts 23H, Memory-Plätze 0000:008CH - 0000:008FH, beinhaltet die Adresse der Routine die die Kontrolle erhält, wenn bei einem character I/O ein CTRL-C oder CTRL-Break erkannt wird und die Break-flag angeschaltet ist.

Dieser Interrupt sollte niemals direkt benutzt werden.

10.38 Interrupt Nr. 24H Critical Error Handler Address

Der Vektor des Interrupts 24H, Memory-Plätze 0000:0090H - 0000:0093H, beinhaltet die Adresse der Routine die die Kontrolle erhält, wenn ein critical error, z.B.hardware error, erkannt wird.

Dieser Interrupt sollte niemals direkt benutzt werden.

10.39 Interrupt Nr. 25H Absolute Disk Read

Der Interrupt 25H liest direkt über das BIOS Daten von einem logischen Sektor einer Platte in das Memory.

- AL = drive number (0 = drive A)
- CX = number of sectors to read
- DX = starting relative (logical) sector number
- DS:BX = segment:offset of DTA

10.40 Interrupt Nr. 26H Absolute Disk Write

Der Interrupt 26H schreibt direkt über das BIOS Daten von der DTA auf einen logischen Sektor.

- AL = drive number (0 = drive A)
- CX = number of sectors to write
- DX = starting relative (logical) sector number
- DS:BX = segment:offset of DTA

10.41 Interrupt Nr. 27H Terminate and Stay Resident

Der Interrupt 27H beendet ein Programm und reserviert Teile oder das ganze zugeteilte Memory, so daß es nicht überschrieben werden kann. Der Interrupt 27H ist nicht so flexibel wie der Interrupt 21H Funktion 31H, da er nur 64Kb zulässt und die Adresse des PSP im CS Register braucht.

- DX = offset des letzten Byte plus 1, relativ zum PSP, des Programms das geschützt werden soll. Andere Aussage ist "size of resident program (bytes)".
- CS = Segment Adresse des PSP

10.42 Interrupt Nr. 28H Idle Interrupt

Die MS-DOS Funktionen, die z.B. auf ein Ereignis (Tastatur) warten, haben zum Teil einen Interrupt 28H Aufruf eingebaut. Der Vektor des Interrupt28H kann also auf ein TSR-Programm zeigen, daß nur zu den Zeiten abgearbeitet wird, zu denen die CPU nicht vom Anwendungsprogramm benötigt wird.

10.43 Interrupt Nr. 29H MS-DOS Fast PUTCHAR

System Console Output. Dieser Interrupt wird von DOS-Outputroutinen benutzt wenn das Bit4 im Attributewort des Characterdevice gesetzt ist.

10.44 Interrupt Nr. 2AH NETBIOS Extended Functions (MS-NET Session)

Der Interrupt 2AH stellt die gleichen Funktionen wie der Interrupt 5CH dem Benutzer zur Verfügung und verlangt die gleichen Voraussetzungen z.B. NCB. Die extended functions sind aber nur mit dem Interrupt 2AH zu erreichen. Um die extended functions des Interrupts 2AH benutzen zu können, muß man vorher einen Digital Session Control Block (DSCB) anlegen.

Field Name	Size	Description
dscb_cmd	1	Command code field.
dscb_err	1	Return code field.
dscb_index	1	Index number of the node entry; used only for DSCB functions 0-7.
dscb_name	16	Specifies the name of the adapter/node name.
dscb_num	2	Specifies the DECnet node address; Bits 31-26 = area, Bits 25-0 = node.
dscb_objtype	1	Indicates DECnet object type of remote adapter.
dscb_acc_len	2	Length of account name if providing optional access control data.
dscb_acc	40	Indicates account name, zero terminated if providing optional access control data.
dscb_pass_len	2	Indicates the password length if providing optional access control data.
dscb_pass	40	Specifies password, zero terminated if providing optional access control data.
dscb_user_len	2	Indicates the user name length.
dscb_user	40	Specifies the user name, zero terminated.

10.44.1 Funktion DCH Extended Functions

Die Register ES:BX zeigen auf die Adresse des DSCB.

Cammand name	Function Code	Description
Installation check	00	Use to check if NETBIOS emulation with the DEC extensions is installed.
Add entry	01	Use to add an entry to remote server table.
Delete entry	02	Delete entry in the remote server table by node number
Delete entry	03	Delete entry in the remote server table by node name
Read entry	04	Read entry in the remote server table by node number
Read entry	05	Read entry in the remote server table by node name
Read entry	06	Read entry in the remote server table by index
Delete all entries	07	Delete all entries in the remote server table
Add remote adapter	21	Add remote adapter name, with optional access control data, to local adapter name table.
Clear remote adapter	22	Remove remote adater name, by name, from local adapter name table
Read remote adapter	23	Read the remote adater name by name, with access control data
Get remote adapter	24	Get the number of entries in the table of remote adapter names
Read all entries	25	Read all remote adapter name entries
Clear all entries	26	Clear all remote adapter name entries

10.45 Interrupt Nr. 2AH MS-network

10.45.1 Funktion 00h Installation check

Register AH is non zero if installed a Microsoft Networks compatible network.

10.45.2 Funktion 01h Execute netbios request

This call is equivalent to invoking INT5C, the NETBIOS interrupt.

ES:BX pointer to NCB

Returns:

AL NetBIOS error code

AH 00h if no error, 01h on error

10.45.3 Funktion 02h Set network printer mode

Unknown

10.45.4 Funktion 03h Check direct I/O

This function determines whether direct transfers to a disk are allowed.

DS:SI pointer to ASCIZ disk device name

Returns:

CF clear if direct disk access allowed

10.45.5 Funktion 04h Execute netbios request

Invoke the NETBIOS handler, optionaly retrying the operation on certain errors.

AL 00h for error retry, 01h for no retry

ES:BX pointer to NCB

Returns:

AX 0000h for no error

AH 01h

AL error code

10.45.6 Funktion 05h Get network resource information

Determine the available amounts of several important network resources.

Returns:

AX reserved

BX number of network names available

CX number of commands (NCBs) available

DX number of sessions available

10.45.7 Funktion 06h Network print-stream control

Specify behavior of redirected network printer output.

AL 01h set concatenation mode

AL 02h set truncation mode

AL 03h flush printer output and start new print job

10.45.8 Funktion 2001h unknown

This function is interpreted by DESQview 2.x

10.45.9 Funktion 2002h unknown

This function is called by MSDOS 3.30 APPEND

10.45.10 Funktion 2003h unknown

This function is called by MSDOS 3.30 APPEND

10.45.11 Funktion 80h Begin DOS critical section

Indicate that an uninterruptible region of code is being entered.

AL critical section number (00h - 0Fh)

01h DOS kernel, SHARE.EXE

02h DOS kernel

05h DOS4+ IFSFUNC

06h DOS4+ IFSFUNC

08h ASSIGN.COM

10.45.12 Funktion 81h End DOS critical section

Indicate that an uninterruptible region of code is being left.

AL critical section number (00h - 0Fh)

10.45.13 Funktion 82h End critical section 0 through 7

Clean up any DOS critical section flags which may have been left set by an aborted process or DOS function call.

10.45.14 Funktion 84h Keyboard busy loop

This is a hook to let other work proceed while waiting for keyboard input.

10.45.15 Funktion 87h Critical section

Specify the start or end of a critical section of code.

AL 00h start, 01h end

This function is called by PRINT.COM

10.45.16 Funktion 89h unknown

ASSIGN uses register AL with 08h.

10.45.17 Funktion C2h unknown

This function is called by APPEND.

10.46 Interrupt Nr. 2BH-2DH MS-DOS

The vectors points at an IRET instruction under DOS 2.0 through 4.01.

10.47 Interrupt Nr. 2BH DOS2.7x only

Get DWIM interpreter Parameter. Der **D**o-**W**hat-**I**-**M**ean Interpreter ist ein ladbares Modul, das von COMMAND.COM benutzt wird, wenn ein unbekannter Befehl vorkommt.

10.48 Interrupt Nr. 2EH DOS Execute Command

Batch benutzt diesen Interrupt.

Force COMMAND.COM to execute a command as if it were typed from the keyboard.

DS:SI pointer to counted CR-terminated command string.

10.49 Interrupt Nr. 2EH Change Master Environment

Mit dem Interrupt 2EH kann die Umgebung des Hauptprogramms geändert werden. DDJ Vol.11 P.854

- DS:DI Pointer zu einem command string (bytecount, ASCII-Text, CR (wird nicht gezählt)).

æ

Kapitel 11

MS-DOS Systemaufrufe cont.

11.1 Interrupt Nr. 2FH Multiplex Interrupt

MS-DOS stell mit dem Multiplex Interrupt einen allgemeinen Mechanismus zur Verfügung mit dem ein Programm die Anwesenheit eines TSR feststellen und mit ihm kommunizieren kann. Dazu wird im Register AH der Identifikationswert und im Register AL die Funktion übergeben. Entspricht der ID Wert nicht dem des augenblicklichen TSR, so wird die Anforderung an den Vorgänger übergeben. Die ID Werte 00H bis 7FH sind für MS-DOS vorgesehen. Der Benutzer sollte die Werte 80H bis 0FFH nehmen.

Wenn der Interrupt 2F mit der Funktion 01H gerufen wird, betrifft es den printer spooler. Andere Funktionen werden z. B. von Append benutzt.

11.1.1 Funktion 00h PRINT.COM

Unterfunktion 80h Give PRINT a time slice

Allow PRINT to execute for a while.

11.1.2 Funktion 01H print spooler call

- AL = 00H get installed status
- AL = 01H submit file to be printed
- AL = 02H remove file from print queue
- AL = 03H cancel all files in queue
- AL = 04H hold print jobs for status read
- AL = 05H end hold for status read

If AL = 01H:

- DS:DX = segment:offset of packet address

If AL = 02H:

- DS:DX = segment:offset of ASCII file specification

If operation is succesful: Carry flag is clear.
If AL was 00H on call:

- AL = status:
- 00H not installed, OK to install
- 01H not installed, not OK to install
- FFH installed

If AL was 04H on call:

- DX = error count
- DS:SI = segment:offset of print queue

If operation is not successful: Carry flag is set

- AX = error code:
- 01H function invalid
- 02H file not found
- 03H path not found
- 04H too many open files
- 05H access denied
- 08H queue full
- 09H spooler busy
- 0CH name too long
- 0FH drive invalid

Unterfunktion 01H

Für die Unterfunktion 01H besteht das Paket aus 5 Bytes. Das erste Byte beinhaltet die Ebene (level) und muß zero sein. In den nächsten 4 Byte steht die Doppelwort Adresse (segment:offset) der ASCII file Specification. Es können keine wildcards (*) angegeben werden. Wenn der file existiert, wird er an das Ende der print queue angehängt.

Unterfunktion 02H

Bei der Unterfunktion 02H sind wildcards erlaubt. Damit kann man mehrere files mit einem call aus der printer queue entfernen.

Unterfunktion 04H

Die Adresse in den Registern DS:SI zeigt auf eine Reihe von 64 Byte Einträgen. Jeder enthält die ASCII file Specification. Der erste ist der der gerade gedruckt wird. Der letzte hat ein zero im ersten Byte.

Unterfunktion 06h Check if error on output device

CF set on error

AX error code

DS:SI pointer to device driver header

CF clear if successful

AX 0000h

11.1.3 Funktion 02h PC LAN Program

Unterfunktion 00h Installation check

Determine whether th PC LAN Program redirector is installed.
AL = FFh if installed.

11.1.4 Funktion 05h Critical error handler

Unterfunktion 00h Installation check

Determine whether code to expand an error number into the corresponding error message has been loaded.

AL 00h not installed, OK to install.

AL 01h not installed, can't install.

AL FFh installed.

Unterfunktion XXh Expand error into string

Convert an error number into the corresponding error message.
DOS 3.x
AL = extended error code (not zero)
DOS 4.x

AL error type; 01h DOS extended error code, 02h parameter error.

BX error code

Returns:

ES:DI pointer to ASCIZ error message (read-only).

11.1.5 Funktion 06h ASSIGN

Unterfunktion 00h Installation check

AL = nonzero if installed.

Unterfunktion 01h Get memory segment

Return a pointer to the drive translation table used by ASSIGN.

ES segment of ASSIGN work area and assignment table.

11.1.6 Funktion 08h DRIVER.SYS

Unterfunktion 00h Installation check

Determine whether the DRIVER.SYS support is present.

AL

00h not installed, OK to install

01h not installed, not OK to install

FFh installed

Unterfunktion 01h Add new block device

Add a new logical drive alias for an existing physical drive.

DS:DI pointer to drive data table

Unterfunktion 02h Execute device driver request

Execute the specified device driver request for a drive alias established by INT2F/AX=0802.

ES:BX pointer to device driver request header

Format of device driver request header:

Offset	Size	Description
00h	Byte	length of request header
01h	Byte	subunit within device driver
02h	Byte	command code
03h	Word	status
		bit15: error
		bit9: busy
		bit8: done
		bit7-0: error code if bit15 set
05h	8 Bytes	reserved

Command code 00h

Offset	Size	Description
0Dh	Byte	number of units (set by driver)
0Eh	DWord	address of first free byte following driver
		(set by driver)
12h	DWord	pointer to BPB (set by block drivers only)
16h	Byte	drive number for first unit of block driver (0=A)

Command code 01h

Offset	Size	Description
0Dh	Byte	media descriptor
0Eh	Byte	return status
		00h don't know
		01h media has not changed
		FFh media has benn changed
0Fh	DWord	pointer to previous volume ID if OPEN/CLOSE/RM bit
		in device header
		set and disk changed (set by driver)

Command code 02h

Offset	Size	Description
0Dh	Byte	media descriptor
0Eh	DWord	transfer address
		pointer to scratch sector if NON-IBM FORMAT bit
		in device header set
		pointer to first FAT sector otherwise
12h	DWord	pointer to BPB (set by driver)

Command code 03h,0Ch

Offset	Size	Description
0Dh	Byte	media descriptor (block devices only)
0Eh	DWord	transfer address
12h	Word	byte count (character devices) or sector count
14h	Word	starting sector number (block devices only)

Command code 04h, 08h, 09h

Offset	Size	Description
0Dh	Byte	media descriptor (block devices only)
0Eh	DWord	transfer address
12h	Word	byte count (character devices) or sector count
14h	Word	starting sector number (block devices only)
16h	DWord	pointer to volume ID if error 0Fh returned

Command code 05h

Offset	Size	Description
0Dh	Byte	byte read from device if BUSY bit clear on return

Command code 06h, 07h, 0Ah, 0Bh

Offset	Size	Description
		no further fields

Command code 10h

Offset	Size	Description
0Dh	Byte	unused
0Eh	DWord	transfer address
12h	Word	byte count

Command code 13h

Offset	Size	Description
0Dh	Byte	category code
		00h unknown
		01h COMn:
		03h CON
		05h LPTn:
		08h disk
0Eh	Byte	function code
0Fh	Dword	apparently unused in DOS 3.3
13h	DWord	pointer to parameter block from INT21/AX=440Dh

Values for command code

Value	Description
00h	INIT
01h	MEDIA CHECK
02h	BUILD BPB
03h	IOCTL INPUT
04h	INPUT
05h	NONDESTRUCTIVE INPUT, NO WAIT
06h	INPUT STATUS
07h	INPUT FLUSH
08h	OUTPUT
09h	OUTPUT WITH VERIFY
0Ah	OUTPUT STATUS
0Bh	OUTPUT FLUSH
0Ch	IOCTL OUTPUT
0Dh	DEVICE OPEN
0Eh	DEVICE CLOSE
0Fh	REMOVABLE MEDIA
10h	OUTPUT UNTIL BUSY
13h	GENERIC IOCTL
17h	GET LOGICAL DEVICE
18h	SET LOGICAL DEVICE
80h	(CD-ROM) READ LONG
81h	(CD-ROM) reserved
82h	(CD-ROM) READ LONG PREFETCH
83h	(CD-ROM) SEEK
84h	(CD-ROM) PLAY AUDIO
85h	(CD-ROM) STOP AUDIO
86h	(CD-ROM) WRITE LONG
87h	(CD-ROM) WRITE LONG VERIFY
88h	(CD-ROM) RESUME AUDIO

Values for error code

Value	Description
00h	write protect violation
01h	unknown unit
02h	drive not ready
03h	unknown command
04h	CRC error
05h	bad drive request structure length
06h	seek error
07h	unknown media
08h	sector not found
09h	printer out of paper
0Ah	write fault
0Bh	read fault
0Ch	reneral failure
0Dh	reserved
0Eh	reserved
0Fh	invalid disk change

Unterfunktion 03h Get drive data table list

Return a pointer to the first in a list of drive data tables decribing the layout of the logical drives supported by the combination of the default disk device driver and aliases established with DRIVER.SYS.

DS:DI pointer to first drive table in list

Format of DOS 3.3 drive table:

Offset	Size	Description
00h	DWord	pointer to next table
04h	Byte	physical unit number (for INT13h)
05h	Byte	logical drive number
06h	19 Bytes	BIOS Parameter Block
19h	Byte	unknown
1Ah	Word	number of DEVICE OPEN calls without corresponding DEVICE CLOSE
1Ch	11 Bytes	volume label or "NO NAME" if none
27h	Byte	terminating null for volume label
28h	Byte	device type
29h	Word	bit flags describing drive bit0: fixed media bit1: door lock supported bit2: used in determining BPB to set for INT21/AX=440Dh bit3: all sectors in a track are the same size bit4: physikal drive has multiple logical units bit5: current logical drive for physical drive bit8: related to disk change detection
2Bh	Word	number of cylinders
2Dh	19 Bytes	BPB for highest capacity supported
40h	3 Bytes	unknown
43h	9 Bytes	file system name, default "NO NAME"
4Ch	Byte	terminating null for filesystem type
4Dh	DWord	time of last access in clock ticks (FFFFFFFFh if never) removable media only

Format of BIOS Parameter Block (BPB):

Offset	Size	Description
00h	Word	bytes per sector
02h	Byte	sectors per cluster, FFh if unknown
03h	Word	number of reserved sectors
05h	Byte	number of FATs
06h	Word	number of root dir entries
08h	Word	total sectors
0Ah	Byte	media descriptor, 00h if unknown
0Bh	Word	sectors per FAT
0Dh	Word	sectors per track
0Fh	Word	number of heads
11h	Word	number of hidden sectors

Format of DOS 4.01 drive table:

Offset	Size	Description
00h	DWord	pointer to next table
04h	Byte	physical unit number (for INT13h)
05h	Byte	logical drive number
06h	19 Bytes	BIOS Parameter Block
19h	9 Bytes	unknown
22h	Byte	device type
23h	Word	bit flags describing drive
		bit0: fixed media
		bit1: door lock supported
		bit2: used in determining BPB to set for INT21/AX=440Dh
		bit3: all sectors in a track are the same size
		bit4: physikal drive has multiple logical units
		bit5: current logical drive for physical drive
		bit8: related to disk change detection
25h	Word	number of cylinders
27h	19 Bytes	BPB for highest capacity supported
3Ah	13 Bytes	unknown
47h	DWord	time of last access in clock ticks (FFFFFFFFh if never)
4Bh	11 Bytes	volume label or "NO NAME"
4Ch	Byte	terminating null for volume label
57h	DWord	serial number
5Bh	8 Bytes	filesystem type ("FAT12" or "FAT16")
63h	Byte	terminating null for volume label

11.1.7 Funktion 10h SHARE

Unterfunktion 00h Installation check

Determine whether SHARE has been loaded.

Returns:

AL 00h not installed, OK to install

AL 01h not installed, not OK to install

AL FFh installed

Unterfunktion 40h unknown

Unterfunktion 80h clear unknown flag

Returns: AL F0h function supported.

Unterfunktion 81h set unknown flag

Returns: AL F0h function supported.

11.1.8 Funktion 11h Network Redirector

Ab Version 3.0 von MS-DOS ist die *Current Directory Structure* (CDS) vorhanden. In **CONFIG.SYS** wird mit **LASTDRIVE=** das letzte gewünschte Laufwerk angegeben. Die CDS wird dann ensprechende Einträge aufweisen. Diese Einträge bestehen aus:

1. dem Pfadeintrag zu dem aktuellen Verzeichnis in dem Laufwerk,
2. den Zustandsbits für dieses Laufwerk und
3. den Laufwerksspezifischen Daten.

Der Pfadeintrag unterliegt den üblichen Beschränkungen. Die laufwerksspezifischen Daten sind z. B. Anzahl der Köpfe, Anzahl der Tracs u.s.w. Die Zustandsbits sind in einem Wort am Ende des Pfadeintrages. Es bedeuten:

Bit	Bedeutung
15	Netzwerk
14	physikalisches Laufwerk
13	über join umgelenkt
12	über subst umgelenkt
11 - 0	z.Z. nicht benutzt

Die Kombination der Bits 15 und 14 bedeutet:

Bit15	Bit14	Bedeutung
0	0	ungültiges Laufwerk
0	1	Laufwerk physikalisch vorhanden
1	0	Laufwerk über File Redirektor ansprechen
1	1	Laufwerk über DFS ansprechen (nur DOS4.x)

Die folgende Tabelle zeigt die Unterfunktionen des Interrupt 2Fh Funktion 11h. In der zweiten Spalte sind die entsprechenden Funktionen des Interrupt 21h zu sehen.

01	3A	Verzeichnis löschen.
03	39	Verzeichnis erzeugen
05	3B	Verzeichnis wechseln
06	10, 3E	Datei schließen
08	14, 21, 3F	Datei lesen
09	15, 22, 40	Datei beschreiben
13	13, 41	Datei löschen
16	0F, 3D	Datei öffnen
17	16, 3C	Datei erzeugen
1B	11, 4E	erste Datei suchen
1C	12, 4F	nächste Datei suchen

Unterfunktion 00h Installation check

Determine whether a network redirctor using the DOS kernel network hooks is installed.

Returns:

AL 00h not installed, OK to install

AL 01h not installed, not OK to install

AL FFh installed

- This function is called by the DOS 3.1+ kernel.
- In DOS 4+, the 11xx calls are all in IFSFUNC.EXE, not in the PC LAN Program redirector.

Unterfunktion 01h Remove remote directory

Remove a directory on a network or installable file system drive.

Called with	
SS SDA first filename pointer SDA first filename buffer SDA CDS pointer	set to DOS CS offset of SDA first filename buffer fully qualified directory name pointer to CDS for drive with dir
Returns	
CF set on error	AX DOS error code

This function is called by the DOS 3.1+ kernel.

Unterfunktion 02h Remove remote directory

Remove a directory on a network or installable file system drive.

Called with	
SS SDA first filename pointer SDA first filename buffer SDA CDS pointer	set to DOS CS offset of SDA first filename buffer fully qualified directory name pointer to CDS for drive with dir
Returns	
CF set on error	AX DOS error code

This function appears to be identical to INT2D/AX=1101h.

Unterfunktion 03h Make remote directory

Create a new directory on a network or installable file system drive.

Called with	
SS SDA first filename pointer SDA CDS pointer	set to DOS CS offset to fully-qualified directory name pointer to CDS for drive with dir
Returns	
CF set on error	AX DOS error code

This function is called by the DOS 3.1+ kernel.

Unterfunktion 04h Make remote directory

Create a new directory on a network or installable file system drive.

Called with	
SS SDA first filename pointer SDA CDS pointer	set to DOS CS offset to fully-qualified directory name pointer to CDS for drive with dir
Returns	
CF set on error	AX DOS error code

This function appears to be identical to INT2D/AX=1103h.

Unterfunktion 05h Chdir

Change the current directory on a network or installable file system drive.

Called with	
SS SDA first filename pointer SDA first filename buffer SDA CDS pointer	set to DOS CS offset of SDA first filename buffer fully-qualified directory name pointer to CDS for drive with dir
Returns	
CF set on error	AX DOS error code

This function is called by the DOS 3.1+ kernel.

Unterfunktion 06h Close remote file

Close a file which was opened on a network or installable file system drive.

Called with	
ES:DI SFT DPB field	pointer to SFT DPB of drive with file
Returns	
CF set on error CF clear if successful	AX DOS error code

This function is called by the DOS 3.1+ kernel.

Unterfunktion 07h Commit remote file

Update the directory entry and flush disk buffers for a file on a network or installable file system drive.

Called with	
ES:DI SFT DPB field	pointer to SFT DPB of drive with file
Returns	
CF set on error CF clear if successful	AX DOS error code

This function is called by the DOS 3.1+ kernel.

Unterfunktion 08h Read from remote file

Read data from a file opened on a network or installable file system drive.

Called with	
ES:DI SFT DPB field CX SS SDA DTA field	pointer to SFT DPB of drive with file number of bytes set to DOS CS pointer to buffer containing data
Returns	
CF set on error CF clear if successful	AX DOS error code CX number of bytes read

This function is called by the DOS 3.1+ kernel.

Unterfunktion 09h Write to remote file

Write data to file opened on a network or installable file system drive.

Called with	
ES:DI	pointer to SFT
SFT DPB field	DPB of drive with file
CX	number of bytes
SS	set to DOS CS
SDA DTA field	pointer to buffer for data
Returns	
CF set on error	AX DOS error code
CF clear if successful	CX number of bytes written

This function is called by the DOS 3.1+ kernel.

Unterfunktion 0Ah Lock region of file

Request that no other processes be allowed access to a portion of the specified file.

Called with	
BX	file handle
CX:DX	starting offset
SI	high word of size
STACK	Word low word of size
ES:DI	pointer to SFT
SFT DPB field	DPB of drive with file
SS	set to DOS CS
Returns	
CF set on error	AL DOS error code
STACK	unchanged

This function is called by the DOS 3.1+ kernel.

Unterfunktion 0Bh Unlock region of file

Allow other processes to access the specified portion of the file.

Called with	
BX	file handle
CX:DX	starting offset
SI	high word of size
STACK	Word low word of size
ES:DI	pointer to SFT for file
SFT DPB field	DPB of drive with file
Returns	
CF set on error	AL DOS error code
STACK	unchanged

This function is called by the DOS 3.1+ kernel.

Unterfunktion 0Ch Get disk space

Get information on allocation size and disk space (free and total) for a network or installable file system drive.

Called with	
ES:DI	pointer to CDS for desired drive.
Returns	
AL	sectors per cluster
AH	unknown
BX	total clusters
CX	bytes per sector
DX	number of available clusters

This function is called by the DOS 3.1+ kernel.

Unterfunktion 0Dh Unknown

The purpose of this function is not known, but appears to be related to file attributes.

Called with	
SDA first filename pointer	pointer to filename
Returns	
unknown	

This function appears to be identical to INT2D/AX=110Fh.

Unterfunktion 0Eh Set remote file's attribute

Change the attributes of a file on a network or installable file system drive.

Called with	
SS	set to DOS CS
SDA first filename pointer	offset of SDA first filename buffer
SDA first filename buffer	fully-qualified file name
SDA CDS pointer	CDS for drive with file
STACK	Word new file attributes
Returns	
CF set on error	AX DOS error code
CF clear if successful	
STACK	unchanged

This function is called by the DOS 3.1+ kernel.

Unterfunktion 0Fh Get remote file's attribute

Get the attributes of a file on a network or installable file system drive.

Called with	
SS	set to DOS CS
SDA first filename pointer	offset of SDA first filename buffer
SDA first filename buffer	fully-qualified file name
SDA CDS pointer	CDS for drive with file
Returns	
CF set on error	AX DOS error code
CF clear if successful	AX file attributes

This function is called by the DOS 3.1+ kernel.

Unterfunktion 10h Unknown

The purpose of this functions is not known, but appears to be related to file attributes.

Called with	
SDA first filename pointer	fully-qualified file name
Returns	
unknown	

This function appears to be identical to INT2D/AX=110Eh.

Unterfunktion 11h Rename remote file

Change the name of a file on a network or installable file system drive.

Called with	
SS	set to DOS CS
DS	set to DOS CS
SDA first filename pointer	offset of SDA first filename buffer
SDA first filename buffer	fully-qualified file name
SDA second filename pointer	offset of SDA new filename buffer
SDA second filename buffer	fully-qualified new file name
SDA CDS pointer	CDS for drive with file
Returns	
CF set on error	AX DOS error code
CF clear if successful	

This function is called by the DOS 3.1+ kernel.

Unterfunktion 12h Unknown

The purpose of this function is not known.

Called with	
SS	set to DOS CS
DS	set to DOS CS
SDA first filename pointer	fully-qualified file name
Returns	
unknown	

Unterfunktion 13h Delete remote file

Remove a file from a network or installable file system drive.

Called with	
SS	set to DOS CS
DS	set to DOS CS
SDA first filename pointer	offset of SDA first filename buffer
SDA first filename buffer	fully-qualified file name
SDA CDS pointer	CDS for drive with file
Returns	
CF set on error	AX DOS error code
CF clear if successful	

This function is called by the DOS 3.1+ kernel.

Unterfunktion 14h Unknown

The purpose of this function is not known.

Called with	
SDA first filename pointer	fully-qualified file name
Returns	
unknown	

Unterfunktion 15h Unknown

The purpose of this function is not known.

Called with	
SS	set to DOS CS
ES:DI	pointer to SFT
Returns	
unknown	

Unterfunktion 16h Open existing remote file

Prepare for access to an existing file located on a network drive.

Called with	
SDA first filename pointer	offset of SDA first filename buffer
SDA first filename buffer	fully-qualified file name
ES:DI	pointer to uninitialized SFT
SS	set to DOS CS
STACKWORD	file open mode
Returns	
CF set on error	AX DOS error code
CF clear if successful	SFT completed
STACK unchanged	

This function is called by the DOS 3.1+ kernel.

Unterfunktion 17h Create/truncate remote file

Create a file on a network drive, or truncate an existing file to zero length.

Called with	
ES:DI	pointer to uninitialized SFT
SS	set to DOS CS
SDA first filename pointer	offset of SDA first filename buffer
SDA first filename buffer	fully-qualified file name
SDA CDS pointer	CDS for drive with file
STACK	WORD file creation mode
Returns	
CF set on error	AX DOS error code
CF clear if successful	SFT completed
STACK unchanged	

This function is called by the DOS 3.1+ kernel.

INT2F/AX=1118h appears to be identical in operation, exept that it is called for drives which do not have a current directory structure (SDA CDS pointer has offset FFFFh).

Unterfunktion 18h Create/truncate file

Create a file on a drive which does not have a current directory structure.

Called with	
ES:DI SS SDA first filename pointer STACK	pointer to uninitialized SFT set to DOS CS fully-qualified file name WORD file creation mode
Returns unknown STACK unchanged	

Unterfunktion 19h Unknown

The purpose of this function is not known.
This function is called by the DOS 3.1+ kernel.
DOS 4.0 IFSFUNC returns with CF set and AX=0003h.

Unterfunktion 1Ah Unknown

The purpose of this function is not known.
DOS 4.0 IFSFUNC returns with CF set and AX=0003h.

Unterfunktion 1Bh Findfirst

Begin a directory search on a network or installable file system drive.

Called with	
SS DS SDA search data block SDA first filename pointer SDA first filename buffer SDA CDS pointer	set to DOS CS set to DOS CS uninitialized 21-byte findfirst search data offset of SDA first filename buffer fully qualified file name CDS for drive with file
Returns CF set on error CF clear if successful SDA search data block completed SDA file found field:	 AX DOS error code (bit 7 of first byte must be set) standard directory entry for file

This function is called by the DOS 3.1+ kernel.

Unterfunktion 1Ch Findnext

Continue a directory search on a network or installable file system drive.

Called with	
SS DS SDA search data block	set to DOS CS set to DOS CS findfirst search data from initial INT2F/AX=111Bh
Returns CF set on error CF clear if successful SDA search data block completed SDA file found field:	 AX DOS error code (bit 7 of first byte must be set) standard directory entry for file

Unterfunktion 1Dh Close all remote files for process

Called with	
SS	set to DOS CS
Returns unknown	

This function is called by the DOS 3.1+ kernel, and closes all FCBs opened by the process (among other actions).

Unterfunktion 1Eh Do redirection

Various subfunctions allow control of network redirection.

Called with		
SS STACK	set to DOS CS WORD	 function to execute
	5F00h BL returns BH	get redirection mode type (03h printer, 04h disk) state (00h off, 01h on)
	5F01h BL returns BH	set redirection mode type (03h printer, 04h disk) state (00h off, 01h on)
	5F02h BX DS:SI ES:DI	get redirection list entry redirection list index pointer to 16-byte local device name buffer pointer to 128-byte network name buffer
	5F03h BL CX DS:SI ES:DI	redirect device device type (see INT21/AX=5F03h) stored parameter value pointer to ASCIZ source device name pointer to destination ASCIZ network path + ASCIZ password
	5F04h DS:SI	cancel redirection pointer to ASCIZ device name or network path
	5F05h BX DS:SI ES:DI	get redirection list extended entry redirection list index pointer to buffer for ASCIZ source device name pointer to buffer for destination ASCIZ network path
	returns BH BL CX BP	 status flag type (03h printer, 04h disk) stored parameter value NETBIOS local session number
	5F06h	appears to be similar to 5F05h
Returns: CF set on error	 AX DOS error code	 STACK unchanged

This function is called by the DOS 3.1+ kernel.

Unterfunktion 1Fh Printer setup

Subfunctions allow getting or setting the network printer setup string or mode.

Called with		
STACK	WORD	function
	5E02h	set printer setup
	5E03h	get printer setup
	5E04h	set printer mode
	5E05h	get printer mode
Returns:		
CF set on error	AX DOS error code	STACK unchanged

This function is called by the DOS 3.1+ kernel.

Unterfunktion 20h Flush all disk buffers

Force an immediate update of the network or installable file system drives drom disk buffers which have not yet been written out.

Called with DS	set to DOS CS
Returns CF clear	successful

This function is called by the DOS 3.1+ kernel.

The current directory structure array pointer and LASTDRIVE= entries of the DOS list of lists are used by the DOS 4 IFSFUNC handler for this call.

Unterfunktion 21h Seek from end of remote file

Called with CX:DX ES:DI SFT DPB field SS	 offset (in bytes) from end pointer to SFT pointer to DPB of drive containing file set to DOS CS
Returns: CF set on error CF clear if successfuk	 AX DOS error code DX:AX new file position

This function is called by the DOS 3.1+ kernel.

Unterfunktion 22h Process termination hook

Inform the network that a process has terminated.

Called with SS	set to DOS CS

This function is called by the DOS 3.1+ kernel.

Unterfunktion 23h Qualify remote filename

Convert a name into a absolute pathname with any network redirections resolved.

Called with DS:SI ES:DI	 pointer to ASCIZ filename to canonicalize pointer to 128-byte buffer for qualified name
Returns: CF set if not resolved	

Unterfunktion 24h Unknown

The purpose of this function is not known.

Called with ES:DI SS	 pointer to SFT set to DOS CS

This function is called by the DOS 3.1+ kernel.

Unterfunktion 25h Redirected printer mode

Set or determine the stata of print streams for the network printer.

Called with STACK	 WORD subfunction
5D07h Returns:	get print stream state DL current state
5D08h	set print stream state DL new state
5D09h	finish print job
Returns: CF set on error STACK	 AX error code unchnged

This function is called by the DOS 3.1+ kernel.

Unterfunktion 26h Unknown

The purpose of this function is not known.

This function is called by the DOS 3.1+ kernel.

Unterfunktion 27h Unused

This function performs no action and returns immediatly.

Unterfunktion 28h Unused

This function performs no action and returns immediatly.

Unterfunktion 29h Unused

This function performs no action and returns immediatly.

Unterfunktion 2Ah Unknown

The purpose of this function is not known.

Unterfunktion 2Bh Unknown

The purpose of this function is not known.

This function is called by the DOS 4.0 kernel.

Unterfunktion 2Ch Unknown

The purpose of this function is not known.

Unterfunktion 2Dh Unknown

The purpose of this function is not known.
This function is called by the DOS 4.0 kernel.

Unterfunktion 2Eh Unknown

The purpose of this function is not known.
This function is called by the DOS 4.0 kernel.

Unterfunktion 2Fh Unknown

The purpose of this function is not known.
This function is called by the DOS 4.0 kernel.

Unterfunktion 30h Get ifsfunc segment

Return the segment of the resident IFSFUNC code.

Returns:	
ES	CS of resident IFSFUNC

11.1.9 Funktion 12h DOS internal services

Unterfunktion 00h Installation check

Determine whether the DOS internal services are present.
Returns FFh in AL for compatibility with other INT2F functions.

Unterfunktion 01h Close current file

Close the file currently being operated on.

Call with:	
SS	set to DOS CS
SDA current SFT pointer	pointer to SFT for file to be closed
Returns:	
ES:DI	pointer to SFT for file

Unterfunktion 02h Get interrupt address

Return a pointer to the interrupt vector corresponding to the given interrupt number.

Call with:	
STACK	WORD vector number
Returns:	
ES:BX	pointer to interrupt vector
STACK	unchanged

Unterfunktion 03h Get DOS data segment

Return the segment of IBMDOS.
Returns in DS segment of IBMDOS.

Unterfunktion 04h Normalize path separator

Convert forward slashes into backslashes.

Call with:	
STACK	WORD character to normalize
Returns:	
AL	normalized character
ZF	set if path separator
STACK	unchanged

Unterfunktion 05h Output character

Send a single character to the standard output of the current process.

Call with:	
STACK	WORD character to output
Returns:	
STACK	unchanged

This function can be called only within a DOS function call.

Unterfunktion 06h Invoke critical error

Call with:	
DI	error code
BP:SI	pointer to device driver header
SS	set to DOS CS
STACK	WORD value to be passed to INT24h in AX
Returns:	
AL	0-3 for Abort, Retry, Ignore, Fail
STACK	unchanged

Unterfunktion 07h Make disk buffer most-recently used

Move specified disk buffer to the end of the disk buffer list (which is kept in reverse order of recency of use).

Call with:	
DS:DI	pointer to disk buffer
Returns:	
	buffer moved to end of list

This function can be called only within a DOS function call.

Unterfunktion 08h Decrement SFT reference count

educe the number of references to the given System File Table by one, setting the count to -1 if there are now no more references to the SFT. Used by network redirectors such as MSCDEX.

Call with:	
ES:DI	pointer to SFT
Returns:	
AX	old value of reference count (before decrement).

The reference count is set to FFFFh to indicate no reference, since 0 indicates that the SFT is not in use. In this case, the caller should set the count to zero after cleaning up.

Unterfunktion 09h Flush and free disk buffer

Force the given disk buffer's contents to disk if it is dirty, and than mark the buffer unused.

Call with:	
DS:DI	pointer to disk buffer

This function can be called only within a DOS function call.

Unterfunktion 0Ah Perform critical error interrupt

Invoke an INT24h, passing the appropriate values for the current drive and operation (stored in the SDA). Used by network redirectors such as MSCDEX.

Call with:	
DS	set to DOS CS
SS	set to DOS CS
STACK	WORD extended error code
Returns:	
AL	user response (00h=ignore, 01h=retry, 02h=abort, 03h=fail)
CF	clear if retry, set otherwise
STACK	unchanged

This function can be called only within a DOS function call, as it uses various fields in the SDA.

Unterfunktion 0Bh Signal sharing violation to user

Produce a critical error interrupt if an attempt was made to open file previously opened in compatibility mode with inheritance allowed.

Call with:	
ES:DI	pointer to system file table entry
DS	set to DOS CS
SS	set to DOS CS
STACK	WORD extended error code
Returns:	
CF set	if no INT24h generated or
	user did not say to retry operation
	AX error code (20h)
CF clear	if user said to retry operation
STACK	unchanged

This function can be called only within a DOS function call.

Unterfunktion 0Ch Set FCB file's owner

Apparently sets the owner of the last-accessed FCB file to the calling process's ID. Used by network redirectors such as MSCDEX.

Call with:	
ES:DI	pointer to SFT for file

Unterfunktion 0Dh Get date and time

Return the current date and time in directory format.

Call with:	
DS	set to DOS CS
Returns:	
AX	current date in packed format
DX	current time in packed format

Unterfunktion 0Eh Mark all disk buffers unreferenced

Clear the "referenced" flag on all disk buffers. This flag is automatically set when a buffer is read or written, and is used in the buffer replacement algorithm. Unreferenced buffers are generally replaced before referenced buffers.

Call with:	
DS	set to DOS CS
Returns:	
DS:DI	pointer to first disk buffer

Unterfunktion 0Fh Make buffer most recently used

Move the specified disk buffer to the end of the buffer chain without flushing it to disk if it is dirty.

Call with:	
DS:DI	pointer to disk buffer
SS	set to DOS CS
Returns:	
DS:DI	pointer to next disk buffer in buffer list

Unterfunktion 10h Find unreferenced disk buffer

Return a pointer to the least-recently used disk buffer which has not been referenced since its first use or the last "mark aal disk buffers unreferenced" call.

Call with:	
DS:DI	pointer to first disk buffer to check
Returns:	
ZF	set if not found (all buffers have been referenced
ZF	clear if found
DS:DI	pointer to next disk buffer in buffer list

Unterfunktion 11h Normalize ASCIZ filename

Copy the given filename, converting it to uppercase and changing forward slashes into backslashes.

Call with:	
DS:SI	pointer to ASCIZ filename to normalize
ES:DI	pointer to buffer for normalized filename
Returns:	
ES:DI	buffer filled

Unterfunktion 12h Get length of ASCIZ string

Return the length of a null-terminated character string.

Call with:	
ES:DI	pointer to ASCIZ string
Returns:	
CX	length of string

Unterfunktion 13h Uppercase character

Return the uppercase equivalent, using the current country's capitalization rules, of the given character.

Call with:	
STACK	WORD character to convert to uppercase
Returns:	
AL	uppercased character
STACK	unchanged

Unterfunktion 14h Compare pointers

Determine whether two FAR pointers are identical.

Call with:	
DS:SI	first pointer
ES:DI	second pointer
Returns:	
ZF	set if pointers are equal
ZF	clear if not equal

Unterfunktion 15h Flush buffer

Force the contents of the specified disk buffer to be written to disk if it is dirty.

Call with:	
DS:DI	pointer to disk buffer
SS	set to DOS CS
STACK	WORD drives for which to skip buffer
Returns:	
STACK	unchanged

Unterfunktion 16h Get address of system file table

Return the address of a system file table entry given its number.

