Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

8 I/O Systeme

8.1 Allgemein

8.1.1 I/O Geräte

Unterscheidung in
- blockorientierte und
- zeichenorientierte Geräte

Ein *blockorientiertes Gerät* speichert oder verarbeitet die Daten in Blöcken, z.B. 512 Bytes oder 32178 Bytes.
Beispiele: Festplatten, Bandgeräte, Zeilendrucker (eine Zeile = 132 Zeichen)

Ein *zeichenorientiertes Gerät* verarbeitet im wesentlichen einzelne Zeichen oder Gruppen unterschiedlicher Größe, z.B. Tastatur, Maus, serielle Schnittstelle.

Die Grenzen zwischen beiden Arten sind jedoch fliessend, dennoch sind die meisten Treiber der einen oder anderen Art zugeordnet.

Einige typische Geräte und ihre Datenraten sind in der folgenden Tabelle aufgeführt.

Tabelle 5: Übersicht I/O Geräte

Gerät	*Datenrate*	*Gerät*	*Datenrate*
Tastatur	10 Bytes/s	40x CDROM	6 MB/s
Maus	100 B/s	Fast Ethernet	12,5 MB/s
56K Modem	7 kB/s	ISA Bus	16,7 MB/s
ISDN mit Kanalbündelung	16 kB/s	EIDE (ATA-2) Festplatte	16.7 MB/s
Laserdrucker	100 kB/s	FireWire (IEEE 1394)	50 MB/s
Scanner	400 kB/s	XGA Monitor	60 MB/s
Ethernet	1,25 MB/s	SCSI Ultra 2 Festplatte	80 MB/s
USB	1.5 MB/s	Ultrium Tape	320 MB/s
Digitaler Camcorder	4 MB/s	PCI Bus	528 MB/s
IDE Festplatte	5 MB/s	Sun Giagplane XP Backbone	20 GB/s

8.1.2 I/O Gerätecontroller

Aufteilung des I/O Gerätes in einen mechanischen (das Gerät selbst) und einen elektronischen Teil (den Gerätecontroller). Dies ist in der folgenden Graphik dargestellt.