Call with:	
BX	system file table entry number
Returns:	
CF	clear if succesful
ES:DI	pointer to file table entry
CF	set if BX greater than FILES=

Unterfunktion 17h Set working drive

Call with:	
SS	set to DOS CS
STACK	WORD drive (0=A; 1=B)
Returns:	
CF	set on error (drive > LASTDRUVE)
CF	clear if successful
DS:SI	pointer to CDS for specified drive
STACK	unchanged

Unterfunktion 18h Get caller's registers

Return a pointer to the stack frame containing the INT21h caller's registers.

Call with:	
Returns:	
DS:SI	pointer to saved caller's AX,BX,CX,DX,SI,DI,BP,DS,ES (ON STACK)

Unterfunktion 19h Set drive

Call with:	
SS	set to DOS CS
STACK	WORD drive (0=default, 1=A)
Returns:	
STACK	unchanged

Unterfunktion 1Ah Get file's drive

Determine which drive a file specifies.

Call with:	
DS:SI	pointer to filename
Returns:	
AL	drive (0=default, 1=A)

Unterfunktion 1Bh Set year/length of february

Specify the current year, and return the length of February in days after storing that length internally.

Call with:	
CL	year - 1980
DS	set to DOS CS
Returns:	
AL	number of days in February

Unterfunktion 1Ch Checksum memory

Compute a checksum of the given range of memory.

Call with:	
DS:SI	pointer to start of memory to checksum
CX	number of bytes
DX	initial checksum
Returns:	
DX	checksum
DS:SI	points beyond checksummed range
AX, CX	destroyed

Unterfunktion 1Dh Sum memory

ADD up the values of a range of bytes until the specified limit is exceeded, and return the value which caused the limit to be exceeded.

Call with:	
DS:SI	pointer to memory to add up
CX	000h
DX	limit
Returns:	
AL	byte which exceeded limit
CX	number of bytes before limit exceeded
DX	remainder after adding first CX bytes
DS:SI	points at byte beyond the one which exceeded the limit

Unterfunktion 1Eh Compare filenames

Determine whether two filenames are identical except for case and forward/backslash differences.

Call with:	
DS:SI	pointer to first ASCIZ filename
ES:DI	pointer to second ASCIZ filename
Returns:	
ZF	set if filenames equivalent
ZF	clear if not

Unterfunktion 1Fh Build current directory structure

Create a new Current Directory Structure for the spezified drive, and return the address of the temporary storage in which it was built.

Call with:	
SS	set to DOS CS
STACK	WORD drive letter
Returns:	
ES:DI	pointer to current directory structure
STACK	unchanged

Unterfunktion 20h Get job file table entry

Given a file handle, return the address of the entry in the Job File Table for that handle in the current process.

Call with:	
BX	file handle
Returns:	
CF	set on error
AL	6 (invalid file handle)
CF	clear if successful
ES:DI	pointer to JFT entry

Unterfunktion 21h Canonicalize file name

Given a file specification, return an absolute pathname which takes into account any renaming due to JOIN, SUBST, ASSIGN, or network redirections.

Call with:	
DS:SI	pointer to file name to be fully qualified
ES:DI	pointer to 128-byte buffer for resulting canonical file name
SS	set to DOS CS
Returns:	
see	INT21/60h

Unterfunktion 22h Set extended error info

Given a set of translation records, set the error class, locus, and suggested action corresponding to the current extended error code.

Call with:	
SDA	error code field set
SS	set to DOS CS
SS:SI	pointer to 4-byte records
	BYTE error code, FFh if last record
	BYTE error class, FFh means don't change
	BYTE suggested action, FFh means don't change
	BYTE error locus, FFh means don't change
Returns:	
SI	destroyed
SDA	error class, error locus, and suggested action fields set

Unterfunktion 23h Check if character device

Determine whether the given name is the name of a character device.

Call with:	
DS	set to DOS CS
SS	set to DOS CS
SDA	eight-character blank-padded name
+218h	DOS 3.10 - 3.30
+22Bh	DOS 4.0x
Returns:	
CF	set if no character device by that name found
CF	clear if found
BH	low byte of device attribute word

Unterfunktion 24h Delay

Perform a sharing retry delay loop.

Call with:	
SS	set to DOS CS
Returns:	
after delay	set by INT21/440B

Unterfunktion 25h Get length of ASCIZ string

Return the length of a null-terminated character string.

Call with:	
DS:SI	pointer to ASCIZ string
Returns:	
CX	length of string

Unterfunktion 26h Open file

Open an existing file with the specified access mode and return a file handle if successful.

Call with:	
CL	access mode
DS:DX	pointer to ASCIZ filename
SS	set to DOS CS
Returns:	
CF	set on error
AL	error code
CF	clear if successful
AX	file handle

Unterfunktion 27h Close file

Close a previously-opened file given its handle.

Call with:	
BX	file handle
SS	set to DOS CS
Returns:	
CF	set on error
AL	6 (invalid file handle)

Unterfunktion 28h Move file pointer

Set the current position in the given file.

Call with:	
BP	4200, 4201, 4202 (see INT21/42)
BX	file handle
CX:DX	offset in bytes
SS	set to DOS CS
Returns:	
CF	set on error
AX	error code (01, 06)
CF	clear if successful
DX:AX	new absolute offset from beginning of file

Unterfunktion 29h Read from file

Read data from a previously-opened file.

Call with:	
BX	file handle
CX	number of bytes to read
DS:DX	pointer to buffer
SS	set to DOS CS
Returns:	
as for	INT21/3F

Unterfunktion 2Ah Set fastopen entry point

Specify the address(es) of the handlers for the FASTOPEN filename cache.

Call with:	
BX	entry point to set (0001 or 0002)
DS:SI	pointer to FASTOPEN entry point
	entry point not set if SI=FFFFh for DOS 4+
Returns:	
CF	set if specified entry point already set

Unterfunktion 2Bh IOCTL

Execute an I/O control function from within network redirector.

Call with:	
BP	44xx
SS	set to DOS CS
additional	registers as for INT21/44xx
Returns:	
as for	INT21/44

Unterfunktion 2Ch Get device chain

Return a pointer to the device driver chain (omitting the NUL device).

Call with:	
Returns:	
BX:AX	pointer to header of second device driver (NUL is first) in driver chain.

Unterfunktion 2Dh Get extended error code

Return the current extended error code.

Call with:	
Returns:	
AX	current extended error code

Unterfunktion 2Eh Get or set error table address

Specify or determine the locations of various tables used to convert error numbers intp error messages.

Call with:	
DL = 00	get standard DOS error table (00 - 12, 50 - 5B)
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 01	set standard DOS error table
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 02	get parameter error table (00 - 0A)
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 03	set parameter error table (00 - 0A)
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 04	get critical/SHARE error table (13 - 2B)
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 05	set critical/SHARE error table (13 - 2B)
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 06	get unknown error table
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 07	set unknown error table
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 08	get unknown error table
Returns:	
ES:DI	pointer to error table
Call with:	
DL = 09	set unknown error table
Returns:	
ES:DI	pointer to error table

Format of error table:

Offset	Size	Description
00h	Byte	FFh
01h	2 Bytes	04h,00h (may be DOS version)
03h	Byte	number of error headers following
04h	2N Words	table of all headers for table

Format of header table:

Offset	Size	Description
00h	WORD	error message number
02h	WORD	offset of error message from start of header

Error messages are count byte followed by message.

Unterfunktion 2Fh Set unknown

Call with:	
Returns:	

11.1.10 Funktion 13h Set disk interrupt handler

Specify the address of the handler for most DOS disk access, and return the old hendler's address.

Call with:	
DS:DX	pointer to interrupt handler disk driver calls on read/write
ES:BX	address to restore INT13 to on system halt
Returns:	
DS:DX	from previous invocation of this function
ES:BX	from previous invocation of this function

11.1.11 Funktion 14h NLSFUNC.COM

Unterfunktion 00h installation check

Determine whether NLSFUNC has been loaded.

Call with:	
Returns:	
AL = 00	not installed, OK to install
AL = 01	not installed, not OK
AL = FF	installed

Unterfunktion 01h change code page

Select a new code page as the default.

Call with:	
DS:SI	pointer to internal code page structure
BX	new code page
DX	country code
Returns:	
AL	status

Format of DOS 3.30 internal code page structure:

Offset	Size	Description
00h	8 Bytes	unknown
08h	64 Bytes	name of country information file
48h	WORD	system code page
4Ah	WORD	number of supported subfunctions
4Ch	5 Bytes	data to return for INT21/6502h
51h	5 Bytes	data to return for INT21/6504h
56h	5 Bytes	data to return for INT21/6505h
5Bh	5 Bytes	data to return for INT21/6506h
60h	5 Bytes	data to return for INT21/6501h

Unterfunktion 02h get country info

Get country-specific information for a country or code page other than the default.

Call with:	
BP	subfunction (see INT21/65)
BX	code page
DX	country code
DS:SI	pointer to internal code page structure (see INT2F1401)
ES:DI	pointer to user buffer
CX	size of user buffer
Returns:	
AL	status

Unterfunktion 03h set country info

Select a new country code as the default.

Call with:	
DS:SI	pointer to internal code page structure (see INT2F1401)
BX	code page
DX	country code
Returns:	
AL	status

Unterfunktion 04h get country info

Return country-specific information for a country other than the current default.

Call with:	
BX	code page
DX	country code
DS:SI	pointer to internal code page structure (see INT2F1401)
ES:DI	pointer to user buffer
Returns:	
AL	status

11.1.12 Funktion 15H MSCDEX

Microsoft's CD-ROM extension Software für CD-ROM Laufwerke. Benutzt die Funktion 11h des Interrupt 2Fh.

Unterfunktion 00H Installation check

Although documented, this function is included because it conflicts with the GRAPHICS.COM installation check.

//this installation check **DOES NOT** folow the format used by other software.

Die Unterfunktion 00H gibt in dem Register BX die Zahl des Drives (3=D), die CD-ROM Devices zugeordnet ist, zurück.

Call with:	
BX	0000h
Returns:	
BX	number of CDROM drive letters used
CX	starting drive letter (0 = A)

Unterfunktion 0BH

Mit der Unterfunktion 0BH kann festgestellt werden, ob ein Drive ein CD-ROM Drive ist.

Unterfunktion 0DH

Mit der Unterfunktion 0DH kann man eine Liste der CD-ROM Laufwerksbuchstaben bekommen.

11.1.13 Funktion 16 undocumented

11.1.14 Funktion 1686H DPMI-Server

Programme können hiermit ermitteln, ob sie im Protected-Mode oder im Real-Mode arbeiten.

Ein: —

Aus: AX = Flag (0 = im Protected-Mode, sonst im Real-Mode oder kein DPMI vorhanden).

11.1.15 Funktion 1687H DPMI-Server

Diese Funktion kann der Client im Real-Mode (bzw. virtuellen 8086-Mode) aufrufen, um die Adresse zu ermitteln, mit der er in den Protected-Mode wechseln kann. Für die Überprüfung der DPMI-Anwesenheit nur eine echte INT-Instruktion benutzen! Gibt der Server im Register SI als Größe 0 zurück, so benötigt er keinen explizit reservierten Speicherplatz für die Verwaltung der Task.

Ein: —

Aus: AX = 0 wenn DPMI installiert ist.

BX = Flags, Bit0 = 1 wenn 32-Bit Applikationen unterstützt werden.

CL = verwendeter Prozessor (2 = 286, 3 = 386, 4 = 486).

DH = Hauptversion

DL = Unterversion

SI = Größe des vom DPMI-Server für jede Task benötigten Speicherblocks in Paragraphen.

ES:DI = Einsprungadresse für den Wechsel

Die zuvor erfragte Adresse muß der Client anspringen, um in den Protected-Mode zu wechseln. Der Client darf danach die Deskriptoren für das Code-, Daten- sowie Stacksegment freigeben, wenn er sie nicht mehr braucht. Den Selektor des PSPs darf er jedoch nicht antasten. Will er das Environment freigeben, so muß dieses vor dem Eintritt in den Protected-Mode geschehen. Galt weiterhin vor dem Aufruf dieser Funktion DS = SS, reserviert der Server nur einen Selektor für beide Segmente.

Ein: AX = Flags (Bit0 = 1 wenn als 32-Bit Applikation ausgeführt werden soll).

ES = Segmentadresse eines Speicherblocks, den der DPMI-Server für Verwaltungsaufgaben benötigt. Danach die zuvor erfragte Adresse anspringen.

Aus: Carry-Flag gesetzt: es ist ein Fehler aufgetreten und das Programm befindet sich weiterhin im Real-Mode.

Carry-Flag nicht gesetzt:

CS = Selektor auf ein Segment, das mit dem Real-Mode Codesegment übereinstimmt.

DS = dto. für Real-Mode DS SS = dto. für Real-Mode SS

ES = Selektor auf das Programm Segment Prefix

FS sowie GS = 0, falls ein 386/486 benutzt wird.

11.1.16 Funktion 19H SHELLB.COM

Diese Angaben sind nur für DOS 4.x gültig!

Unterfunktion 00H Installation check

Determine wether SHELLB has been loaded.

Call with:	
Returns:	
AL = 00	not installed
AL = FF	installed

Unterfunktion 01H SHELLB.COM - SHELLC.EXE interface

Inform SHELLB of SHELLC's address, and return the location of a workspace for SHELLC.

Call with:	
BL	00 if SHELLC transient
	01 if SHELLC resident
DS:DX	pointer to far call entry point for resident SHELLC.EXE
Returns:	
ES:DI	pointer to SHELLC.EXE workspace within SHELLB.COM

Unterfunktion 02H SHELLB.COM - COMMAND.COM interface

Get the next line which COMMAND.COM should execute in preference to reading a command from the current batch file.

Call with:	
ES:DI	pointer to ASCIZ full filename of current batch file, with at least the final filename element uppercased.
DS:DX	pointer to buffer for results
Returns:	
AL = 00	failed
AL = FF	success

Unterfunktion 03H SHELLB.COM - COMMAND.COM

Determine whether Programm Start Command is attempting to re-execute the current batch file.

Call with:	
ES:DI	pointer to ASCIZ batch file name as for INT2F/1902
Returns:	
AL = FF	if quoted batch filename matches last SHELLB parameter
AL = 00	if it does not

Unterfunktion 04H SHELLB.COM transient to TSR interface

Determine the name of the batch file from which the DOS shell is executing.

Call with:	
Returns:	
ES:DI	pointer to name of current shell batch file
	WORD number of bytes of name following
	BYTEs (8 max) uppercase name of shell batch file

11.1.17 Funktion 1AH ANSI.SYS

Unterfunktion 00H installation check

Determine whether ANSI.SYS is present. Returns AL = FFh if installed.

Unterfunktion 01H get/set display information

This is presumable the DOS IOCTL interface to ANSI.SYS.

Call with:	
CL	7Fh for GET
	5Fh for SET
DS:DX	pointer to parm blocks as for INT21/440C, CX=037Fh/035Fh respectively
Returns:	
CF	set on error
AX	error code (many nonstandard
CF	clear if successful
AX	destroid

Unterfunktion 02H misc request

Get or set miscellaneous ANSI.SYS flags.

Call with:	
DS:DX	pointer to parameter block
Returns:	

Format of parameter block:

Offset	Size	Description
00	BYTE	subfunction
		00 set/reset interlock
		01 get /L flag
01	Byte	interlock state
		00 reset
		01 set
02	BYTE	(returned)
		00 if /L not in effect
		01 if /L in effect

This interlock prevents some of the ANSI.SYS post-processing in its hook onto INT10 function 00 mode set.

11.1.18 Funktion 1BH XMA2EMS.SYS

Unterfunktion 00H installation check

Determine whether XMA2EMS.SYS has been loaded. Returns AL = FFh if installed.

Unterfunktion 01H get hidden frame info

Determine information which XMA2EMS.SYS hides from regular EMS function calls.

Call with:	
DI	hidden physical page number
Returns:	
AX	FFFFh if failed (no such hidden page)
AX	0000 if OK, then
	ES segment of page frame
	DI physical page number

Unterfunktion FFH unknown

The purpose of this function is not known, but it appears likely to be the same as INT2F/1B01.

This function is called by FASTOPEN.

11.1.19 Funktion 40H OS/2 box

Unterfunktion 01H switches DOS to background

Unterfunktion 02H switches DOS to foreground

11.1.20 Funktion 43H XMM-Driver

Die Funktion 43H ist nur gültig, wenn der XMM-Driver installiert ist.

Unterfunktion 00H

Mit der Unterfunktion 00H wird die Anwesenheit des Drivers getestet. Steht danach im Register AL der Wert 80H, so ist der Driver installiert.

Unterfunktion 10H

Mit der Unterfunktion 10H kann die Adresse des Funktionsdispatchers erfragt werden. Sie steht dann in ES:BX und wird mit einem far-call benutzt.

11.1.21 XMM-Funktionen

Die mit der Unterfunktion 10H der Funktion 43H des Interrupts 2FH erhaltene Adresse muß als DWORD PTR für den Call zu dem XMS-Funktionen benutzt werden.

Funktion 00H get XMS version number

Die XMS-Version sowie die interne Revisionsversion des Treibers erfragen. Zusätzlich wird angegeben, ob der High Memory Bereich existiert.

- AX = XMS-Version als 16-Bit BCD-kodierte Zahl.
- BX = interne Revisionsnummer des Treibers.
- DX = 01H, falls HMA existiert, ansonsten 00H.

Funktion 01H request high memory area

Versuchen, HMA für den eigenen Bedarf anzumelden. Der angemeldete Bedarf wird mit dem /HMA-MIN=Parameter des XMS-Treibers verglichen und HMA nur freigegeben, wenn der Wert in DX $\geq$ HMA-MIN ist. Auf diese Weise ist eine optimale Ausnutzung durch große TSRs gewährleistet.

DX = vom Programm benötigter HMA-Speicherplatz. Applikationen sollten hier FFFFH angeben, TSR-Programme den tatsächlichen Bedarf.

Funktion 02H release high memory area

Rückgabe des mit der Funktion 01H reservierten HMA an den XMS-Treiber. Der Bereich steht damit anderen Applikationen zur Verfügung.

Funktion 03H global enable A20

Adreßleitung A20 freischalten, so daß diese vom 80286/80386 gesteuert wird. Diese Funktion sollte nur von Programmen benutzt werden, die HMA reserviert haben. Bei NEAT-chip-sets geht das 'direkt' schneller.

Funktion 04H global disable A20

Die Adreßleitung A20 wieder Sperren.

Funktion 05H local enable A20

Adreßleitung A20 freigeben. Diese Funktion wird von Programmen benutzt, die direkten Zugriff auf Extended Memory (nicht zwingend HMA) benötigen. Local Enable A20 erhöht bei jedem Aufruf einen internen Zähler. Nur wenn dieser Zähler Null ist, wird A20 durch Local Enable A20 freigegeben. Im Gegensatz dazu sperrt Local Disable A20 die Adreßleitung nur dann, wenn der Zähler den Wert Eins hat. Auf diese Weise sind verschachtelte Aufrufe der Funktionen 05H und 06H möglich.

Funktion 06H local disable A20

Gegenstück zur Funktion 05H zum Sperren der Adreßleitung A20. Die Funktion 05H muß immer durch einen Aufruf dieser Funktion rückgängig gemacht werden, bevor das Programm beendet wird.

Funktion 07H query A20

Status der Adreßleitung A20 abfragen. Falls A20 freigegeben ist, kehrt die Funktion ohne Fehler zurück.

- AX = 01H, falls A20 freigegeben, sonst
- AX = 00H und BL = 00H, falls gesperrt. Andere Werte für BL bedeuten Fehler.

Funktion 08H query free extended memory

Die Länge des größten freien zusammenhängenden Speicherblocks und den gesamten freien Speicherplatz erfragen. HMA wird niemals dazugerechnet, auch wenn momentan nicht belegt.

- AX = Länge des größten Blocks
- DX = insgesamt freier Speicherplatz

Funktion 09H allocate extended memory block

Einen Speicherblock im Extended Memory reservieren und ein Handle für die Benutzung dieses Blocks zurückgeben.

- DX = benötigter Speicherplatz in KBytes
- DX = Handle

Funktion 0AH free extended memory block

Einen zuvor mit der Funktion 09H reservierten Speicherblock wieder freigeben. Der Block darf nicht verriegelt sein. Im Register DX steht die Handle.

Funktion 0BH move extended memory block

Einen Speicherblock zwischen Extended und konventionellen Speicher bzw. innerhalb des Extended oder konventionellen Speichers verschieben. Die Länge des Blocks muß **gerade** sein! Auf geraden Adressen ausgerichtete Speicherblöcke erhöhen zudem die Übertragungsgeschwindigkeit. Falls sich Quelle und Ziel überlappen, sind nur Verschiebe-Aktionen mit Quelladresse kleiner Zieladresse garantiert. A20 muß nicht extra aktiviert werden (der Zustand wird automatisch wiederhergestellt). Die Funktion garantiert außerdem eine ausreichende Zahl von Interrupt-Fenstern, damit keine Interrupts verloren gehen.

- DS:SI = Zeiger auf folgende Struktur:
  - DD? = Länge des zu bearbeitenden Blocks
  - DW? = Handle Quelle oder Null, falls konventioneller Speicher
  - DD? = Quelloffset bzw. Segment:Offset-Zeiger im Intel-Format, falls das Quellhandle Null ist.
  - DW? = Handle Ziel oder Null
  - DD? = Zieloffset bzw. Segment:Offset

Funktion 0CH lock extended memory block

Verriegelt einen Speicherblock im Extended Memory gegen Verschieben durch den XMM. Normalerweise verschiebt der XMM bei Bedarf die Speicherblöcke so, daß genügend Speicherplatz für Anfragen zur Verfügung stehen. Für spezielle Zwecke, wie DMA, ist es nötig, den Block zu verriegeln.

- DX = Handle des zu verriegelnden Blocks
- DX:BX = 32-Bit lineare Adresse des Speicherblocks

Funktion 0DH unlock extended memory block

Entriegeln eines Speicherblocks. Da sich der Block damit ab sofort verschieben kann, sind alle Zeiger darauf ungültig und sollten nicht länger benutzt werden.

DX = Handle des zu entriegelnden Blocks.

Funktion 0EH get EMB handle information

Alle Extended Memory Blocks (EMB) enthalten zusätzlich Informationen, die mit dieser Funktion erfragt werden können.

- DX = Handle des zu fragenden Blocks
- BH = Anzahl der Verriegelungen
- BL = Anzahl der noch freien Handles
- DX = Blocklänge in KBytes

Funktion 0FH reallocate extended memory block

Die Speichergröße eines reservierten und nicht verriegelten Speicherblocks verändern.

- BX = neue Größe des Speicerblocks in KByte
- DX = Handle eines entriegelten Blocks

Funktion 10H request upper memory block

Einen Speicherblock zwischen den Adaptern reservieren. UMB's dienen der Verwaltung zusätzlichen Speicherplatzes zwischen 640 k und 1MB.

- DX = benötigter Speicherplatz in Paragrphen
- BX = Segmentadresse des UMB's
- DX = Größe des angeforderten Blocks in Paragraphen, falls die Funktion ordnungsgemäß ausgeführt werden konnte, ansonsten die Größe des größten noch freien UMB's in Paragraphen.

Funktion 11H release upper memory block

Einen mit der Funktion 10H zuvor reservierten UMB wieder freigeben. Im Register DX steht die Segmentadresse des UMB's.

XMS-Fehlercodes

Die Fehlercodes befinden sich im Register BL.

- 80H = die gewünschte Funktion ist nicht implementiert.
- 81H = ein VDISK-Gerät wurde entdeckt. Aus Sicherheitsgründen wird die Funktion nicht ausgeführt, damit es nicht zum Überschreiben von Daten kommt.
- 82H = ein Fehler beim Schalten der Adreßleitung A20 ist eingetreten.
- 8EH = genereller Treiberfehler.
- 8FH = nicht behebbarer Treiberfehler aufgetreten.
- 90H = es existiert kein high memory Bereich.
- 91H = HMA ist momentan nicht verfügbar.
- 92H = der benötigte Speicherplatz für Request High Memory Area ist kleiner als durch den Schalte /HMAMIN= des XMM vorgegeben. Im Sinne einer besseren Ausnutzung des HMA wird die Benutzung daher abgelehnt.
- 93H = HMA ist nicht belegt.
- 94H = die Adreßleitung A20 ist weiterhin freigegeben.
- A0H = kein Extended Memory mehr frei.
- A1H = alle verfügbaren Handles sind bereits vergeben. Es können maximal 255 Handles vergeben werden. Im Normalfall reserviert der XMM jedoch nur Platz für z. B. 32 Handles. Mit einem optionalen Schalter /NUMHANDLES=nn läßt sich die Anzahl der Handles erhöhen.
- A2H = das übergebene Handle ist ungültig.
- A3H = das Handle für die Quelle ist ungültig.
- A4H = Offset für die Quelle ist ungültig.
- A5H = das Handle für das Ziel ist ungültig.
- A6H = Offset für das Ziel ist ungültig.

- A7H = die Länge des zu übertragenden Blocks ist ungültig.
- A8H = Quelle und Ziel überlappen sich. Es sind nur Verschiebungen von niedrigeren zu höheren Adressen erlaubt, wenn Ziel und Quelle sich überlappen.
- A9H = Parity-Fehler aufgetreten.
- AAH = der angegebene Speicherblock ist nicht verriegelt.
- ABH = der angegebene Speicherblock ist verriegelt.
- ACH = der Zähler für die Anzahl der Verriegelungen ist übergelaufen.
- ADH = der Verriegelungsversuch ist fehlgeschlagen.
- B0H = ein kleiner UMB ist verfügbar.
- B1H = es ist kein UMB mehr frei.
- B2H = angegebene UMB-Segmentadresse ist ungültig.

11.1.22 Funktion 4A CHECK

10h Check Smartdrive

Wenn AX=4A10h und BX=00h ist, wird, wenn Smartdrive4.0 geladen ist, AX=BABEh zurückgegeben.

11.1.23 Funktion 7A IPX Internetwork Packet Exchange

Steht nach dem Aufruf der Funktion 7A Unterfunktion 00, also AX=7A00h, des Interrupt 2Fh in dem Register AL 077h, so ist IPX vorhanden und in den Registern ES:DI steht die Einsprungadresse, auf die sich die folgenden Unterfunktionen beziehen.

Packet Header

Offset	Field	Size
0	Checksum	2 Bytes
2	Length	2 Bytes
4	Transport Control	1 Byte
5	Packet Type	1 Byte
6	Destination Network	4 Bytes
10	Destination Node	6 Bytes
16	Destination Socket	2 Bytes
18	Source Network	4 Bytes
22	Source Node	6 Bytes
28	Source Socket	2 Bytes
ab 30	Data	

Destination Network = 0 = same Network.

Destination Node = FFFFFFFFFFFFh = all Nodes.

Destination Socket = 456h = Diagnostic Socket.

Packet Type

0	Unknown Packet Type
1	Routing Information Packet
2	Echo Packet
3	Error Packet
4	Packet Exchange Packet
5	Sequenced Packet Protocol
16-31	Experimental Protocols

Event Control Block

Offset	Field	Size	Type
0	Link Address	4 Bytes	Pointer
4	ESR Address	4 Bytes	Pointer
8	In Use Flag	1 Byte	
9	Completion Code	1 Byte	
10	Socket Number	2 Bytes	high-low
12	IPX Workspace	4 Bytes	
16	Driver Workspace	12 Bytes	
28	Immediate Address	6 Bytes	high-low
34	Fragment Count	2 Bytes	low-high
36	Fragment Address 1	4 Bytes	Pointer
40	Fragment Size 1	2 Bytes	low-high
42	Fragment Address 2	4 Bytes	
46	Fragment Address 2	2 Bytes	

Reply Buffer

Offset	Inhalt	Type
0	Netzweradresse	Byte[4]
4	Knotenadresse	Byte[6]

Request Buffer

Offset	Inhalt	Type
0	Zielnetzwerk	Byte[4]
4	Zielknoten	Byte[6]
10	Ziel-Socket	Byte[2]

Unterfunktion 00 Open Socket

Aufruf:
- BX = 00 Diese Funktion.
- AL = 00 geöffnet bleiben bis zum expliziten Schließen oder bis das Programm beendet wird.
- AL = FF geöffnet bleiben bis zum expliziten Schließen (für TSR).

- DX = gewünschte Socket-Nummer: 0000 = dynamische Zuweisung; 0XXXX = Socket XXXX öffnen.

Rückgabe:
- AL = Function completion Code:
  * 00 = Socket wie gewünscht geöffnet.
  * FF = gewünschtes Socket bereits geöffnet.
  * FE = kein Socket mehr verfügbar.
- DX = zugewiesene Socket-Nummer.

Unterfunktion 01 Close Socket

Aufruf:
- BX = 01 Diese Funktion.
- DX = Nummer des zu schließenden Sockets.

Rückgabe: keine.

Unterfunktion 02 Get Local Target

Aufruf:
- BX = 02 Diese Funktion.
- ES:SI = Zeiger auf den request buffer.
  * Offset 00 = Zielnetzwerk (Byte[4]).
  * Offset 04 = Zielknoten (Byte[6]).
  * Offset 10 = Ziel-Socket (Byte[2]).
- ES:DI = Zeiger auf den reply buffer.

Rückgabe:
- AL = completion code:
  * 00 = erfolgreich.
  * FA = keine Verbindung zum Zielknoten.
- CX = Transportdauer in Timer-Ticks.
- request buffer = Offset 0 local Target Byte[6].

Unterfunktion 03 Send Packet

Aufruf: BX = 03 Diese Funktion.
ES:SI = Pointer zum zugehörigen ECB.

Rückgabe: keine.

Unterfunktion 04 Listen for Packet

Aufruf: BX = 04 diese Funktion.
ES:SI = Pointer zum zugehörigen ECB.

Rückgabe: keine.

Unterfunktion 05 Schedule Event

Aufruf: BX = 05 diese Funktion.
AX = Wartezeit in Timer-Ticks.
ES:SI = Pointer zum zugehörigen ECB.

Rückgabe: keine.

Unterfunktion 06 Cancel Event

Aufruf: BX = 06 diese Funktion.
ES:SI = Pointer zum zugehörigen ECB.

Rückgabe: AL = Function Completen Code:
- 00 = erfolgreich
- F9 = löschen nicht möglich
- FF = ECB existiert nicht

Unterfunktion 08 Get Internal Market

Aufruf: BX = 08 diese Funktion

Rückgabe: AX = Intervall-Marker

Der Intervall-Marker wird im Rhytmus des Timer-Ticks (=18mal/s) inkrementiert.

Unterfunktion 09 Get Internetwork Address

Aufruf: BX = 09 diese Funktion.

Rückgabe: ES:DI Zeiger auf den Reply Buffer.

Unterfunktion 0A Relinquish Control

Aufruf: BX = 0A diese Funktion.

Rückgabe: Keine.

Dieser Aufruf kann die CPU, auf der die Anwendung läuft, zeitweise anderen Prozessen zur Verfügung stellen.

Unterfunktion 0B Disconnect from Target

Aufruf: BX = 0B diese Funktion.
ES:DI = Zeiger auf den request buffer.

Rückgabe: keine.

11.1.24 Funktion AEh Installable Commands

Unterfunktion 00h

Unterfunktion 01h

11.1.25 Funktion B7H Append

Unterfunktion 00H Check Install

- AH = 0 = nicht installiert
- AH = 1 = installiert

Unterfunktion 02H Get Version

Im Register AX steht der Versionscode, für MS-DOS eine -1.

Unterfunktion 04H Get Path

In den Registern ES:DI steht die Adresse des Pfades.

Unterfunktion 06H Get State

Im Register BX steht der Status.

Unterfunktion 07H Set State

Im Register BX steht der Status.

Unterfunktion 11H Set Return Found Name Mode

11.2 Interrupt Nr. 30H MS-DOS

11.3 Interrupt Nr. 31H DOS Protected Mode Interface

Nach dem VCPI (virtual control program interface) wurde Mitte 1990 das DOS Protected Mode Interface (DPMI) von z. B. Borland, Intel, Microsoft definiert.

DPMI macht fast alle Fähigkeiten des 80386-Prozessors unter DOS nutzbar. Als *Client* wird ein Programm bezeichnet, daß die DPMI-Funktionen nutzt. Ein *Server* stellt die Funktionen zur Verfügung. Bei einigen Funktionen unterscheidet man zwischen 16- und 32-Bit Anwendung. Es wird dann z. B. ES:(E)DI angegeben. Der Aufruf des DPMI erfolgt mit der Funktionsnummer im Register AX durch den Interrupt 31H, der nur im Protected-Mode zur Verfügung steht.

Um Real-Mode Software aus dem Protected-Mode aufzurufen, verwendet das DPMI die folgende Real-Mode Call Struktur.

SS	30h
SP	2Eh
CS	2Ch
IP	2Ah
GS	28h
AS	26h
DS	24h
ES	22h
Flags	20h
EAX	1Ch
ECX	18h
EDX	14h
EBX	10h
EBP	08h
ESI	04h
EDI	00h

æ

Wenn ein Ausnahme-Interrupt auftritt, hinterläßt das DPMI einige Informationen auf dem Stack, die eine geeignete Behandlungsroutine auswerten kann.

SS	SS
SP	ESP
FLAGS	EFLAGS
CS	CS
IP	EIP
Error	Error Code
Ret CS	Return CS
Ret IP	Return EIP

æ

11.3.1 Funktion 000H Allocate LDT Descriptors

Einen oder mehrere Deskriptoren in der LDT dieser Task anfordern. Den Inhalt der Deskriptoren muß das Programm selbst initialisieren. Aus Sicherheitsgründen legt das DPMI dort die Beschreibung für ein Datensegment ab Adresse 0h mit einem Limit Null ab. Fordert der Client mehrere Deskriptoren an, so enthält AX den Selektor des ersten Deskriptors. Um den nächsten Selektor zu erhalten, muß er zum Inhalt von AX den von der Funktion 3h (Get Next Selector Increment Value) zurückgelieferten Wert addieren.

Ein: CX = Anzahl der anzufordernden Deskriptoren.

Aus: AX = erster reservierter Selektor.

11.3.2 Funktion 001H Free LDT Descriptor

Einen einzelnen LDT-Dekriptor wieder freigeben. Mit dieser Funktion darf der Client nur die Deskriptoren wieder zurückgeben, die er mit den Funktionen 0h, Ah sowie Dh angefordert hat. Die beim Wechsel in den Protected-Mode automatisch erstellten Deskriptoren für CS, DS und SS kann er allerdings mit dieser Dienstleistung ebenfalls wieder freigeben.

Ein: BX = zurückzugebender Selektor.

Aus: —

11.3.3 Funktion 002H Segment to Descriptor

Eine Segmentadresse des Real-Mode in einen Deskriptor für ein 64 KByte großes Datensegment mit der entsprechenden Basisadresse konvertieren. Mehrfache Anforderungen mit der gleichen Real-Mode Segmentadresse ergebebn den gleichen Selektor.

Ein: BX = Real-Mode Segmentadresse.

Aus: AX = Selektor für ein Segment mit der angegebenen Startadresse.

11.3.4 Funktion 003H Get next Selector invrement value

Hat der Client mit der Funktion 0H mehrere Deskriptoren angefordert, so muß er den von dieser Funktion zurückgegebenen Wert benutzen, um zum jeweils nächsten Selektor zu gelangen.

Ein: —

Aus: AX = Zu einem Selektor zu addierender Wert, um zum nachfolgenden Selektor zu gelangen.

11.3.5 Funktion 006H Get Segment Base Address

Die Basisadresse eines Segments erfragen.

Ein: BX = Selektor

Aus: CX:DX = lineare 32-Bit Adresse des Segmentbeginns (CX = High-Word).

11.3.6 Funktion 007H Set Segment Base Address

Die Basisadresse eines Segments ändern.

Ein: BX = Selektor.
CX:DX = neue 32-Bit Basisadresse des Segments.

Aus: —

11.3.7 Funktion 008H Set Segment Limit

Das Segmentlimit für das angegebene Segment neu setzen.

Ein: BX = Selektor.
CX:DX = neues 32-Bit Segmentlimit.

Aus: —

11.3.8 Funktion 009H Set Descriptor Access Rights

Die Zugriffsrechte und Typeninformationen eines Deskriptors ändern.

Ein: BX = Selektor.
CL = Zugriffsrechte/Selektortyp.
CH = zusätzliche Zufriffsrechte und Typinformationen für den 80386.

Aus: —

11.3.9 Funktion 00AH Create Code Segment Alias Descriptor

Einen Datensegmentdeskriptor mit der gleichen Basisadresse und Größe wie ein gegebenes Codesegment kreieren.

Ein: BX = Selektor eines Codesegments.

Aus: AX = Neuer Selektor für Alias-Segment.

11.3.10 Funktion 00BH Get Descriptor

Einen LDT-Deskriptor in einen Buffer kopieren.

Ein: BX = Selektor.
ES:(E)DI = Zeiger auf einen 8 Byte großen Buffer, in den der Descriptor kopiert wird.

Aus: —

11.3.11 Funktion 00CH Set Descriptor

Einen LDT-Deskriptor mit Daten direkt aus einem Buffer initialisieren.

Ein: BX = Selektor.
ES:(E)DI = Zeiger auf einen 8 Byte großen Buffer, der einen Deskriptor enthält.

Aus: —

11.3.12 Funktion 00DH Allocate Specific LDT Descriptor

Einen einzelnen LDT Deskriptor in der LDT gezielt anfordern.

Ein: BX = Selektor.

Aus: —

Verwaltung des DOS-Speichers.

11.3.13 Funktion 100H Allocate DOS Memory Block

Aus dem MS-DOS-Speicherpool einen Block anfordern.

Ein: BX = Größe des gewünschten Speicherblocks in Paragraphen (16 Byte).

Aus: Carry Flag gelöscht:
AX = Real-Mode Segmentadresse des reservierten Blocks.
DX = Selektor, um auf diesen Speicher zuzugreifen.
Carry Flag gesetzt:
AX = DOS-Fehlercode (7 = MCBs zerstört, 8 = nicht genügend Speicher vorhanden).
BX = Größe des größten momentan freien Speicherblocks in Paragraphen.

11.3.14 Funktion 101H Free DOS Memory Block

Einen Block aus dem MS-DOS-Speicherpool wieder zurückgeben. Alle damit verbundenen Deskriptoren gibt der Server automatisch frei.

Ein: DX = Selektor des zurückzugebenden Blocks.

Aus: falls Carry Flag gesetzt:
AX = DOS-Fehlercode (7 = MCBs zerstört, 9 = falsches Segment angegeben.

11.3.15 Funktion 102H Resize DOS Memory Block

Die Größe eines dem MS-DOS-Speicherpools entstammenden Blocks ändern.

Ein: BX = Neue Größe des Speicherblocks in Paragraphen.
DX = Selektor des zu verändernden Blocks.

Aus: falls Carry Flag gesetzt:
AX = DOS-Fehlercode (7, 8 oder 9).
BX = Maximale Größe dieses Blocks.

Interrupt Diesntleistungen.

11.3.16 Funktion 200H Get Real Mode Interrupt Vector

Die Adresse eines Real-Mode Interrupt-Handlers für die aktuelle Task erfragen. DPMI unterstützt alle 256 Interruptvektoren.

Ein: BL = Interruptnummer.

Aus: CX:DX = Adresse im Real-Mode-Format (Segment:Offset) des Real-Mode Interrupt-Handlers.

11.3.17 Funktion 201H Set Real Mode Interrupt Vector

Einen Real-Mode Interruptvektor für diese Task neu setzen. Handelt es sich dabei um einen Handler für Hardware-Interrupts, so muß der dafür benutzte Programmcode und die während der Behandlung angesrochene Datenbereich gegen Auslagerung verriegelt sein.

Ein: BL = Interruptnummer.
CX:DX = Adresse eines Real-Mode Interrupt-Handlers im Format Segment:Offset.

Aus: —

11.3.18 Funktion 202H Get Processor Exeption Handler Vector

Die Adresse einer Behandlungsroutine für Ausnahmezustände (Interrupts 00h bis 1Fh) erfragen. Diese Routinen unterscheidet das DPMI selbständig von den entsprechenden Hardware-Interruptroutinen, die dank IBM ebenfalls in diesem Bereich liegen.

Ein: BL = Nummer eines Ausnahme-/Fehlerinterrupts (00H bis 1FH).

Aus: CX:(E)DX = Adresse im Format Selektor:Offset des Exception-Handlers.

11.3.19 Funktion 203H Set Processor Exeption Handler Vector

Die Adresse für eine Behandlungsroutine von Prozessor-Ausnahmezuständen setzen. Der Server reicht einen Ausnahme-Interrupt an die Kette der Behandlungsroutinen weiter. Ein Handler kann entweder mit einem RETF zurückkehren oder aber den nächsten Handler in der Kette anspringen. Erreicht ein solcher Interrupt den DPMI-Standard-Handler, so ruft dieser für die Ausnahmezustände 0 bis 5 sowie 7 den entsprechenden Interrupt-Handler auf. Alle anderen Ausnahmen führen zur Beendigung der Applikation.

Ein: BL = Nummer eines Ausnahme-/Fehlerinterrupts (00H bis 1FH).
CX:(E)DX = Neue Adresse des Handlers im Format Selektor:Offset.

Aus: —

11.3.20 Funktion 204H Get Protected Mode Interrupt Vector

Die Adresse einer Interruptroutine für den Protected-Mode erfragen.

Ein: BL = Interruptnummer.

Aus: CX:(E)DX = Selektor:Offset der Interruptroutine.

11.3.21 Funktion 205H Set Protected Mode Interrupt Vector

Die Adresse für eine Protected-Mode Interruptroutine setzen.

Ein: BL = Interruptnummer.
CX:(E)DX = Selektor:Offset der neuen Interruptroutine.

Aus: —

CPU-Moduswechsel.

11.3.22 Funktion 300H Simulate Real Mode Interrupt

Eine Interruptroutine des Real-Mode aufrufen. Der Server ignoriert die in der Datenstruktur angegebenen Werte für CS:IP und verwendet statt dessen die Werte des entsprechenden Real-Mode Interruptvectors. Sind die Felder SS:SP null, so liefert das DPMI einen Stack mit ca. 30 Worten Platz. Der Aufruf des Interrupthandlers erfolgt mit gelöschtem Trace- und Interrupt-Flag. Die eventuell auf den Protected-Mode Stack verfrachteten Parameter mu"d der Aufrufer selbst später wieder entfernen.

Ein: BL = Interruptnummer.
BH = Flags (Bit0 = 1: Interruptcontroller zurücksetzen und A20 sperren; wird im virtuellen 8086-Modus ignoriert).
CX = Anzahl der Worte, die vom Protected-Mode-Stack zum im Real-Mode benutzten Stack kopiert werden.
ES:(E)DI = Selektor:Offset der Real-Mode Datenstruktur.

Aus: ES:(E)DI = Zeiger auf die modifizierte Datenstruktur.

11.3.23 Funktion 301H Call Real Mode Procedure with FAR Return Frame

Eine Routine im Real-Mode aufrufen, die mit einem RETF zurückkehrt. Die Zieladresse ist in den Feldern CS:IP der Datenstruktur angegeben. Ansonsten gelten die für die Funktion 300h aufgeführten Hinweise.

Ein: BL = Interruptnummer.
BH = Flags (Bit0 = 1: Interruptcontroller zurücksetzen und A20 sperren; wird im virtuellen 8086-Modus ignoriert).
CX = Anzahl der Worte, die vom Protected-Mode-Stack zum im Real-Mode benutzten Stack kopiert werden.
ES:(E)DI = Selektor:Offset der Real-Mode Datenstruktur.

Aus: ES:(E)DI = Zeiger auf die modifizierte Datenstruktur.

11.3.24 Funktion 302H Call Real Mode Procedure with IRET Frame

Eine Routine im Real-Mode aufrufen, die mit einem IRET zurückkehrt. Ansonsten gelten die für die Funktion 301h aufgeführten Hinweise.

Ein: BH = Flags (Bit0 = 1: Interruptcontroller zurücksetzen und A20 sperren; wird im virtuellen 8086-Modus ignoriert).

CX = Anzahl der Worte, die vom Protected-Mode-Stack zum im Real-Mode benutzten Stack kopiert werden.

ES:(E)DI = Selektor:Offset der Real-Mode Datenstruktur.

Aus: ES:(E)DI = Zeiger auf die modifizierte Datenstruktur.

11.3.25 Funktion 303H Allocate Real Mode Call-Back Address

Um eine Protected-Mode Routine vom Real-Mode aus aufrufen zu können, einen *Call-Back* anfordern.

Ein: DS:(E)SI = Selektor:Offset der aufzurufenden Protected-Mode Routine.

ES:(E)DI = Selektor:Offset der Real-Mode Datenstruktur.

Aus: CX:DX = Adresse des "Call Back" im Real-Mode Format Segment:Offset.

Eine Protected-Mode Routine vom Real-Mode aus aufrufen. Diese Routine muß die Felder CS:IP in der Datenstruktur verändern, sonst führt der Call-Back in eine Endlosschleife. Der Rücksprung zum Aufrufer erfolgt mittels IRET-Befehl. Ohne weitere Vorsichtsmaßnahmen ist ein Call-Back zudem nicht reentrant, da jeder Aufruf die gleiche Real-Mode Datenstruktur benutzt. Ist die Wiedereintrittsfähigkeit jedoch notwendig, so muß die Routine die Datenstruktur an einen sicheren Platz kopieren und bei ihrer Rückkehr ES:(E)DI dorthin zeigen lassen, damit das DPMI die Register ordnungsgemäß wiederherstellen kann.

Ein: DS:(E)SI = Selektor:Offset des Real-Mode Stacks SS:SP

ES:(E)DI = Selektor:Offset der Real-Mode Datenstruktur.

SS:(E)SP = verriegelter Protected-Mode Stack. Interrupts gesperrt.

Aus: ES:(E)DI = Selektor:Offset auf die für die Rückkehr zu benutzende Real-Mode Datenstruktur.

11.3.26 Funktion 304H Free Real Mode Call-Back Address

Einen mit der Funktion 303h angeforderten Call-Back wieder an das DPMI zurückgeben.

Ein: CX:DX = Real-Mode Adresse eunes Call-Back.

Aus: —

11.3.27 Funktion 305H Get State Save/Restore Addresses

Die Adressen der Routine erfragen, die den aktuellen Zustand einer Rask sichern oder wiederherstellen können. Diese Routinen sind von spezieller Bedeutung für den durch die Funktion 306h angebotenen Dienst oder aber, wenn die Behandlung eines Hardware-Interrupts DOS-Aufrufe ausführt. Falls die jeweilige DPMI Implementation das Sichern des Zustands nicht nötig hat, liefert sie als Buffergröße zurück. Auch wenn die Routinen so keine Funktion haben, darf sie der Client anspringen.

Ein: —

Aus: AX = Größe des für die Zustandssicherung benötigten Buffers in Bytes.

BX:CX = Real-Mode Adresse zum Sichern/Wiederherstellen des Task-Zustands.

SI:(E)DI = Protected-Mode Adresse zum Sichern/Wiederherstellen des Task-Zustands.

Task-Zustand sichernwiederherstellen.

Ein: AL = 0: Zustand sichern; 1: wiederherstellen.

ES:(E)DI = Zeiger auf einen Buffer, der die Daten aufnimmt (Adresse im Format der jeweiligen CPU-Betriebsart).

11.3.28 Funktion 306H Get RAW Mode Switch Address

Die Einsprungadressen für die Routinen zum Moduswechsel erfragen. Soweit möglich, sollte ein Client nur die Dienstleistungen 300h bis 304h für Moduswechsel benutzen.

Ein: —

Aus: BX:CX = Einsprungadresse zum Wechseln vom Real-Mode in den Protected-Mode.

SI:(E)DI = Einsprungadresse für den Wechsel vom Protected-Mode in den Real-Mode.

Moduswechsel durchführen. Die entsprechende, von der Funktion 306h erfragte Routine mit einem FAR-JMP aufrufen, um den CPU-Modus zu wechseln. Es bleibt nur der Inhalt des (E)BP-Registers erhalten. Die Segmentregister FS und GS werden bei 32-Bit Implementationen auf Null gesetzt.

Ein: AX = neuer Wert für DS.

CX = neuer Wert für ES.

DX = neuer Wert für SS.

(E)BX = neuer Wert für (E)SP.

SI = neuer Wert für CS

(E)DI = neuer Wert für (E)IP.

11.3.29 Funktion 400H Get Version

Informationen über die DPMI-Implementation beziehungsweise -Konfiguration erfragen.

Ein: —

Aus: AX = Versionsnummer binär (AH = high, AL = minor).

BX = Flags

- Bit0 = 1:DPMI-Implementation für den 80386
- Bit1 = 1: an Real-Mode Software weitergereichte Interrupts werden im Real-Mode bearbeitet.
- Bit1 = 0: an Real-Mode Software weitergereichte Interrupts werden im virtuellen 8086-Modus bearbeitet.
- Bit2 = 1: virtuelle Speicherverwaltung vorhanden.

CL = Prozessortyp (2 = 286, 3 = 386, 4 = 486).

DH = erster Interruptvektor des Master-Interrupt Kontrollers.

DL = erster Interruptvektor des Slave-Interrupt Kontrollers.

11.3.30 Funktion 500H Get Free Memory Information

Informationen über die aktuelle Speicherbelegung ermitteln. DPMI-Implementationen, die keine virtuelle Speicherverwaltung bieten, füllen nur das erste DWORD aus. Alle anderen Einträge setzen sie auf -1. Prinzipiell kann jede DPMI-Implementation einzelne Informationen verweigern, indem es den entsprechenden Eintrag mit -1 initialisiert.

Ein: ES:(E)DI = Selektor:Offset auf einen 48 Byte großen Buffer mit folgender Struktur:

- DWORD ? = Größter freier Block in Bytes.
- DWORD ? = Anzahl der freien unverriegelten Seiten.
- DWORD ? = Größter Block in Seiten, der angefordert und verriegelt werden kann.
- DWORD ? = Linearer Adreßraum in Seiten.
- DWORD ? = Anzahl der unverriegelten Seiten.
- DWORD ? = Anzahl der unbenutzten physikalischen Seiten.
- DWORD ? = Anzahl der physikalischen Seiten insgesamt.
- DWORD ? = Freier linearer Adreßraum in Seiten.
- DWORD ? = Größe der Swap-Datei in Seiten.

Aus: —

11.3.31 Funktion 501H Allocate Memory Block

Einen Speicherblock anfordern. Die zum Ansprechen des Speichers notwendigen Deskriptoren muß das Programm selbst reservieren und initialisieren. Da DPMI-Implementationen sehr oft die Anforderungen auf die volle Seitengröße (4 KByte) aufrunden, empfiehlt es sich, gleich einen Speicher in Vielfachen von 4 KByte anzufordern. Der reservierte Bereich ist unverriegelt.

Ein: BX:CX = Größe des Speicherblocks in Bytes.

Aus: BX:CX = Lineare Adresse des reservierten Blocks.

SI:DI: = Handle des Speicherblocks.

11.3.32 Funktion 502H Free Memory Block

Einen zuvor mit der Funktion 501h angeforderten Speicherblock wieder freigeben.

Ein: SI:DI = Handle des zurückzugebenden Blocks.

Aus: —

11.3.33 Funktion 503H Resize Memory Block

Die Größe eines mit der Funktion 501h reservierten Speicherblocks ändern. Sowohl die lineare Adresse des Blocks als auch die Handle können sich verändern. Das Verkleinern eines Blocks auf 0 Byte ist nicht möglich.

Ein: BX:CX = Neue Größe des Speicherblocks.

SI:DI = Handle des Speicherblocks.

Aus: BX:CX = Neue lineare Adresse des Blocks.

SI:DI = Neue Handle des Speicherblocks.

Virtuelle Speicherverwaltung.

11.3.34 Funktion 600H Lock Linear Region

Einen Speicherbereich gegen das Auslagern auf die Festplatte verriegeln.

Ein: BX:CX = lineare Startadresse des Bereichs.

SI:DI = Länge des Bereichs.

Aus: —

11.3.35 Funktion 601H Unlock Linear Region

Einen zuvor mit der Dienstleistung 600h verriegelten Speicherbereich wieder entriegeln. Das DPMI zählt die Verriegelungs- und Entriegelungs-Aufrufe. Eine Seite kann es nur dann auslagern, wenn der Zähler auf Null steht.

Ein: BX:CX = lineare Startadresse des Bereichs.

SI:DI = Länge des Bereichs.

Aus: —

11.3.36 Funktion 602H Mark Real Mode Region as pageable

Einen Speicherbereich innerhalb des ersten Megabytes als auslagerbar kennzeichnen. Ob der Server diese Seite auslagern kann, hält hier nur eine einfache Flag fest, so daß der Client diese Funktion sowie die Dienstleistung 603h jeweils nur einmal aufrufen darf.

Ein: BX:CX = lineare Startadresse des Bereichs.

SI:DI = Länge des Bereichs.

Aus: —

11.3.37 Funktion 603H Relock Real Mode Region

Einen Speicherbereich innerhalb des ersten Magabyte wieder verriegeln.

Ein: BX:CX = lineare Startadresse des Bereichs.
SI:DI = Länge des Bereichs.

Aus: —

11.3.38 Funktion 604H Get Page Size

Die Größe einer Speicherseite in Bytes ermitteln.

Ein: —

Aus: BX:CX = Größe einer Speicherseite.

11.3.39 Funktion 702H Mark Page as Demand Paging Candidate

Einen Speicherbereich derart markieren, daß das DPMI diesen bei der nächsten Gelegenheit bevorzugt auf die Festplatte auslagert.

Ein: BX:CX = Startadresse des zu markierenden Bereichs.
SI:DI = Länge des Bereichs.

Aus: —

11.3.40 Funktion 703H Discard Page Contents

Einen Speicherbereich als unwichtig markieren, so daß die Speicherverwaltung diesen nicht auf Platte auslagert, sondern bei Bedarf überschreibt.

Ein: BX:CX = Startadresse des zu markierenden Bereichs.
SI:DI = Länge des Bereichs.

Aus: —

11.3.41 Funktion 800H Physical Address Mapping

Die lineare Adresse einer physikalischen Adresse erfragen, um auf diese Weise auf Geräte zuzugreifen, deren Steuerung über Speicheradressen und nicht durch Ein-/Ausgabeports erfolgt. Nicht in jeder Implementation vorhanden.

Ein: BX:CX = Physikalische Adresse.
SI:DI = Größe des Bereichs in Byte.

Aus: BX:CX = entsprechende lineare Adresse.

11.3.42 Funktion 900H Get and disable Virtual Interrupt State

Die virtuelle Interruptflag zurücksetzen, so daß Harware-Interrupts das Programm nicht mehr erreichen.

Ein: —

Aus: AL = Zustand der Flag vor dem Aufruf (0 = gesperrt).

11.3.43 Funktion 901H Get and enable Virtual Interrupt State

Die virtuelle Interruptflag setzen.

Ein: —

Aus: AL = Alter Zustand der Flag.

11.3.44 Funktion 902H Get Virtual Interrupt State

Den aktuellen Zustand ser virtuellen Interrupt-Flag erfragen.

Ein: —

Aus: AL = Aktueller Zustand.

Verkäuferspezifische APIs.

11.3.45 Funktion A00H Get Vendor Specific API Entry Point

Die Einsprungadresse für verkäuferspezifische DPMI-Erweiterungen ermitteln.

Ein: DS:(E)SI = Zeiger auf einen durch ein Nullbyte beendeten String.

Aus: ES:(E)DI = Einsprungadresse in das API.

11.3.46 Funktion B00H Set Debug Watchpoint

Einen Hardware-Watchpoint mit den angegebenen Kriterien an eine bestimmte Adresse setzen.

Ein: BX:CX = Lineare Adresse eines Watchpoints.
DL = Größe des Watchpoints in Byte (1, 2 oder 4).
DH = Art des Watchpoints.

- 0 = Auslösen bei Befehlsausführung.
- 1 = Lesender Zugriff.
- 2 = Lesender oder schreibender Zugriff.

Aus: BX = Handle des Watchpoints.

11.3.47 Funktion B01H Clear Debug Watchpoint

Einen Watchpoint löschen.

Ein: BX = Handle des Watchpoints.

Aus: —

11.3.48 Funktion B02H Get State of Debug Watchpoint

Überprüfen, ob ein bestimmter Watchpoint ausgeführt wurde. Wenn eine Aplikation mehrere Watchpoints angefordert hat, muß sie mit dieser Funktion feststellen, welcher Watchpoint der Auslöser des Debugging-Interrupts war.

Ein: BX = Handle eines Watchpoints.

Aus: AX = Statusflag (Bit 0 ist auf 1 gesetzt, wenn der Watchpoint ausgelöst wurde).

11.3.49 Funktion B03H Reset Debug Watchpoint

Den Status eines Watchpoints, nachdem er ausgelöst wurde, zurücksetzen.

Ein: BX = Handle eines Watchpoints.

Aus: —

11.4 Interrupt Nr. 32H MS-DOS

11.5 Interrupt Nr. 33H Mouse

Mouse-Funktionen werden mit dem Interrupt 33H aufgerufen.
Die Funktion wird im Register AX übergeben. Die anderen Parameter, so weit nötig, werden in den Registern AX, BX, CX oder DX übergeben und nicht auf Gültigkeit geprüft. Ergebnisse werden auch nur in den Registern AX, BX, CX und DX abgelegt.

11.5.1 Funktion 00H Reset Genius Mouse Driver

Diese Funktion wird benutzt um festzustellen, ob der Driver installiert ist und ihn in den Ausgangszustand zu versetzen.

Input: AX = 0H

Return: AX = mouse state (-1: installiert; 0: nicht installiert)

BX = number of bottons

Der Ausgangszustand für die Genius Mouse und Driver ist:

internal cursor flag	-1 (cursor concealed, unsichtbar)
graphics cursor shape	horizontal oval
text cursor	reverse video
user-defined call mask	all zeroes
light pen emulation mode	enabled
vertical mouse motion/pixel ratio	16 to 8
horizontal mouse motion/pixel	8 to 8
vertical min/max cursor coordinates	0/current display mode y values minus 1
horizontal min/max cursor coordinates	0/current display mode x values minus 1

11.5.2 Funktion 01H Enable Cursor Display

Die interne cursor flag ist von der Initialisierung her -1. Mit der Funktion 01H wird der Zähler der flag erhöht und damit der cursor sichtbar. Eine weitere Erhöhung des Zählers erfolgt nicht.

Input: AX = 1H

Return: none

11.5.3 Funktion 02H Disable Cursor Display

Mit dieser Funktion kann der cursor unsichtbar gemacht werden. Sie sollte immer vor einer Veränderung des Bildschirminhalts benutzt werden. Danach muß dann die Funktion 01H benutzt werden um den cursor sichtbar zu machen. Wenn das Programm den Bildschirmmodus ändert, wird diese Funktion immer aufgerufen. Am Ende eines Programms sollte diese Funktion benutzt werden, damit nichts auf dem Bildschirm zurückbleibt.

Input: AX = 2H

Return: none

11.5.4 Funktion 03H Read Cursor

Diese Funktion liest den Status der Mouse-Tasten und die Position des cursor. Der Status ist integer.

Input: AX = 3H

Return: BX = button Status (Bit = 1 wenn gedrückt)

- Bit0 = linke Taste
- Bit1 = rechte Taste
- Bit2 = mittlere Taste

CX = horizontal cursor coordinate

DX = vertical cursor coordinate

11.5.5 Funktion 04H Set Cursor

Die Werte müssen innerhalb des virtuellen Bildschirms sein. Sie werden auf den nächstmöglichen Wert für den augenblicklichen Bildschirm gerundet.

Input: AX = 4H

CX = new horizontal cursor coordinate

DX = new vertical cursor coordinate

Return: none

11.5.6 Funktion 05H Read Button Press State

Input: AX = 5H

BX = button status (links = 0, rechts = 1, mitte = 2)

Return: AX = button status

BX = number of buttons pressed

CX = horizontal cursor coordinate at last press

DX = vertical cursor coordinate at last press

11.5.7 Funktion 06H Read Button Release State

Input: AX = 6H

BX = button status (links = 0, rechts = 1, mitte = 2)

Return: AX = button status

BX = number of buttons released

CX = horizontal cursor coordinate at last release

DX = vertical cursor coordinate at last release

11.5.8 Funktion 07H Define Horizontal Range

Input: AX = 7H

CX = minimum horizontal cursor coordinate

DX = maximum horizontal cursor coordinate

Return: none

11.5.9 Funktion 08H Define Vertikal Range

Input: AX = 8H

CX = minimum vertical cursor coordinate

DX = maximum vertical cursor coordinate

Return: none

11.5.10 Funktion 09H Define Cursor Graphics Style

Input: AX = 9H

BX = horizontal cursor hot spot

CX = vertical cursor hot spot

DX = pointer to screen and cursor mask

Return: none

11.5.11 Funktion 10H Define Cursor Text Style

Input: AX = 10H

BX = select cursor (0 = software text; 1 = hardware text)

CX = screen mask value/scan line start

DX = cursor mask value/scan line stop

Return: none

11.5.12 Funktion 11H Read Motion Number

Input: AX = 11H

Return: CX = horizontal number

DX = vertical number

11.5.13 Funktion 12H Define Event Handler Entry Location

Input: AX = 12H

CX = call mask

ES:DX = pointer to event handler routine

Return: none

11.5.14 Funktion 13H Enable Light Pen Emulation

Input: AX = 13H

Return: none

11.5.15 Funktion 14H Disbale Light Pen Emulation

Input: AX = 14H

Return: none

11.5.16 Funktion 15H Define Sensitivity

Input: AX = 15H

CX = horizontal mouse motion counts to pixel ratio

DX = vertical mouse motion counts to pixel ratio

Return: none

11.5.17 Funktion 16H Disable Cursor Display in Special Range

Input: AX = 16H

ES:DX = pointer to special range

Return: none

11.5.18 Funktion 19H Define Double-Speed Threshold

Input: AX = 19H

DX = threshold speed in mouse motions/second

Return: none

11.5.19 Funktion 20H Swap Event Handler Entry Location

Input: AX = 20H

CX = new call mask

ES:DX = new pointer to event handler routine

Return: CX = old call mask

ES:DX = old pointer to event handler routine

11.5.20 Funktion 21H Get Mouse Driver State Storage Size

Input: AX = 21H

Return: BX = buffer size required for mouse driver state

11.5.21 Funktion 22H Save Mouse Driver State

Input: AX = 22H

ES:DX = pointer to the buffer

Return: none

11.5.22 Funktion 23H Restore Mose Driver State

Input: AX = 23H

ES:DX = pointer to the buffer

Return: none

11.5.23 Funktion 29H Set CRT Page Number

Input: AX = 29H

BX = CRT page for mouse cursor display

Return: none

11.5.24 Funktion 30H Read CRT Page Number

Input: AX = 30H

Return: BX = CRT page number of current cursor display

Nur mit IBM-EGA:

11.5.25 Funktion F0H Retrieve Single Data

Input: AX = F0H

BX = index number

DX = port number

Return: BL = retrieved data from EGA register

11.5.26 Funktion F1H Save Single Data

Input: AX = F1H

DX = port number

– non-single register only
BL = index number
BH = data
– single register only
BL = data
BH = disregard

Return: none

11.5.27 Funktion F2H Retrieve Registers on a Specified Port

Input: AX = F2H

CH = starting index number

CL = number of registers

DX = port number

ES:BX = destination of returned data

Return: returned data to destionation address

11.5.28 Funktion F3H Save Registers on a Specified Port

Input: AX = F3H

CH = starting index number

CL = number of registers

DX = port number

ES:BX address source of incomming data

Return: none

11.5.29 Funktion F4H Retrieve Several Registers Data

Input: AX = F4H

CX = number of registers (more than 1)

ES:BX = address of register packet

- Byte1-2 port address
- Byte3 index number
- Byte4 returned data

Return: returned data is saved into byte 4

11.5.30 Funktion F5H Save Several Registers Data

Input: AX = F5H

CX = number of registers

ES:BX = address of register packet

- Byte1-2 port address
- Byte3 index number
- Byte4 output data

Return: none

11.5.31 Funktion F6H Reset All Registers as Initial Values

Input: AX = F6H

Return: none

11.5.32 Funktion F7H Set Initial Values

Input: AX = F7H

DX = port number

ES:BX = table of output data

Return:

11.5.33 Funktion FAH Get Version Number of Driver

Input: AX = FAH

BX = 00H

Return: ES:BX = pointer to genius mouse driver version number

11.6 Interrupt Nr. 34H-3E MS-DOS

11.7 Interrupt Nr. 3FH VROOMM Overlay Manager

Virtual Runtime Object-Oriented Memory Manager.

Speicherverwaltung für die dynamischen Ein- und Auslagerungsvorgänge von Turbo c++.

11.8 Interrupt Nr. 40H Reserved (Diskette Int. redir)

Der Interrupt 40H wird vom Interrupt 13H benutzt, wenn eine Anforderung kleiner 80H (Floppy) erkannt wurde.

Dieser Interrupt sollte niemals direkt benutzt werden.

11.9 Interrupt Nr. 41H und 46H Fixed Disk Parameters

Diese Vektoren zeigen auf die Parameter Tabellen der Winchesterlaufwerke im ROM. Der POST wählt über die Angaben im CMOS die richtige Tabelle aus.

11.10 Interrupt Nr. 42H EGA-BIOS

Revectored.

11.11 Interrupt Nr. 43H EGA Graphics char

11.12 Interrupt Nr. 44H Novell Netware API

Bei dem PCjr und EGA 1st 128 Character map.

11.13 Interrupt Nr. 47H SQL Base API

11.14 Interrupt Nr. 48H PCjr Keyboard translate

11.15 Interrupt Nr. 49H Keyboard-supplement devices

In Peter Norton's Programmer's Guide findet man "points to translation table for keyboard-supplement devices".

PCjr scan code translation table.

11.16 Interrupt Nr. 4AH Interval Interrupt

Dieser Interrupt wird innerhalb des Interrupts 70H aufgerufen, wenn vorher im CMOS-Register B das Bit 5 gesetzt wurde. Dieser Interrupt wird nur ausgeführt, wenn nicht das Bit 6 im CMOS-Register B gesetzt wurde.

11.17 Interrupt Nr. 4BH QEMM386

11.18 Interrupt Nr. 50H — 57H DESQview IRQ0 — 7

11.19 Interrupt Nr. 51H Mouse Functions

11.20 Interrupt Nr. 59H GSS Graphic Interface

11.21 Interrupt Nr. 5AH Cluster Adapter BIOS

11.22 Interrupt Nr. 5BH Cluster Adapter

11.23 Interrupt Nr. 5CH NETBIOS Emulation Interface (NETBIOS Session)

Zuerst muß ein Network Control Block (NCB) angelegt werden.

Field Name	Size	Description
ncb_command	1	Command code field. Wenn das höchste Bit (80H) gesetzt ist, wird eine asynchrone Operation ausgeführt. Sonst eine synchrone.
ncb_retcode	1	Return code field. FFH = Command pending 00H = Command completed other values = error
ncb_lsn	1	Local session number field.
ncb_num	1	Local adapter name.
ncb_buffer	4	Pointer to message buffer.
ncb_length	2	Buffer length field.
ncb_callname	16	Local or remote adapter name.
ncb_name	16	Name in local adapter name table.
ncb_rto	1	Receive timeout value, in 500-millisecond intervals.
ncb_sto	1	Send timeout not used.
ncb_post	4	Pointer to post (callback) routine.
ncb_lana_num	1	Adapter number for first adapter. Usually zero.
ncb_cmd_cplt	1	Command status field. FFH = Command pending 00H = Command completed other values = error
ncb_reserve	14	Reserved field. Must be present.

11.23.1 NETBIOS Emulator Commands

Die meisten commands sind in synchroner und asynchroner Form vorhanden.

General Commands	Reset	Reset local adapter.
	Cancel	Cancel previously issued command.
	Adapter Status	Obtain adapter status of local or remot node.
Name Support	Add Name	Add a name to local adapter name table.
	Add Group Name	Add group name to local adapter name table.
	Find Name	Find the location of an adapter name on the network.
	Delete Name	Delete name from local adapter name table.
Session Support	Call	Create a virtual circuit with local or remote node.
	Listen	Listen for connect request from another node.
	Hang Up	Close the virtual circuit.
	Send	Send data on the virtual circuit.
	Chain Send	Send two buffers together on the virtual circuit.
	Receive	Receive data on a virtual circuit.
	Receive Any	Receive data from any virtual circuit.
	Session Status	Receive status on all active sessions.
Datagram Support	Send Datagram	Send Datagram message to name or group name.
	Send Broadcast Datagram	Send broadcast message to any name or group name.
	Receive Datagram	Receive Datagram message.
	Receive Broadcast Datagram	Receive broadcast Datagram from any name.

Reset

Dieser command reseted den localen status und cleared die name und session tables.

Command code:	32H - Synchronous form only.
Input fields:	ncb_lsn number of sessions supported. ncb_num number of command blocks supported. ncb_lana_num number of the adapter to reset.
Output fields:	ncb_retcode.
Final return codes:	00H - success. 03H - invalid command. 23H - invalid number in ncb_lana_num. 4XH - network error. 80H + nnH - DECnet error.

Cancel

Dieser command hebt einen vorhergehenden auf, wenn der noch nicht abgearbeitet wurde.

Command code:	35H - Synchronous form only.
Input fields:	ncb_lana_num ncb_buffer contains the address of the NCB to cancel.
Output fields:	ncb_retcode
Final return codes:	00H - success. 03H - invalid command. 23H - invalid number in ncb_lana_num. 24H - command completed during cancel. 26H - the requested command is invalid to cancel. 4XH - network error. 80H + nnH - DECnet error.

Adapter Status

Mit diesem command kann man den Status des localen oder den eines remote Adapters erhalten. Der Status wird in den Buffer geschrieben. Er muß mindestens 60 Byte lang sein. Das Maximum ist 348.

Command code:	33H - Synchronous form. B3H - Asynchronous form.
Input fields:	ncb_buffer the address of the status. ncb_length the length of the status buffer supplied to the call. ncb_callname is either a remote name or an * for local. ncb_post address of the callback routine. ncb_lana_num
Output fields:	ncb_length updated. ncb_retcode
Final return codes:	00H - success. 01H - illegal buffer length. 03H - invalid command. 05H - command timed out. 06H - buffer too small. 0BH - command canceled. 19H - duplicate name found. 21H - interfac busy. 22H - maximum number of outstanding commands. 23H - invalid number in ncb_lana_num. 4XH - network error. 80H + nnH - DECnet error.

Add Name

Mit diesem command wird ein name in die local adapter name table eingetragen.

Command code:	30H - Synchronous form. B0H - Asynchronous form.
Input fields:	ncb_name the name to be added. ncb_post address of the callback routine (async.). ncb_lana_num
Output fields:	ncb_num used to reference the name for datagram and receive any commands. ncb_retcode
Final return codes:	00H - success. 03H - invalid command. 0DH - name already in local name table. 0EH - local name table full. 15H - name begins with * or 00H. 16H - name being used by remote system. 19H - duplicate name found. 21H - interfac busy. 22H - maximum number of outstanding commands. 23H - invalid number in ncb_lana_num. 4XH - network error. 80H + nnH - DECnet error.

Add Group Name

Mit diesem Befehl wird ein 16-character group name in der adapter name table abgelegt.

Command code:	36H - Synchronous form. B6H - Asynchronous form.
Input fields:	ncb_name the name to be added. ncb_post address of the callback routine (async.). ncb_lana_num
Output fields:	ncb_num used to reference the name for datagram commands. ncb_retcode
Final return codes:	00H - success. 03H - invalid command. 0DH - name already in local name table. 0EH - local name table full. 15H - name begins with * or 00H. 16H - name being used by remote system. 19H - duplicate name found. 21H - interfac busy. 22H - maximum number of outstanding commands. 23H - invalid number in ncb_lana_num. 4XH - network error. 80H + nnH - DECnet error.

Find Name

Mit diesem Befehl wird der Platz im Netzwerk für den 16 character name gefunden.

Command code:	F8H - Synchronous form. 78H - Asynchronous form.
Input fields:	ncb_lana_num ncb_length length of data in ncb_buffer ncb_buffer ncb_post ncb_callname
Output fields:	ncb_retcode ncb_reserve ncb_length
Final return codes:	00H - success. 03H - invalid command. 21H - interfac busy. 22H - maximum number of outstanding commands. 23H - invalid number in ncb_lana_num. 4XH - network error. 80H + nnH - DECnet error.

Delete Name

Mit diesem Befehl wird der 16 character name aus der local adapter name table ausgetragen.

Command code:	31H - Synchronous form. B1H - Asynchronous form.
Input fields:	ncb_name the name being deleted. ncb_post the address of a callback routine. ncb_lana_num
Output fields:	ncb_retcode

Final return codes:	00H - success.
	03H - invalid command.
	0FH - command completed, name has active sessions;
	the name is flagged as de-registered.
	15H - name not found.
	21H - interfac busy.
	22H - maximum number of outstanding commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Call

Mit diesem Befehl wird ein virtual circuit erzeugt mit einem local oder remote name.

Command code:	10H - Synchronous form.
	90H - Asynchronous form.
Input fields:	ncb_callname the 16-character name being called.
	ncb_name the local adapter name.
	ncb_rto the receive time out to be used in 500 ms intervals.
	ncb_sto not supported.
	ncb_post address of callback routine.
	ncb_lana_num
Output fields:	ncb_retcode
	ncb_lsn the virtual circuit ID to be used for subsequent
	commands relating to this circuit.
Final return codes:	00H - success.
	03H - invalid command.
	05H - connect timed out.
	09H - no resource available.
	0BH - command canceled.
	11H - maximum number of sessions.
	12H - remote system rejected connection.
	14H - called name unknown.
	15H - invalid name.
	19H - name conflict detected.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Listen

Mit diesem Befehl kann man auf eine Verbindungsanfrage eines anderen Adapters hören.

Command code:	11H - Synchronous form.
	91H - Asynchronous form.
Input fields:	ncb_callname either an asterisk if the listen is for
	anyone, or a specific adapter name.
	ncb_name the local name for which the listen is being issued.
	ncb_rto a timeout in 500 ms intervals.
	ncb_sto not used.
	ncb_post address of callback routine.
	ncb_lana_num

Output fields:	ncb_lsn number of virual circuit, 1 to 254.
	ncb_callname the name that connected.
	ncb_retcode
Final return codes:	00H - success.
	03H - invalid command.
	0BH - command canceled.
	11H - maximum number of sessions.
	15H - name was not in local table.
	17H - name deleted.
	19H - name conflict detected.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Hang Up

Mit diesem Befehl wird eine Verbindung zwischen zwei Adaptern aufgelöst.

Command code:	12H - Synchronous form.
	92H - Asynchronous form.
Input fields:	ncb_lsn specifies the virtual circuit to hung up.
	ncb_post address of callback routine.
	ncb_lana_num
Output fields:	ncb_retcode
Final return codes:	00H - success.
	03H - invalid command.
	08H - invalid virtual circuit.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Send

Mit diesem Befehl können Daten über ein virtual circuit das im ncb_field eines CALL- oder LISTEN-Befehls angegeben wurde gesendet werden.

Command code:	14H - Synchronous form.
	94H - Asynchronous form.
Input fields:	ncb_lsn specifies the virtual circuit on which to send.
	ncb_buffer address of the data to send.
	ncb_length length of data in ncb_buffer.
	ncb_post address of callback routine.
	ncb_lana_num
Output fields:	ncb_retcode

Final return codes:	00H - success.
	03H - invalid command.
	08H - illegal virtual circuit number.
	0AH - virtual circuit closed.
	0BH - command canceled.
	18H - virtual circuit aborted.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	80H + nnH - DECnet error.

Chain Send

Mit diesem Befehl kann ein Datenbuffer an den Vorgänger zur Übertragung angehängt werden.

Command code:	17H - Synchronous form.
	97H - Asynchronous form.
Input fields:	ncb_lsn specifies the virtual circuit on which to send.
	ncb_buffer address of first buffer.
	ncb_length length of first buffer.
	ncb_callname contains the following:
	2 byte - length of second buffer.
	4 byte - address of second buffer.
	ncb_post address of callback routine.
	ncb_lana_num
Output fields:	ncb_retcode
Final return codes:	00H - success.
	03H - invalid command.
	08H - illegal virtual circuit number.
	18H - session aborted.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Receive

Mit diesem Befehl können Daten von einem angegebenen virtual circuit angenommen werden.

Command code:	15H - Synchronous form.
	95H - Asynchronous form.

Input fields:	ncb_lsn specifies the virtual circuit
	on which to receive.
	ncb_buffer address receive data buffer.
	ncb_length size of the receive buffer.
	ncb_post address of callback routine.
	ncb_lana_num

Output fields:	ncb_retcode
	ncb_length the number of bytes returned.

Final return codes:	00H - success.
	03H - invalid command.
	05H - command timed out.
	06H - message incomplete.
	08H - illegal virtual circuit number.
	0AH - session closed.
	0BH - command canceled.
	18H - session ended abnormally.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Receive Any

Mit diesem Befehl können Daten von allen virtual circuit Verbindungen empfangen werden.

Command code:	16H - Synchronous form.
	96H - Asynchronous form.
Input fields:	ncb_buffer address receive data buffer.
	ncb_length length of receive buffer.
	ncb_num virtual circuit number, or FFH if any circuit.
	ncb_post address of callback routine.
	ncb_lana_num
Output fields:	ncb_lsn the virtual circuit number
	that the data was received on.
	ncb_num the adapter name number if FFH was specified
	as the input value for this field.
	ncb_retcode
Final return codes:	00H - success.
	03H - invalid command.
	06H - message incomplete.
	0AH - session closed.
	0BH - command canceled.
	13H - illegal name number.
	17H - name deleted.
	18H - session ended abnormally.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Session Status

Mit diesem Befehl kann der Status für einen angegebenen name von allen aktiven sessions erhalten werden.

Command code:	34H - Synchronous form.
	B4H - Asynchronous form.
Input fields:	ncb_buffer address of data buffer.
	ncb_length size of buffer
	ncb_name either a specific name or ∗.
	ncb_post address of callback routine.
	ncb_lana_num.
Output fields:	ncb_length returns the number of bytes written to the buffer.
	ncb_retcode
Final return codes:	00H - success.
	01H - illegal buffer length.
	03H - invalid command.
	06H - message incomplete.
	15H - name not found.
	19H - name conflict detected.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Send Datagram

Mit diesem Befehl kann eine datagram message zu einem einzelnen oder einem Gruppen name gesendet werden, wenn sie einen Befehl receive datagram vorher abgesetzt hatten.

Command code:	20H - Synchronous form.
	A0H - Asynchronous form.
Input fields:	ncb_buffer address of data buffer to transmit.
	ncb_length number of bytes to transmit.
	ncb_num is the local name number that is requesting the send datagram.
	ncb_callname specifies either the 16-byte name or group name to which the message is being sent.
	ncb_post address of callback routine.
	ncb_lana_num.
Output fields:	ncb_retcode.
Final return codes:	00H - success.
	01H - illegal buffer length.
	03H - invalid command.
	13H - illegal name number.
	19H - name conflict detected.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Send Broadcast Datagram

Mit diesem Befehl kann eine message an alle gesendet werden, die den Befehl receive broadcast message vorher abgesetzt hatten.

Command code:	22H - Synchronous form.
	A2H - Asynchronous form.
Input fields:	ncb_buffer address of data buffer to transmit.
	ncb_length number of bytes to transmit.
	ncb_num is the local name number issuing the message.
	ncb_post address of callback routine.
	ncb_lana_num.
Output fields:	ncb_retcode.
Final return codes:	00H - success.
	01H - illegal buffer length.
	03H - invalid command.
	13H - illegal name number.
	19H - name conflict detected.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Receive Datagram

Mit diesem Befehl kann jede datagram message empfangen werden.

Command code:	21H - Synchronous form.
	A1H - Asynchronous form.
Input fields:	ncb_buffer address of the buffer into which the received data is placed.
	ncb_length is the size of the receive buffer.
	ncb_num is the local name number for the receive.
	ncb_post address of callback routine.
	ncb_lana_num.
Output fields:	ncb_length the size of the received message.
	ncb_callname the name of the message originator.
	ncb_retcode.
Final return codes:	00H - success.
	03H - invalid command.
	06H - message incomplete.
	0BH - command canceled.
	13H - illegal name nunber.
	17H - name deleted.
	19H - name conflict detected.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

Receive Broadcast Datagram

Mit diesem Befehl kann eine message von jedem empfangen werden der einen Befehl send broadcast datagram abgesetzt hat.

Command code:	23H - Synchronous form.
	A3H - Asynchronous form.
Input fields:	ncb_buffer address of data buffer.
	ncb_length is the size of the receive buffer.
	ncb_num is the local name number issuing the request.
	ncb_post address of callback routine.
	ncb_lana_num.
Output fields:	ncb_length number of bytes returned in buffer.
	ncb_callname the name of the message originator.
	ncb_retcode.
Final return codes:	00H - success.
	03H - invalid command.
	06H - message incomplete.
	0BH - command canceled.
	13H - illegal name number.
	17H - name deleted.
	19H - name conflict detected.
	21H - interface busy.
	22H - too many commands.
	23H - invalid number in ncb_lana_num.
	4XH - network error.
	80H + nnH - DECnet error.

11.24 Interrupt Nr. 60H Netze, PC_IPC, SYS_PROF

11.25 Interrupt Nr. 63H, 64H, 65H, 66H werden von T^3 benutzt

Durch die Benutzung von Interrupt 64h kann T^3 nicht zusammen mit einem Novell-Netzwerk laufen!

11.26 Interrupt Nr. 64H Novell Low-Level API

11.27 Interrupt Nr. 67H LIM EMS VCPI Driver

Expanded Memory Manager definiert von Lotus, Intel und Mocrosoft. Hat "Untermieter" bekommen, z. B. VCPI (Virtual Control Program Interface).

11.27.1 Funktion 40H Get Manager Status

Der Status der Hardware und Software wird im Register AH zurück gegeben.

11.27.2 Funktion 41H Get Page Frame Segment

Im Register BX steht die Adresse des EMM page frame, wenn im Statusregister AH 00H ist.

11.27.3 Funktion 42H Get Expanded Memory Pages

Im Register BX steht die Zahl der nicht belegten EMM Seiten, im Register DX steht die Zahl der vorhandenen Seiten.

11.27.4 Funktion 43H Allocate Expanded Memory

Wenn der Status (AX=00H) null ist, wird im Register DX die handle zurückgegeben. Im Register BX muß die Zahl der logischen Seiten stehen, die belegt werden sollen. Diese Funktion ist vergleichbar mit einem file-open.

11.27.5 Funktion 44H Map Memory

Diese Funktion mapped eine der logischen Seiten in eine der vier EMM Seiten.

- AL = physical page (0 - 3)
- BX = logical page (0 - n-1)
- DX = EMM handle

11.27.6 Funktion 45H Release Handle and Memory

Im Register DX steht die EMM handle, die aufgegeben werden soll. Diese Funktion ist einem file-close zu vergleichen.

11.27.7 Funktion 46H Get EMM Version

Wenn der Status (AH=00H) null ist so steht im Register AL die EMM Version. Die höherwertigen 4Bit geben in BCD die Hauptversion und der Rest, auch in BCD, die Unterversion an.

11.27.8 Funktion 47H Save Mapping Context

Diese Funktion wird von Interrupt Handlern und residenten Treibern benutzt wenn sie beim initialisieren das expanded memory benötigen.

11.27.9 Funktion 48H Restore Mapping Context

Diese Funktion wird von Interrupt Handlern und residenten Treibern benutzt wenn sie vorher die Funktion 47H benutzt haben.

11.27.10 Funktion 4BH Get Number of EMM Handles

Wenn der Status (AH=00H) null ist steht im Register BX die Anzahl der aktiven handles. Diese Zahl ist maximal 255.

11.27.11 Funktion 4CH Get Pages Owned by Handle

Im Register BX steht die Zahl der logischen Seiten der im Register DX angegebenen handle. Die Zahl ist zwischen 1 und 512 da eine EMM handle nie keine Seite belegt haben kann.