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

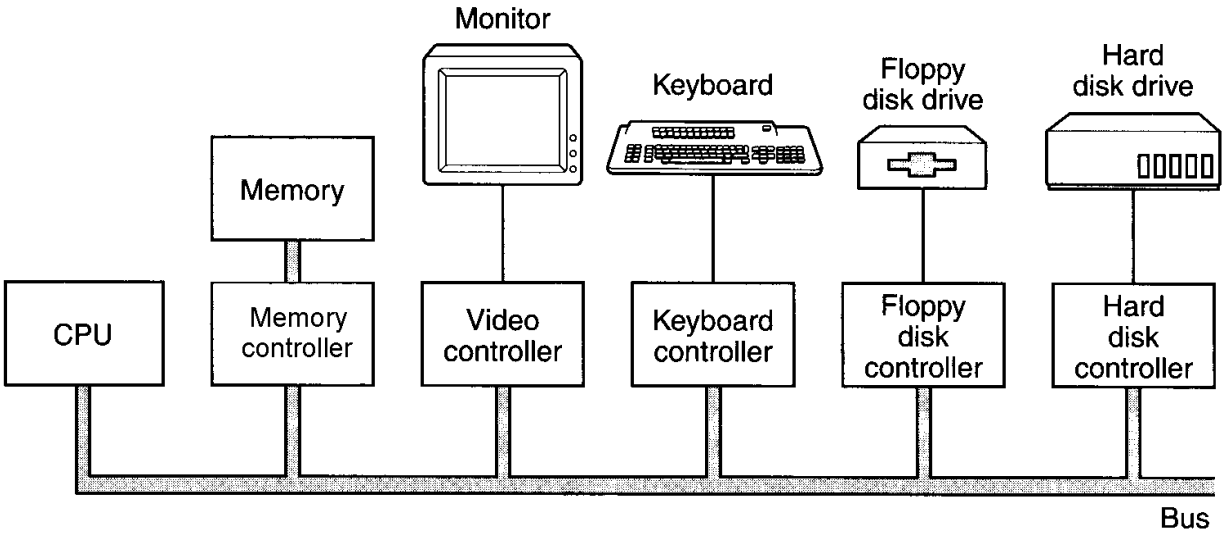


Abbildung 87: Übersicht Rechneraufbau

Beispiele:
- Lesen von Festplatte: Der Gerätecontroller liest die Bit-Folgen von der Festplatte ein, überprüft die Checksumme (ECC) und gibt einen Rückgabestatus und den eingelesenen Block zurück
- Serielle Schnittstelle: Ein Zeichen wird in den Controller geschrieben, der schickt dieses dann Bit-für-Bit (mit Start-, Stop- und Parity-Bit) auf die Leitung

8.1.3 Adressierungsarten von I/O Geräten

Generell gibt es zwei unterschiedliche Adressierungsarten für I/O Zugriffe:

1. *Getrennte CPU Befehle für Speicher- und I/O-Gerätezugriff*:
   - Eingabe
     ```
     IN REG, PORT
     ```
   - Ausgabe
     ```
     OUT REG, PORT
     ```

   Dann sind die folgenden Befehle unterschiedlich:
   ```
   IN R0, 4    ; lese I/O Port 4 in Register R0 ein
   MOV R0, 4   ; lese Speicherzelle 4 in Register R0 ein
   ```

2. *Speicher-gemapptes I/O (Memory-Mapped I/O)*: Die Geräteadressen werden in den Adressraum des Speichers eingeblendet.
   - Der Adressraum wird aufgeteilt in Speicher und I/O Adressen
   - Die MMU muss Adresszugriffe entsprechend erkennen und umsetzen (gehört diese Adresse zum Speicher oder zum I/O Adressbereich?)

Beispiel Embedded Bereich (siehe auch Abschnitt 7.1.5): Gegeben sei ein Prozessor, der im Bereich 0xFFF00- 0xFFFFF Spezialregister zur Ansteuerung von I/O Geräten hat. Für den Zugriff auf I/O und Speicher wird derselbe Befehl verwendet:

```
MOV R0, 0xFFFD1   ; lese serielles Empfangsregister nach R0 ein
MOV R0, 0xD0004   ; lese Speicherzelle 0xD0004 in Register R0 ein
```

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

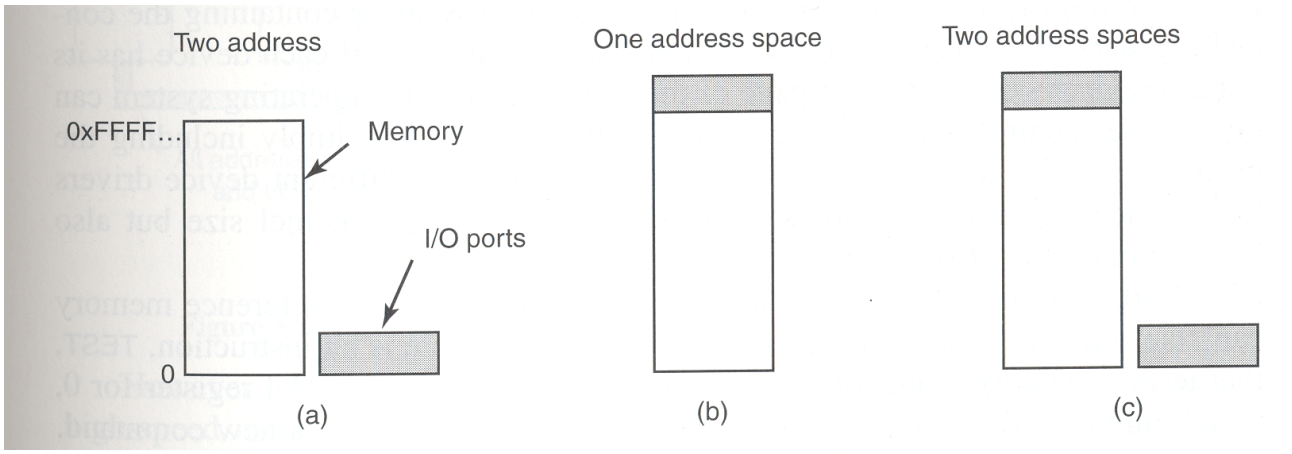


Abbildung 88: Adressierungsarten für I/O Geräte (nach [1]): a) getrennte Befehle; b) Memory-Mapped I/O; c) gemischt;