11.27.12 Funktion 4DH Get Pages for all Handles

Diese Funktion schreibt in ein Array die Zahl der aktiven handles und die dazugehörenden Seiten. Da die Zahl der handles nicht größer als 255 werden kann, reicht ein Array mit 1020 Byte aus. Jeweils das erste Wort eines Doppelworteintrags beinhaltet die handle, das zweite die Anzahl der Seiten.

- Call:
  - DI = offset of array to receive information
  - ES = array segment
- Returns:
  - AH = Status
  - BX = number of active EMM handles

11.27.13 Funktion 4EH Get/Set Page Map

Diese Funktion wurde bei der EMM Version 3.2 hinzugefügt. Sie soll multitasking unterstützen und nicht vom Benutzer gebraucht werden.

- Unterfunktion 00H = get mapping registers into array
- Unterfunktion 01H = set mapping registers into array
- Unterfunktion 02H = get and set mapping registers in one operation

- Unterfunktion 03H = return needed size of page-mapping array
- Call:
  - AL = Unterfunktionsnummer
  - DS:DI = Array holding mapping information (01H,02H)
  - ES:DI = Array to receive information (00H,02H)
- Returns:
  - AH = status
  - AL = bytes in page-mapping array (03H)
  - Array pointed to by ES:DI receives mapping information for subfunctions 00H and 02H.

11.27.14 Funktion 4FH Get partial page map

Unterfunktion 00H

Sichere nur den Zustand der physikalischen Seiten, deren Segmentadressen in der Tabelle angegeben sind. Die Register DS:SI zeigen auf ein Feld dessen erstes Wort die Anzahl der folgenden Einträge zeigt. Das folgende Wort-Array enthält die Segmentadressen der zu sichernden Seiten. Die Register ES:DI zeigen auf ein Zustandsfeld, dessen Größe mit der Unterfunktion 02H ermittelt werden kann.

Unterfunktion 01H

Den Zustand der Hardware entsprechend der in dem Zustandsfeld abgelegten Daten wiederherstellen. In den Registern DS:DI steht ein Zeiger, der auf ein Zustandsfeld zeigt, das zuvor mit der Unterfunktion 00H erstellt wurde.

Unterfunktion 02H

Diese Unterfunktion ermittelt die Größe eines Zustandsfeldes, das Platz für die angegebene Anzahl Seiten bietet. Im Register BX steht die Anzahl der zu sichernden physikalischen Seiten. Danach ist im Register AL die Größe eines entsprechenden Zustandsfeldes.

11.27.15 Funktion 50H Map/unmap multiple handle pages

Unterfunktion 00H

Ein- oder Ausblenden mehrerer logischer Seiten in physikalische Seiten. Im Gegensatz zur Funktion 44H können mehrere Seiten auf einmal eingeblendet werden.

- CX = Anzahl der Einträge in der Tabelle
- DX = Handle
- DS:DI = Zeiger auf Array mit folgender Struktur:
  - Wort = logische Seitennummer der einzublendenden Seite
  - Wort = physikalische Seite

Unterfunktion 01H

Ein- oder Ausblenden mit Angabe der Segmentadressen.

- CX = Anzahl der Einträge in der Tabelle
- DX = Handle
- DS:DI = Zeiger auf Array mit folgender Struktur:
  - Wort = logische Seitennummer der einzublendenden Seitennummer
  - Wort = Segmentadresse der physikalischen Seite

11.27.16 Funktion 51H Reallocate pages

Ändern der durch ein Handle belegten logischen Seiten. Auf diese Weise kann die Menge der benutzten EMS-Seiten während des Programmlaufs dynamisch geändert werden. Erlaubt ist auch die Reservierung von null Seiten. Das Handle wird dabei nicht an den EMM zurückgegeben, sondern bleibt dem Programm zur Verfügung.

- BX = Neue Anzahl der unter diesem Handle zu reservierenden Seiten.
- DX = Handle
- BX = Anzahl der tatsächlich unter diesem Handle zur Verfügung stehenden Seiten.

11.27.17 Funktion 52H Get handle attribute

Unterfunktion 00H

Ermitteln der Resetfestigkeit eines Handlers. Falls das Handle resetfest ist, übersteht es einen Neustart des Systems unbeschadet. Im Register AL steht dann die Kennung mit 0 für nicht resetfest und mit 1 für resetfest. Im Register DX muß die Handle stehen.

Unterfunktion 01H

Mit der Unterfunktion 01H kann man die Resetfestigkeit eines Handles setzen. Im Register DX steht die Handle und im Register BL steht 0 für nicht resetfest oder 1 für resetfest.

Unterfunktion 02H

Ermitteln, ob der EMM und die dazugehörige Hardware resetfeste Handles unterstützt. Nicht alle Karten unterstützen diese Fähigkeit, die sehr stark vom Design der Hardware abhängt (eigener refresh u.a.). Der Treiber gibt dann im Register AL die Kennung zurück mit 0 für Resetfestigkeit wird nicht unterstützt oder mit 1 für Resetfestigkeit wird unterstützt.

11.27.18 Funktion 53H Get handle name

Unter LIM4.0 ist es möglich, einem Handle auch einen Namen zu geben, der aus acht Zeichen besteht.

Unterfunktion 00H Get name

Diese Unterfunktion legt den Namen im mit ES:DI angegebenen Speicherplatz ab. Im Register DX steht die Handle Nummer.

Unterfunktion 01H Set name

Die Register DS:DI zeigen auf einen 8-Byte langen Namen (mit null aufgefüllt). Im Register DX steht die Handle Nummer. Ein Name kann nicht mehrmals an unterschiedliche Handles vergeben werden.

11.27.19 Funktion 54H Get handle directory

Die Größe eines Eintrags ist 10 Byte (1 Wort für die Handle und 8 Byte für den Namen). Die maximale Größe der Tabelle beträgt, da maximal 255 Handles zugelassen, 2550 Byte.

Unterfunktion 00H Get handle directory

In den Registern ES:DI steht ein Zeiger auf ein Array mit der directory Struktur. In dem Register AL steht danach die Anzahl der offenen Handles.

Unterfunktion 01H Search for named handles

In den Registern DS:DI steht ein Zeiger, der auf einen 8 Byte langen Namen weist. Danach steht in dem Register DX die Handle für den angegebenen Namen.

Unterfunktion 02H Get total handles

In dem Register BX steht danach die maximale Anzahl Handles, inklusive Systemhandle 0, die von dem EMM unterstützt werden.

11.27.20 Funktion 55H Alter page map and jump

Die in dem Datenfeld angegebenen logischen Seiten einblenden und danach die angegebene Adresse mittels JMP FAR Befehls anspringen. Der alte Zustand des Fensters geht durch das Einblenden verloren und muß gegebenenfalls vorher explicit gesichert werden. Es dürfen auch null Seiten ein-/ausgeblendet werden. Alle Register, soweit sie nicht vom EMM benötigte Parameter enthalten, werden beim Anspringen der angegebenen Adresse nicht verändert. Die Flags werden ebenfalls nicht verändert.

- AL = 0 Die jeweilige physikalische Seite wird über ihre Nummer angesprungen.
- AL = 1 statt dessen über die Segmentadresse die physikalische Seite ansprechen.
- DX = Handle
- DS:SI = Zeiger auf folgende Struktur:
  - DD?= Adresse der anzuspringenden Routine
  - DB?= Anzahl der Einträge im Array
  - DD?= Zeiger auf das Array mit folgender Struktur:
    - DW?= Nummer der logischen Seite, die einzublenden ist.
    - DW?= Nummer oder Segmentadresse der physikalischen Seite, in die die logische Seite einzublenden ist.

11.27.21 Funktion 56H Alter page map and call

Die in dem Datenfeld angegebenen logischen Seiten einblenden und danach die angegebene Adresse mittels CALL FAR aufrufen. Bei der Rückkehr wird der Zustand des Fensters mit Hilfe des Zeigers auf die wiederherzustellenden Seiten restauriert. Zu beachten ist, daß der EMM selbst Daten auf dem Stack ablegt, die er für die Rückkehr benötigt. Dieser zusätzliche Speicherbedarf ist daher zu berücksichtigen. Die Rückkehr aus der aufgerufenen Routine erfolgt mittels RETF. Analog der Funktion 55H bleiben die Register, die keine Parameter enthalten, sowie die Flags erhalten.

- AL =
  - 0= Die jweilige physikalische Seite wird über ihre Nummer angesprochen.
  - 1= statt dessen über die Segmentadresse die physikalische Seite ansprechen.
- DX = Handle
- DS:SI = Zeiger auf folgende Struktur:
  - DD?= Adresse der anzuspringenden Routine.
  - DB?= Anzahl der Einträge im Array mit den neu einzublendenden Seiten.
  - DD?= Zeiger auf dieses Array.
  - DB?= Anzahl der Einträge im Array mit den nach RETF wieder einzublendenden Seiten.
  - DD?= Zeiger auf dieses Array.
  - DW?= Nummer der logischen Seite.
  - DW?= Nummer oder Segmentadresse der physikalischen Seite.

Unterfunktion 02H Get page map stack space size

Mit dieser Unterfunktion kann der Stackverbrauch erfragt werden. In dem Register BX steht die Anzahl der Bytes, die auf dem Stack abgelegt wurden einschlieslich der Rückkehradresse.

11.27.22 Funktion 57H Move memory region

Verschieben bzw. Austauschen von Speicherbereichen, die sich sowohl im EMS-Speicher als auch im konventionellen Speicher befinden dürfen. Der Vorteil dieser Funktionen liegt darin, daß der Fensterkontext nach der Ausführung automatisch wiederhergestellt wird, keine Ein-/Ausblende-Aktionen vom Programm auszuführen sind und auch Speicher einfach innerhalb des EMS-Speichers kopiert bzw. ausgetauscht werden kann.

- AL =
  - 00H = move
  - 01H = exchange
- DS:SI = Zeiger auf folgende Struktur:
  - DD?= Länge des Bereichs in Byte, der verschoben werden soll.
  - DB?= Speichertype der Quelle:
    * 0 = konventionell
    * 1 = expanded
  - DW?= Handle, falls Quelle = expanded, ansonsten auf null setzen.

  - DW?= Offset des Quellbereichs (0 bis 3FFFH bzw. 0FFFFH).
  - DW?= Segmentadresse der Quelle, falls im konventionellen Speicher, ansonsten logische Seitennummer.
  - DB?= Typ des Zielbereichs:
    * 0 = konventionell
    * 1 = expanded
  - DW?= Handle, falls Ziel = expanded.
  - DW?= Offset des Zielbereichs.
  - DW?= Segmentadresse bzw. logische Seitennummer des Zielbereichs.

11.27.23 Funktion 58H Get mappable physical address array

Unterfunktion 00H

Eine Tabelle mit den Segmentadressen aller vorhandenen physikalischen Seiten mit der dazugehörigen Seitennummer im durch ES:DI bezeichneten Speicherbereich ablegen. Die Tabelle ist aufsteigend nach Segmentadressen sortiert (und **nicht** nach Seitennummern!).

- ES:DI = Zeiger auf ein Feld mit der folgenden Struktur:
  - DW? = Segmentadresse einer physikalischen Seite.
  - DW? = Seitennummer der physikalischen Seite.
- CX = Anzahl der Einträge in der Tabelle.

Unterfunktion 01H

Get mappable physical address array entries. Anzahl der Tabelleneinträge für die Funktion 5800H ermitteln. Diese Zahl mit der Größe der Struktur (4) multipliziert ergibt die Größe des benötigten Speicherplatzes.
CX = Anzahl der Einträge in der Tabelle.

11.27.24 Funktion 59H Get hardware configuration array

Unterfunktion 00

Struktur mit der Beschreibung der Expanded-Memory-Hardware im Speicher ab ES:DI ablegen. Diese Funktion kann jederzeit durch das Betriebssystem gesperrt werden.

- ES:DI = Zeiger auf folgende Struktur:
  - DW? = Größe der intern verwendeten Seiten (raw pages) in Paragraphen.
  - DW? = Anzahl der zusätzlichen Registersätze zum Ein-/Ausblenden (mapping register set).
  - DW? = Größe eines Speicherbereichs, der den Zustand der Hardware (mapping context) aufnehmen kann. Der Wert entspricht dem der Funktion 4E03H.
  - DW? = Anzahl der Registersätze, die mit DMA-Kanälen gekoppelt werden können. Diese Registersätze ermöglichen DMA-Transfer direkt in den EMS-Speicher, während das EMS-Fenster von der Software für andere Dinge belegt ist.
  - DW? =
    * 0 = DMA-Registersätze verhalten sich so, wie durch Funktion 5BxxH beschrieben.
    * 1 = Nur ein DMA-Registersatz, alle Kanäle werden damit gekoppelt.

Unterfunktion 01H

Get unallocated page count. Die EMS-Hardware kann auch kleinere Seiten intern verwenden, die kleiner als 16 KByte sind. Nach außen hin sind jedoch nur Standardseiten von 16 KByte Größe sichtbar. Ein Betriebssystem kann jedoch zum Ein-/Auslagern von Programmen auch kleinere Seitengrößen verwenden.

- BX = Anzahl der momentanen freien Subseiten.
- DX = Anzahl der Subseiten insgesamt.

11.27.25 Funktion 5AH Allocate pages

Unterfunktion 00H allocate standard pages

Analog der Funktion 43H Speicherplatz im EMS-Speicher resrvieren. Bei dieser Funktion ist es aber erlaubt, auch null Seiten zu reservieren.

- BX = Anzahl der zu reservierenden Seiten (16 KByte).
- DX = Handle

Unterfunktion 01H allocate raw pages

Ähnlich der Funktion 5A00H, nur daß nun logische Seiten reserviert werden, deren Größe ein Teil von 16 KByte beträgt. Der EMM erkennt dann selbständig an dem Handle, ob die gewählte Funktion mit Standardseiten oder Subseiten arbeitet.

- BX = Anzahl der zu reservierenden Subseiten.
- DX = Handle

11.27.26 Funktion 5BH Alternate map register set

Unterfunktion 00H get alternate map register set

Zeiger auf einen Speicherbereich mit gesichertem Hardware-Zustand holen bzw. Nummer des benutzten alternativen Registersatzes.

Wenn das Register BL gleich 0 ist, ist in den Registern ES:DI der Zeiger.

Ist das Register BL ungleich 0, ist es die Nummer des alternativen Registersatzes.

Unterfunktion 01H set alternate map register set

Alternativen Registersatz aktivieren oder einen alten Hardware-Zustand wiederherstellen. Ist der alternative Registersatz, angegeben im Register BL, nicht die Nummer Null, so wird für das Ein-/Ausblenden der angegebene Registersatz benutzt, ansonsten liefert der durch ES:DI adressierte Speicherbereich, der zuvor mit der Funktion Get Page Map initialisiert wurde, die notwendigen Daten.

Unterfunktion 02H get alternate map save array size

Größe des Speicherbereichs für die Funktion 5BxxH ermitteln. Der Wert in Bytes steht in dem Register DX.

Unterfunktion 03H allocate alternate map register set

Alternativen Registersatz anfordern. Falls keine zusätzlichen Registersätze unterstützt werden, ist BL = 0. In diesem Fall sollte die "get alternate map"-Funktion aufgerufen werden.

Unterfunktion 04H deallocate alternate register set

Den im Register BL angegebenen Registersatz dem EMM wieder zur Verfügung stellen. Der Registersatz Null, der Standardsatz, kann nicht freigegeben werden.

Unterfunktion 05H allocate DMA register set

Einen DMA-Registersatz vom EMM anfordern. Die Nummer des DMA-Registersatzes steht dann in dem Register BL. Mit diesen DMA-Registersätzen sind DMA-Übertragungen in den EMS-Speicher möglich, ohne daß die betroffenen Seiten auch im Fenster eingeblendet sein müssen.

Unterfunktion 06H enable DMA on alternate map register set

Einen bestimmten DMA-Kanal mit einem DMA-Registersatz koppeln.

- BL = DMA-Registersatz
- DL = DMA-Kanal

Unterfunktion 07H disable DMA on alternate map register set

DMA-Übertragung mit einem Registersatz wieder sperren. Der DMA-Registersatz wird in dem Register BL angegeben.

Unterfunktion 08H deallocate DMA register set

Einen zuvor mit der Funktion 5B03H reservierten DMA-Registersatz wieder an den EMM zurückgeben. Der DMA-Registersatz wird in dem Register BL angegeben.

11.27.27 Funktion 5CH Prepare expanded memory hardware for warm boot

Bereitet die EMS-Hardware auf einen Warmstart vor.

11.27.28 Funktion 5DH OS/E function set

Unterfunktion 00H Enable OS/E function set

Freigabe der Ausführung der Funktionen 59xxH, 5BxxH sowie 5DxxH. Mit dieser und der nächsten Funktion kann ein(e) Betriebssystem/Umgebung den Zugriff anderen Programmen verwehren, so daß es nicht zu Problemen z. B. beim Multitasking (via Ein-Ausblenden von EMS-Seiten im Bereich unterhalb der 640-KByte-Grenze) kommt.

- Ein:BX,CX = Schlüssel (**nicht** jedoch beim ersten Aufruf!)
- Aus:BX,CX = Schlüssel (nur beim ersten Aufruf!)

Unterfunktion 01H disable OS/E function set

Sperren der Ausführung der Funktionen 59xxH, 5BxxH und 5DxxH.

- Ein:BX,CX = Schlüssel (**nicht** jedoch beim ersten Aufruf!)
- Aus:BX,CX = Schlüssel (nur beim ersten Aufruf!)

Unterfunktion 02H return access key

Rückgabe des Zugriffschlüssels für die Funktionen 5D00H und 5D01H an den EMM. Nach dem Aufruf stehen diese beiden Funktionen nach dem Motto "Wer zuerst kommt ... " dem nächsten zur Verfügung. Der Schlüssel steht in den Registern BX und CX.

11.27.29 Funktion DEH (VCPI virtual control program interface)

Unterfunktion 00H VCPI Presense Detection

- Ein: —
- Aus: Falls VCPI vorhanden ist, ist AH = 0 und BH = Versionsnummer, BL = Unterversion.

Unterfunktion 01H Get Protected Mode Interface

- Ein: ES:DI = Zeiger auf einen 4 KByte großen Buffer für eine Seitentabelle.
  DS:SI = Zeiger auf drei Einträge in der GDT des Klienten.
- Aus: DI = Zeiger auf den ersten nicht mehr durch diese Funktion initialisierten Eintrag in der Seitentabelle.
  EBX = Offset des Einsprungpunkts in den VCPI-Code für den Protected Mode.

Unterfunktion 02H Get Maximum Physical Mamory Address

- Ein: —
- Aus: EDX = Physicalische Adresse der höchsten 4KByte-Seite, die der Server jemals reservieren wird.

Unterfunktion 03H Get Number of Free 4K Pages

- Ein: —
- Aus: EDX = Anzahl freier Seiten.

Unterfunktion 04H Allocate a 4K Page

- Ein: —
- Aus: EDX = Physikalische Adresse der reservierten Seite.

Unterfunktion 05H Free a 4K Page

- Ein: EDX = Physikalische Adresse der zurückzugebenden Seite.
- Aus: —

Unterfunktion 06H Get Physical Address of 4K Page in first Megabyte

- Ein: CX = Seitennummer.
- Aus: EDX = Physikalische Adresse der Seite.

Unterfunktion 07H Read CR0

- Ein: —
- Aus: EBX = Inhalt des CR0-Registers.

Unterfunktion 08H Read Debug Registers

- Ein: ES:DI = Zeiger auf 8 DWORDs zur Aufnahme der Inhalte der Debugging-Register.
- Aus: —

Unterfunktion 09H Load Debug Registers

- Ein: ES:DI = Zeiger auf 8 DWORDs, die die neuen Inhalte für die Debugging-Register enthalten.
- Aus: —

Unterfunktion 0AH Get 8259A Interrupt Vector Mappings

- Ein: —
- Aus: BX = erster Interruptvektor des Master-Interrupt-Controllers.
  CX = erster Interruptvektor des Slave-Interrupt-Controllers.

Unterfunktion 0BH Set 8259A Interrupt Vector Mappings

- Ein: BX = erster Interruptvektor des Master-Interrupt-Controllers.
  CX = erster Interruptvektor des Slave-Interrupt-Controllers.
- Aus: —

Unterfunktion 0CH Switch From V86 Mode to Protected Mode

- Ein: ESI = lineare Adresse der folgenden Datenstruktur unterhalb der 1-MByte-Grenze:

  DWORD ?= neuer Inhalt für CR3.

  DWORD ?= lineare Adresse (unterhalb 1 MB) des 6-Byte-Wertes (PWORD), der in das GDTR geladen wird.

  DWORD ?= lineare Adresse (unterhalb 1 MB) des 6-Byte-Wertes (PWORD), der in das IDTR geladen wird.

  WORD ?= Selektor für LDTR.

  WORD ?= Selektor für TR.

  PWORD ?= im Protected Mode anzuspringende Zieladresse. Interrupts gesperrt.
- Aus: —

Unterfunktion 0DH Switch From Protected Mode to V86 Mode

- Ein: Selektor eines Datensegments, das bei 0 beginnt und mindestens so groß ist, wie der durch die Funktion 01H initialisierte Adressraum in der Seitentabelle.

  SS:ESP = Zeiger auf folgende Struktur unterhalb der 1-MByte-Grenze:

  QWORD ?= Rückkehradresse (nicht benötigt).

  DWORD ?= EIP

  DWORD ?= CS

  DWORD ?= reserviert für EFLAGS

  DWORD ?= ESP

  DWORD ?= SS

  DWORD ?= ES

  DWORD ?= DS

  DWORD ?= FS

  DWORD ?= GS

  Interrupts gesperrt.
- Aus: —

11.27.30 EMM Error Codes

- 00H Function was successful.
- 80H Internal error in the EMM software.
- 81H Malfunction in the expanded memory hardware.
- 82H EMM is busy. Nut Versionen vor LIM 3.2.
- 83H Invalid expanded memory handle.
- 84H Function requested by the application is not supported by the EMM. Erlaubt sind in LIM 4.0 Funktionen von 40H bis 5EH.
- 85H No more expanded memory handles available.

- 86H Error in save or restore of mapping context.
- 87H Allocation request specified more logical pages than are available in the system; no pages were allocated.
- 88H Allocation request specified more logical pages than are currently available in the system (the request does not exeed the physical pages that exist, but some are alredy allocated to other handles); no pages were allocated.
- 89H Zero pages cannot be allocated.
- 8AH Logical page request for mapping is outside the range of pages assigned to the handle.
- 8BH Illegal physical page number in mapping request (not in the range 0-3).
- 8CH Save area for mapping context id full.
- 8DH Save of mapping context failed because save area already contains a context associated with the requested handle.
- 8EH Restore of mapping context failed because save area does not contain a context for the requested handle.
- 8FH Subfunction parameter not defined.
- 90H Die Kennung für die Resetfestigkeit ist ungültig (muß 0 oder 1 sein).
- 91H Die Hardware/Software unterstützt keine resetfesten Handles.
- 92H Der Quell- und Zielbereich des Kopiervorgangs überlappen sich, so daß daher ein Teil der Quelle verloren ging. Die Operation wurde trotzdem vollständig durchgeführt.
- 93H Der Bereich der reservierten Speicherseiten im Expanded Memory wurde von der Quelle oder dem Ziel überschritten.
- 94H Konventioneller Speicher und der Expanded-Memory-Bereich überlappen sich.
- 95H Offset für eine logische Seite liegt außerhalb des erlaubten Bereichs von 0 bis 3FFFH.
- 96H Länge überschreitet die 1-MByte-Grenze.
- 97H Quell- und Zielbereich haben das gleiche Handle und überlappen sich.
- 98H Der Typ eines Quell- bzw. Zielbereichs ist ungültig bzw. wird nicht unterstützt.
- 9AH Der angegebene alternative Map-Register-Satz wird nicht unterstützt.
- 9BH Alle alternativen Map/DMA-Register-Sätze sind momentan vergeben.
- 9CH Alternative Map/DMA-Register-Sätze werden nicht unterstützt, und der angegebene Registersatz ist nicht Nummer Null.
- 9DH Obwohl alternative Map/DMA-Register-Sätze unterstützt werden, ist der angegebene Registersatz entweder nicht definiert, nicht zugeteilt oder der momentan zugeteilte Registersatz.
- 9EH Dedizierte DMA-Kanäle werden nicht unterstützt.
- 9FH Der angegebene dedizierte DMA-Kanal ist ungültig.
- A0H Zu dem angegebenen Namen konnte kein Handle gefunden werden.
- A1H Der vergebene Name existiert bereits.

- A2H Der auszutauschende oder zu verschiebende Bereich würde über die 1-MByte-Grenze an den Speicheranfang überlaufen, daher wurde die Operation nicht ausgeführt.
- A3H Die Daten der angegebenen Struktur sind entweder zerstört oder bedeutungslos.
- A4H Die gewählte Funktion wurde vom Betriebssystem/der Umgebung gesperrt.

11.28 Interrupt Nr. 68H APPC/PC

11.29 Interrupt Nr. 69H Command Terminal Access Interface (DECnet/PCSA multiplex

Die CTERM services werden mit dem Interrupt 69H erreicht. Zuerst muß aber die Größe des Session Control Blocks (SCB) mit der Unterfunktion 0AH festgestellt und mit dem Systemaufruf Interrupt 21H Funktion 48H Memory zugeteilt werden.

11.29.1 Funktion 01H CTERM Funktionen

Unterfunktion	Code
Installation check	00
Send Data	01
Read Data	02
Status	03
DECnet Status	04
Open Session	05
Close Session	06
Get SCB Size	0A
Get DECnet Socket	0B
Reserved	0C
Reserved	0D
Reserved	0E
CTERM Deinstall Driver	0F

Wird der schedular gestartet, so wird der Interrupt 69H belegt. Dieser wird nach dem Starten von DLL an ihn abgegeben und der Interrupt 0BH benutzt (IRQ3).

11.30 Interrupt Nr. 6AH OPTHELP

11.31 Interrupt Nr. 6AH Local Area Transport Access Interface

Die LAT services werden mit dem Interrupt 6AH erreicht. Die Funktion steht im Register AH, eine Unterfunktion gegebenenfalls im Register AL. Im Register DH muß immer FFH stehen. Am Anfang einer Session muß der Benutzer sich mit der 'open_function' eine handle geben lassen und nur diese für diese Session benutzen.

Die LAT Funktionen:

01H	Send Data
02H	Read Data
03H	Get Status
D0H	Open/Close
D1H	Send Break Signal
D2H	Reserved
D3H	Reset LAT Counters
D4H	Copy LAT Counters
D5H	Get Next LAT Service
D6H	LAT Service Table Reset

11.31.1 Funktion 01H Send Data

Mit der Funktion 01H wird das Zeichen im Register AL zum LAT driver geschickt. Im Register DL muß die Handle stehen. Konnte das Zeichen abgeliefert werden, so ist das Register AH leer. Ist das Bit 7 gesetzt, konnte die Funktion das Zeichen nicht in die queue des LAT drivers stecken.

11.31.2 Funktion 02H Read Data

Mit der Funktion 02H wird ein Zeichen von der Handle im Register DL in das Register AL gelesen. Ist in dem Register AH das Bit 7 gesetzt, so wurde kein Zeichen gelesen.

11.31.3 Funktion 03H Get LAT Status

Mit der Funktion 03H kann der Status der LAT session mit der Handle im Register DL in das Register AH gelesen werden. Die Registerpaare ES:BX und ES:DX sind reserviert.

AH Bit 7	Reserved
AH Bit 6	Reserved
AH Bit 5	Transmit buffer empty
AH Bit 4	Reserved
AH Bit 3	Session in start state
AH Bit 2	Session not active
AH Bit 1	Unable to queue transmit data
AH Bit 0	Receive data available

11.31.4 Funktion D0H Open/Close Session

Unterfunktion FFH Open Session

Mit dieser Funktion werden buffers geöffnet, eine Handle vergeben und eine LAT session gestartet. Die Funktion steht im Register AH und die Unterfunktion im Register AL. Das Register DH wurde mit FFH geladen. In ES:BX steht ein long pointer zu einem LAT session control block (SCB). In ES:DI steht ein long pointer zu dem password string. Im Register DL wird die Handle zurückgegeben, die für diese session zuständig ist. Im Register AH steht dann der Status:

AH Bit 7-5	Reserved virtual circuit
AH Bit 3	Data buffer specification error
AH Bit 2	No more sessions available
AH Bit 1	No more virtual circuit blocks
AH Bit 0	Service not in table or name error

Unterfunktion 00H Close Session

Mit dieser Funktion wird eine Session mit der Handle im Register DL geschlossen und die Buffer freigegeben selbst wenn noch nicht alle Zeichen abgearbeitet sind.

11.31.5 Funktion D1H Send Break Signal

Diese Funktion entspricht dem Break Signal eines Modems. Im Register DL muß die Handle stehen. Ist das Bit 7 im Register AH gesetzt, so war die Funktion nicht in der Lage das Signal zu der Handle zu senden.

11.31.6 Funktion D3H Reset LAT Counters

Mit dieser Funktion werden die LAT counters nach null gesetzt.

11.31.7 Funktion D4H Copy LAT Counters

Mit dieser Funktion werden die LAT counters in den angegeben buffer kopiert. Im Register CX steht die Größe des Buffers in Bytes. In den Registern ES:BX steht die Anfangsadresse. Ist danach im Register AX eine FFFFH, so war die Buffergröße nicht ausreichend.

11.31.8 Funktion D5H Get Next LAT Service Name

Mit dieser Funktion kann die LAT service table gelesen werden. Wenn im Register AX eine FFFFH erscheint, so ist das Ende der Tabelle erreicht. In den Registern ES:BX muß ein long pointer zu einem Buffer für den service name stehen mit mindestens 17 Bytes.

11.31.9 Funktion D6H Reset Service Table

Mit dieser Funktion wird die service table zurück gesetzt und die LAT counters mit null beschrieben. Im Register AX wird die Anzahl der services der table zurückgegeben. Steht im Register BX eine FFFFH war die Tabelle übergelaufen.

11.32 Interrupt Nr. 6BH Ungermann-Bass NETCI

11.33 Interrupt Nr. 6BH DECnet-DOS old port driver

11.34 Interrupt Nr. 6CH DOS 3.2 Echtzeituhr

11.35 Interrupt Nr. 6CH DECnet-DOS SCH

Der Interrupt 6CH wird von dem schedular benutzt.

11.36 Interrupt Nr. 6DH für VGA reserviert

11.37 Interrupt Nr. 6DH DECnet-DOS old DLL

Der Interrupt 6DH wird vom DLL benutzt.

11.38 Interrupt Nr. 6EH DECnet-DOS Network Process

Das Programm sendet Daten zum DNP über den Interrupt 6EH. Es werden Funktionsaufrufe zum öffnen und schliessen von Verbindungen und zum senden und empfangen von Daten benutzt. Vorher muß aber überprüft werden ob der DNP vorhanden ist. Dazu wird mit dem Interrupt 21H und der Funktion 35H der Vektor des Interrupts 6EH festgestellt. Die fünf Bytes vor dem Pointer in ES:BX (Vektor) enthalten den Namen und die Version, also in rückwärtiger Reihenfolge P, N, D, minor-, major-version. Für jeden Aufruf von Interrupt 6EH muß ein Network I/O Control Block (NIOCB) angelegt werden. Danach kann der Interrupt 6EH mit DEH im Register AH, 1 im Register AL und der NIOCB Adresse in den Registern DS:DX gerufen werden.

NIOCB Data Fields:

NIOCB Field	Size	Description
io_fcode	1	Function code field.
io_socket	2	Socket number field.
io_flags	2	Flag option field.
io_status	2	Command status field. 0 = call completed successfully -1 = error completing call -2 = call is pending
io_errno	2	Error code field.
io_psize	2	Parameter list size or buffer size.
io_resv	1	Reserved. Must be zero.
io_callback	4	Callback address.
attach_dn	12	Attach function data structure.
io_buffer	4	Data buffer address.
listen_dn	2	Listen data structure.
localinfo_dn	20	local node information data structure.
select_dn	16	Select data I/O descriptors.
shutdown_dn	2	Shut down logical link.
sioctl_dn	8	Socket definitions.
sockaddr_dn	26	Socket definitions.
sockopt_dn	12	Socket options.
buffer_dn	6	Data buffer descriptor.

11.38.1 Funktion DEH (DNP) Unterfunktion 1H (NIOCB)

DECnet-DOS Network Process Calls

Function Name	Function Code	Description
ABORT	10	Disconnect all logical links and detach the sockets that do not have the option SO_KEEPALIVE set.
ACCEPT	5	Accept an incoming connection request on a socket, and return a socket number.
ATTACH	0	Create a socket and attach a socket number.
BIND	2	Assign an object name or number to a socket.
CANCEL	20	Cancel previous function request.
CONNECT	4	Initiate a connection request on a socket.
DETACH	1	Disconnect all associated active logical links, and detach the specified socket and any associated sockets, only if the option SO_KEEPALIVE is not set.
DISCONNECT	6	Disconnect from the peer socket, and terminate the logical link connection.
GETSOCKOPT	26	Get the information associated with the socket option.
LISTEN	3	Listen for pending connections on a socket.
LOCALINFO	22	Retrieve network information for the local node.
PEERADDR	16	Retrieve information about your peer socket.
RCVD	8	Receive data on a specified socket.
RCVOOB	13	Receive out-of-band message on a specified socket.
SELECT	23	Check the I/O status of the network socket.
SEND	9	Send data on a specified socket.
SENDOOB	14	Send out-of-band messages on a specified socket.
SETSOCKOPT	25	Set options associated with sockets.
SHUTDOWN	7	Shut down part or all of a full duplex logical link connection.
SIOCTL	24	Control the operations of open sockets.
SOCKADDR	15	Retrieve information set by the BIND call for the specified socket.

11.39 Interrupt Nr. 6FH Novell PCOXAPI, 10-NET

11.40 Interrupt Nr. 70H Real Time Clock

Diese BIOS-Routine bedient die periodischen und Alarm Interrupts des CMOS-Timers. Er wird mit einer Frequenz von 1.024 kHz getaktet. Für die periodischen Interrupts ergibt sich daraus 1024 Interrupts pro Sekunde. Für die Alarm Funktion wird ein Interrupt nach Ablauf der Zeit erzeugt.
Der Interrupt ist erlaubt wenn die Alarm oder Event Funktion benutzt wurde. Bei der Event Funktion wird der wait counter herunter gezählt und bei Null die angegebene Memoryzelle mit 80H beschrieben. Für die Alarm Funktion muß vorher die korrekte Adresse für den Interrupt 4AH bestimmt werden. Danach kann erst der real time clock alarm mit Interrupt 1AH Funktion 06H gesetzt werden.
Zuerst wird das CMOS-Register C (flags) getestet. Wenn nicht die Bits 5 und 6 (60H) gesetzt sind ist es kein gültiger RTC-Interrupt. Ist Bit 6 gesetzt, so wird von dem Zähler RTC_LOW 976 (1/1024) abgezogen. Ist der Wert des 32 Bit zählers kleiner null geworden, wird in den Speicher auf den das Wort user_flag zeigt eine 80H geschrieben. Ist aber das Bit 5 gesetzt, so wird der Interrupt erlaubt und ein Interrupt 4AH erzeugt.

11.41 Interrupt Nr. 71H Redir to IRQ2

11.42 Interrupt Nr. 72H User

Der Interrupt Nr. 72H wird von der IRQ-Leitung 10 belegt.

11.43 Interrupt Nr. 73H User

Der Interrupt Nr. 73H wird von der IRQ-Leitung 11 belegt.

11.44 Interrupt Nr. 74H User

Der Interrupt Nr. 74H wird von der IRQ-Leitung 12 belegt.

IBM Mouse event.

11.45 Interrupt Nr. 75H Coprocessor

Der Interrupt 75H schreibt in die Adresse F0H eine Null und cleared damit den Coprozessor. Zu den beiden Interrupt-Kontrolern wird ein EOI Signal gesendet. Danach wird ein INT02 (NMI) ausgeführt. Wenn der nicht mit vernünftigen code überschrieben wurde, wird nur ein parity-check ausgeführt (IBM kompatibel). IRQ13-Leitung.

11.46 Interrupt Nr. 76H Disk Hardware Interrupt

Der Interrupt 76H setzt in HF_INT_FLAG die done mask FFH. Danach werden die Interrupt Kontroler zurückgesetzt und ein Interrupt 15H mit der Funktion 91H und der Unterfunktion 00H ausgeführt. IRQ14-Leitung.

11.47 Interrupt Nr. 77H User

Der Interrupt Nr. 77H wird von der IRQ-Leitung 15 belegt.

11.48 Interrupt Nr. 7AH Novell, AutoCAD, 3270

11.49 Interrupt Nr. 7BH Btrieve API

11.50 Interrupt Nr. 7CH IBM REXX88PC

11.51 Interrupt Nr. 7EH NET.EXE

Ist ein Interrupt für Päckchentreiber. Er wird z. B. von dem TSR-Programm 3C503.COM benutzt. Das ist Voraussetzung für PCS-Linda, ein System zur Parallelisierung von Programmen auf MS-DOS PC's mit Turbo-Pascal.

11.52 Interrupt Nr. 7FH 8514/A, Halo88, HLCAPI

11.53 Interrupt Nr. 80H — 85H BASIC

11.54 Interrupt Nr. 86H BASIC, NETBIOS

11.55 Interrupt Nr. 87H — A3 BASIC

11.56 Interrupt Nr. A4H BASIC, Right-Hand Man

11.57 Interrupt Nr. B3H BASIC, Zipkey

11.58 Interrupt Nr. E0H BASIC, CP/M-86

11.59 Interrupt Nr. E4H BASIC, Logitech Modula C

11.60 Interrupt Nr. EBH BASIC, EMS-Fenster

11.61 Interrupt Nr. ECH BASIC, DOSEDIT

Basic (welche Funktion ?) und DOSEDIT (welche Version ?) benutzen den gleichen Interrupt wie DLL von DECnet-DOS.

11.62 Interrupt Nr. ECH DECnet-DOS DLL

Dieser Interrupt wird vom DLL benutzt.

11.63 Interrupt Nr. EFH FASTOPEN

DECnet-DOS benutzt mit DLL auch den Interrupt EFH.

11.64 Interrupt Nr. EFH DECnet-DOS DLL

Dieser Interrupt wird vom DLL benutzt.

11.65 Interrupt Nr. F1H EMMXXXX0

11.66 Interrupt Nr. F3H EMMXXXX0

11.67 Interrupt Nr. F4H VGA Video-RAM

11.68 Interrupt Nr. F8H MS-DOS

11.69 Interrupt Nr. FAH MS-DOS

11.70 Interrupt Nr. FEH — FFH

Absturz nach Rückkehr aus dem Protected-Mode emmxxxx0. æ

Kapitel 12

Memory Management

Personal Computer, die MS-DOS kompatibel sind, können mit drei unterschiedlichen Arten von Random-Access Memory (RAM) bestückt sein:

- herkömmliches Memory (conventional)
- ausgedehntes Memory (expanded)
- verlängertes Memory (extended)

Alle MS-DOS Rechner müssen mindestens etwas conventional Memory besitzen. Die Anwesenheit von expanded oder extended Memory hängt von den installierten Hardware Options und dem vorhandenen Processor Typ ab. Jede Memory Art hat eigene Fähigkeiten, Merkmale und Grenzen. Jede hat auch eigene Management Techniken.

12.1 Conventional Memory

Conventional Memory ist der Begriff für das bis zu 1MB direkt adressierbare Memory der Prozessoren 8086/8088 oder der Prozessoren 80286/80386 im real mode (8086-emulation mode). Physikalische Adressen für das conventional Memory werden von einem 16-Bit Segment Register, das als Basis Register wirkt und eine Paragraph Adresse enthält, kombiniert mit einem 16-Bit Offset, der in einem Index Register steht oder von der ausführbaren Instruktion benutzt wird, erzeugt.

Auf IBM-PCs und kompatiblen werden von MS-DOS und den Programmen nur die unteren 640KB oder weniger benutzt. Der Bereich oberhalb von 640KB ist zwischen den ROM (read-only memory) chips, die primitive Device Handler und Test-Programme enthalten, und den RAM und ROM chips auf den zusätzlichen Boards, für Zwischenspeicherung oder device-abhängige Routinen, aufgeteilt.

Die unteren 640KB, die von MS-DOS verwaltet werden, können in drei Gruppen unterteilt werden:

- Die Interrupt Vector Table
- Die Operating System Area
- Die (überschreibbare) Programm Area

Die Interrupt Vector Table benutzt die untersten 1024 Bytes des Memory (00000-003FFH). Diese Adressen und die Länge sind in den Prozessor eingebaut und können nicht geändert werden. Jedes Doppelwort in der Tabelle wird Interrupt Vector genannt und beinhaltet das Segment und den Offset der Adresse einer Interrupt Handler Routine für eine dazugehörende Hardware oder Software Interrupt Nummer. Interrupt Handler sind normalerweise im Operating System eingebaut, können aber auch vom Benutzer Programm zur Verfügung gestellt werden. Interrupt Vectoren, die weder von einer Software noch einer Hardware benutzt werden, werden vom Operating System initialisiert, so daß sie zu einer einfachen Interrupt Return Instruction (IRET) zeigen oder es wird eine ERROR Message angezeigt.

Die Operating System Area beginnt unmittelbar oberhalb der Interrupt Vector Tabelle mit System-Tabellen und -Buffern, zusätzlichen installierbaren Device Drivern, die im CONFIG.SYS file angegeben wurden, und dem residenten Teil des COMMAND.COM Command Interpreter. Die Größe des vom Betriebssystem belegten Memory hängt von der Version von MS-DOS, der Anzahl der Disk Buffer, und der Zahl und Größe der installierten Driver ab.

Der Rest des Memory bis maximal 640KB wird Transient Program Area (TPA) genannt. Externe MS-DOS Befehle, z.B. CHKDSK, werden in die TPA geladen. Der ladbare Teil des Command Interpreters (COMMAND.COM) wird auch in die TPA geladen.

Die TPA wird unterteilt und dann Memory Arena genannt. Jedes Teil wird Arena Entry genannt. Jedes Entry besteht aus mindestens einem Paragraph (16-Bytes) oder mehreren bis hin zur ganzen TPA. Jedem Entry ist ein Arena Entry Header vorangestellt, in dem sich Informationen über die Größe und den Status des Entries befinden. Jedesmal wenn ein Programm beendet, ein Programm geladen und mit der EXEC-Funktion ausgeführt, eine memory allocation, modification oder release Funktion benutzt wird, untersucht MS-DOS die Arena Entry Headers. Ist einer dieser Entries beschädigt, so kehrt MS-DOS mit einem Error zu dem rufenden Programm zurück. Ist dieses Programm der COMMAND.COM, so wird die Message "Memory allocation error" angezeigt und das System angehalten.

12.1.1 MS-DOS Unterstützung

Der MS-DOS Kern stellt drei Funktionen mit dem Interrupt 21H für das Memory Management der TPA zu Verfügung:

- Funktion 48H (Allocate Memory Block)
- Funktion 49H (Freee Memory Block)
- Funktion 4AH (Resize Memory Block)

Diese drei Funktionen können von Programmen, dem Command Processor und MS-DOS benutzt werden, um die Arena Entries dynamisch zu verwalten. Wenn eine memory allocation Anforderung erzeugt wird, so wird die Kette der Arena Entries untersucht, um eine passende Lücke zu finden. Dafür stehen dem Memory Manager drei Strategien zur Verfügung:

- First fit – es wird der erste Entry an der niedrigsten Adresse genommen, der die Anforderung erfüllt.
- Best fit – es wird der Entry benutzt, der die Anforderung mit dem kleinsten Memory erfüllt.
- Last fit – es wird der Entry an der höchsten Adresse benutzt, der die Anforderung erfüllt.

Wenn der ausgewählte Entry größer ist als benötigt, so wird der Arena Entry geteilt und der Anforderung die genaue Größe zugeteilt. Der Rest erhält einen neuen Entry und wird mit "unowned" markiert, so daß er von einer anderen Anforderung benutzt werden kann.

Untersuchungen über Belegungsstrategien haben gezeigt, daß first-fit die effektivste Methode ist. Ab Version 3.0 von MS-DOS kann das Programm mit Funktion 58H des Interrupt 21H eine andere Strategie für den Memory Manager auswählen.

12.1.2 Benutzung der memory management Funktionen

Wenn ein Programm gestartet wird, so sind bereits zwei Arena Entries reserviert. In dem ersten findet man die ganze Umgebungsinformation, in dem zweiten PSP, Code, Data und Stack. Die Größe des zweiten Entries hängt von dem Typ des Programms (.COM oder .EXE) ab. Typischerweise erhält ein .COM Programm das ganze Memory des ersten Arena Entries, das den Programminhalt, 256 Byte für den PSP und mindestens 2 Bytes für den Stack aufnehmen kann. Da die TPA selten vor dem Laden eines Programms fragmentiert ist, so wird das .COM Programm das vorhandene Memory erhalten und es in einen relativ kleinen Teil für die Umgebung und einen sehr großen Teil für das Programm aufteilen.

Die Zuteilung von Memory wird bei einem .EXE Programm von den Feldern MINALLOC und MAXALLOC bestimmt. Mit MINALLOC wird dem MS-DOS Lader mitgeteilt, wieviel Memory-Paragraphen mindestens zugeteilt werden müssen, zusätzlich zu dem vorhanden Code und den Daten. In MAXALLOC steht die Zahl der Paragafen, die zugeteilt werden können, falls vorhanden. Wenn nicht angegeben, so setzt der Microsoft Linker MAXALLOC=FFFFH, was 1MB entspricht. Also teilt sich ein .EXE Programm genauso das ganze vorhandene Memory zu, wie ein .COM Programm. Deshalb sollte man mit /CPARMAXALLOC dem Linker einen kleineren Wert geben oder EXEMOD benutzen, das mit Microsofts Compilern geliefert wird.

12.2 Expanded Memory

Die Expanded Memory Spezifikation (EMS), Enhanced expanded Memory Spezifikation (EEMS), ist eine funktionale Definition eines bank-switched Memory Subsystems. Sie besteht aus benutzerinstallierbaren Boards und einem residenten Treiber, Expanded Memory Manager (EMM), der vom Hersteller mitgeliefert wird. Bis zu 8MB können in jedem PC eingesetzt werden. Das Expanded Memory kann von Programmen in Pages zu 16KB genutzt werden. Maximal 4 Pages werden in einen zusammenhängenden Memory-Bereich oberhalb von 640KB (page frame) gemapped. Die Adresse des Page Frame kann vom Benutzer festgelegt werden, um nicht mit anderen Optionen (ROM) zu Konflikten zu führen.

12.2.1 Expanded Memory Manager

Der EMM stellt ein hardware-unabhängiges Interface zwischen dem Programm und dem(n) expanded memory Board(s) zur Verfügung. Der EMM wird vom Hersteller als installierbarer Character-Device Driver geliefert und mit dem Befehl 'device=' im CONFIG.SYS in das lauffähige MS-DOS gelinkt.

Der EMM besteht aus den Teilen Driver und Manager. Der Driver verhält sich wie ein Driver und stellt auch Funktionen wie 'Initialisieren' und 'Output Status' zur Verfügung. Der Manager hat mehrere Klassen von Services:

- Status of the expanded memory subsystem
- Allocation of expanded memory pages
- Mapping of logical pages into physical memory
- Deallocation of expanded memory pages
- Support for multitasking operating systems
- Diagnostic routines

Programme können mit dem EMM direkt über den Interrupt 67H kommunizieren. Der MS-DOS Kern nimmt nicht Teil an der Verteilung des Memories und benutzt auch selbst nichts davon.

12.2.2 Anwesenheitsprüfung

Das Programm, das den EMM benutzen will, muß zuerst dessen Anwesenheit und Funktion prüfen. Danach muß mit dem Managerteil des EMM jedes Board geprüft werden. Es sind zwei Möglichkeiten für die Anwesenheitsprüfung vorhanden:

- Erstens — Mit dem Interrupt 21H Funktion 3DH wird ein Request zum vorbelegten Namen EMMXXXX0 gemacht. Wenn der Request erfolgreich ist, kann eine von zwei Voraussetzungen dafür vorhanden sein. Entweder ist der Driver vorhanden oder es ist ein File in dem augenblicklichen Verzeichnis vorhanden. Um festzustellen, ob die erhaltene Handle zu einem File oder einem Device gehöhrt, können die System-Calls IOCTL Get Device Information (Interrupt 21H, Funktion 44H, Subfunktion 00H) und Check Output Status (Interrupt 21H, Funktion 44H, Subfunktion 07H) benutzt werden.
- Zweitens — Der Vector des Interrupt 67H kann benutzt werden, um den Header des Device Drivers zu untersuchen. Dazu kann Interrupt 21H Funktion 35H (Get Interrupt Vector) benutzt werden. Ab dem Offset 0AH des Device Headers befindet sich der String EMMXXXX0. Diese Methode ist viel kürzer als die erste, benutzt aber Memory ausserhalb des Programmbereichs.

12.2.3 Benutzung von expanded memory

Nachdem die Benutzbarkeit des EMM festgestellt wurde, kann das Programm am MS-DOS vorbei über den Interrupt 67H mit dem EMM kommunizieren. Dafür stehen folgende Funktionen zur Verfügung:

- 40H — Get Manager Status
- 41H — Get Page Frame Segment
- 42H — Get Expanded Memory Pages
- 43H — Allocate Expanded Memory
- 44H — Map Memory
- 45H — Release Handle and Memory
- 46H — Get EMM Version
- 47H — Save Mapping Context
- 48H — Restore Mapping Context
- 4BH — Get Number of EMM Handles
- 4CH — Get Pages Owned by Handle
- 4DH — Get Pages for All Handles
- 4EH — Get/Set Page Map

Die Funktionen 49H und 4AH sind zwar in EMS Ver.3.0 definiert, später aber als reserviert ausgewiesen worden.

12.3 Extended Memory

Als extended Memory wird der Bereich oberhalb von 1MB bezeichnet. Er ist mit den Prozessoren 80286 und 80386 im protected mode linear adressierbar. Im Gegensatz zu dem expanded Memory wird kein Manager-Programm benötigt. Protected mode Operating Systeme wie XENIX oder OS/2 können diesen Bereich auch für die Ausführung von Programmen benutzen. Programme unter MS-DOS können diesen Bereich weder zur Ausführung noch direkt für Daten nutzen.

IBM PC/AT kompatible haben in ihrem ROM BIOS zwei Funktionen des Interrupt 15H für die Nutzung des extended Memory in real address mode:

- 87H — Move Extended Memory Block
- 88H — Obtain Size of Extended Memory

Damit kann eine schnelle Zwischenspeicherung von Informationen erreicht werden (RAMdisk). Da kein ERROR vorhanden ist, kann ein Programm auch den Bereich einer RAMdisk benutzen. Diese Funktionen sind auch nicht mit der compatibility Box von OS/2 kompatibel.
Im protected mode, der für diese Arbeiten erforderlich ist, ist der Interrupt abgeschaltet da die Vector Tabellen sonst anders interpretiert würden. Deshalb muß man sich über die Zeiten, die benötigt werden, informieren. Zum Beispiel braucht ein Zeichen von der Tastatur ca. 1ms bei 9600 Baud. Ein Block-move mit 512 Byte braucht auch ca. 1ms. Ein Block mit 64k Byte aber schon zwischen 37ms (beide even) und 72ms (beide odd). Dadurch könnte ein Buffer überlaufen ohne bemerkt zu werden.

æ

Kapitel 13

MS-DOS Device Drivers

Das BIOS des MS-DOS ist configurierbar, d.h. neue Driver können eingeflochten und vorhandene entlassen werden. Das bezieht sich auf sogenannte 'residente' Driver. Nicht residente Driver werden im CONFIG.SYS mit 'device=' zur Bootzeit dem System bekannt gemacht.

Zur Bootzeit müssen mindestens fünf residente Driver vorhanden sein. Sie befinden sich in einer 'linked list', d.h. im 'header' befindet sich ein DWORD Pointer zum nächsten. Der letzte Pointer ist -1,-1 (alle bits on).

Jeder Driver hat zwei 'entry points' in Vorbereitung auf multitasking:

- strategy entry point
- interrupt entry point

Zur Zeit ruft MS-DOS diese Entries direkt nacheinander auf.

In einer multitasking Umgebung soll die Strategieroutine den Request intern queuen und sofort zurückkehren. Die Interruptroutine erledigt dann den Rest. Ist sie fertig, so flagged sie sich als 'done'. Periodisch wird MS-DOS diese flags prüfen und dann den wartenden Process aufwecken.

Werden Anforderungen auf diese Art gequeued werden, ist es nicht mehr möglich, Informationen über Register auszutauschen. Dafür werden 'Request Pakete' benutzt. Sie sind von variabler Größe und Format und bestehen aus zwei Teilen:

- static request header section , für alle requests im gleichen Format.
- eine section mit Informationen, die für die Anforderung typisch ist.

Ein Driver wird mit einem Pointer zu einem Paket gerufen. In multitasking Versionen wird dieses Paket in einer globalen Liste ausstehender requests eingereiht werden.

Zur Zeit (MS-DOS 3.1) steht nur ein request an. Die strategie routine muß die Adresse des paketes an einer festen Adresse ablegen und sofort zurückkehren. Die Interruptroutine verarbeitet die Informationen in dem Paket und entlässt dann den Interrupt.

Ein Device Driver, der mit SYSINIT installiert werden soll, muß den Device Driver Header am Anfang des Files haben. Das Link-Feld muß mit -1 initialisiert werden, weil SYSINIT es mit den richtigen Informationen füllt. Driver im EXE-Format können nicht mit SYSINIT (config.sys) geladen werden, weil zu der Zeit der EXE-Loader nicht vorhanden ist (command.com). Der Driver muß mit EXE2BIN zu BIN oder SYS gewandelt werden.

13.1 Device Driver Format

Ein Device Driver ist ein ProgrammSegment, daß für die Kommunikation zwischen DOS und der Hardware verantwortlich ist. Am Anfang ist ein spezieller Kopf der es als Device Driver kenntlich macht, Entry Points definiert und Attribut Bits setzt. Es benutzt kein Program Segment Prefix (PSP), muß deshalb ein ORG 0 oder keinen ORG-Befehl haben.
Es werden zwei Arten von Device Drivern unterschieden:

- Character Device Drivers
- Block Device Drivers

Character Devices führen seriellen Zugriff auf das Device aus, z. B. Console, Communication Port oder Drucker. Sie haben einen Namen (CON) und Programme können Channels (handles oder FCB's) öffnen und benutzen.
Block Devices sind z. B. die Disk's in dem System. Sie können in einem strukturierten Stück, physikalischer Sector, zufällig zugreifen. Sie haben keinen Namen und können deshalb nicht direkt geöffnet werden. Daführ haben sie eine Unit Nummer und einen Drive Buchstaben.

13.2 Device Driver Besonderheiten

Am Anfang eines installierbaren Device Drivers muß ein Device Header vorhanden sein. Der Code darf nicht bei 100H, sondern muß bei 0 beginnen (ORG 0). Wenn kein weiterer Driver vorhanden ist, muß im Link-Feld eine -1 stehen. Bei einem Character Device muß der name 8 Buchstaben lang sein, nötigenfalls muß mit blank (20H) aufgefüllt werden. Installierbare Device Driver werden zuerst behandelt, d.h. vorhandene Driver können überschrieben werden.
Residente Disk Driver können nicht in dieser Weise überschrieben werden.
Installierbare Device Driver können überall im Memory zu liegen kommen. Deshalb soll man bei 'far memory references' besonders gut aufpassen.

13.2.1 Device Strategy Routine

Diese Routine wird bei jedem service request von MS-DOS gerufen. Da zur Zeit nur ein request anstehen darf, braucht sie nur den Pointer zum 'request packet' zur Weiterverarbeitung durch die Interrupt Routine zu retten.

13.2.2 Device Interrupt Routine

Diese Routine beinhaltet allen Code zur Abarbeitung des Service Requests. Hier kann die Hardware direkt angesprochen oder ROM BIOS Calls benutzt werden. Error handling und exit routinen sollten hier vorgesehen werden.

13.3 Installieren von Device Drivern

MS-DOS kann zur Boot Zeit neue Driver dynamisch installieren. Der initialization Code im io.sys benutzt dazu die Informationen im config.sys.
MS-DOS ruft die Driver in folgender Reihenfolge auf:

- MS-DOS macht einen far call zum strategy entry.
- MS-DOS übergibt der strategy routine driver Information in einem request header.
- MS-DOS macht einen far call zu der interrupt routine.

Diese Structur ist entworfen worden, um eine zukünftige multitasking Umgebung unterstützen zu können.

13.4 Device Headers

Ein device header wird am Anfang jedes device drivers benötigt. Ein header sieht wie folgt aus:

- DWORD Pointer zum nächsten device. Normalerweise -1.
- WORD Attributes (s. unten).
- WORD Pointer zum strategy entry point.
- WORD Pointer zum interrupt entry point.
- 8-BYTE Character device name field. Für block devices ist das erste Byte die Zahl der units.

Bedeutung der Bits im Attribut word.

- Bit 15 = 1 für character device; = 0 für block device.
- Bit 14 = 1 für IOCTL Unterstützung.
- Bit 13 = 1 für output til busy für character device; = 1 für NON FAT ID für block device.
- Bit 12 reserviert, muß 0 sein.
- Bit 11 = 1 wenn OPEN/CLOSE/RM unterstützt.
- Bit 10 bis 05 reserviert, muß 0 sein.
- Bit 03 =1 if intended current CLOCK.
- Bit 02 =1 if intended current NUL.
- Bit 01 =1 if intended current STO.
- Bit 00 =1 if intended current STI.

Die device entry points sind Worte, sie müssen also Offsets zum Segment sein, daß auf die Tabelle weist.

13.4.1 Next Device Field

Der Pointer zum 'next device header field' ist ein double word (Offset gefolgt von Segment). Wenn MS-DOS den driver läd setzt es den pointer zu dem nächsten driver in der system liste. Wenn mehrere driver in dem gleichen file vorhanden sind, so muß das erste word der Offset zum nächsten header sein. Der letzte muß mit -1 abgeschlossen werden.

13.4.2 Attribute Field

Das Attribut Field wird benutzt, um den Typ des Device festzustellen, für den dieser Driver zuständig ist. Zusätzlich zur Unterscheidung zwischen block und character device werden diese Bits bei bestimmten character devices für spezielle Behandlung der devices benutzt. Ist der andere Typ bei der Erleuterung der Bedeutung der Bit's nicht angegeben, so muß das Bit 0 sein.
Das NON FAT ID Bit für block device wirkt auf die Bedienung des BUILD BPB (BIOS Parameter Block) device call. Dieses Bit hat eine andere Bedeutung für character device. Es wird dann vorausgesetzt, daß OUTPUT UNTIL BUSY benutzt werden kann.
Das IOCTL Bit hat für beide Typen von device Bedeutung. Es können dann Controll character des device behandelt werden mit INT 21 Function 44 code xx. Nur device, die controll strings verarbeiten können, sollten in dem driver dieses Bit gesetzt bekommen.
Das OPEN/CLOSE/RM Bit sagt dem MS-DOS 3.x, daß dieser driver zusätzliche MS-DOS Funktionen unterstützt. In früheren Versionen ist es reserviert und auf null gesetzt. Alle neueren driver sollten die calls OPEN, CLOSE und REMOVABLE MEDIA unterstützen und deshalb dieses Bit gesetzt haben.

13.4.3 Strategy und Interrupt Routinen

Diese beiden Worte beinhaltn die pointer zu den entry points der strategy und interrupt routine. Sie sind ein Wort lang und müssen deshalb in dem selben Segment sein, in dem der device header ist.

13.4.4 Name Field

In diesem 8-Bit Feld steht der Nahme des character device, aufgefüllt mit blanc, oder die Zahl der units eines block device.

13.5 Request Header

Wenn MS-DOS einen device driver ruft um eine Funktion auszuführen, wird ein request header in ES:BX übergeben. Es ist ein Festlängen header gefolgt von Daten, die zu der auszuführenden Funktion gehöhren. Der device driver muß den Maschinen Status festhalten, d. h. alle Register und Flags müssen am Anfang gerettet und am Ende restauriert werden. Auf dem Stack stehen ca. 20 Plätze zur Verfügung. Wird mehr gebraucht, so muß der driver einen eigenen Stack anlegen.

Request Header ⇒

- BYTE Length of record.
- BYTE Unit code ; subunit für block device ; keine Bedeutung für character device.
- BYTE Command code.
- WORD Status.
- 8 BYTEs Reserved.

13.5.1 Length of Record

Dieses Feld ist mit der Länge, in Bytes, des request headers belegt.

13.5.2 Unit Code Field

Ein block device driver kann mehrere units bedienen. Zur Unterscheidung, welche unit gerade bedient werden soll, wird hier die Nummer eingetragen.

13.5.3 Command Code Field

Im command code field des request headers können folgende Werte stehen:

Command Code 0 INIT

Command Code 1 MEDIA CHECK (block devices only)

Command Code 2 BUILD BPB (block devices only)

Command Code 3 IOCTL Input (nur bei IOCTL unterstützenden device)

Command Code 4 INPUT (read)

Command Code 5 NON-DESTRUCTIVE INPUT NO WAIT (Char devs only)

Command Code 6 INPUT STATUS (Char devs only)

Command Code 7 INPUT FLUSH (Char devs only)

Command Code 8 OUTPUT (write)

Command Code 9 OUTPUT (write) with verify

Command Code 10 OUTPUT STATUS (Char devs only)

Command Code 11 OUTPUT FLUSH (Char devs only)

Command Code 12 IOCTL OUTPUT (nur bei IOCTL unterstützenden device)

Command Code 13 DEVICE OPEN (nur wenn OPEN/CLOSE/RM Bit gesetzt ist)

Command Code 14 DEVICE CLOSE (nur wenn OPEN/CLOSE/RM Bit gesetzt ist)

Command Code 15 REMOVABLE MEDIA (nur wenn OPEN/CLOSE/RM Bit gesetzt ist)

Command Code 16 OUTPUT UNTIL BUSY (nur wenn NON FAT ID Bit gesetzt ist und character device driver)

Command Code 19 Generic IOCTL Request

Command Code 23 Get Logical Device (block device only)

Command Code 24 Set Logical Device (block device only)

13.5.4 Status Field

Am Anfang ist der Inhalt des Status Wortes gleich null. Es wird beim zurückkehren gesetzt. Bit 9 ist das Busy-Bit, das nur bei status oder removable media calls gesetzt wird. Bit 8 ist das done-bit, das nur nach erfolgreichem Ende der Aufgabe gesetzt wird. Im anderen Fall wird das Error-Bit (Bit 15) gesetzt. Die Bits 0 bis 7 halten dann den Error-Code:

Error code 0 Write protect violation

Error code 1 Unknown unit

Error code 2 Drive not ready

Error code 3 Unknown command

Error code 4 CRC error

Error code 5 Bad drive request structure length

Error code 6 Seek error

Error code 7 Unknown media

Error code 8 Sector not found

Error code 9 Printer out of paper

Error code A Write fault

Error code B Read fault

Error code C General failure

Error code D Reserved

Error code E Reserved

Error code F Invalid disk change

13.6 Device Driver Functions

Device driver können eine oder mehrere der folgenden Funktionen ausführen. Sie können auch mit command codes in bestimmten Fällen unterteilt sein.

- INIT
- MEDIA CHECK
- BUILD BPB
- READ OR WRITE
- NON DESTRUCTIVE READ NO WAIT
- OPEN OR CLOSE
- REMOVABLE MEDIA
- STATUS
- FLUSH

Wenn die strategy routine gerufen wird, so steht in ES:BX der pointer zum request header. Die strategy routine speichert den pointer (in der queue) zwischen. Die interrupt routine benutzt den pointer. Der command code in dem request header zeigt die Funktion, die ausgeführt werden soll, an. Alle DWORD pointer werden in der Reihenfolge Offset vor Segment gespeichert.

13.6.1 INIT

Command code=0

INIT – ES:BX ⇒

- 13-BYTE Request header
- BYTE Number of units
- DWORD End Address
- DWORD Pointer to BPB array (not set by character device)
- BYTE Block device number

Eine der Funktionen, die für einen device driver definiert sind, ist INIT. Diese routine wird beim Installieren ausgeführt. Die INIT routine muß als DWORD pointer die END ADDRESS des residenten Teils des drivers zurückgeben. Damit kann der Rest des drivers gelöscht werden um Platz zu sparen.

Die Anzahl der units, die end address und der BPB pointer müssen vom driver gesetzt werden. Jedoch wird der BPB pointer beim installieren dazu benutzt, um auf das Zeichen nach dem "=" in der Zeile im config.sys zu zeigen. Damit kann man dem driver parameter übergeben.

Für block device schreibt MS-DOS in das "block device number Byte" die Zahl, die für den driver zuständig ist (A=0, B=1).

Auch installierbare character devices müssen den end address parameter zurück geben. Er wird als pointer zum ersten benutzbaren Byte nach dem driver verstanden. Vorhandener initialisierungs code wird überschrieben.

Mit dem INIT-Code kann auch der Vektor des Device in der Interrupt-Vektor-Tabelle gesetzt werden.

13.6.2 MEDIA CHECK

Command code = 1

MEDIA CHECK – ES:BX ⇒

- 13-BYTE Request header
- BYTE Media descriptor from BPB
- BYTE Returned
- Returned DWORD pointer to previous Volume ID if bit 11 set and Disk changed is returned.

13.6.3 BUILD BPB (BIOS Parameter Block)

Command code = 2

BUILD BPB – ES:BX ⇒

- 13-BYTE Request header
- BYTE Media descriptor from BPB
- DWORD Transfer address (points to one sector worth of scratch space or first sector of FAT depending on the value of Bit 13 in the device attribute word).
- DWORD Pointer to BPB

Diese Funktion wird nur von block devices benutzt.

13.6.4 READ or WRITE

Command code = 3,4,8,9,12 und 16

READ or WRITE ES:BX ⇒

- 13-BYTE Request header
- BYTE Media descriptor from BPB
- DWORD Tranfers address
- WORD Byte/sector count
- WORD Starting sector number (Ignored on character device)
- Returned DWORD pointer to requested Volume ID if error 0FH

Command code 3 IOCTL READ

Command code 4 READ (block oder character)

Command code 8 WRITE (block oder character)

Command code 9 WRITE WITH VERIFY

Command code 12 IOCTL WRITE

Command code 16 OUTPUT TIL BUSY (character devices only)

Der driver muß in Abhängigkeit von dem command code read oder write ausführen. Block device lesen oder schreiben sectoren; character device lesen oder schreiben bytes. Er muß auch das Status word und die Zahl der erfolgreichen byte oder sector transfers setzen. Error-Bit und -code alleine genügt nicht.

13.6.5 NON DESTRUCTIVE READ NO WAIT

Command code = 5

ES:BX ⇒

- 13-BYTE Request header
- BYTE read from device

Dieser call erlaubt MS-DOS einen input character voraus zu sehen. Das device setzt das done bit im status wort.

Wenn das character device das busy bit als 0 zurück gibt, dann wird das nächste byte gelesen. Dieses Byte wird nicht aus dem buffer (non destructive) entfernt. Wenn das busy bit gleich 1 ist, sind keine character im buffer.

13.6.6 OPEN or CLOSE

Command code = 13 und 14

OPEN oder CLOSE – ES:BX ⇒

- 13-BYTE Static request header

Diese Funktion wird von MS-DOS nur dann ausgeführt, wenn im device driver header das attribute bit OPEN/CLOSE/RM gesetzt wurde.

13.6.7 REMOVABLE MEDIA

Command code = 15

REMOVABLE MEDIA – ES:BX ⇒

- 13-BYTE Static request header

Diese Funktion wird von MS-DOS nur dann ausgeführt, wenn im device driver header das attribute bit OPEN/CLOSE/RM gesetzt wurde.

13.6.8 STATUS

Command code = 6 UND 10

STATUS – ES:BX ⇒

- 13-BYTE request header

Status zeigt an, ob data für input oder output warten. Der driver muß das status wort und das busy bit wie folgt setzen:

- Für output auf character device: Wenn der driver bit 9 zu 1 setzt, weiss MS-DOS, das ein write request (falls gemacht) auf Abarbeitung wartet. Wenn es 0 ist, ist kein request anhängig und ein neuer würde sofort ausgeführt.
- Für input von character device mit buffer: Eine 1 zeigt an, daß kein character im buffer vorhanden ist und ein request würde zum device gehen. Eine 0 zeigt an, daß im buffer characters vorhanden sind. MS-DOS geht davon aus, daß alle character device einen input type-ahead buffer haben. Device, die keinen buffer haben, sollen busy = 0 setzen, damit DOS nicht auf etwas wartet, was nicht sein kann.

13.6.9 FLUSH

Command code = 7 UND 11

FLUSH – ES:BX ⇒

- 13-BYTE request header

Der flush call sagt dem driver, daß er alle anstehenden requests löschen soll.

13.7 MEDIA DESCRIPTOR BYTE

Inhalte der media descritor byte und FAT ID byte haben für MS-DOS keine Bedeutung. Sie werden an den driver übergeben, der sie auswertet.

13.8 MEDIA DESCRIPTOR TABLE

Das MS-DOS file system benutzt eine linked list von pointers (einen für jeden cluster), die file allocation table (FAT) genannt wird. Unbenutzte cluster werden mit 0 markiert und end of file mit FFF oder FFFF bei Einheiten mit 16-Bit FAT Einträgen.

Um mehr Flexibilität in der Unterstützung unterschiedlicher Plattenformate zu geben, wird der BPB ein Teil des boot sectors.

- 3 BYTE near jump to boot code
- 8 BYTES OEM name and version
- WORD bytes per sector- - - - - - - - - - - - - - -
- BYTE sectors per allocation unit
- WORD reserved sectors
- BYTE number of FAT's
- WORD number of root dir entries
- WORD number of sectors in logical image
- BYTE media descriptor
- WORD number of FAT sectors- - - - - - - - - - - -
- WORD sectors per track
- WORD number of heads
- WORD number of hidden sectors

Die letzten drei Worte werden nur vom driver benutzt, um das Medium zu verstehen (40 track double sided - 80 track single sided).

13.9 THE CLOCK DEVICE

MS-DOS nimmt an, daß eine clock im system vorhanden ist. Es kann eine CMOS real-time clock oder ein interval timer sein, der zur boot Zeit von dem Benutzer gestartet wird. Das clock device ist ein normales character device, nur das es mit einem bit im attribut word des drivers identifiziert wird. Das clock device benutzt 6 Byte.

- low byte days since 1-1-80
- hi byte days since 1-1-80
- minutes
- hours
- sec/100
- seconds

13.10 Zusammenfassung

Die folgenden Punkte sollen verdeutlichen, was bei einem write request auf ein block device alles getan wird:

- MS-DOS schreibt ein request Paket in einen reservierten Bereich des Memory.
- MS-DOS ruft den driver mit dem strategy entry point.
- Der driver rettet ES und BX (pointer to request packet) und macht einen FAR return.
- MS-DOS ruft den driver mit dem interrupt entry point.
- Der driver liest aus dem Paket den command code (Offset 2) und stellt einen write request fest. Der command code wird zu einem Index in eine command tabelle gewandelt und die zuständige routine ausgeführt.
- Der driver liest die unit nummer (Offset 1).
- Da es ein write command ist, muß der driver auch die transfer adresse (Offset 14), den sector count (Offset 18) und den start sector (Offset 20) in dem request paket benutzen.
- Der driver übersetzt die first logical sector number in track, head und sector nummer.
- Der driver schreibt die angegebene Anzahl sectoren von der transfer adresse zur Platte. Es können mehrere operationen erfolgen.
- Wenn alles abgearbeitet ist, muß der driver das status word schreiben. Nicht nur das done bit sondern auch den sector count im request packet.
- Falls ein Fehler aufgetreten ist, so muß nicht nur das error bit und der error-code gesetzt werden, sondern auch der sector count.
- Der driver macht dann ein FAR return zu MS-DOS.

Der driver muß alle Register und Flags retten. Die direction flag und das interrupt enable bit sind sehr kritisch. Wenn der interrupt entry point des drivers gerufen wird, hat MS-DOS ca. 40 Byte auf dem Stack frei. Der driver muß einen eigenen Stack anlegen, wenn er viel mit dem Stack arbeiten will.

13.11 Skelett eines device drivers

Im Folgenden wird ein Gerüst eines character device drivers dargestellt. Nähere Erläuterungen sind im vorhergehenden chapter zu sehen.

- Header
  1. WORD address of next driver, initial -1
  2. WORD attribut bit's
  3. WORD strategy routine entry
  4. WORD service routine entry
  5. BYTE's 8-byte name field
- command tables
  1. WORD address of read routine
  2. WORD address od write routine
  3. WORD address of init routine
  4. WORD address of exit routine
- DWORD SAVE ;zwei worte zum retten des pointers des request pakets.
- STRATEGY z. B.
  MOV WORD PTR CS:[SAVE],BX
  MOV WORD PTR CS:[SAVE+2],ES
  RET
- SERVER-ENTRY ;zuerst alle Register auf den Stack retten. Die angeforderte routine selectieren und anspringen, gegebenenfalls die error routine.
- ERROR-entries ;hier werden die error-bits gesetzt.
- EXIT-ENTRIES ;hier wird der interrupt verlassen.
- es folgen die command routinen.

æ

Kapitel 14

Multitasking

Das BIOS des IBM-PC-AT besitzt eine Möglichkeit zur Unterstützung von multitasking Anwendungen. Haken und Ösen ("Hooks") werden für einen Vielfachaufgabenverteiler ("multitasking dispatcher") bereitgestellt. Immer wenn im BIOS eine busy oder wait loop ausgeführt wird, wird dem Programm ein "Haken" zur Verfügung gestellt mit dem es aus der loop ausbrechen kann. Auch wenn das BIOS einen Interrupt bedient wird eine entsprechende wait loop ausgeführt und ein neuer "Haken" zur Verfügung gestellt. Somit kann ein Programm geschrieben werden, daß einen "device driver" beinhaltet. Das folgende ist nur im "real address mode" gültig und muß Aufgabe des Programms sein um diese Unterstützung zu erlauben.

- Das Programm ist verantwortlich für die serielle Bearbeitung der Aufgaben des "device drivers". Das BIOS ist nicht "reentrant'.

- Das Programm ist verantwortlich für die Anpassung übereinstimmender "wait" und "post" calls.

14.1 Interfaces

Dem multitasking dispatcher werden vier Schnittstellen zur Verfügung gestellt.

14.1.1 Startup

Zuerst muß sich das Programm in den Interrupt 15 einhaken. Der dispatcher ist verantwortlich dafür nur die Funtionen 90 und 91 zu suchen. Alle anderen muß er an den vorhergehenden Benutzer des Interrupt 15 weitergeben. Dies kann mit einem JMP oder CALL geschehen. Ist der Funktioncode 90 oder 91 wird der dispatcher die entsprechende Arbeit veranlassen und mit IRET beenden.

14.1.2 Serialization

Das multitasking System muß ein serielles abarbeiten der Aufgaben für den device driver sicherstellen. Mehrfaches gleichzeitiges Benutzen des gleichen drivers durch unterschiedliche Aufgaben werden zu schwerwiegenden Fehlern führen.

14.1.3 Wait (Busy)

Immer wenn das BIOS eine busy loop beginnen will, wird ein Interrupt 15 mit der Funktion 90 in Register AH gemacht. Dies gibt eine "wait" Bedingung bekannt. Zu dieser Zeit kann der Dispatcher den Zustand der task retten und eine andere starten. Dieses erlaubt überlappende Ausführung von tasks wenn die Hardware beschäftigt ist. Mit folgender Ergänzung zum BIOS wurde dies ermöglicht:

```
MOV AX,90XXH ;wait code in AH and type code in AL

INT 15H ;issue call

JC TIMEOUT  ;optional: for timeout or if carry is set, time-out occured

NORMAL TIMEOUT LOGIC ;normal time-out
```

14.1.4 Post (Interrupt)

Immer wenn das BIOS eine interrupt flag für eine busy loop gesetzt hat, wird ein Interrupt 15 mit einem Funktioscode 91 im Register AH ausgelöst. Dies gibt eine "post" Bedingung bekannt. An diesem Punkt wird der Dispatcher den Status der Task auf "ready to run" setzen und zu der interrupt routine zurück kehren. Mit folgender Ergänzung zum BIOS wurde dies ermöglicht:

```
MOV AX,91XXH ;post code AH and type code AL

INT 15H ;issue call
```

14.2 Classes

Die folgenden Arten von wait loops werden unterstützt:

- Die Funktionen 00H bis 7FH sind hintereinander wiederverwendbar. Der Zugriff auf das BIOS muß für diese Funktionen auf eine Task zu jeder Zeit begrenzt werden.
- Die Funktionen 80H bis BFH sind reentrant. Es ist keine Begrenzung der Zahl der Tasks für einen Zugriff auf das device vorhanden.
- Die Funktionen C0H bis FFH sind non-interrupt. Es ist kein entsprechender Interrupt für die wait loop vorhanden. Deshalb muß der Dispatcher festlegen was diese Bedingung erfüllt, um die loop zu verlassen.

14.2.1 Function code classes

Type code(AL) 00H–7FH hintereinander wiederbenutzbare device. Der serielle Zugriff muß sichergestellt sein.

Type code(AL) 80H–BFH reentrant devices. Mit ES:BX werden unterschiedliche calls festgestellt. Mehrfache calls sind gleichzeitig erlaubt.

Type code(AL) C0H–FFH wait only calls. Nur time-out calls. Die zeit wird mit der Funktionsnummer in AL festgelegt.

14.2.2 Function code assignments

Das Folgende gilt nur für IBM-AT BIOS. Die Zeiten sind nur angenähert.

Type code (AL) 00H 6 sekunden fixed disk

Type code (AL) 01H 2 sekunden diskette

Type code (AL) 02H keine Zeit für keyboard

Type code (AL) FDH 1 sekunde für motor start bei write

Type code (AL) FDH 625 millisekunden für motor start bei read

Type code (AL) FEH 18 sekunden für printer

Die asynchrone Unterstützung wurde nicht übernommen. Der seriell/parallel Adapter erzeugt zwar Interrupts, doch das BIOS unterstützt sie nicht. Deshalb muß diese Unterstützung in dem multitasking Betriebssystem vorgesehen werden.

14.3 Time-Outs

Um time-outs geeignet unterstützen zu können, muß der Dispatcher sich über die Zeit bewußt sein. Wenn ein device in eine busy loop eintritt wird es dort eine bestimmte Zeit verbringen, bevor ein error erzeugt wird. Der Dispatcher kehrt zu der BIOS wait loop zurück mit gesetztem carry bit wenn ein time-out erzeugt wurde.

14.4 Machine-sensitive code

Wenn Programme besondere Vorteile bestimmter Maschinen benutzen wollen, müssen sie den augenblicklichen Maschinentype feststellen. Er wird mit Interrupt 15 Funktion AH erhalten. Das erste Byte zeigt den Type und das zweite die Serie.

- FF 00 IBM Personal Computer
- FE 00 IBM Personal Computer XT
- FE 00 IBM Portable Personal Computer
- FD 00 IBM PCjr
- FC 00 IBM Personal Computer AT
- FB 00 IBM Personal Computer XT mit 256/640 system platine

IBM wird für neue device den Maschinentyp festlegen. æ

Kapitel 15

DECnet-DOS und PCSA

Die Version 3.0 solte nur in Verbindung mit VMS Version 5.1 und höher, also z. Z. nur auf MSFD1F, benutzt werden.

Die Unterschiede zwischen DEPCA und NI5010 von Interlan sind unbedingt zu beachten! Die DEPCA-Mouse kann eine serielle Schnittstelle für andere Aufgaben frei machen. Sie benutzt den IRQ 2 und muß mit dem DEC-mouse-driver betrieben werden! Die DEPCA-Turbo Netzwerkkarte hat keinen Mouse-Achluß mehr, dafür können aber die oberen (9 – 15) Interruptleitungen selektiert werden, da es eine 16-Bit Karte ist.

15.1 Netzwerke

Als unterste Ebene wird der Scheduler (SCHPC) benutzt. Er benutzt die Interrupts 1AH (Alarm), 69H (Cterm), 6CH (SCH) und 70H (RTC). Wird der 69H an nachfolgende abgegeben, so wird er durch 0BH ersetzt.

Darüber befindet sich der data link layer (DLL) der für die unterschiedlichsten Hardware-Interface vorhanden ist. Er benutzt die Interrupts 69H (Cterm), 6DH (DLL), ECH (DLL) und EFH (DLL). Der Interrupt 69H wird an die folgenden Aufgaben, z.B. CTERM oder LAST, abgegeben.

Der DLL wird gleichberechtigt von LAT mit Interrupt 6AH oder DNP mit Interrupt 6EH benutzt.

SETHOST kann beide, LAT und DNP, benutzen.

CTERM, NFT, NDU und LAST benutzen nur DNP.

LAD, mit Interrupt 13H (disk services), benutzt nur LAST und damit DNP.

Session benutzt die Interrupts 2A (Netbios) und 5C (Netbios).

Redir braucht Session und benutzt die Interrupts 05H (Print Screen), 09H (Keyboard), 17H (Printer), 21H (System), 28H (Idle) und 2F (Multiplex).

Nicht alle Teile werden gleichzeitig benötigt. Will man nur Terminal im LAN sein, so reichen rund 65 000 Byte, SCHPC (3104) und DLL (50480) und LAT (11840). Terminal im WAN kosten rund 142 000 Byte, SCHPC und DLL und DNPETHPC (51 424) und CTERM (37 152). CTERM kann mit /r (remove) aus dem Memory entfernt werden und sollte deshalb immer als letztes aufgerufen werden, um Löcher zu vermeiden. Für rund 166 000 Byte, SCHPC und DLL und DNPETHPC und LAST (8528) und LAD (5216) und SESSION (23 488) und REDIR (24352), kann man Disks, Files und Queus auf den dafür eingerichteten Hosts benutzen.

DLL ist hardware-abhängig. Es muß der für die Karte zuständige DLL, z.B. DLLDEPCA oder DLLMICOM oder DLL3COM, benutzt werden. Dabei muß man auf die Einstellungen in MSNET.INI, z. B. IRQ=3, achten. Sonst müssen die Angaben beim Aufruf gemacht werden.

Der Schalter /R: muß für MICOM auf 1 gesetzt sein und "executor receive pipe quota" sollte 1 sein, weil "nack quota" null ist. Bei circuit sollte "multicast" und "service" disabled sein. Der Schalter /M: sollte auf D gesetzt sein, um multicast zu verbieten.

15.2 DECnet-DOS

DECnet-DOS besteht aus den Driverprogrammen NDDRV.SYS (network disk) und NPDRV.SYS (network printer), die im CONFIG.SYS in das System eingebunden werden, den Transportprogrammen z. B. NFT, den Dienstprogrammen z. B. NDU und Hilfsprogrammen z. B. DTS.

Zusätzlich zu den Driver-Programmen müssen die Programme für die unterschiedlichen Level des Netzwerkes gestartet worden sein. In MSNET.INI sind einige Startproceduren vorhanden, die um andere erweiter werden können, oder es werden Batchfiles angelegt, die den Rechner für die beabsichtigte Arbeit vorbereiten.

Das Program DECNET ist etwas für Oberflächenfreaks!

15.3 PCSA

Für PCSA muß der LADDRV (local area disk driver) in CONFIG.SYS, gegebenenfalls mit dem Schalter /d=N, wobei N die Zahl der Laufwerke ist, angegeben sein. Dann werden N Laufwerke vorbereitet die mit NET USE bzw. USE an Services der VAX angebunden werden können. Es wird zwischen öffentlichen und privaten, Disk und File Services unterschieden. Öffentliche Services, z. B. PCTEX, werden vom Systemmanager eingerichtet und verwaltet. Private Services kann sich jeder im Rahmen seiner Plattenkapazität, nur der benötigte Platz wird angerechnet, selber einrichten. Der File STARTNET.BAT sollte den eigenen Erfordernissen angepasst werden. Es ist zwischen verbrauchtem Plattenplatz und Zeit für das Laden über das Netz zu wählen. Auch der File AUTOUSER.BAT in der eigenen Dirctory auf der VAX muß überarbeitet werden. Vorallem wenn im PC mehrere Laufwerksbezeichnungen, große Platten mit mehreren Partitionen, vergeben wurden, können Probleme mit der starren Aufteilung der physikalischen und logischen Laufwerke und deren Zuordnung zu Disk- und File-Services bzw. zu private und public Services auftreten. Es ist nicht im Sinne von der Firma DEC die PC's mit Plattenkapazität auszustatten.