Für effektiven Speicherzugriff werden heutzutage spezielle hoch-performante Speicherbusse verwendet, die nicht für I/O geeignet sind. Deshalb werden z.B. beim Pentium 2 Busse eingesetzt, ein schneller für Speicherzugriffe und ein etwas langsamerer zur Anbindung vom PCI-Bus (siehe folgende Abbildung und Abbildung 7: Pentium Busstruktur (aus [1])).

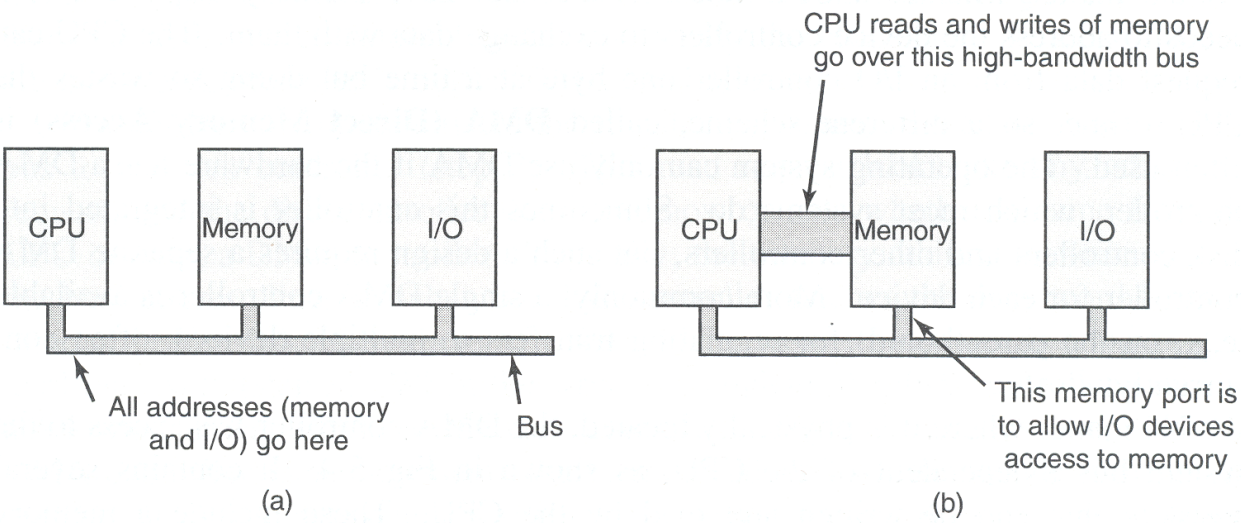


Abbildung 89: Busarchitektur für Memory-Mapped-I/O

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

I/O address range (hexadecimal)	device
000-00F	DMA controller
020-021	interrupt controller
040-043	timer
200-20F	game controller
2F8-2FF	serial port (secondary)
320-32F	hard-disk controller
378-37F	parallel port
3D0-3DF	graphics controller
3F0-3F7	diskette-drive controller
3F8-3FF	serial port (primary)

Abbildung 90: I/O Adressen eines PCs (nicht vollständig)

8.1.4 Polling

Prinzip: der I/O Controller hat ein Status-Bit, mit dem die Beendigung der Aktion angezeigt wird.
Ablauf:
1. Gerät ist bereit.
2. CPU schreibt ein Kommando ins Kommando-Byte (und ggfs. ein Datenbyte ins Datenregister) und setzt das Command-Ready-Bit.
3. Das I/O-Gerät liest das Kommando und das Datenbyte, setzt das Busy-Bit auf 1 und beginnt die Abarbeitung der Aufgabe.
4. Derweil pollt die CPU regelmässig das Status-Bit
5. Wenn der Controller fertig ist, werden das Command-Ready und das Busy-Bit zurückgesetzt; ggfs. wird ein Fehlerbit gesetzt.
6. Die CPU bemerkt die Beendigung und holt ggfs. das Ergebnis ab.