15.4 Überblick

Das Bild 15.1 auf der Seite 459 zeigt die Zusammenhänge der Komponenten der PCSA auf.

Zusätzlich zu der fertigen Software kann man sich mit Hilfe der Komponenten auch eigene Übertragungsmöglichkeiten schaffen. Dem Benutzer stehen dafür die Interrupts 2AH und 5CH zur Verfügung.

15.5 Terminal

Der PC kann mit den entsprechenden Programmen Terminal an den unterschiedlichsten Rechnern sein. Die Tastaturbelegung und die Darstellung auf dem Bildschirm hängt von den verwendeten Protokollen ab. Nicht alles ist möglich und nicht alles mögliche ist realisiert!

DEC liefert SETHOST für die Protokolle CTERM und LAT mit der Netzwerksoftware mit.

15.5.1 CTERM

Das Command Terminal Protokol (CTERM) ist ein Software-Modul, das für Terminalemulatoren und andere Programme zur Kommunikation mit Host-Systemen in einem DECnet^{TM}-Netzwerk vorgesehen ist. Das Host-System muß ein DECnet Phase IV Knoten sein und das CTERM DECnet-Objekt muß instaliert sein. Dann stellt es WAN-Services (wide area network) dem Benutzer zur Verfügung. Benutzt werden sie über den Interrupt INT69H. CTERM hat dann Zugriff auf das Netz über DNP und DLL. Das Benutzerprogramm kann dann

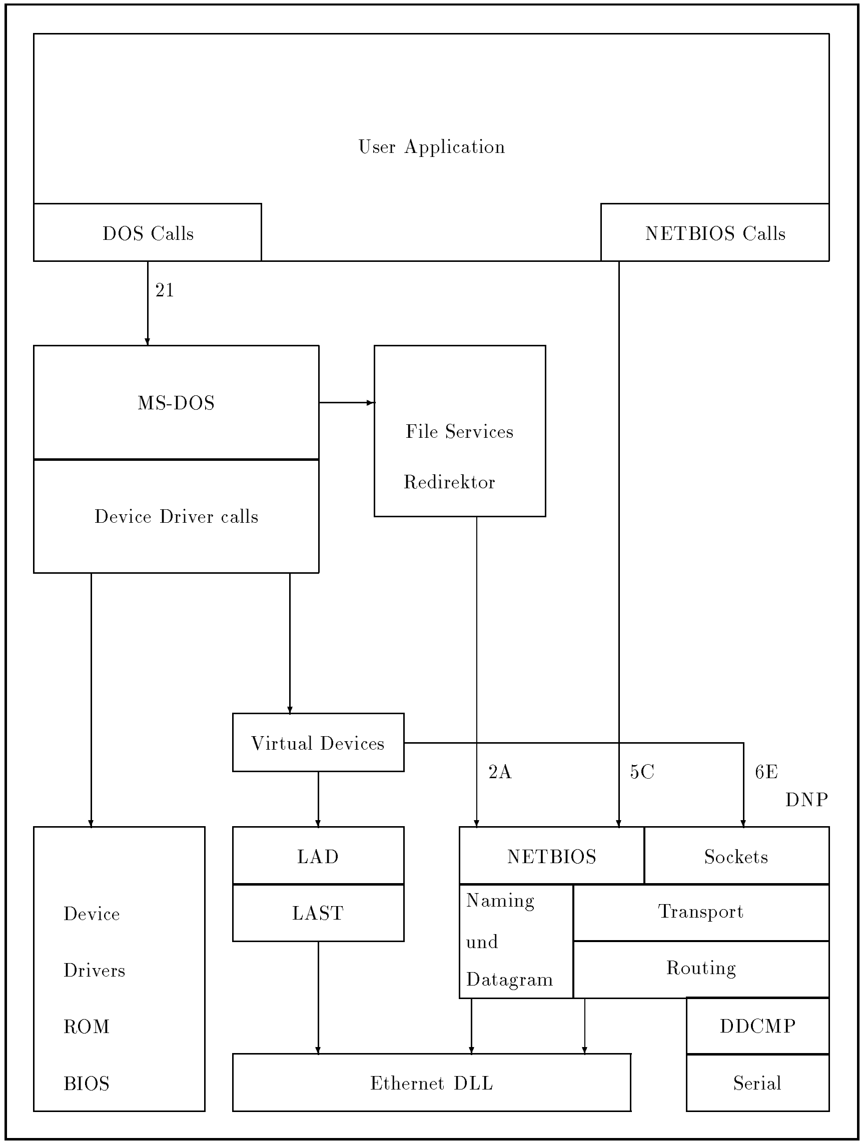


Abbildung 15.1: Zasammenhänge der Netzwerk-Komponenten

- eine Sitzung eröffnen,
- Zeichen senden,
- Zeichen empfangen,
- eine Sitzung beenden.

CTERM ist ein TSR-Programm (terminate and stay resident) für MS-DOS. Der Terminal-Emulator SETHOST lädt es in das Memory und entläßt es auch wieder. Man kann das auch von den DOS-Kommandozeile aus tun. Dafür hat CTERM einige Schalter (/U unload, /D disable, /R remove, /T trace), die in DEBUG.DOC des DECnetPCSA Client for DOS kit beschrieben sind [1].

CTERM benutzt zum Informationsaustausch zwischen Benutzerprogramm und Host-System eine Datenstruktur, den CTERM Session Control Block (SCB). Zuvor muß das Programm aber die Größe des SCB's feststellen um genügend Platz im Memory zur Verfügung stellen zu können. Dazu wird die Funktion 0AH (GetSCBSize) des Interrupts 67H benutzt.

Danach muß eine Session mit der Funktion 05H des Interrupt 69H eröffnet werden. Dieser Funktion muß ein Pointer zu dem SCB-Buffer und ein Pointer zu einem ASCII String, der die Node-Information enthällt, übergeben werden. Zurück erhällt man eine (16-bit integer) "handle", die für die weiteren calls benötigt wird.

Installation Check

Die Funktion 00H des Interrupt 69H stellt fest, ob der CTERM-Driver installiert ist. Der Real-Time-Scheduler (SCH) muß vorher installiert sein.

On Entry:

AH = 01, modul identification
AL = 00, function code

On Return:

succeeded:
ES:BX = far address of module entry point
failed:
AL = 00, CTERM not installed
AL = FFH, CTERM already installed

Send Char

Mit der Funktion 01H des Interrupt 69H werden Daten zum CTERM-Driver geschickt.

On Entry:

AH = 01, modul identification
AL = 01, function code
BL = character to be sent
DX = session handle returned on the OPEN SESSION call

On Return:

succeeded:
AH = 0, character sent
failed:
AH Bit 7= 1, unable to send character

[1] Warum denn auch da, wo die CTERM Command Line beschrieben wird?

Read Char

Mit der Funktion 02H des Interrupt 69H werden Zeichen vom CTERM-Driver empfangen.

On Entry:

AH = 01, modul identification
AL = 02, function code
DX = session handle returned on the OPEN SESSION call

On Return:

succeeded:
AL = character received
failed:
AH Bit 7= 1, no character read

Status

Mit der Funktion 03H des Interrupt 69H wird der Status des CTERM-Drivers erhalten.

On Entry:

AH = 01, modul identification
AL = 03, function code
DX = session handle returned on the OPEN SESSION call

On Return:

succeeded:
AX = status word
failed:
AX = status word

- Bit 7 Session has been aborted
- Bit 6 DECnet error (use Funktion 04H)
- Bit 5 reserved
- Bit 4 reserved
- Bit 3 reserved
- Bit 2 reserved
- Bit 1 reserved
- Bit 0 Receive data available

DECnet Status

Mit der Funktion 04H des Interrupt 69H wird der DECnet Status erhalten. Der 2 Byte Error-Code wird im Anhang G des Programmers Reference Manual erklärt.

On Entry:

AH = 01, modul identification

AL = 04, function code

DX = session handle returned on the OPEN SESSION call

On Return:

AX = DECnet error codes

Open Session

Mit der Funktion 05H des Interrupt 69H wird eine Sitzung eröffnet. Der Zeiger auf eine mit null abgeschlossene Zeichenkette wird in den Registern DS:BX und der Zeiger zu dem Session Control Block (SCB) in den Registern ES:DX übergeben. In dem Register AX steht dann die handle, die dann für diese Sitzung gilt.

On Entry:

AH = 01, modul identification

AL = 05, function code

DS:BX = long pointer to ASCIIZ string nodespec

ES:DX = long pointer to the CTERM SCB

On Return:

succeeded:

AX = an integer greater than 0; handle for subsequent calls to use this session

failed:

AX = an integer less than or equal to zero; CTERM driver reason code (refer to function 03H: Status for a list of reason codes)

Close Session

Mit der Funktion 06H des Interrupt 69H wird die mit der handle in dem Register DL bezeichnete Sitzung abgeschlossen.

On Entry:

AH = 01, modul identification

AL = 06, function code

DX = session handle returned on the OPEN SESSION call

On Return:

succeeded:

AH = 0

failed:

AH = 0CH, session does not exist

Get SCB Size

Mit der Funktion 0AH des Interrupt 69H wird die Größe des SCB in dem Register AX erhalten.

On Entry:

AH = 01, modul identification

AL = 0A, function code

On Return:

succeeded:

AX = size of SCB to allocate

Get DECnet Socket

Mit der Funktion 0BH des Interrupt 69H wird die Socket-Nummer, die mit dieser Sitzung verbunden ist, in dem Register DX erhalten. Sie wird benötigt, wenn man die DECnet-Library benutzen will.

On Entry:

AH = 01, modul identification

AL = 0B, function code

On Return:

succeeded:

AX = an integer greater than 0; DECnet socket number for this session

failed:

AX = 00, no match

Unload

Mit der Funktion 0FH des Interrupt 69H wird der CTERM-Driver aus dem Memory entfernt. Der Driver kann nur entfernt werden, wenn er der letzte in der Reihe der TSR-Programme ist [2].

On Entry:

AH = 01, modul identification

AL = 0F, function code

On Return:

succeeded:

AH = 0

failed:

AH = FFH, could not deinstall CTERM

[2] Sonst entstehen Löcher.

15.5.2 LAT

Das Local Area Transport (LAT) Modul kann von Terminal-Emulatoren und anderen Programmen über den Interrupt 6AH benutzt werden. LAT setzt einen geladenen DLL voraus.

LAT kann von der DOS-Ebene aus mit folgenden Schaltern gestartet werden:

- /D:nn

  Mit diesem Schalter kann die LAT Service Direktory um nn Einträge vergrössert werden. Vorgegeben sind 10 Einträge zu jeweils 47 Byte.

- /G: 1,2,3,32...

  Mit diesem Schalter können die LAT-Gruppen erlaubt werden, von denen multicast erwünscht ist. Ohne diesen Schalter sind alle erlaubt [3]. Die GWDG ist 1, die Strömungsforschung ist 3.

- /N

  Dieser Schalter erlaubt LAT multicast abzuschalten. Es wird durch den NCP Befehl DISABLE MULTICAST nicht berührt.

- /R:n

  Der Schalter setzt den Retransmitfaktor auf n. Erlaubt sind 4 bis 255.

- /U

  Mit diesem Schalter wird LAT aus dem Memory entfernt. Anstehende Sitzungen werden unterbrochen. Befindet sich LAT im Expanded Memory (EMS), kann es mit diesem Schalter nicht entfernt werden.

Send Char

Mit der Funktion 01H des Interrupt 6AH

On Entry:

AL = character to be send

AH = 01

DH = FFH

DL = session handle returned from the Open Session Call

On Return:

succeeded:

AH = 0

failed:

AH Bit 7 = 1, unable to queue character

[3] und sorgen damit für overhead.

Read Char

Mit der Funktion 02H des Interrupt 6AH

On Entry:

AH = 02

DH = FFH

DL = session handle returned from the Open Session Call

On Return:

succeeded:

AL = received character

failed:

AH Bit 7 = 1, no character read

Get Status

Mit der Funktion 03H des Interrupt 6AH

On Entry:

AH = 03

DH = FFH

DL = session handle returned from the Open Session Call

On Return:

AH = status byte

- Bit 7 reserved
- Bit 6 reserved
- Bit 5 Transmit buffer empty
- Bit 4 reserved
- Bit 3 Session in start state
- Bit 2 Session not active
- Bit 1 Unable to queue transmit data
- Bit 0 Receive data available

ES:BX = reserved

ES:DX = reserved

Open Session

Mit der Funktion D0H des Interrupt 6AH

On Entry:

AH = D0H

AL = FFH

ES:BX = long pointer to LAT Session Control Block (SCB)

DH = FFH

On Return:

succeeded:

AH = 0

failed:

AH =

- Bit 7 - 5 = Reseved virtual circuit
- Bit 3 = Data buffer specification error
- Bit 2 = No more sessions available
- Bit 1 = No more virtual circuit blocks
- Bit 0 = Service not in table or name error

DL = session handle for subsecuent service requests over this session connection

Close Session

Mit der Funktion D0H des Interrupt 6AH

On Entry:

AH = D0H

AL = 00H

DH = FFH

DL = session handle returned from the Open Session Call

On Return:

AX = 00H, no error

AX = 01H, no such active session

AX = 02H, session not in running state; try again after a short delay

Send Break Signal

Mit der Funktion D1H des Interrupt 6AH

On Entry:

AH = D1H

DH = FFH

DL = session handle returned from the Open Session Call

On Return:

succeeded:

AH = 0

failed:

AH Bit 7 = 1, unable to send break signal

Reset LAT Counters

Mit der Funktion D3H des Interrupt 6AH

On Entry:

AH = D3H

DH = FFH

On Return:

LAT counters set to zero

Copy LAT Counters

Mit der Funktion D4H des Interrupt 6AH

On Entry:

AX = D4H

CX = maximum size of buffer, in bytes

DH = FFH

ES:BX = location of buffer to contain counters

On Return:

succeeded:

AX = 0, counters copied into buffer

failed:

AX = FFFFH, buffer size not sufficient

Get next LAT Service Name

Mit der Funktion D5H des Interrupt 6AH

On Entry:

AH = D5H

DH = FFH

ES:BX = long pointer to the buffer for the returned service name; the buffer must be al least 17 bytes long.

On Return:

succeeded:

AX = 0

failed:

AX = FFFFH, end of table - no service name available

ES:BX = long pointer to the service name terminated by a zero byte.

LAT Service Table Reset

Mit der Funktion D6H des Interrupt 6AH

On Entry:

AH = D6H

DH = FFH

On Return:

AX = number of services entered into the service table; this number varies depending on whether or not the service table has been filled.

BX = 0, service table not overflowed

BX = FFFFH, service table has overflowed

15.5.3 SETHOST

15.6 Filetransfer

Bei Filetransfer zwischen unterschiedlichen Rechnern und Betriebssystemen sind einige grundsätzliche Überlegungen zu beachten. Sinnvoll können auf der anderen Seite eigentlich nur Source-Codes sein; binäre Files nur dann, wenn sie mit einem Programm formatiert verarbeitet werden sollen. Auch dabei ist gegebenenfalls auf den Unterschied zwischen den beiden Filesystemen Rücksicht zu nehmen. Ein Programm mit z. B. 512 Byte Verarbeitungsbreite wird bei einem Angebot von 128 Byte pro Block gegebenenfalls streiken und sicherlich nur unsinnige Fehlermeldungen produzieren!
Bei dem transparenten Filetransfer der PCSA muß man ganz besonders auf den Inhalt der Files achten. Ein .COM-File kann ein ausführbares File eines PC sein oder eine Batch-Procedure für die VAX.

15.6.1 NFT

Das Netzwerk File Transfer Programm erlaubt es, Files zwischen dem eigenen Rechner und anderen Rechnern auszutauschen. Es können auch noch andere, auf files bezogene Arbeiten mit NFT erledigt werden. Es ist auch ein funktionsgleiches Programm für MS-Windows vorhanden.

APPEND

Der Inhalt eines oder mehrerer input-files wird an das Ende eines bestehenden output-files angehängt.
APPEND /switch input-files output-file

/switch
- /ASCII
- /DELETE
- /IMAGE
- /NOLOG
- /PRINT
- /SUBMIT

input-files Inputfile-Namen müssen durch ein Komma getrennt werden. Sie dürfen nur acht Zeichen lang sein. Der Filetype (extension) darf nur drei Zeichen lang sein.

output-file An diesen File wird/werden der/die File(s) angehängt.

Input und output müssen auf unterschiedlichen Rechnern sein. Zu den Filenamen gehört gegebenenfalls der komplette Weg (path, Plattenlaufwerk, Unterverzeichiss) und/oder die Login-Informationen auf dem Knoten:
NFT> APPEND daten1,daten2 MSFD1A"username password"::[.messdat]messung12.dat;3

COPY

Mit dem NFT-Befehl *COPY* können files von dem einen zu dem anderen Rechner übertragen werden. Weiterverarbeitende Software kann einen besonderen switch erfordern, da die Filesysteme unterschiedlich sein können und die verwendeten Protokolle nicht alle Möglichkeiten erfassen.
COPY /switch input-files output-file

/switch
- to remote node
  - /ALLOCATION=nnn
  - /ASCII
  - /BLOCK sollte bei DVI-Files in Verbindung mit PS-Drivern immer verwendet werden!
  - /CC=xxx
  - /DELETE
  - /FIXED=nnn
  - /IMAGE
  - /LSA
  - /MACY11
  - /MRS=nnn
  - /NOLOG
  - /NOSPAN
  - /PRINT
  - /SUBMIT
  - /VARIABLE
  - /VFC=nnn
- from remote node
  - /ASCII
  - /BLOCK
  - /IMAGE
  - /NOCONVERT
  - /NOLOG

input-files Sollen mehrere Files kopiert werden, sind die Namen mit Komma zu trennen.

output-file Gibt die Output-Files an.

Input und output müssen auf unterschiedlichen Rechnern sein. Zu den Filenamen gehört gegebenenfalls der komplette Weg (path, Plattenlaufwerk, Unterverzeichiss) und/oder die Login-Informationen auf dem Knoten:
NFT> COPY daten1*,daten2* MSFD1A"username password"::[.messdat]*.dat

DELETE

Mit DELETE wird ein File oder eine Gruppe von Files bei Benutzung von "Wildcards (*)" aus dem Verzeichniss gelöscht.
DELETE /switch file-spec

/switch /NOLOG

file-spec muß ein erlaubter File-Name sein.

Zu den Filenamen gehört gegebenenfalls der komplette Weg (path, Plattenlaufwerk, Unterverzeichiss) und/oder die Login-Informationen auf dem Knoten:
NFT> DELETE MSFD1A"username password"::[.messdat]*.dat;*

DIRECTORY

Mit diesem Befehl kann man sich ein Listing des Verzeichnisseintrags geben lassen.
DIRECTORY /switch file-spec

/switch /BRIEF oder /FULL

file-spec muß ein erlaubter File-Name sein.

Zu den Filenamen gehört gegebenenfalls der komplette Weg (path, Plattenlaufwerk, Unterverzeichiss) und/oder die Login-Informationen auf dem Knoten:

NFT> DIRECTORY MSFD1A"username password"::[.messdat.h2o]messung12.dat;3

Wildcards (*.*) werden verwendet, wenn nichts anderes angegeben wurde. Von ULTRIX werden dann nur Files gelistet, die einen Punkt in dem Namen haben. Will man bei ULTRIX alle gelistet haben, muß man einen einfachen Stern (*) angeben.

EXIT

Dieser Befehl beendet den NFT. Es kann auch CTRL Z gefolgt von RETURN benutzt werden.

HELP

Mit diesem Befehl erhält man eine Liste der Befehle und Schalter, für die weitere Hilfe zur Verfügung steht. Diese weitergehende Information erhält man, wenn man HELP gefolgt von dem gewünschten WORT eingibt.
NFT> HELP /switch — command-verb

/switch jeder, in der mit HELP erhaltenen Liste, angegebene Schalter.

command-verb jedes zugelassene, in der mit HELP erhaltenen Liste, NFT Kommando.

PRINT

Dieser Befehl reiht einen File des Remote-node in die Druckerwarteschlange des Remote-node ein.
NFT> PRINT /switch file-spec

/switch /DELETE oder /NOLOG

file-spec muß ein auf dem Remote-node vorhandener File sein.

NFT> PRINT MSFD1A"username password"::[.messdat.h2o]messung12.dat;2

SET

Mit dem Befehl SET kann man für diese Sitzung dem NFT die Node-Information abkürzen.
NFT> SET node-spec:: file-spec

node-spec ist die Knoteninformartion inklusive Zugriffsberechtigung.

file-spec ist gegebenenfalls die Platte, Direktory, Subdirectory ...

NFT> SET MSFD1A"username password"::[.messdat.h2o]

Dann wird mit "copy *.* MSFD1A::" alles nach subdir h2o gebracht.

NFT> PRINT MSFD1A"username password"::

Dann wird mit "copy *.* MSFD1A::" alles in die Hauptdir gebracht.

SHOW

Mit diesem Befehl kann man sich die mit dem Befehl SET eingetragenen Informationen ansehen.
NFT> SHOW node-name::

node-name:: muß ein erlaubter Knotenname sein.

node-name:: wenn nicht angegeben, werden alle Einträge gelistet.

SUBMIT

Mit diesem Befehl kann auf einem dafür geeigneten Knoten eine dort vorhandene Batchprozedure zur Abarbeitung in die Warteschlange eingetragen werden.
NFT> SUBMIT /switch remote-filespec

/switch /NOLOG

remote-filespec es kann auch die mit SET abgekürzte Form benutzt werden.

TYPE

Es wird der angegebene File auf dem Bildschirm angezeigt. Es sollten nur ASCII-Files genommen werden. Nichtdruckbare Zeichen werden den dazwischenliegenden Protokollen und Tabellen entsprechend dargestellt.
NFT> TYPE file-spec

file-spec es kann auch die mit SET abgekürzte Form benutzt werden.

15.7 Dienstprogramme

15.7.1 NDU

Mit der Network Device Utility können virtuelle Platten und Printer auf dem anderen Rechner erzeugt und benutzt werden.

CLOSE

Der NDU-Befehl *close* beendet die Verbindung zwischen dem angegebenen local drive oder printer (NPRN) und dem virtuellen disk file auf dem anderen Rechner.

NDU> CLOSE DRIVE drive

NDU> CLOSE NODE node-name NDISK file-name

NDU> CLOSE PRINTER

drive ist das alphabetische Zeichen des Drive, mit dem gearbeitet werden soll.

node-name kann bis zu 6 alphanumerische Zeichen, davon mindestens ein alphabetisches, haben. Es kann auch die node-address in der Form area.node angegeben werden, dabei kann die area von 1 bis 63 und die node von 1 bis 1023 gehen.

file-name kann bis zu 8 alphanumerischen Zeichen sein. Device- und Direktory-Informationen können bis zu einer Gesamtlänge von 127 Zeichen enthalten sein.

CREATE

Der NDU-Befehl *create* erzeugt und formattiert auf dem anderen Rechner eine Platte in der angegebenen Größe als File mit dem angegebenen Namen und stellt eine Verbindung her zu dem angegebenen drive.

NDU> CREATE PRINTER NODE node-name	NPRINT file-name	USER access-info
NDU> CREATE NODE node-name	NDISK file-name	DRIVE drive
		USER access-info
		ACCESS RW/RO
		ALLOCATION n
		MAX n

node-name ist der Name des Knotens auf dem das virtuelle Laufwerk oder Drucker eingerichtet werden soll.

file-name ist der Name des Files, in dem die Disk abgebildet werden soll. Als Filetype (extension) sollte DSK benutzt werden um Verwechselungen vorzubeugen. Wird für einen Drucker kein Filename angegeben, so wird einer erzeugt.

drive ist der Buchstabe eines möglichen Laufwerks. NDU SHOW listet die möglichen. Wird nichts angegeben, wird der nächst mögliche genommen.

access-info ist die Information, die zum Einloggen in den Knoten erforderlich ist. Wird nichts angegeben, wird die Information in der permanenten Datenbasis (NCP DEFINE ... ACCESS) genommen.

RW/RO read/write kann nur von einem benutzt werden. read/only kann von mehr als einem gleichzeitig benutzt werden (bezieht sich nur auf die Disk!).

ALLOCATION n bestimmt die anfängliche Größe des Files in Blocks.

MAX n bestimmt die maximale Größe, auf die das File ausgeweitet werden kann.

DELETE

Der NDU-Befehl *delete* löscht den Eintrag des angegebenen Files in dem Verzeichniss des anderen Rechners und gibt den drive zur weiteren Benutzung frei.

NDU> DELETE	NODE node-name	NDISK file-name	USER access-info
	DRIVE drive		

node-name ist der Name des Knotens auf dem das virtuelle Laufwerk sich befindet.

file-name ist der Name des Files der gelöscht werden soll.

access-info ist die Information, die zum Einloggen in den Knoten erforderlich ist. Wird nichts angegeben, wird die Information in der permanenten Datenbasis (NCP DEFINE ... ACCESS) genommen.

drive ist das alphabetische Zeichen des Drive, mit dem gearbeitet werden soll.

HELP

NDU> HELP	command-verb
	entity

command-verb ist der NDU-Befehl, für den Hilfe erwartet wird.

entity ist der Schalter, für den mehr Auskunft erhofft wird.

OPEN

NDU> OPEN PRINTER NODE node-name	NPRINT file-name	USER access-info
NDU> OPEN NODE node-name	NDISK file-name	DRIVE drive
		USER access-info
		ACCESS RW/RO

node-name ist der Name des Knotens auf dem das virtuelle Laufwerk

file-name ist der Name des Files, in dem die Disk abgebildet wurde. Wurde für einen Drucker kein Name angegeben, wird einer vergeben.

drive ist der Buchstabe eines möglichen Laufwerks. NDU SHOW listet die möglichen. Wird nichts angegeben, wird der nächst mögliche genommen.

access-info ist die Information, die zum Einloggen in den Knoten erforderlich ist. Wird nichts angegeben, wird die Information in der permanenten Datenbasis (NCP DEFINE ... ACCESS) genommen.

RW/RO read/write kann nur von einem benutzt werden. read/only kann von mehr als einem gleichzeitig benutzt werden (bezieht sich nur auf die Disk!).

SHOW

Der NDU-Befehl *SHOW* zeigt den Status aller virtuellen disks und printer.

15.7.2 FAL

Wenn der File Access Listener auf dem eigenen Rechner läuft, kann man diesen von anderen Rechnern aus erreichen.

15.7.3 SPAWNER

Wenn der Spawner gestartet ist, wartet er im Vordergrund auf Aufgaben, die ihm über das Netz zugetragen werden. Die Aufgaben, die er erledigen kann, müssen vorher mit NCP DEFINE OBJECT in den File DECOBJ.DAT eingetragen worden sein. DEC stellt die Objects FAL und DTR zur Verfügung.

15.7.4 TFA Transparent File Access

15.7.5 TTT Transparent Task to Task

15.7.6 C-Library

15.7.7 Assembler Library

15.7.8 NETBIOS

15.8 Hilfsprogramme

15.8.1 Memory Management

Zur Unterstützung der Verwaltung des Memories ist das MEMMAN Programm vorhanden. Damit lassen sich auch die Netzwerkomponenten aus dem Memory entfernen. Alle Aufrufe können mit /B für *brief* oder /F für it full benutzt werden.

MEMMAN /E Expanded Memory

Dieser Befehl zeigt eine Tabelle mit den folgenden Teilen:

Driver version Ist die Versionsnummer des EMM-Drivers.

Page frame address Ein *page frame* ist ein Fenster, durch das auf das expanded Memory zugegriffen werden kann. Die gezeigte Adresse ist der Anfang des Fensters.

Expanded memory size Ist die Größe des expanded Memory in *pages* und in Byte. Eine *page* ist 16KBytes.

Expanded memory available Ist die Größe des vefügbaren Memories.

Handle Ist eine Identifizierung die von einem Programm benutzt wird, um auf einen Block des zugewiesenen expanded memories zuzugreifen.

Page Ist die Größe des für diese handle zugewiesenen expanded memory in pages. Eine page ist 16KBytes.

Size Ist die Größe des zugewiesenen expanded Memory in Bytes für diese handle.

Attributes Sind die Charakteristiken des expanded memory für diese handle (volatile (flüchtig) und non-volatile).

Handle name Ist der Name, den ein Programm der handle gegeben hat. Nichtdruckbare Zeichen werden hexadezimal dargestellt.

MEMMAN /H Help

Die Schalter /H und /? sind gleichwertig. Es wird eine Liste der möglichen Schalter angezeigt.

MEMMAN /M Memory Map

Es wird eine Liste der Memorybenutzung gezeigt. Mit dem Schalter /F wird die Liste ausführlicher.

PSP Ist die Prozess Adresse oder ID.

Bytes Ist die Größe des Blocks.

Owner Ist der Prozess.

Command line Diese Spalte kann folgende Bedeutung haben:

- Die in der Kommandozeile dem Prozessnamen folgenden Schalter ganz oder teilweise.
- Wenn dort <ENVIRONMENT> steht, ist das Memory Teil eines anderen Prozesses.
- Wenn dort <DATA> steht, sind dort Daten eines anderen Prozesses.

Hooked interrupts Ist eine Lister der von diesem Prozess verwendeten Interrupts.

MEMMAN /S Summary

Zeigt eine Liste des ganzen Memories. Mit dem Schalter /F wird sie ausführlicher. Es werden dann nicht nur conventional, expanded und extended Memory gezeigt, sonden auch RAM und ROM in Adaptern zwischen den Adressen A000 und F000.

MEMMAN /U Unload

Wenn z. B. mit STARTNET.BAT der PCSA Mark gesetzt wurde, kann mit /U das Memory bis dorthin befreit werden und dann von anderen Programmen genutzt werden.

MEMMAN /X Extended Memory

Der Schalter /X zeigt Informationen über das extended Memory. Der XMS-Driver, z. B. HIMEM.SYS, muß in CONFIG.SYS angegeben sein.

15.8.2 MAIL

EXIT

HELP

MAIL

SEND

SHOW DEFAULTS

15.8.3 NCP

Das Netzwerk Control Programm erlaubt das Netzwerkmanagement und einige Tests der Hard- und Software.

NCP> CLEAR

Der NCP-Befehl *clear* entfernt die entsprechenden Angaben aus der volatile database, also nur für diese Sitzung. Mit *purge* können sie aus der permanenten Datenbasis entfernt werden.

Local oder *remote adapter name* ist die Bezeichnung eines Benutzerprogramms, das NETBIOS benutzt wie z. B. DNP (DECnet Network Process).

CLEAR	EXECUTOR	RECEIVE PASSWORD
		TRANSMIT PASSWORD
	LOCAL-ADAPTER-NAME	name BYTE 16 nn
	KNOWN LOCAL-ADAPTER-NAMES	
	REMOTE-ADAPTER-NAME	name
	KNOWN REMOTE-ADAPTER-NAMES	

RECEIVE PASSWORD removes the password the executor must receive from the adjacent node in order to exchange messages with the adjacent node. The password is a character string consisting eight alphanumeric characters.

TRANSMIT PASSWORD removes the password the executor must transmit to the adjacent node in order to exchange messages with the adjacent node. The password is a character string consisting eight alphanumeric characters.

name is a 1 to 16 character ASCII printable string that specifies the name of the local or remote application you want to remove from the volatile database. The characters are case sensitive with use of quotation marks.

nn specifies a value for Byte 16. This value can be any decimal value from 0 to 255.

NCP> COPY

COPY	NODE	node-id	FROM	remote-node-id
		area.⋆	FROM	remote-node-id
	KNOWN	NODES	FROM	remote-node-id
			TO	file-id

node-id Name oder Adresse, z. B. **MSFD5** oder **3.36**.

remote-node-id Name oder Adresse, z. B. **MSFD1F** oder **3.91**.

area.⋆ Area ist der Bereich des Netzes, über das Auskunft erhalten werden soll. Der Stern ist eine "wildcard" die nur funktioniert in Netzen, in denen das Network-Management das zugelassen hat.

file-id Ist der Name des files, in den die Information koiert werden soll.

NCP> DEFINE

Mit DEFINE wird die permanente Datenbasis geändert. Diese Änderungen werden erst beim nächsten booten übernommen. Mit SET werden die jetzt gültigen, nur für *diese* Sitzung in der volatile Datenbasis, Parameter geändert.

DEFINE ACCESS USER user-id PASSWORD password access-type

- *username* ist ein bis zu 39 Zeichen langer string.
- *password* ist ein bis zu 39 Zeichen langer string.
- *access-type* beschreibt die Art des Zugriffs auf die Informationen auf diesem Knoten.
  - NONE kein Zugriff ist default.
  - READ nur lesen.
  - WRITE nur schreiben.
  - ALL lesen und schreiben.

DEFINE CIRCUIT
- HELLO TIMER *seconds* gibt die Frequenz der routing messages zu dem adjacent node an (normaly 30).
- MULTICAST-LISTENER DISABLED/ENABLED gibt an, ob der circuit auf multicast Anforderungen antworten soll.
- OWNER DECNET/DLX/MOP für DECNET und PCSA muß hier DECNET eingetragen sein.
- SERVICE DISABLED/ENABLED erlaubt die Benutzung für z. B. Loopback-Tests vom Netz her.
- STATE ON/OFF off = nichts geht mehr!

DEFINE EXECUTOR Wer mit diesem Befehl spielt, muß damit rechnen, daß die anderen Netzwerkbenutzer ihm den Kopf waschen! Die Änderungen werden erst mit dem nächsten boot gültig. Sonst muß man SET benutzen!

- ADDRESS *node-address* gibt die eigene Adresse an. Wird vom Netzwerkverwalter vergeben und besteht aus der Areanummer und der Rechnernummer.
- CONFIDENCE TIMER *number* Maximal 32K, default ist 15 [4]. Eine zusätzliche Zeit, die ablaufen muß, bevor ein link als nicht erreichbar erklärt wird.

[4] Ist ja eine tolle Erklärung!

- DELAY FACTOR *number* Nach Ablauf dieser Zeit wird die Message erneut an den Knoten gesendet. Die Zeit wird in 1/16 Sekunden gemessen. Number ist eine dezimale Zahl zwischen 16 und 4095. Die default number ist 32, also 32/16=2 Sekunden [5].
- DELAY WEIGHT *number* is the weight to a current round trip to a remote node when updating the estimated round trip delay to a node. The number is decimal in the range 1 to 255. The default is 1.
- IDENTIFICATION *id-string* specifies a text string for the executor node. The string can consist of 1 to 32 alphanumeric characters. You must use a set of double quotation marks to delimet any string containing blanks or tabs. You cannot use a single set of quotation marks within the identification string.

  You can use the IDENTIFICATION parameter to identify the executor node to other nodes in the network.
- INACTIVITY TIMER *seconds* is the maximum duration of inactivity (no data in either direction) on a logical link before the node checks to see if the logical link still works. If no activity occurs within the minimum number of seconds, the node generates traffic to test the link. The value range is 1 to 32K. The default is 30 [6].
- INCOMING TIMER *seconds* is the maximum duration between the time a connection request is received at the executor node and the time that it takes to accept or reject the request. If the request is not accepted or rejected by the user program within the specified number of seconds, the connection request is automatically rejected. If no value is set, there is no timer. The value range is 1 to 32K. The default is 45.
- MAXIMUM BUFFERS *number* specifies the maximum number of large network communication buffers available for use by DECnet-DOS. The valid range is 1 to 255. The default is six times the number of links. The actual number of buffers is retricted by available memory. This affects the network response time and memory usage. The default is 16.
- MAXIMUM LINKS *number* specifies the number of active logical links for the executor node. The valid range is 1 to 32. The default is 4.
- NAK QUOTA *number* This is the number of out-of-order segments to receive before sending a negative acknowledge (NAK) out on the logical link. This number must be less than or equal to the RECEIVE PIPE QUOTA. The value range for this quota is 0 to 4095. The default is 0 (or 6 for slower devices and processors) [7].
- NAME *node-name* Ist der Name des Knotens. Wird vom System-Manager vergeben und muß 1 - 6 Zeichen betragen mit mindestens einem alphabetischen.
- OUTGOING TIMER *seconds* Ist die Zeit zwischen der Anforderung und dem Zustandekommen einer Verbindung. Der Bereich ist von 1 bis 65535. Voreingestellt ist 60. [8]
- RECEIVE PASSWORD *password* Acht Zeichen, die der Anforderer senden muß, um überhaupt eine Antwort zu bekommen. Wie wird den das auf der VAX gesetzt?
- RECEIVE PIPE QUOTA *number* is the number of segments the network will receive for a link by flow control. The valid range is 1 to 65535. The default is 6.
- RETRANSMIT FACTOR *number* is the maximum number of times that the executor node will restart the retransmission timer when it expires. If the number is exceeded, the logical link is disconnected. The range is 1 to 65535. The default is 12.
- BUFFER SIZE *number* allows you to define the maximum buffersize for Wide Area Network messages. These buffers are for messages within the local area network. The maximum value is 1498 bytes. This is also the default value.

[5] Die Leitung muß doch dicht zu kriegen sein!
[6] Ist sehr nützlich für die Auslastung des Netzwerks.!
[7] Was ist ein slower device (processor), slower als was?
[8] Da muß aber ein sehr großes ????? ran.

- SEGMENT BUFFER SIZE *number* specifies the size of the segment buffers in bytes. These buffers are used for transmitting DECnet messages from the local area network. The size is a decimal integer in the range of 1 to 16383 for DDCMP communications and 1 to 1466 for Ethernet. The default is the user buffer size +17.
- STATE ON/OFF Nur im state off können bestimmte settings vorgenommen werden. State off wird erst nach dem booten wirksam.
- TRANSMIT PASSWORD *password* ist bis zu acht Zeichen, die zu dem Empfänger geschickt werden, um eine Verbindung aufzubauen. Wie kann die VAX das für die einzelnen Rechner unterscheiden?
- TRANSMIT PIPE QUOTA *number* ist die Zahl der segmente, die der driver in das Netzt schickt, bevor er den Benutzer blockiert. Der Bereich ist 1 bis 65535. Die Voreinstellung ist 6.
- AUTOBOOT ENABLED/DISABLED This parameter is used by MS-NET. The default is ENABLED.

DEFINE LINE *line-id* ist ASYNC-1 oder ETHER-1. Da es nur jeweils eine line gibt, ist dieser Parameter optional.

- DEVICE device-id ist COM-1 oder COM-2 für DDCMP. Circuit muß im state off sein.
- MODEM NULL/FULL Wird der Parameter mit NULL angegeben, so werden die Modem Kontroll-Signale nicht berücksichtigt. Bei FULL wird die line abgeschaltet, wenn die Verbindung unterbrochen wird.
- RECEIVE BUFFERS n Eine Erhöhung der Zahl, die die Installationsprozedur setzt, beeinflußt das Zeitverhalten des Netzes und den Verbrauch von local Memory. Erlaubte Zahlen sind 1 bis 1024.
- SPEED baud-rate Ist die Geschwindigkeit der Leitung bei DDCMP von 50 bis 19 200.
- STATE ON/OFF Definiert den Zustand der line nach dem nächsten Start. Manche Parameter können nur im state off gesetzt oder geändert werden.

DEFINE NODE node-address NAME node-name access-info LAT-HOST MS-NET

- node-address besteht aus Areanummer und Knotennummer. Wird vom System-Manager vergeben.
- node-name besteht aus bis zu 6 alphanumerischen Zeichen mit mindestens einem Buchstaben.
- access-info setzt die Zugriffsrechte für den Knoten in der Form:
  - USER *user-name* ist eine Zeichenkette aus 1 bis 39 alphanumerischen Zeichen die den Benutzer auf dem Knoten bestimmen.
  - PASSWORD *password* ist eine Zeichenkette aus 1 bis 39 alphanumerischen Zeichen die die Zugriffsrechte des Benutzers auf dem Knoten regeln.
  - ACCOUNT *account* ist eine Zeichenkette aus 1 bis 39 alphanumerischen Zeichen, die bei uns zur Zeit nicht benutzt werden.
- LAT-HOST gibt an, daß der Knoten ein LAT host ist.
- MS-NET gibt an, daß der Knoten ein MS-NET server ist. Wird PCSA benutzt, wird ein Eintrag in die NETBIOS server volatile database vorgenommen.

DEFINE OBJECT object-name NUMBER n FILE file-name ARGUMENTS xxx

Es werden Objekte für den SPAWNER definiert und in DECOBJ.DAT eingetragen.

- object-name ist ein DECnet object mit maximal 16 Zeichen.
- n ist die Nummer des Objekts im Bereich 0 bis 255.
- file-name ist ein Kommandofilename mit der Filenamenerweiterung .CMD. Es können Pfadbezeichnungen verwendet werden.

- xxx sind Kommandozeilen Argumente für das mit FILE angegebene Programm. Sind Leerzeichen vorhanden, muß der ganze string in Anführungszeichen gesetzt werden.

DEFINE REMOTE-ADAPTER-NAME

NCP> EXIT

Der Befehl EXIT beendet das Programm NCP. Es kann auch CTRL/Z Ret. benutzt werden.

NCP> HELP

HELP zeigt Informationen über die optional angegebenen Befehle und gegebenenfalls die angegebenen Unterteilungen der Befehle.

NCP> LIST

List zeigt die in der permanent database eingetragenen Parameter. Es können benutzt werden:

LIST	ACCESS
	CIRCUIT
	EXECUTOR
	LINE
	NODE
	OBJECT
	REMOTE-ADAPTER-NAME

NCP> LIST ACCESS

NCP> LIST CIRCUIT

NCP> LIST EXECUTOR

NCP> LIST LINE

NCP> LIST NODE

NCP> LIST OBJECT

NCP> LIST REMOTE-ADAPTER-NAME

NCP> LOOP

LOOP	CIRCUIT
	EXECUTOR
	LINE CONTROLLER
	NODE

NCP> MIRROR

NCP> MONITOR

NCP> PURGE

PURGE	ACCESS
	EXECUTOR
	NODE
	OBJECT
	REMOTE-ADAPTER-NAME

NCP> READ

NCP> SET

SET CIRCUIT
ECHO
EXECUTOR
LINE
LINK
LOCAL-ADAPTER-NAME
PAUSE
REMOTE-ADAPTER-NAME

NCP> SHOW

SHOW ACTIVE
ADJACENT
CIRCUIT
EXECUTOR
LAT
LINE
LOCAL-ADAPTER-NAME
NODE
REMOTE-ADAPTER-NAME

NCP> TELL

NCP> ZERO

ZERO CIRCUIT
EXECUTOR
LAT
LINE

15.9 PCSA Network Commands

PCSA kann nur mit den Knoten benutzt werden, auf denen auch PCSA vorhanden ist, z. Z. nur MSFD1F. Die Services müssen vom System-Manager eingetragen sein.

Die NET-Befehle sind im MSNET.INI definiert. Sie benutzen LDU (nur bis V. 2.2).

15.9.1 Befehle allgemeiner Art

Befehle allgemeiner Art sind Befehle, die sich nicht direkt auf Disk-, File- oder Print-Services beziehen.

BCAST

Sends a Broadcast message to clients on the network.

BCAST [/C] node [,node] |filename message

node ist der Name des Knotens, der die Message erhalten soll. Kann auch ein Gruppenname sein, wenn alle Gruppenmitglieder sie erhalten sollen.

filename ist der Name eines Files, der einspaltige Einträge von Knotennamen enthält. Maximal 200 Einträge sind möglich. Die Extension **muß** .DIS sein, kann in der Angabe zu dem Befehl entfallen.

message ist die Mitteilung, also kein Brief.

/C informiert über **nicht** gesendete Messages.

In dem File BCAST.ID können 12 Zeichen zur persönlichen Kennung abgelegt werden, die mit dem eigenen Knotennamen zu der Message gegeben werden.

LATCP

Manages the Local Area Transport (LAT) and creates preferred services.

ADD Zeigt die zur Zeit verfügbaren Server an. Das Erstellen der Liste dauert ca. 2 Minuten, da sich die Server nur gelegentlich [9] melden. Aus dieser Liste können die bevorzugten, bei denen man Zugriffsrechte hat, ausgewählt und in eine locale Tabelle übernommen werden.

ADD LPTn

DEFINE FALLBACK (ON|OFF) Normalerweise wird nur der Eintrag in der Service-Tabelle mit dem höchsten rating für den gewünschten Service benutzt. In einem cluster kann der gleiche Service mit unterschiedlichen rating auf unterschiedlichen Rechnern ahgeboten werden. Dann wird bei FALLBACK ON die Service-Tabelle abgearbeitet, bis eine Verbindung zustande kommt.

DEFINE GROUP CODES

DEFINE LPTn THROTTLE x Sind mehr als ein örtlicher Drucker vorhanden, kann mit der Angabe von x für die Größe des zu dem Drucker LPTn zu übetragenden Blocks ein Geschwindigkeitsunterschied der Drucker vermindert werden, indem der langsamere Drucker kleinere Blöcke übertragen bekommt, an denen er genau so lange arbeitet, wie der schnellere mit größeren Blöcken.

DEFINE MAX CIRCUITS n Mit n wird die Anzahl der Knoten-zu-Knoten Verbindungen angegeben. Der Wert ist zwischen 1 und 32, voreingestellt ist 4.

DEFINE MULTICAST (ON|OFF)

DEFINE RETRANSMIT LIMIT

DEFINE SCB

DEFINE SEARCH

DEFINE SERVICE TABLE

DEFINE SESSION THROTTLE

DELETE

DELETE LPTn

EXIT

HELP

LIST SERVICES

SHOW

[9] könnte ruhig noch etwas länger sein!

SHOW CHARACTERISTICS

SHOW CIRCUITS

SHOW COUNTERS

SHOW PORTS

SHOW SESSIONS

ZERO COUNTERS

EMSLOAD

Mit dem Befehl EMSLOAD können PCSA-Komponenten in das expanded Memory geladen werden. An den Schaltern der Komponenten ändert sich nichts.

LOGIN (LOGON)

Mit LOGIN wird eine Verbindung zu der persönlichen Direktory auf einem VMS- oder ULTRIX-Server hergestellt. Der File AUTOUSER.BAT wird abgearbeitet. Er ist editierbar.

LOGIN nodename username [password | ⋆] [/VIRTUAL]

nodename ist der Knotenname zu dem die Verbindung hergestellt werden soll.

username ist der Benutzername auf diesem Knoten.

password Ist das Passwort ein Stern, dann wird das Passwort abgefragt und die Antwort nicht auf dem Bildschirm ausgegeben.

/virtual Wenn der drive nicht angegeben wird, wird die Umgebungsvariable _VP genommen (nur vor V 3.0). Bei Version 3.0 wird im AUTOUSER.BAT die Umgebungsvariable _SYSD gesetzt und die Login-Schalter entfallen.

/init_user

/init_workstation

/update_system

/help

NET CLEAR

NET CLEAR nodename

nodename ist der DECnet Knotenname, der aus dem File DECNODE.DAT entfernt werden soll.

NET DEFINE

Dieser Befehl ist in der Version 3.0 nicht mehr verfügbar!
NET DEFINE name address

name Ist die Bezeichnung eines Knotens (Rechner), der mit der Adresse, die vom Systemmanager vergeben wurde, in Verbindung bringt.

address Ist die Adresse des Knotens, die mit dem Namen in Verbindung gebracht werden soll. In unserem Netzt fangen alle Adressen mit 3 an, da wir in dem Göttinger Netz die Area 3 sind.

NET HELP

NET HELP [command]
ATTRIB
CLEAR
COMMANDS
CONTINUE
CREATE
DEFINE
DELETE
DISK
DISMOUNT
DLL
DLL802
DNP
EMSLOAD
ERROR
FILE
HELP
LAD
LAST
LAT
LIST
LOAD
MEMMAN
MODIFY
MOUNT
PASSWORD
PAUSE
PERMIT
PRINT
REDIR
SAVE
SCH
START
TEST
TIME
USE
ZERO

NET LIST

Zeigt eine Liste der dem Rechner bekannten Knoten und Informationen über jeden einzelnen.
NET LIST

NET LOAD

NET SAVE

NET TEST

Ist vergleichbar mit NCP LOOP. Testet die Verbindung zu dem angegebenen Knoten.
NET TEST name

name ist der Rechner, der die Spiegelung vornimmt.

NETTIME

NET TIME [nodename]

nodename ist der Knotenname zu dem die Verbindung hergestellt werden soll.

TRCV

SETLOGON

SETNAME

15.9.2 Befehle für File Services

NET ATTRIB

Damit können die ACL (access control list) Zugriffsrechte für einen oder eine Gruppe von Benutzern geändert werden. Jeder Eintrag ist ein ACE (access control entry).

NET ATTRIB drv:[\\path] file [identifiers] [/query]
[/access=rights]
[/bottom]
[/remove]

drv: ist der den drive kennzeichnende Buchstabe, den man bei dem USE-Befehl benutzt oder bekommen hat.

path ist gegebenenfalls ein auf dem Server zu benutzender Pfad.

file mit Typeerweiterung ist die Datei, deren Attribute geändert werden sollen.

identifiers see your system administrator!

/query es werden die fehlenden Informationen abgefragt.

/access=option ist eine der folgende:

1. Read für nur lesen.
2. Write für lesen und schreiben.
3. None für keinen Zugriff.

/bottom Jeder ACE wird am Anfang der ACL eingetragen. Mit diesem Schalter kann der neue an das Ende gesetzt werden, so daß ältere Einträge wirksam bleiben.

/remove Hiermit werden ACE's aus der ACL ausgetragen.

NET CONTINUE

Hebt die zeitweilige Unterbrechung des Befehls NET PAUSE auf.

NET CONTINUE [drdr]|[prdr]

drdr stellt alle File Services wieder her.

prdr stellt alle Printer Services wider her.

NET FILE SERVICES

Mit Version 3 nicht benutzbar. Zu benutzen ist **use \\node /show**. Damit werden die auf dem Knoten *node* vorhandenen File Services angegeben.

NET FILE SERVICES \\node[%username] [password]

node

username

password

NET PASSWORD

Damit kann das Passwort auf dem Knoten *node* geändert werden. Es gelten die gleichen Bedingungen wie auf der VAX selber!

NET PASSWORD \\node[%username]

node ist der Knoten, auf dem das Password geändert werden soll.

username ist der Benutzer, für den die Änderung gelten soll.

NET PAUSE

Soll der lokale Drucker an LPT1: benutzt werden, das Device LPT1: aber auf einen Netzwerkdrucker zeigt, kann man mit dem Befehl *NET PAUSE* die Netzwerkverbindung unterbrechen. Mit dem Befehl *NET CONTINUE* kann sie wieder hergestellt werden.

NET PAUSE drdr|prdr

DRDR unterbricht alle File Services.

PRDR unterbricht alle Printer Services.

NET PRINT

Mit *NET PRINT* wird der File PRINTQ aus der Directory DECNET geladen. Damit geht das Printen erheblich schneller als mit dem DOS Befehl Print. MS-NET.INI wird benutzt. Daher können nur die der Druckerschlange zugeordneten settings benutzt werden. Werden andere gewünscht, so muß man alle gleichzeitig angeben. Nicht jeder Schalter ist für jedes device gültig oder sinnvoll. Der Schalter /COPIES:n z. B. kann bei der PS-Druckerschlange **nicht** benutzt werden!

NET PRINT \\nodename
NET PRINT [filename] device

nodename ist der Knotenname, zu dem eine Verbindung besteht.

filename ist der File, der gedruckt werden soll.

device ist LPT1, LPT2 oder LPT3. Devices werden immer mit Doppelpunkt angegeben!

Die logischen Drucker können mit dem Befehl *NET PRINT LPTn: /SET* für bestimmte Aufgaben vorbereitet werden. Die gesetzten kann man sich mit *NET PRINT dev: /SHOW* anzeigen lassen. Wird die Angabe des device fortgelassen, werden alle aktivierten Druckerwarteschlangen angezeigt. Zur Verfügung stehen:

/* das augenblickliche setting wird benutzt.

/AFTER=time Verzögert das Drucken bis zu der angegebenen Zeit in der Form: dd-mmm--yyyy:hh:mm, wobei:

- dd der Tag in der Form 01, 02, ..., 31 ist.
- mmm der Monat in der Form Jan, Feb, ..., Dec ist.
- yyyy das Jahr in der Form 1988, ..., 1992 ist.
- hh die Stunde in der Form 00, ..., 23 ist.
- mm die Minute in der Form 00, ..., 59 ist.

/BURST druckt über die Perforation bei endlosem Papier, um die einzelnen Jobs zu unterscheiden. Bei Laserdruckern mit Einzelblatt nicht sinnvoll.

/NOBUSRT ist die Standardeinstellung.

/CHARACTERISTICS=(code,*ldots*) die vom Systemverwalter definierten characteristics kann man sich mit dem *SHOW QUEUE/CHAR* Befehl ansehen.

/COPIES=n ist die Anzahl Kopien, bis zu 255, die erzeugt werden.

/FLAG ist Standard und erzeugt eine zusätzliche erste Seite mit Jobinformationen und Benutzernamen.

/NOFLAG sollte man nur nehmen, wenn man seine Zettel suchen will!

/FORM=type die vom Systemverwalter eingerichteten forms können mit dem Befehl show queue/form erhalten werden.

/HEADER druckt eine Kopzeile auf jede Seite.

/NOHEADER ist der Standard.

/JOB_COUNT=n ist eine Zahl zwischen 1 und 255. Entspricht dem */COPOY*

/LOWERCASE zeigt an, daß der Druck mit Kleinschreibung erfolgen soll.

/NOLOWERCASE ist der Standard.

/NAME=job-name kann 1 bis 39 alphanumerische Zeichen sein und wird auf der Flag Page gedruckt und im Queuelisting gezeigt.

/NOTE=string bis zu 255 Zeichen, die auf der flag page gedruckt werden.

/OPERATOR=string bis zu 255 Zeichen, die dem Operator übermittelt werden, sobald der Druck anfängt.

/PAGES=lowlim,uplim damit können die Anfangs- und Endseite eines längeren Files angegeben werden. Fehlt die erste Angabe (lowlim), wird ab der ersten gedruckt. Kennt man die Seitenzahl der letzten nicht, kann als uplim eine 0 angegeben werden.

/PARAMETER=(param,*ldots*) können jeweils 255 Zeichen enthalten. Acht Parameter sind möglich. Die Summe aller Zeichen in allen Parametern muß kleiner 255 sein. Sie müssen durch Kommata getrennt sein. Die Parameter hängen von dem Drucker ab (Papierfach, Orientierung, Verkleinerung). /PARAM="UNNUMBER_UP=2" druckt bei einem PrintServer40 zwei Seiten verkleinert auf einer Seite ab.

/PASSALL

/NOPASSALL

/PRIORITY=n legt die Priorität zwischen 0 (kleinste) und 255 (höchste) fest.

/RESTART startet den Ausdruck erneut nach einem Systemabsturz oder einem STOP/QUEUE.

/NORESTART startet nicht erneut.

/SETUP=(module,*ldots*) kopiert die angegebenen Module aus der "device control library" in den programmierbaren Drucker.

/SPACE erzeugt "double space".

/NOSPACE normal.

/TRAILER erzeugt eine Folgeseite mit Jobinformationen.

/NOTRAILER Standard.

NET USE

Mit diesem Befehl kann eine Verbindung zu einem File oder Print Service aufgebaut werden.

```
NET USE drive: [\\node\]service [password|⋆ [/V]
                                            [/F|/S]
                                            [/RW|/RO]
                                            [/X]
```

drive sollte immer in der allgemeinen Form *?:* angegeben werden. Damit wird die nächstfolgende logische Drivebezeichnung vergeben. Wird dem Befehl */env=var* angefügt, wird die Drivebezeichnung in der vorhandenen Umgebungsvariablen *var* einschließlich Doppelpunkt abgelegt und man kann die Variable *var* benutzen.

node ist zur Zeit nur die MSFD1F.

service ist eine der Druckerschlangen, die eigene Directory oder ein mit dem Befehl *use* *MSFD1F* \ */show* angegebener Service.

password sollte aus Sicherheitsgründen immer mit * angegeben werden. Wird für *public* Services nicht benötigt.

Die oben angeführten und nicht erläuterten Schalter sind nicht mehr gültig oder haben mit den File Srvices nichts zu tun und werden an anderer Stelle erläutert.

PERMIT

Erlaubt eine Verbindung zwischen zwei PC's.

```
PERMIT alias=drive:\path [/R] [/W] [/C] nodename|⋆
```

alias ist der Name, der den ganzen Pfad zu einer Direktorie darstellt, die mit einem anderen PC geteilt werden soll.

drive: ist die Laufwerksbezeichnung, die für diese Directory über das Netz angeboten werden soll.

path ist der ganze Pfad zu der Directory, die über das Netz angeboten werden soll.

/R erlaubt nur lesen.

/W erlaubt schreiben. /C muß auch angegeben werden, wenn editieren erlaubt sein soll (Backupfiles).

/C erlaubt das Eröffnen neuer Files.

nodename ist der Knotenname mit dem die Disk oder die Direktory geteilt werden soll (gemeinsam benutzt). Wird ein * angegeben, können alle sie benutzen.

15.9.3 Befehle für Virtual Disk Services

Öffentliche Disk Services werden vom Systemmanager zur Verfügung gestellt. Welche Disk Services zur Zeit zur Verfügung stehen, kann man mit dem Befehl USE
MSFD1F /SHOW /VIRTUAL erfahren.
Einen Teil seines Plattenplatzes kann man selbst verwalten. Das kann mit den Befehlen *NET CREATE* und *NET DELETE* geschehen.
Einen eigenen Disk Service kann man mit der folgenden Befehlsfolge erzuegen und benutzen.

1. **NET CREATE \\MSFD1F%user ⋆ /QUE /FILE=filename.DSK**
2. **NET MOUNT \\MSFD1F\servicename%user ⋆ /QUE /FILE=filename.DSK**
3. **USE ?: \\MSFD1F\servicename%user ⋆/virtual**

Für *user* und * muß der Benutzername und das Passwort eingegeben werden. Der Parameter /QUE sollte angegeben werden, um die restlichen Parameter angeben zu können. Der anzulegende File sollte die Erweiterung DSK erhalten. Bei dem zweiten Befehl kann für den einzurichtenden Service ein Passwort angegeben werden. War das der Fall, so muß es auch mit dem USE Befehl benutzt werden.

NET CREATE

Mit diesem Befehl wird ein neuer File, der immer die Extension *.DSK* haben sollte. Er ist für den PC eine Disk, für die VAX ein Kontainerfile. Dieser kann mit der VAX-Utility *PCDISK* bearbeitet werden.

```
NET  CREATE  \\node[%username]  [password|⋆]  [/query]
                                              /FILE=file-spec
                                              [/size=n]
                                              [/type=class]
                                              [/alloc=n]
```

node ist der Knoten, auf dem man Plattenplatz vergeben kann, z. Z. nur MSFD1F.

username ist der Benutzername, der auf diesem Knoten den Plattenplatz zur Verfügung stellt.

password ist das Passwort für diesen Benutzer.

/query sollte immer angegeben werden, da dann die fehlenden Informationen abgefragt werden. Diese können nicht mit dem Cursor beantwortet werden, sondern müssen eingetippt werden.

/file=file-spec muß angegeben werden und immer mit .dsk enden.

/type=class kann nur **USER** sein, da die anderen entsprechende Privilegien benötigen.

/size=n ist die maximale Größe für die Disk.

/alloc=n ist die Anzahl der Blöcke, die angelegt werden sollen.

NET DELETE

```
NET  DELETE  \\node[%username]  [password|⋆]  /FILE=file -spec
                                              [/QUERY]
                                              [/TYPE=class]
```

node

username ist der Benutzername, der auf diesem Knoten den Plattenplatz zur Verfügung stellt.

password ist das Passwort für diesen Benutzer.

/QUERY es werden die fehlenden Informationen abgefragt.

/FILE=file-spec

TYPE=class

NET DISK SERVICES

```
NET  DISK  SERVICES  \\node%username  [password|⋆]
NET  DISK  SERVICES  service
```

node

username

password ist das Passwort für diesen Benutzer.

service

NET DISMOUNT

```
NET  DISMOUNT  [drive:]  \\node\service[%username]  [password|⋆]
                                                    [/QUERY]
                                                    [/CLUSTER]
                                                    [/TEMPORARY]
                                                    [/TYPE=class]
```

drive

node

service

username ist der Benutzername, der auf diesem Knoten den Plattenplatz zur Verfügung stellt.

password ist das Passwort für diesen Benutzer.

/QUERY es werden die fehlenden Informationen abgefragt.

/CLUSTER

/TEMPORARY

/TYPE=class

NET ERROR

```
NET  ERROR
```

NET MODIFY

```
NET  MODIFY  \\node\service[%username]  [password| ⋆  [/QUERY]
                                                      /FILE=file-spec
                                                      [/TYPE=class]
                                                      [/CONNECTIONS=n]
                                                      [/EXTENSION=n]
                                                      [/PASSWORD=string]
                                                      [/NOPASSWORD]
                                                      [/RATING=n]
```

node

service

username ist der Benutzername, der auf diesem Knoten den Plattenplatz zur Verfügung stellt.

password ist das Passwort für diesen Benutzer.

/QUERY es werden die fehlenden Informationen abgefragt.

/FILE=file-spec

/TYPE=class

/CONNECTIONS

/EXTENSION

/NOPASSWORD

/PASSWORD

/RATING

NET MOUNT

NET MODIFY \\node\service[%username] [password| ⋆ [/QUERY]
/FILE=file-spec
[/TYPE=class]
[/CONNECTIONS=n]
[/PASSWORD=string]
[/RO|/RW]
[/TEMPORARY]
[/RATING=n]

node

service

username ist der Benutzername, der auf diesem Knoten den Plattenplatz zur Verfügung stellt.

password ist das Passwort für diesen Benutzer.

/QUERY es werden die fehlenden Informationen abgefragt.

/FILE=file-spec

/TYPE=class

/CONNECTIONS

/PASSWORD

/RATING

/RO

/RW

/TEMPORARY

NET USE

NET USE drive: [\\node\]service [password|⋆ [/V]
[/F|/S]
[/RW|/RO]
[/X]

drive

node

service

password ist das Passwort für diesen Benutzer.

/V

/F

/S

/RW

/RO

/X

NET ZERO LAD

NET ZERO LAD

15.9.4 Befehle zum Starten des Netzwerks

NET START LAD

NET START LAD [/A:D|E|-1]
[/IRQ:n]
[/N:name]
[/M:D|E]
[/R:n]
[/C:D|E]
[/T:n]
[/W:n]
[/G:n]

/A:

/IRQ:n

/N:name

/M:D|E

/R:n

/C:D|E

/T:n

/W:n

/G:n

NET START LAT

NET START LAT [/IRQ:N]
[/D:size]
[/N]
[/R:retransmission]
[/G:group]
[/T:n]

IRQ:n

/D:size

/N

/R:retransmission

/T:n

/G:group

NET START RDR

NET START RDR [/IRQ:n]
[/M:D]
[/MSN:n]
[/REM:n]
[/NBS:n]
[/CMD:n]
[/LAN:n]
[/NAM:Y|N]
[/I2A:Y|N]
[/Pn:x]
[/S:n]
[/T:n]

/IRQ:n

/M:D|E

/MSN:n

/REM:n

/NBS:n

/CMD:n

/LAN:n

/NAM:Y|N

/I2A:Y|N

/Pn:x

/S:n

/T:n

NET START RDR802

NET START RDR [/IRQ:n]
[/M:D]
[/MSN:n]
[/REM:n]
[/NBS:n]
[/CMD:n]
[/LAN:n]
[/NAM:Y|N]
[/I2A:Y|N]
[/Pn:x]
[/S:n]
[/T:n]

/IRQ:n

/M:D|E

/MSN:n

/REM:n

/NBS:n

/CMD:n

/LAN:n

/NAM:Y|N

/I2A:Y|N

/Pn:x

/S:n

/T:n

15.9.5 Befehle zum Beenden des Netzwerks

Diese Befehle sind in Versionen vor 3.0 nicht vorhanden! Nur CTERM lässt sich mit dem Schalter /r (remove) entfernen. Wird eine andere Umgebung, z. B. für T^3, benötigt, muß CONFIG.SYS und AUTOEXEC.BAT erneuert und dann gebootet werden. Bei DECPCA-Karten **muß** Reset betätigt werden. Bei MICOM-Karten kann das Program *REBOOT* benutzt werden.

Mit Version 3.0 wird ein STOPNET.BAT zur Verfügung gestellt. Es darf, wie das STARTNET.BAT, nur aus der Root-Directory, also C:, benutzt werden!

Mit der Version 4.0 befinden sich die .BAT-Files in der DECnet Directory. PCAPP muß unbedingt im path enthalten sein weil mehrere Aplikationen sich dort befinden. æ

Kapitel 16

Ethernet

Zu einem Ethernet-Knoten gehören die Teile:

AUI Attachment Unit Interface

DTE Data Terminal Equipment

MAU Media Access Unit

Durch die immer höhere Integration werden jetzt auch Chips angeboten, die für mehrere unterschiedliche Prozessoren gedacht sind. So der Am79C900 von Advanced Micro Devices, dessen Vorgänger auf der DEPCA ist, für die 29000, 80X86 und 680X0. Er wird ILACC (Integrated Local Area Communications Controller) genannt. Er beinhaltet die Funktionen des Am7990 (LANCE) und des AM7992A (SIA).

16.1 Pin Beschreibung

Die Pin-Belegung variiert mit den Gehäuseformen.

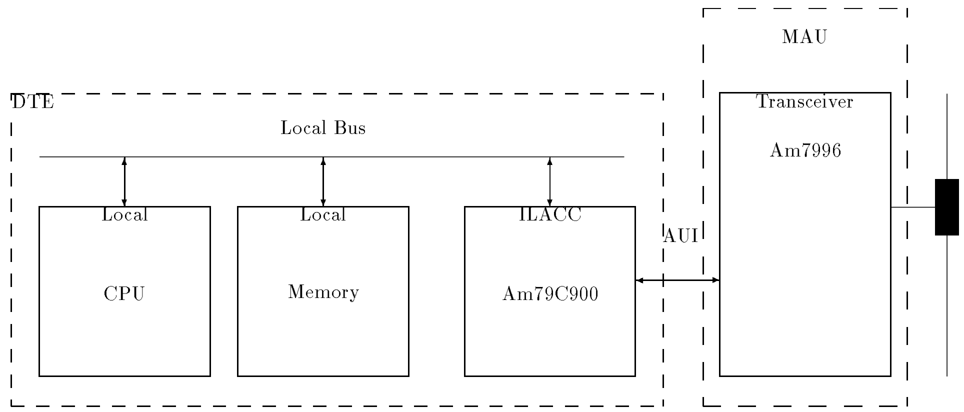


Abbildung 16.1: Ethernet Anschlußeinheit

ALE,AS Address Latch Enable/Address Strobe (Input/Output, High Impedance).

Wird zum demultiplexen des DAL-Bus benutzt und definiert den Adressteil des Memory-Zyklus. AS ist die logische Inversion des ALE. Die Polarität des Pins ist über ACON (CSR3 Bit 1) programmierbar:

ACON = 0 : ALE (die fallende Flanke fängt die Adresse).

ACON = 1 : AS (die steigende Flanke fängt die Adresse).

Wird als Input während der Bus-Verteilung benutzt, um das Ende des vorhergegangenen Bus-Master-Transfers festzustellen.

BCLK Bus Clock (Input).

BCLK ist die Clock-Frequenz für das Microprozessor-Interface. BCLK muß mindestens 1,2 mal die Frequenz des Netzwerkes, in dem Bereich von 2 bis 20 MHz, sein. Wenn das interne SIA benutzt wird, ist dies 0,6 mal die Frequenz von XTAL1/2. Für ein Netzwerk mit einer Datenrate von 10 Mbits/s ergibt sich also XCLK = 20 MHz, BCLK(min) = 12 MHz, BCLK(max) = 20 MHz.

BGACK,MASTER Bus Grant Acknowledge (Input/Output, Open Drain).

Wird nur in 680X0 Systemen benutzt.

MASTER Master (Output, Open Drain).

Gültig wenn der ILACC der Bus-Master (BACON = 00) ist, um die Data/Address Bustreiber zu erlauben.

C/D Control/Data Select (Input).

Wird während Slave-Zyklen benutzt um zwischen Control und Data Informationen zu unterscheiden.

C/D = 0 : Data Port benutzt.

C/D = 1 : Register Address Port benutzt.

Der Input wird ignoriert, wenn der ILACC Bus-Master ist.

CDT Collision Detect (Input).

Wenn dieser Pin aktiviert wird sind mehrere Knoten auf dem Medium vorhanden. Dieser Pin wird nicht gebraucht und sollte nach low gelegt werden, wenn das interne SIA (PORTSEL = 0, CSR15 Bit 8) benutzt wird.

COLL (+),(-) Collision Detect (Input).

Ein differentieller Input in Ethernet/802.3 Anwendungen. Wenn das interne SIA (PORTSEL = 1, CSR15 Bit 8) nicht benutzt wird, sollten diese Eingänge auf Ground gelegt werden.

CRS Carrier Sense (Input).

Muß belegt werden, wenn erlaubte Daten von einem externen Transceiver empfangen werden. CRS sollte nach low gelegt werden, wenn das interne SIA (PORTSEL = 0, CSR15 Bit 8) benutzt wird.

CS Chip Select (Input).

Muß in Verbindung mit C/D benutzt werden, um and die internen Register zu kommen. Wird bei Bus-Master Zyklen ignoriert.

$DAL_0 - DAL_1, BE_2 - BE_3$ Data/Address/Byte enable lines (Input/Output, High Impedance).

BACON = 00 und Master-Zyclus. Während des Adressteils des Memory-Transfers führen die BE Pins die Memory Bank Selekt Signale in einem 80X86 System. Während des Datenteils des Memory-Transfers führen die Pins die Daten und werden dann mit $DAL_0 und DAL_1$ bezeichnet.

BACON = 00 und Slave-Zyklus. Es sind Daten bzw. Adress Pins.

Data/Address lines Data/Address lines (Input/Output, High Impedance).

BACON = 01 und Master-Zyclus. Während des Adressteils des Memory Transfers führen die Pins die Adressen $A_1 und A_0$ in einem 680X0 System. Während des Datenteils des Memory-Transfers führen die Pins die Daten und werden dann mit $DAL_0 und DAL_1$ bezeichnet.

BACON = 01 und Slave-Zyklus. Es sind Daten bzw. Adress Pins.

$DAL_2 - DAL_{31}$ Data/Address Lines (Input/Output, High Impedance).

Während des Adressteils des Memory Transfer Zyklus führen die Pins die Adress Informationen. Während des Datenteils führen sie die Daten.

DALI Data/Address Line In (Output).

Signal zum Freischalten des Datenweges zum ILACC. Aktive in Master und Slave Zyklen.

DALO Data/Address Line Out (Output).

Signal zum Freischalten des Datenweges vom/zum ILACC. Aktive in Master und Slave Zyklen.

DAS Data Strobe (Input/Output, High Impedance).

Definiert den Datenteil eines Bustransfers. Input bei Bus Slave Zyklen, Output bei Bus Master Zyklen.

HLDA,BUSACK Hold Acknowledge (Input).

Reaktion eines anderen potentiellen Bus-Master, daß die Busbelegung aufgegeben wurde; in einer 80x86 Umgebung (BACON=00).

Bus Acknowledge (Input).

In einer 680x0 Umgebung (BACON=01)

HOLD,BUSREQ Hold (Output, Open Drain).

Bus Request (Output, Open Drain).

INTR Interrupt (Output, Open Drain).

RCV (+),(-) Receive Data (Input).

R/W Read/Write (Input/Output, High Impedance).

READYL Ready Low (Input/Output, Open Drain).

RESET System Reset (Input).

Reset setzt die interne Logic zurück. Alle Ausgänge werden hochohmig geschaltet.

RINTR Receive Interrupt (Output, Open Drain).

RTS Request To Send (Output).

RxC Receive Clock (Input).

RxD Receive Data (Input).

$SIZ_0 - SIZ_1, BE_0 - BE_1$ Size (Output, High Impedance).

TxC Transmit Clock (Input).

TxD Transmit Data (Output).

VCC Power supply (6 Pins).

+5-V Spannungsversorgung für die interne Interface Logig und die I/O Pins.

VSS Ground (10 Pins).

0-V Referenz für die interne Interface Logig und die I/O Pins.

AVCC Analog Power supply.

+5-V Spannungsversorgung für den analogen Teil des internen SIA.

AVSS Analog Ground Reference.

0-V Referenz für die analogen Funktionen des internen SIA.

XCLK Clock (Output).

XMIT (+), (-) Transmit Data (Output).

Ein differentieller Output um Manchester encoded Daten vom SIA zu senden. Es werden pseudo-ECL Pegel verwendet.

XTAL (1), (2) Crystal Oscillator (Input).

NC No Connection. Do not connect. Im Gegensatz zu Anderen dürfen diese Anschlüße nicht als Stützpunkt verwendet werden!

16.2 Funktionen

16.2.1 General

Der ILACC wird programmiert durch 59 Kontroll- und Status-Register (CSR's) auf dem Chip und Datenstrukturen im Memory. Er kann sowohl selbständig die Buffer verwalten als auch Datenpakete vom und zum Netz bearbeiten. Es werden drei Memorystrukturen gebraucht:

1. Initialization Block

   Sieben 32-Bit Memoryeinträge beginnend an einer Langwort-Grenze für die notwendigen Parameter.

   - Mode of Operation
   - Physical Address
   - Logical Address Filter
   - Pointers to Receive and Transmit Descriptor Rings
   - Number of Entries in Receive and Transmit Descriptor Rings

2. Receive and Transmit Descriptor Rings

   Zwei zusammenhängende Ringstrukturen im Memory für die Kontrolle der Receive- und Transmit-Pakete. Sie bestehen aus:

   - Adresse eines Datenbuffers
   - Status und Error Informationen für den Buffer
   - Länge des Buffers

3. Data Buffers

   Data Buffers sind Memorybereiche, die für das Zwischenspeichern der Pakete, die an einer beliebigen Wortgrenze beginnen können, vorzusehen sind. Jeder Buffer muß einen zusammenhängenden Adressbereich umfassen. Jeder Buffer kann im Memory beliebig plaziert werden.

   Die Programmierreihenfolgen kann folgend zusammengefaßt werden:

   (a) Die CSR's des ILACC's müssen mit Daten versehen werden, um den Initialisierungs-Block im Memory bekannt zu machen. Die Byte Control, Byte Addressing, Address Latch Enable und Bus Arbitration Modes müssen definiert werden; also die Umgebung angegeben werden (80x86, 680x0, 29000).
   (b) Die letztgenannten müssen initialisiert werden, mit Daten belegt.
   (c) Der Initialisierungs-Block muss über DMA geladen werden.
   (d) Zugang zu den Descriptor Ringen und den Daten-Buffern herstellen.

   Der ILACC hat einen 32-Bit breiten Adressbereich, mit dem er mit DMA als Master auf den ganzen Speicher des Prozessors zugreifen kann.

   Interrupts zu dem Prozessor werden in den folgenden Fällen erzeugt.

   (a) Nach dem Abarbeiten der Initialisierungs Routine.
   (b) Wenn ein Datenpaket erhalten wurde.
   (c) Am Anfang eines zu sendenden Paketes.
   (d) Wenn das zu sendende Paket abgearbeitet wurde.
   (e) Transmitter time-out Fehler.
   (f) Bei einem fehlenden Paket.
   (g) Bei einem Memory Fehler.
   (h) Ende eines loopback Test.

16.2.2 ILACC Bus Configuration

Der ILACC hat 32-Bit Daten- und Address-Bus. Mit den BACON Bits kann zwischen Litle-Endian und Big-Endian (680X0 und 80X86) unterschieden werden.

16.2.3 Bus Cycles

Abhängig von der Benutzung kann der ILACC als 'bus slave' (memory or I/O mapped) oder als 'bus master' (DMA) Device arbeiten.

Bus Slave Cycles

Slave Zyklen werden von dem Prozessor ausgeführt um den ILACC zu initialisieren oder den Status abzufragen. Der Host kann Schreib- oder Lesezugriff zu den CSR's bekommen, wenn er das Signal CS auf 0 legt. Dadurch wird der ILACC in den bus slave mode geschaltet. Zuerst muß die Adresse des gewünschten Registers in den Register Address Pointer (RAP) geschrieben werden. Dann kann auf das Register über das Data Port zugegriffen werden. Im slave mode werden nur die DAL_{0-15} benutzt.

Bus Acquisition

Der Buszugriff kann über zwei oder drei Signale gesteuert werden. Das bestimmen die Bus Acquisition Control (BACON) Bits in CSR4 wie folgt:

BACON	Bus-Config.
00	80X86
01	680X0
10	sereved
11	reserved

Bei 80X86 Prozessoren wird die Kontrolle mit zwei Leitungen ausgeübt. HOLD (hold request) und HLDA (hold acknowledge).
Bei 680X0 Prozessoren wird die Kontrolle mit drei Leitungen ausgeübt. BUSREQ (Bus Request), BUSACK (Bus Acknowledge) und BGACK (Bus Grant Acknowledge).
Der ILACC belegt den Bus wenn empfangene Pakete in das Memory gebracht werden sollen, er nach einer Transmitmessage sucht oder einen zu sendenden Buffer transferiert.
Wenn der ILACC den Bus freigibt und es sind schon oder noch 16 Byte in dem FIFO, wird der Bus erst nach 4 Busclock für empfangen und 10 Busclock für senden erneut belegt.

Bus Master Cycles

Als DMA Transfer kommen in Frage:

1. single-cycle DMA
2. dual-cycle DMA
3. burst-cycle DMA

Single-cycle DMA Transfers
Nachdem der ILACC den Bus belegt hat, benutzt er eine Lang-Wort-Adresse und liest oder schreibt. Danach wird der Bus freigegeben.
Dual-cycle DMA Transfer
Nachdem der ILACC den Bus belegt hat, benutzt er eine Lang-Wort-Adresse und liest oder schreibt. Danach wird die gleiche Operation (lesen oder schreiben) mit einer anderen Adresse, nicht unbedingt der folgenden, ausgeführt. Danach wird der Bus freigegeben.
Burst-cycle DMA Transfer
Nachdem der ILACC den Bus belegt hat, wird eine Reihe von gleichen Transfers mit Langwort-Adressen ausgeführt. Die Länge hängt von DMAPLUS (CSR4, Bit14) ab. Ist es 0, werden nicht mehr als 4 Transfers gemacht. Ist es 1, wird bis zu der oberen oder unteren Grenze des FIFO gearbeitet.

Bus Master Transactions

1. initialization block read access
2. descriptor read/write access
3. data buffer read/write access

Summary of 32-Bit Memory Transfer

Serial Interface

Data Flow Overview

Error Reporting and Diagnostic

Diagnostics

Cyclic Redundancy Check (CRC)

16.3 Programmable Resources

Im Folgenden werden die Kontrol- und Statusregister und die Daten-Strukturen im Memory beschrieben, die nötig sind, um den ILACC zu programmieren.

16.3.1 User programmable Registers

Auf die internen Register kann nur mit einer Doppel-Operation erfolgen. Zuerst muß in das Register Adress Port (RAP) die Adresse des gewünschten Registers geschriben werden. Dann kann des angewählte Register über das Data-Port gelesen oder geschrieben werden.

Der C/D Pin erlaubt die Wahl zwischen Data-Port und RAP:

C/D = 0 = Data Port

C/D = 1 = Register Address Port

Die oberen 16-Bit des RAP sind nicht definiert, von den unteren werden nur die Bits 0 bis 5 für die Selektierung des Control und Status Registers (CSR) benutzt.

16.4 Control and Status Registers

Auf die CSR's kann nur über das Data-Port (C/D=0) zugegriffen werden. Zuvor musste die Adresse des gewünschten in das RAP (C/D=1) geschrieben werden.

Unabhängig vom Zustand des STOP-Bits kann nur auf die Register CSR0 und CSR3-4 zugegriffen werden. Für CSR1-2 muß vorher das STOP-Bit im CSR0 gesetzt wirden sein.

Für alle untergeorneten Operationen wird der Datentransfer über die unteren 2 Byte (DAL_{15-0}) abgewickelt. Die ($DAL31-16$) sind nicht definiert.

Die Adressen 5 - 58 können nur gelesen werden, vorausgesetzt, das STOP-Bit ist gesetzt.

RAP	CSR	Bedeutung
00	CSR0	Control und Status Register 0
01	CSR1	Control und Status Register 1
02	CSR2	Control und Status Register 2
03	CSR3	Control und Status Register 3
04	CSR4	Control und Status Register 4
05	CSR5	Transmit Collision Count
06	CSR6	RX/TX Descritor Table Length
07	CSR7	Receive Coll./Runt Packet Cnt.
08	CSR8	Logical Address Filter 15-0
09	CSR9	Logical Address Filter 31-16
00	CSR10	Logical Address Filter 47-32
11	CSR11	Logical Address Filter 63-48
12	CSR12	Physical Address 15-0
13	CSR13	Physical Address 31-16
14	CSR14	Physical Address 47-32
15	CSR15	Mode Register
16	CSR16	Initialization Block Address 15-0
17	CSR17	Initialization Block Address 31-16
18	CSR18	Current RX Buffer Address 15-0
19	CSR19	Current RX Buffer Address 31-16
20	CSR20	Current TX Buffer Address 15-0
21	CSR21	Current TX Buffer Address 31-16
22	CSR22	Next RX/TX Buffer Address 15-0
23	CSR23	Next RX/TX Buffer Address 31-16
24	CSR24	Base Address of RX Ring 15-0
25	CSR25	Base Address of RX Ring 31-16
26	CSR26	Next Address of RX Ring 15-0
27	CSR27	Next Address of RX Ring 31-16
28	CSR28	Current Address of RX Ring 15-0
29	CSR29	Current Address of RX Ring 31-16
30	CSR30	Base Address of TX Ring 15-0
31	CSR31	Base Address of TX Ring 31-16
32	CSR32	Next Address of TX Ring 15-0
33	CSR33	Next Address of TX Ring 31-16
34	CSR34	Current Address of TX Ring 15-0
35	CSR35	Current Address of TX Ring 31-16
36	CSR36	Runt Backup Buffer Address 15-0
37	CSR37	Runt Backup Buffer Address 31-16
38	CSR38	Retry Backup Buffer Address 15-0
39	CSR39	Retry Backup Buffer Address 31-16
40	CSR40	Current RX Byte Count
42	CSR42	Current TX Byte Count
44	CSR44	Next RX/TX Byte Count
46	CSR46	Poll Time Count
48	CSR48	HI Address init Block
50	CSR50	Current RX Status
52	CSR52	Current TX Status
54	CSR54	Next RX/TX Status
56	CSR56	Runt Backup Byte Count
58	CSR58	Retry Backup Byte Count

Tabelle 16.1: ILACC Control und Status Register

Control and Status Register 0 (CSR0)

Der ILACC erneuert den Inhalt des Registers durch eine OR Operation des alten mit dem neuen Zustand. Die Bedeutung der Bits ist:

15 ERR

14 BABL

13 CERR

12 MISS

11 MERR

10 RINT

09 TINT

08 IDON

07 INTR

06 INEA

05 RXON

04 TXON

03 TDMD

02 STOP

01 STRT

00 INIT

Control and Status Register 1 (CSR1)

15 - 00 IADR Die unteren 16 Bit der ersten Adresse des Initialisierungs Blocks. Wird mit CSR2 zur Basis Adresse des Initialisierungs Blocks. In einer Byte Maschine müssen bei einer 32-Bit Adresse die unteren beiden natürlich nicht gesetzt werden!

Control and Status Register 2 (CSR2)

15 - 00 IADR Die oberen 16 Bit der ersten Adresse des Initialisierungs Blocks. Wird mit CSR1 zur Basis Adresse des Initialisierungs Blocks.