Beispiel: Das Schreiben einer Speicherzelle in einem EEPROM dauert typischerweise einige Millisekunden. Die CPU wartet solange, bis das Busy-Bit zurück gesetzt wird, bevor sie den nächsten Schreibvorgang startet.

8.1.5 Interrupts

- Über eine Interrupt-Request-Line kann das Gerät einen Interrupt auf der CPU auslösen
- Nummer des Interrupts am Adress-Bus -> mittels Vektor-Interrupt-Tabelle wird die entsprechende Interrupt-Routine aufgerufen
- Kann **maskable** (abschaltbar) oder **unmaskable** (nicht abschaltbar) sein

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

- Verschiedene Interrupts können unterschiedliche Prioritäten haben, d.h. ein niederpriorer Interrupt wird von einem höher-prioren Interrupt unterbrochen

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Abbildung 91: Vektor-Interrupttabelle des PCs

8.1.6 Direkter Speicherzugriff (Direct Memory Access, DMA)

Um die CPU beim Datentransfer von und zu den I/O Geräten zu entlasten, wird ein DMA-Controller eingesetzt, der Zugriff auf den Speicher und die I/O Geräte hat. Er kann unabhängig von der CPU einen Datentransfer durchführen und die Daten in den Speicher schreiben bzw. von dort lesen.

Der allgemeine Ablauf ist wie folgt (lesender Zugriff):

1. Die CPU startet einen DMA-Transfer, indem sie den Controller entsprechend programmiert. Wichtige Daten sind Anfangsadresse im Speicher, Adresse der Daten im I/O Gerät, Anzahl der Bytes, sowie weitere Kontrollbefehle
2. Die DMA fordert ein oder mehrere Bytes vom I/O Gerät an
3. Das I/O Gerät legt die geforderten Bytes auf den PCI Bus
4. Die DMA liest die Bytes vom Gerät und schreibt sie an die gewünschte Stelle im Speicher.Gegebenfalls werden die Schritte 2. bis 4. wiederholt, bis alle Daten in den Speicher transferiert wurden.
5. Ist der Transfer beendet, so löst die DMA einen Interrupt in der CPU aus und benachrichtigt diese, dass die Daten nun im Speicher vorliegen.