Control and Status Register 3 (CSR3)

15 RES

14 BABLM

13 RES

12 MISSM

11 MERRM

10 RINTM

09 TINTM

08 IDONM

07 -03 RES

02 BSWP

01 ACON

00 RES

Control and Status Register 4 (CSR4)

15 RES

14 DMAPLUS

13 - 08 RES

07 - 06 BACON

05 - 04 RES

03 TXSTRT

02 TXSTRTM

01 LBE

00 LBEM

16.5 User-readable Registers

Transmit Collision Count (CSR5)

RX/TX Descriptor Table Length (CSR6)

Receive Collision/Runt Packet Count (CSR7)

Logical Address Filter (CSR8 - 11)

Physical Address Register (CSR12 - 14)

Mode Register (CSR15)

Initialization Block Address (CSR16 - 17)

Current RX Buffer Address (CSR18 - 19)

Current TX Buffer Address (CSR20 - 21)

Next RX/TX Buffer Address (CSR22 - 23)

Base Address of RX Ring (CSR24 - 25)

Next Address of RX Ring (CSR26 - 27)

Current Address of RX Ring (CSR28 - 29)

Base Address of TX Ring (CSR30 - 31)

Next Address of TX Ring (CSR32 - 33)

Current Address of TX Ring (CSR34 - 35)

Runt Backup Buffer Address (CSR36 - 37)

Current RX Byte Count (CSR40)

Current TX Byte Count (CSR42)

Next RX/TX Byte Count (CSR44)

Poll Time Count (CSR46)

Hi Address Init Block (CSR48)

Current RX Status (CSR50)

Current TX Status (CSR52)

Next RX/TX Status (CSR54)

Runt Backup Byte Count (CSR56)

Retry Backup Byte Count (CSR58)

16.6 Initialization

æ

Anhang A

Abkürzungen

ACK	**ACK**nowledgement. Kontrollcode und Bezeichnung für erfolgreich abgeschlossenen Empfang.
ACR	fixed disk **A**dapter **C**ontroller **R**egister
ALE	**A**ddress **L**atch **E**nable
ANSI	**A**merican **N**ational **S**tandards organisation
API	**A**nwendungs-**P**rogramm-**I**nterface.
ARB	**A**ccess **R**ights **B**yte (part of 80286/80386 instruction)
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
ASIC	**A**pplication-**S**pecific **I**ntegrated **C**ircuit
ASR	fixed disk **A**dapter **S**tatus **R**egister
AT	**A**dvanced **T**echnology
b	Binary
BAT	**B**asic **A**ssurance **T**est (Keyboard diagnostic)
BCD	**B**inary **C**oded **D**ecimal
BIOS	**B**asic **I**nput/**O**utput **S**ystem
bps	**B**it **P**er **S**econd
BSC	**B**inary **S**ynchronous **C**ommunication.
CAS	**C**olumn **A**ddress **S**trobe
CCB	**C**ommand **C**ontrol **B**lock
CGA	**C**olor **G**raphics **A**dapter
CLK	**CL**oc**K** line on a mocropocesseor
CMOS	**C**omplementary **M**etal **O**xide **S**emiconductor
CRC	**C**yclic **R**edundancy **C**heck
CSB	**C**ommand **S**pecify **B**lock

CSMA/CD	**C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection.
CTS	**C**lear **T**o **S**end
DAC	**D**igital-to-**A**nalog **C**onverter
DCC	**D**isplay **C**ombination **C**ode
DIN	**D**eutsche **I**ndustrie **N**orm
DIP	**D**ual **I**nline **P**ackage
DMA	**D**irect **M**emory **A**ccess
DPL	**D**escriptor **P**rivileg **L**evel
DSR	**D**ata **S**et **R**eady
DSR	**D**evice **S**ervice **R**outine
DTE	**D**ata **T**erminal **E**quipment
DTR	**D**ata **T**erminal **R**eady
DVI	**D**igital **V**ideo **I**nteractive
EA	**E**ffective **A**ddress
EBCDIC	**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange
ECC	**E**rror **C**hecking and **C**orrection
ECMA	**E**uropean **C**omputer **M**anufacturing **A**ssociation
EGA	**E**nhanced **G**raphics **A**dapter
EIA	**E**lectronic **I**ndustries **A**ssociation
EISA	**E**xtended **I**ndustry **S**tandard **A**rchitecture
EOI	**E**nd **O**f **I**nterrupt
EPL	**E**xtended **P**rivilege **L**evel
EPROM	**E**rasable **P**rogrammeble **R**ead-**O**nly **M**emory
ESDI	**E**nhanced **S**mall **D**evice **I**nterface
ETB	**E**nd of **T**ransmission **B**lock
FCB	**F**ormat **C**ontrol **B**lock — disk I/O data structure
FDDI	**F**ibre **D**istributed **D**ata **I**nterface
GDT	**G**lobal **D**escriptor **T**able
h	hexadecimal, hier meist H.
ICW	**I**nterrupt **C**ontrol **W**ord
IDE	**I**ntegrated **D**rive **E**lectronics (AT-Bus disk)
IEEE	**I**nstitute of **E**lectrical and **E**lectronic **E**ngineers.
IEEE 802.1	Netzwerk-Management und Bridge-Standards.

IEEE 802.2	Logical Link Control
IEEE 802.3	CSMA/CD (Ethernet)-Netzwerke.
IEEE 802.4	Token-Bus-Netzwerk.
IEEE 802.5	Token-Ring-Netzwerk.
IEEE 802.6	Metropolitan Area Networks (MAN).
IEEE 802.9	Integrated Voice Data (IVD) Netzwerke.
INT	**INT**errupt
I/O	**I**nput/**O**utput
IRQ	**I**nterrupt **R**e**Q**uest line
IRET	**RET**urn from an **I**nterrupt
ISA	**I**ndustry **S**tandard **A**rchitecture
ISR	**I**nterrupt **S**ervice **R**outine
ISR	**I**nterrupt **S**tatus **R**egister
K	Kilo, hier meist gleich 1024
KB	KiloByte oder Bit
Kbs	Kilobits per second
LAN	**L**ocal **A**rea **N**etwork.
LDT	**L**ocal **D**escriptor **T**able
LDTR	**L**ocal **D**escriptor **T**able **R**egister
LED	**L**ight-**E**mitting **D**iode
LID	**L**ogical **ID**
LLC	**L**ogical **L**ink **C**ontrol.
LSB	**L**east **S**ignificant **B**yte oder auch **B**it
LSI	**L**arge **S**cale **I**ntegration
M	Mega, hier entsprechend 1024 x 1024
MAC	**M**edia **A**ccess **C**ontrol.
MAN	**M**etropolitan **A**rea **N**etwork.
MB	MegaByte oder MegaBit
MDA	**M**onochrome **D**isplay **A**dapter
MFM	**M**odified **F**requency **M**odulation
MHz	Megaherz, hier ist Mega = 1 Mill.
MMU	**M**emory **M**anagement **U**nit
MSB	**M**ost **S**ignificant **B**yte oder **B**it

NCB	**N**etwork **C**ontrol **B**lock
NDIS	**N**etwork **D**river **I**nterface **S**pecification.
NMI	**N**on**M**askable **I**nterrupt
OCW	**O**peration **C**ontrol **W**ord des Interruptkontrollers
OS/2	**O**perating **S**ystem/**2** oder Operating $\frac{System}{2}$
PABX	**P**rivate **A**utomatic **B**ranch **E**xchange. Fernsprech-Nebenstellenanlage.
PC	**P**ersonal **C**omputer
PCLK	**P**eripheral **CL**oc**K**
PEL	**P**icture **EL**ement
PGA	**P**rofessional **G**raphics **A**dapter
PIC	**P**rogrammable **I**nterrupt **C**ontroller
PIO	**P**rogrammed **I**nput/**O**utput
PIT	**P**rogrammable **I**nterrupt **T**imer
POR	**P**ower-**O**n **R**eset
POST	**P**ower-**O**n **S**elf **T**est
RAM	**R**andom **A**ccess **M**emory
RAS	**R**ow **A**ddress **S**trobe
RGB	**R**ed-**G**reen-**B**lue
RI	**R**ing **I**ndicator
RLL	**R**un **L**ength **L**imited
ROM	**R**ead-**O**nly **M**emory
RTC	**R**eal **T**ime **C**lock
RTS	**R**equest **T**o **S**end
SCSI	**S**mall **C**omputer **S**ystems **I**nterface
SDLC	**S**ynchronous **D**ata **L**ink **C**ommunications
SIMM	**S**ingle **I**nline **M**emory **M**odule
SIP	**S**ingle **I**nline **P**ackage
SMD	**S**urface-**M**ount **D**evice
SSB	**S**ense **S**ummary **B**lock — a disk I/O structure
TCP/IP	**T**ransmission **C**ontrol **P**rogram/**I**nternet **P**rotocol.
TSS	**T**ask **S**tate **S**egment
TTL	**T**ransistor-**T**ransistor **L**ogic
UART	**U**niversal **A**synchronous **R**eceiver/**T**ransmitter
UTP	**U**nshielded **T**wisted **P**air.
VGA	**V**ideo **G**raphics **A**rray
VLSI	**V**ery **L**arge **S**cale **I**ntegration
WAN	**W**ide **A**rea **N**etwork.
XNS	**X**erox **N**etwork **S**ystem.
XT	e**X**tended **T**echnologie

Anhang B

Glossary

Access control Zugriffssteuerung. Mit der Zugriffssteuerung werden die konkurrierenden Netzwerkzugriffe (contention) aufgelöst. In einem effektiven Netzwerkentwurf stellt die Zugriffssteuerung für alle Benutzer ausreichende Übertragungsdiesnte sicher.

Adapter card

Adapter ROM

Address bus

ASCII American Standard Code for Information Interchange. Zuordnung bestimmter Ziffernfolgen zu entsprechenden Zeichen.

Arbitration

Asynchron Datenübertragung ohne zusätzliche Leitungen zur Synchronisation. Anfang und Ende eines Zeichens werden durch Start- und Stopbits gekennzeichnet.

AT-Bus 16-Bit breites Bus-System für PCs mit 80286- und 80386-Prozessoren

AUI **A**ttachment **U**nit **I**nterface für Ethernet.

Backbone Bei größeren vernetzten Systemen, die unterschiedliche oder gleiche Netzstrukturen heben, ist eine besondere Verwaltungsstruktur für den Informationsaustausch zwischen den Netzgruppen zu schaffen.

Bandbreite In der Kommunikationstechnik beschreibt die Bandbreite, welcher Frequenzbereich auf einem Verbindungsweg übertragen werden kann. Die Übertragungskapazität hängt direkt von der Bandbreite ab. Je größer die Bandbreite, desto größer ist das Informationvolumen, das in der Zeiteinheit über die Leitung geschickt werden kann.

Baseband Basisband. Übertragungsart, bei der Daten von verschiedenen Benutzern seriell in zeitlicher Reihenfolge (time sharing) über eine gemeinsame Datenleitung gegeben werden. Die Daten der verschiedenen Benutzer sind nach den Regeln des System ineinander verschachtelt (interlaced).

BIOS Basic Input Output System.

BIOS service

Bit Eine Abkürzung für "binary digit" (Binärziffer). Bit ist die kleinste Informationseinheit für Digitaldaten, wobei zwischen zwei Zuständen gewählt werden kann — zum Beispiel zwischen "0" und "1", "hoch" und "tief", "aus" und "an".

BPS

BCC

BOOT

Broadband Breitband. Eine Übertragungsart, in der Daten von verschiedenen Benutzern in verschiedenen Frequenzkanälen übertragen werden und somit eine Datenleitung gemeinsam genutzt werden kann.

Buffer

Burst mode

Bus Ein Bus besteht physikalisch aus mehreren parallelen Leitungen zur Übnertragung von Signalen und Versorgungsspannungen. Man unterscheidet zwischen uni- und bidirektionalem Bus.

BHE

Byte

Cache

CPU

Channel

CCB

CSB

CMOS

Configuration

Contention Streit. Ereignet sich, wenn zwei oder mehrere Benutzer sich um einen Zugriff auf ein einziges Übertragungsmedium streiten.

Conventional memory

CRC Cyclic redundancy Check. Methode zur Berechnung von Redundanzinformationen eines Datenblocks.

Cylinder

DAC

Dämpfung Leistungsabfall entlang einer Leitung für Maße wie Volt, Strom oder Signalstärke auf einem Übertragungsmedium, gemessen in Dezibel pro Kilometer.

Decode-Error Wenn ein Phasenfehler größer ist als 50 Prozent des gesamten Lesefensters, findet der Flußwechsel außerhalb der gültigen Decodierzelle statt.

Default

DMA Direct Memory Access, Direktspeicherzugriff.

DOS **D**isk **O**perating **S**ystem. Ungünstigste aller Lösungsmöglichkeiten.

DMA Controller

DMA Device

DIP **D**ual **I**nline **P**ackage. Gehäuseform mit zwei Pin-Reihen parallel zur Längsachse des Chips.

DRAM **D**ynamic **R**andom **A**ccess **M**emory. Da der Kondensator, der die Ladung der Information enthält, nicht Verlustlos ist, muß von Zeit zu Zeit die Ladung erneuert (refreshed) werden.

ECC

Einphaszeit Diese Zeit gibt an, wie lange es dauert, bis sich die PLL-Schaltung einer Änderung der Eingangsfrequenz anpaßt. Bei Floppy-Disk-Controllern ist diese Einphaszeit bei jedem Lese- oder Schreibvorgang zur Synchronisation von Lesedaten von Bedeutung.Während eines Synchronisiervorganges werden Sync-Bytes gelesen. Diese erzeugen konstante Lesedatenimpulse. Eine Adreßmarke folgt den Sync-Bytes. Aus der Definition für die Spurformate abgeleitet, muß die Einphaszeit kleiner als 384 μs (=6x64μs in FM oder 12x32μs in MFM) sein. Wird diese Einphaszeit überschritten, können die Daten auf der Diskette nicht mehr korrekt gelesen werden.

EPROM **E**lectrical **P**rogrammable **R**ead **O**nly **M**emory.

Ethernet Ein Datenübertragungsprotokoll mit Busstruktur und CSMA/CD, das die Plazierung und das Wiederauffinden von Daten spezifiziert und stellt heute den IEEE-Standard 802.3 dar. Ethernet wird von einigen hüheren Protokollen, z. B. TCP/IP, DECnet, als Transportbasis genutzt. Die Übertragungsgeschwindigkeit beträgt 10 Mbps.

EP Extra Pulse. Im Gegensatz zum Missing-Pulse (MP) ist der EP ein "magnetischer Stachel" auf der ansonsten homogen-aktiven magnetischen Oberfläche. Wie der MP ist der EP in der Regel nur einige bis einige -zig Amplituden (2F) groß. Wie MP-Errors wird ein EP durch Analog-Messung lokalisiert. Hier wird jedoch die zu prüfende Spur zunächst mit Gleichstrom gelöscht (DC-erased). Danach sollte die Spur bis auf das Eigenrauschen magnetisch still sein. Die Schwelle für das Feststellen eines EP bei einer Messung ist programmierbar. Sie wird angegeben in Prozent von der TAA. Ursache für EP können Fehler in der Beschichtung (Inhomogenität) oder im Untergrund (Substrat) sein. Gebräuchliche Werte für MP-Treshold sind je nach Hersteller zwischen 20 Prozent und 35 Prozent.

Expanded memory ein durch *Bank Switching* geschaffener Speicher.

EMS **E**xpanded **M**emory **S**ecification. Von Lotus, Intel und Microsoft entwickelt.

EEMS

Extended memory Memory mit Adressen oberhalb 1-MByte.

FDDI Fibre distributed data interface.Ein Hochgeschwindigkeitsnetzwerk in Ringstruktur mit gegenläufigen Glasfaserdoppelleitungen, für Übertragungsgeschwindigkeiten von 100 Mbps, das mit dem Zugriffsverfahren Token Passing arbeitet. In einem Netz können maximal 500 Knoten installiert werden — Knotenabstand max. 2 km.

Fixed disk nicht auswechselbarer Massenspeicher.

FTAM File transfer, access and management. Ein ISO/OSI-Standard der Schicht 7, der den Filetransfer zwischen Anwendungen unter Nutzung der durch die Schichten 1 bis 6 bereitgestellten Arbeitseinheiten und Protokollen ermöglicht.

Gateway Eine Hard- und Software-Einrichtung, um verschiedene Netzwerke miteinander zu verbinden, die nicht mit gleichen oder ähnlichen Protokollen arbeiten.

Halbduplex Übertragung nur über eine Leitung, die jedoch von beiden Teilnehmern in beiden Richtungen (zu verschiedenen Zeiten) verwendet wird.

Hardware ist alles, was man anfassen kann.

Head Kopf, der über einem magnetisierbaren Medium schwebt.

High Memory die oberen 384 KByte des 1-MByte Real-Mode Speicherbereichs. Wenn vorhanden, kann der Inhalt der ROM's dorthin kopiert werden. Zwischen die ROM-Bereiche können TSR-Programme oder Systemteile gelegt werden.

ID Inner-Diameter. Bestimmt den Punkt der innersten Spur eines Mediums. Er gilt als kritischer Bereich eines Mediums, da hoer Aufzeichnungsdichte und Bitshif-Effekt am höchsten sind. Außerdem ist die Flughöhe des Kopfes bei ID, bedingt durch die geringe Windgeschwindigkeit, am niedrigsten.

Interrupt Unterbrechung oder Unterbrechungsanfordrung.

ISO/OSI-Referenzmodell OSI-Schichtenmodell, 7-Schichten- oder Ebenen-Modell. Das von der ISO formulierte Modell legt in sieben Funktionsebenen fest, wie grundsätzliche Funktionen aller Kommunikationsvorgänge in der Datenverarbeitung und Telekommunikation vonstatten gehen sollen. Die Ebenen sind (von unten nach oben): physical, data link, network, transport, session, presentation, application.

Known state

Level 1 relay Andere Bezeichnung für Repeater.

Level 2 relay Andere Bezeichnung für Bridge.

Level 3 relay Andere Bezeichnung für Router.

Level 7 relay Andere Bezeichnung für Gateway.

Low Level Format

Main Memory

Margin-Test Der Margin-Test ist der aussagefähigste Test, um die Wahrscheinlichkeit für das Auftreten zukünftiger Fehler zu prognostizieren. Grundlage für das Aufzeichnen bei einer Festplatte ist die Zeicodierung der Flußwechsel auf der magnetischen Oberfläche. Jeder gültige Flußwechsel, der bei der Decodierung ein Bit generieren soll, muß innerhalb eines vorgegebenen Zeitfensters stattfinden. Die ideale Lage des Flußwechsels liegt genau in der Mitte dieses Fensters. Der Flußwechsel findet in der Praxis in den seltensten Fällen genau in dieser Idealposition statt. Durch kleinste Schwankungen von Drehzahl und Phasenlage der Spindel, durch magnetische Inhomogenitäten in der magnetischen Beschichtung, durch den Bitshift-Effekt und andere mechanische und elektrische Einflüsse auf Head/Disk-Assembly und den Lese-/Decodierkanal des Festplatten-Laufwerks, findet der Flußwechsel mehr oder weniger weit entfernt vor oder nach dem Idealzeitpunkt statt. Angenommen, das Lesefenster ist 100 ns breit. Der Ideal-Zeitpunkt liegt genau in der Mitte bei Zeitpunkt 0. Wenn der Lese-/Decodierkanal des Laufwerks in der Lage ist, alle Flußwechsel im Lesefenster zu decodieren, die mit ± 40 ns vom Idealzeitpunkt entfernt stattfinden, so existiert jeweils am vorderen und hinteren Ende des Fensters ein 10 ns breiter Rand (Early-/Late-Margin), der das aktive Fenster auf 80 ns verschmälert. Je weiter nun der tatsächliche Flußwechsel vom Idealzeitpunkt in Richtung linker oder rechter Margin stattfindet, desto größer ist die Gefahr, bei späteren Leseversuchen den Flußwechsel aus dem Fenster zu verlieren und damit einen Lesefehler in Kauf zu nehmen.

MAU **M**edium **A**ttachment **U**nit. Besteht nach IEEE802.3 aus zwei Komponenten: dem Transceiver (die Sende- und Empfangseinheit) und dem Anschluß and das Kabel. Zu den Aufgaben des Transceivers gehört die Überwachung des Kabels und das Registrieren von Kollisionen.

MP Missing Pulse. Ein MP ist ein Harderror auf der Plattenoberfläche. Er tritt auf, wenn auf der ansonsten homogen-aktiven magnetischen Beschichtung ein blinder (nichtmagnetischer) Fleck ist. Ein MP ist gewöhnlich nur einige bis einige -zig Amplituden (2F) groß. Festgestellt wird ein MP durch Analog-Messung. Der Maßstab die TAA. Die Schwelle (Treshold) für das Auftreten eines MP während der Messung ist programmierbar. Sie wird angegeben in Prozent von der TAA. Ursache für MP können Fehler in der Beschichtung (Inhomogenität) oder im Untergrund (Substrat) sein. Gemessen wird immer mit 2F. Gebräuchliche Werte für MP-Treshold liegen je nach Hersteller zwischen 50 Prozent und 90 Prozent.

NMOD Negativ-Modulation. Bezeichnet das Maß des Amplitudenabfalls beim Lesen einer Spur, die zuvor mit 2F beschrieben wurde. NMOD tritt auf, wenn die magnetische Induktion im Lesekopf durch Inhomogenität der Beschichtung, durch Verwellung oder Verspannung der Scheibenoberfläche absinkt. Maßstab für NMOD ist die TAA.

OD Outer-Diameter. Bestimmt den Punkt der Äußersten Spur eines Mediums.

Page Mode Eine page ist eine Zeile innerhalb der matrixförmigen Anordnung eines Speichers. Der Pagemode gestattet das serielle Lesen/Schreiben einer Zeile. Die "Row Addrress" bleibt dabei konstant und nur die "Column Address" wird geändert.

Peakshift Bitshift-Effekt. Zei magnetische Domänen, die auf der magnetischen Beschichtung direkt nebeneinander liegen, üben eine gegenseitige Kraft auf sich aus und verursachen ein Auseinanderdriften der einzelnen Domänen. Dieser Trend ist um so stärker, je näher die Domänen beieinanderliegen und je größer die relative Feldstärke der einzelnen Domänen ist. Der Bitshift-Effekt tritt am äußerten Rand (OD) am schwächsten, in der innersten Spur (ID) am stärksten auf. Im ID-Bereich ist die Aufzeichnungsdichte am höchsten und damit der mittlere Abstand zwischen den Domänen am geringsten. Zum anderen wird durch die geringe Flughöhe des Kopfes, und dadurch bedingt durch den höheren magnetischen Fluß beim Beschreiben, die relative Feldstärke der eonzelnen Domänen am größten. Da alle Informationen auf der Plattenoberfläche zeitcodiert sind, sorgt das Verschieben der Domänen für Ungenauigkeiten während der Decodierung. Bei einem Festplattenlaufwerk wird durch konstruktive Maßnahmen dem Bitshift-Effekt entgegengewirkt (Write-Precompensation, Reduced-Write-Current).

Phase-Error Phasenfehler. Die Zeitdifferenz zwischen dem Zeitpunkt des tatsächlich detektierten Datenimpuls (Flußwechsel) und dem theoretischschen Idealzeitpunkt. Der Idealzeitpunkt liegt gewöhnlich exakt in der Mitte des in der Decodierschaltung generierten Lesefensters. Phasenfehler bestehen aus zwei Komponenten: einem Langzeitphasenfehler, der sich als Mittelwert einer großen Anzahl von Datenimpulsen bildet, und dem kurzzeitigen Phasenfehler, der bei einzelnen Datenimpulsen auftreten kann und in bezug auf den Mittelwert gemessen wird. Langzeit-Phasenfehler werden in der PLL des Lesedecoders ausgeregelt.

Pit Vertiefung in der Oberfläche einer CD-ROM.

PMOD Positiv-Modulation. Bezeichnet das Maß des Amplitudenanstiegs beim Lesen einer Spur, die zuvor mit 2F beschrieben wurde. PMOD tritt auf, wenn die magnetische Induktion im Lesekopf durch Inhomogenität der Beschichtung, durch Verwellung oder Verspannung der Scheibenoberfläche ansteigt. Maßstab für PMOD ist die TAA.

Protokoll Ein prototkoll definiert Aufbau-, Überwachungs- und Abbaumodalitäten einer Datenverbindung, es enthält Regelungen über Datenformate, zeitliche Abläufe und die Art der Fehlerbehandlung beim Datenaustausch zwischen Komputersystemen.

Read-Channel Der Begriff Lesekanal beinhaltet Medium, Köpfe, alle Schaltungen von den Köpfen bis einschließlich Decoder, Pulse-Detection-Schaltung und PLL-Schaltung. Bei ST-5-6-Laufwerken ist ein Teil des Lesekanals auf dem Controller, während bei intelligenteren Laufwerken wie SCSI oder ESDI auch Decoder und PLL auf dem Festplattenlaufwerk integriert sind.

Recording-Density Aufzeichnungsdichte. Die Aufzeichnungsdichte ist ein entscheidendes Kriterium bei der Leistungsbeschreibung eines Festplattensystems. Höhere Aufzeichnungsdichten ermöglichen mehr Speicherkapazität und höhere Datentransferrate bei gleicher Nutzfläche. Durchschnittliche Werte bei heutigen (1987/88) Plattensystemen liegen, abhängig von der Geräteklasse, etwa zwischen 12 und 24 $MBit/Zoll^2$. Mit der Aufzeichnungsdichte steigt auch der Bitshift-Effekt.

RES Resolution. Ist ein Index für die Fähigkeit des Mediums, hohe Frequenzen aufzuzeichnen. Wenn eine Spur mit einer bestimmten Frequenz (1F) beschrieben und gelesen wird, reduziert sich bei doppelter Schreibfrequenz (2F) die TAA um einen meßbaren Wert. Das Verhältnis 2F/1F bestimmt die RES. Gängige Werte sind 80 bis 95 Prozent. Die RES wird im kritischen Bereich (ID) gemessen.

TAA bei 1F = 95 mV

TAA bei 2F = 84 mV

RES = $\frac{2F}{1F} = \frac{84mV}{95mV} = 83\ \%$

RJ45 ist eine 8-polige Steckverbindung für nichtabgeschirmte Zweidrahtleitungen (UTP = unshielded twisted pair). Pinbelegung:

Pin 1 Transmit Data +

Pin 2 Transmit Data -

Pin 3 Receive Data +

Pin 6 Receive Data -

Pin 4, 5, 7, 8 nicht belegt.

SIP Single-in-Line-Package. Gehäuseform mit einer Pin-Reihe an einer Kante.

SIPP Single-in-Line-Pin-Package. Modul, meist mit Speichern versehen, mit einer Reihe Pin's.

SIMM Single-in-Line-Memory-Module. Modul mit direkt Steckverbindung.

S/N Signal-/Noise-Ratio. Bezeichnet den Störspannungsabstand und gibt das Verhältnis zwischen maximalem Nutzsignal und Eigenrauschen an.

TAA Track-Average-Amplitude. Durchschnittliche Spur-Amplitude. Die TAA ist für fast alle Messungen an Festplattensystemen Basis für die verschiedenen Meßaussagen. Die Schwelle (Treshold) für verschiedene Fehlerbedingungen wird in der Regel in Prozent von TAA angegeben.

ZIP Zig-Zag-in-Line-Package. Gehäuseform mit Anschlüssen an der Kante, die versetzt nach den Seiten gebogen dann zwei Reihen ergeben.

1F Bezeichnet die niedrigst mögliche Aufzeichnungsfrequenz für ein Festplattenlaufwerk.

2F Bezeichnet die höchstmögliche Aufzeichnungsfrequenz für ein Festplattenlaufwerk.

Anhang C

Chips

C.1 Chipsets und Chips

HT15 Headlands chip für 386SX.

CL-Px0070 Video window generator (Cirrus).

Am286ZX kompletter AT von AMD!

386WB Write Back Chipsatz von OPTi mit 20 - 50 MHz. Der Write-back-Cache-Controller ist ausgelegt für 64 bis 512 KByte. Damit werden 22,3 MIPS bei 50 MHz erreicht.

38600SX/DX C & T (10% better performance).

38605SX/DX C & T (40% more speed).

ELEATsx Chipset von C & T für Desktop Computer.

CHIPSClock Chipset von C & T fr Desktop und Notebook Computer.

CHIPSlite Chipset von C & T für Notebook Computer.

PEAKsx Chipset von C & T für Desktop Computer.

PEAK/DM Chipset von C & T für 80386-Rechner mit Cache.

PAEK386 Chipset von C & T für 80386-Rechner mit Zweiweg-assoziativ-Cache.

SCAT Single-Chip-AT-System-Controller 82C235.

VL82C486 Single chip für 486.

SCATsx Single-Chip-AT-System-Controller 82C836.

TACT84500 EISA Chipset

HT216 VGA chip von Headland

μPD765 Floppy-Controller Standard!

M1101 High Speed XT Peripheral Chip.

M1201 System Controller für 10MHz 8086.

M1203 Address und Data Bus Buffer.

DS1287 Dallas Semiconductor equivalent to Motorola's MC146818A Real Time Clock.

M2105 Mouse Controller.

M2107 I/O & Subsystem Controller.

M2101 8 Bit μP Mouse Controller.

M2201A Bi-direktional Printer Adapter Interface.

M2201 Printer Adapter Interface (PAI).

M2203/5 Laser Printer Chip Set.

M2301 XT-IOB Chip (2 UART, 1 RTC).

M2303 AT-IOB Chip (2 16450).

M3107 MDA+CGA+Hercules Chip.

M3113/5 MCGA & Hercules Graphics Controller set.

M3205 Color Palette.

M5103 Floppy Disk Interface & Data Separator.

NCR5380 SCSI-Controller asynchron (ASI).

NCR53C400 SCSI-Controller (CPU-unterstützt).

NCR53C94 SCSI-Controller (DMA-slave).

NCR53C700 SCSI-Controller (DMA-Master mit 86C70 ISA-controller).

NCR53C720 SCSI-Controller mit cache-line-burst-mode (4 mal 16 Byte burst ergibt 105 MByte/s).

CF58209 Monochrome Graphic Chip.

SCX6212 Monochrome Graphic Chip.

MC68HC99 SCSI-Controller mit ESDI-Mode (24 MHz).

IDT71589 Cache-RAM für 50MHz i486

IDT7252 BiFIFO

μPD72061 Hard Disk Controller (HDC).

μPD72120 Advanced Graphics Display Controller (AGDC).

μPD7262 ESDI-Controller, Revision F vom Februar 1986. (18 Mbit/s)

μPD72064 Floppy-Disk-Controller (765 kompatibel).

μPD72067 Advanced-Floppy-Disk-Controller.

μPD72111 SCSI-Controller, 1,5MB/s asynchron, 4MB/s synchron.

VM7301 Dataseparator, synthesizer, encoder-decoder (24MBit/s)

VM7401 Pulse detector, servo peak detector (24MBit/s)

Am7990 LANCE Ethernetcontroller

Am79C900 ILACC Ethernetcontroller (Mode 0 = LANCE).

82F010 CMOS Flash Memory

8237 DMA Controller

8254 Timer

8259 Interrupt Controller

8742 Keyboard Controller PS/2-compatible.

82077AA Floppy-Disk-Controller

82C100 Peripheral controller

VL82C106 Combo (2 x COM, LPT, Keybd, IDE).

VL82C107 SCAMP combi chip

82C206 Peripheral controller

82C235 Single-Chip-AT-System-Controller SCAT.

VL82C310 AT midrange SCAMP-LT

VL82C330 Systemcontroller

VL82C331 ISA-Bus-Controller

VL82C332 Daten-Puffer

82350

82351 EISA Local I/O

82352 EISA Bus Buffer

82353 EISA Bus Master 16 Bit

82355 Bus Master Interface Controller

82357 Integrated System Peripheral

82358 EISA Bus Controller (EBC)

82359 EISA DRAM-Controller bis 256 MB

82360SL Peripherie für 80386SL

82385 Cache controller

82C401 Clock für 82C451 VGA-, 82C455 und 82C456 VGA-Flatpanel-Controller.

82C402 Clock für 82C450 und 82C452 VGA-Controller.

82C403 Clock für 82C480 8514/AT-kompatiblen Grafik-Controller.

82C426 CGA-Flatpanel-Controller.

82C450 VGA-Controller.

82C451 VGA-Controller.

82C452 VGA-Controller.

82C455 VGA-Controller.

82C456 VGA-Controller für Flatpanel mit 64 Graustufen.

82C480 Grafik-Controller 8514 kompatibel.

82B484 Support Chip für 8514/A.

82490DX Second-Level-Cache für 82486 50MHz

82495DX Second-Level-Cache-Controller für 82486 50MHz

82C5086 SCSI Schnittstellen Controller mit 64KB FIFO und 5,3 Mbyte/s.

82C5087 SCSI-Protokoll-Controller mit 4/5,3 MB/s und Funktionen Command Queuing, Q-Tag-Messages und Wide-SCSI.

SAB82532 Enhanced Serial Communications Controller 2 Channel (ESCC2).

82C601 E/A Controller

82C641 Power-Control-Unit.

82706 VGA Controller

82C710 Multi-Function-Controller.

82C711 Single-Chip-Universal-I/O-Controller.

82750PB Pixel-Prozessor für DVI.

82750DB Display-Prozessor für DVI.

82C765 Floppy-Controller (1 MBits/s!). C & T Megazelle.

82786 Display-Prozessor (2,5 MPixel/s).

82C835 Zweiweg-assoziativ-Cache-Controller.

82C836 Single-Chip-AT-System-Controller SCATsx.

82C862 Schnittstellen-Controller für Floppy, Winchester, Maus, Gameport, zwei serielle und eine parallele.

82961KA Printer controll chip for i960 systems.

DP8490 SCSI-Controller (4 MBytes/s) mit Parity und Extra-Mode-Bit (EASI).

TC85M911F Bus-Master und DMA-Controller für 80486.

TC85M921F Address- und Daten-Buffer für 80486.

TC85M931F Memory- und Bus-Controller für 80486.

TC85M951F Peripheral Support für 80486.

8680 PC on a chip.

MB86961 Ethernet controller

ADG9000 Hard Disk Interface (HDI).

SL9011 System-Controller 386DX

SL9020 Daten-Controller 386DX

SL9025 Adreß-Controller 386DX

SL9X5X Speicher-Controller 386DX

SL9X52 System and memory controller

SL9X56 System memory and address controller (plus cache interface).

92C28 Ethernet-Controller

Am95C71 Video-Data-Compression/Expansion-Prozessor (VCEP).

Am95C75 Raster-Printer-Controller (RPC) 400 dpi; 40 Seiten/m.

AM9590 Für vier Festplatten und Disketten. Interfaces: ESDI, ST506, ST412 und double-density-Floppy.

16V8 20 pin mask PAL.

20V8 24 pin mask PAL.

C.2 FDDI

Fiber **D**istributed **D**ata **I**nterface. Mit der Datenrate von 100 MBit/s, kurzen Abständen zwischen den Paketen und einer kleinen Paket-Mindestgröße ist das FDDI ein ideales Netzwerk für deterministische Echtzeit-Anwendungen. Der Paket-Abstand von 8 Byte und die Mindestlänge der Rahmen von 9 Byte sind für Sprache und Prozeßsteuerung gut geeignet. Das andere Extrem, die maximale Rahmenlänge von 4500 Byte befriedigt Aplikationen wie Grafik, die den Transfer großer Datenblöcke bewältigen müssen.

C.2.1 Physikalische Schicht

Die physikalische Schicht spezifiziert das elektrische und optische Interface zum Ring. Das elektrische Interface umfaßt Ausfiltern von Verletzungen des Protokolls, Codieren, Taktrückgewinnung, elastisches Puffern empfangener Daten, Decodieren, Ausgleichen sowie Vorkehrungen für die physikalische Rekonfiguration. Die optischen Spezifikationen enthalten den optischen Treiber, den optischen Empfänger, optische Bypass-Schalter, Steckverbinder und die optische Faser.

C.2.2 Gleichstromkompensierte RLL 4B/5B-Codierung

Zuerst wird der vom Senderteil des FORMAC (Fiber Optic Ring Media Access Controller) kommende Datenstrom einer 4-Bit/5-Bit-Wandlung unterworfen. 4B/5B-Code hat einen Wirkungsgrad von 80 Prozent, denn zu jeweils 4 Bit eines Datensymbols kommt ein weiteres Bit. Der FDDI-Code ist ein RLL-3-Code. Das bedeutet, es gibt höchstens drei aufeinanderfolgende Nullen. Die Ausnahmen sind der Connection Management Line State Symbolstrom, wie HALT, das vier aufeinanderfolgende Nullen nach jeder Eins hat, und der Master Line State Symbolstrom, der zwischen Einsen neun Nullen aufweist.
Die zweite Codewandlung ist die Transformation von NRZ in NRZI. Sie wird mit den seriellen NRZ-Daten ausgeführt, die aus dem der 4B/5B-Wandlung folgenden Parallel-/Serien-Umsetzer kommen. NRZI-Daten sind durch einen Pegelwechsel für jede Eins im NRZ-Datenstrom und das Ausbleiben eines Pegelwechsels für jede Null im Datenstrom gekennzeichnet. Dieses Schema übermittelt zusammen mit den Daten auch die Information über den Codiertakt. RLL und NRZ-in-NRZI-Transformation stellen sicher, daß nicht mehr als drei Codiertakte ohne Pegelwechsel vorkommen. Das erleichtert der Phasenschleife des Empfängers im EDS, den Takt aus den eintreffenden Daten zu regenerieren und die Synchronisierung während der Dauer des Datenstroms zu halten.
Ein 4B/5B-codierter NRZI-Datenstrom ist auch gleichstromkompensiert. Der mittlere Gleichstrom am Medium beträgt nicht mehr als ± 10 Prozent. Der optische Empfänger besitzt einen Equalizer, um die Dispersion auf der optischen Faser auszugleichen, und die Filterwirkung des Equalizers führt zu einer Verzerrung, wenn im seriellen NRZI-Datenstrom eine Gleichstrom-Komponente vorhanden ist. Die Gleichstromfreiheit unterstützt auch die automatische Verstärkungsregelung des optischen Empfängers.

C.2.3 Interface zum Knoten-Rechner

Der ENDEC besitzt ein Interface zum Knoten-Prozessor für die Initialisierung, die Ausgabe von Line State Symbolen auf das Medium und die Übergabe des aus dem empfangenen Datenstrom decodierten Line State Status zum Management der Verbindungen sowie für die Diagnose des Knoten und die Rekonfiguration des Rings bei Fehlern. Für Diagnosen werden interne Schleifen am Busanschluß des FORMAC über den Multiplexer MUX2 und über den seriellen NRZI-Ausgang unterstützt. Zu den Vorkehrungen für die Rekonfiguration gehören der interne MUX1, der zwischen zwei Send-Eingangskanälen wählt und eine Regenerator-Betriebsart, bei der die empfangenen Daten auf den eigenen Send-Ausgang zurückgeführt werden.

C.2.4 Takt-Rückgewinnung

Der EDS (ERX) muß eine besonders leistungsfähige PLL haben. Sie muß aus dem empfangenen Datenstrom den Takt mit hoher Zuverlässigkeit regenerieren, selbst bei einem mit ±3,2 ns Jitter behafteten Signal mit 125 MBit/s, bei dem die Periode eines Bits nur 8 ns beträgt. Die PLL ist intern entweder auf die Halt-Symbole (00100) oder die Master Line State Symbolpaare (00100 und 00000) gelockt, je nachdem, ob der Knoten eine Station mit einem Einfach- oder mit einem Dopplel-Anschluß ist. Die PLL wahrt die Synchronisierung, sobald die Paket- oder Idle(11111)-Symbolströme kontinuierlich empfangen werden. Die PLL benutzt nur ein einziges externes Bauelement. Selbst wenn am Eingang des EDS kein Signal anliegt, weicht die PLL-Frequenz niemals um mehr als $100 * 10^{-6}$ von der Mittenfrequenz ab.

C.2.5 Die Byte-Erkennung

Bevor eine Decodierung möglich ist, sind im empfangenen Signal die Bytegrenzen zu bestimmen. Beim Reset des ENDEC können die Initialisierungs-Symbolströme wie HALT und Master erkannt werden, sobald die PLL einrastet, ohne Rücksicht bei welchem Bit eines Bytes die Bytegrenze festgelegt wird. Beim Empfang von Rahmen vom FORMAC bestimmt der ENDEC die Grenzen anhand des ersten Bytes in diesem Rahmen. Das erste Byte ist der Start-Delimiter. Es ist durchaus möglich, daß wegen der veränderlichen Zahl von Präambel-Bits zwischen den Rahmen die Bytegrenzen für aufeinanderfolgende Rahmen unterschiedlich sein können. Die Präambel-Bits werden ausgegeben, um die Synchronisierung der PLL zwischen den Rahmen zu wahren und sie dienen auch zum Ausgleich von Differenzen zwischen dem empfangenen Takt und dem localen Takt.

C.2.6 FDDI-Chips

GA1110E Tacktverteiler

GA1210E Tacktverteiler mit Tacktverdopplung.

16G041-H Tacktrückgewinnung mit PLL zwischen 100 und 625 MHz.

16G052 Sender für SONET (Synchronous Optical Network)(1 244,15 MBit/s).

16G053 Empänger für SONET (Synchronous Optical Network)(1 244,15 MBit/s).

16G054 Sender für ESCON

16G055 Empänger für ESCON

GA22V10 PLD mit Signaldurchlauf 7.5 ns (111 MHz).

GA23SV8 PLD (166 MHz).

Am79C81 RAM Buffer Controller (RBC)

Am79C82 Data Path Controller (DPC)

Am79C83 Fiber Optic Ring Media Access Controller

Am79C830 *Formac* fiber optic ring media access controller.

Am7984 Endec (ENcoder/DECoder) Transmitter (ETX)

Am7985 Endec Receiver (ERX), Datan-Separator (EDS).

Am79C864 *Amphy*

Am79865 *PDT*transmitter

Am79866 *PDR*receiver

FCG-68836 FDDI-ClockGenerator

ELM-68837 Elasticity-Buffer- und Link-Manager-Baustein.

MAC-68838 Media-Access-Controller

FSI-68839 FDDI-System-Interface

æ