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

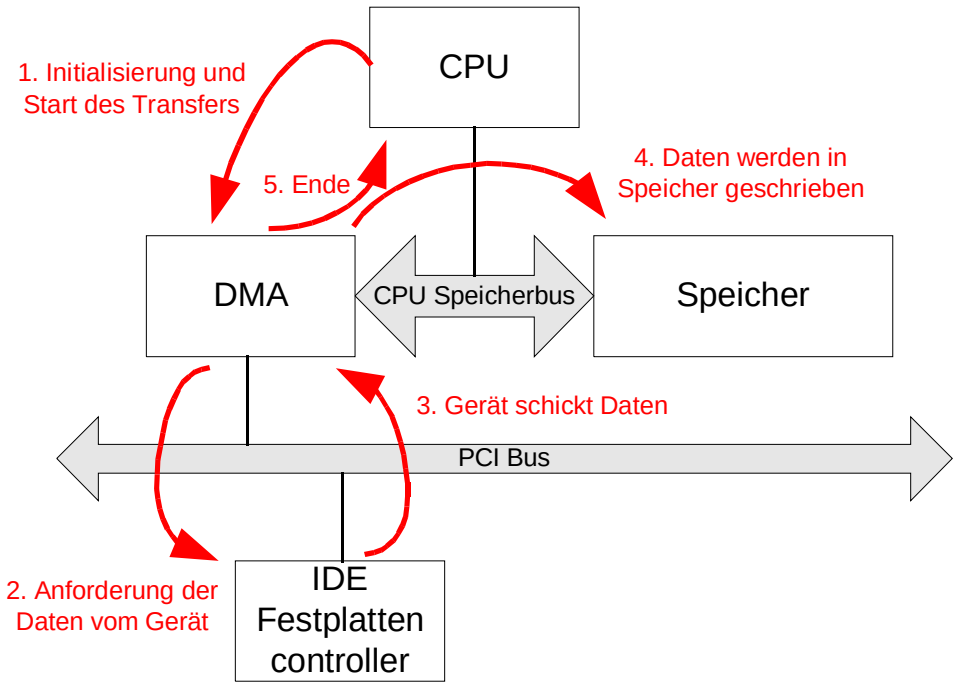


Abbildung 92: Schema DMA

Wichtig: Das virtuelle Speichermanagement muss auf die DMA abgestimmt sein. Es muss vermieden werden, dass eine Seite aus dem Hauptspeicher entfernt wird, während die DMA darauf zugreift. Seiten mit DMA-Zugriff müssen deshalb im Speichermanagement gesperrt werden.

8.2 I/O Software Schichten

Generell ist es bei der Entwicklung eines Betriebssystems sinnvoll, eine möglichst einheitliche Treiberarchitektur zu schaffen. Aufgrund der schnellen Hardware-Entwicklungszeiten müssen immer neue Treiber ins Betriebssystem eingebunden werden. Standardisierte Schnittstellen bieten die Möglichkeit, leicht neue Treiber hinzuzufügen, ohne andere Teile des OS ändern zu müssen.
Ein Betriebssystem besteht typischerweise aus 4 Software-Schichten:

User-Level I/O Software
Geräte-unabhängige Betriebssystem-Software
Gerätetreiber
Interrupt-Handler
Hardware

8.2.1 Interrupt Handler

Unterste Ebene der I/O-Verarbeitung:
- Interrupt-Routine, wird direkt als Interrupt ausgeführt

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

- *Input:* Liest Geräte-Daten / Stati ein und schreibt diese in einen Puffer des Gerätetreibers
- *Output:* Entnimmt neue Daten / Kommandos aus den Puffern des Gerätetreibers und schickt sie an das Gerät
- Interrupts sind schwierig zu programmieren:
    - sehr Hardware-nah, genaue Kenntnisse der CPU und der Hardware sind nötig
    - Synchronisierung mit dem Rest des Betriebssystems "per Hand" nötig, typischerweise nur wenige Betriebssystemmittel zur Verfügung
    - Interrupts dürfen keine Seitenfehler erzeugen -> sonst Blockade des Betriebssystems möglich!

8.2.2 Gerätetreiber

Geräte-spezifischer Code zur Kontrolle des Gerätes, wird typischerweise vom Hersteller der Hardware mitgeliefert

- Für jeden Typ von Hardware ein eigener Treiber, z.B. jeweils einer für Maus, Joystick, Tastatur, Modem, ISDN-Karte
- Manchmal ein Treiber für mehrere ähnliche Geräte, z.B. SCSI-Kontroller für mehrere, unterschiedliche Festplatten
- Treiber muss im Kernel laufen, da Hardware-Zugriffe nötig sind (-> Sicherheitsproblem! Ein schlechter Treiber kann die Stabilität des OS gefährden!)
- Typischerweise ist ein Treiber entweder block-orientiert oder zeichen-orientiert.
- Treiber können fest in den Kernel kompiliert sein (frühere Unix-Varianten) oder dynamisch nachgeladen werden. Letzteres kann entweder zum System-Startup geschehen (dann ist nach Hardware-Updates ein Reboot nötig), oder zur Laufzeit, z.B. USB-Treiber für Kameras.

Typische Aufgaben eines Gerätetreibers (Interface):

- Treiber-Initialisierung und Ende-Routinen
- I/O Systemaufrufe: open/close
- Geräte-Kontrollfunktionen, read/write, etc.
- Datentransfer-Routinen
- Timer (Zeitdienste)
- und weitere

Gerätetreiber und DRM (Digital Rights Management)

Ausgehend von der Musikindustrie gibt es Bestrebungen, ein DRM auf den Rechnern zu etablieren. Dabei handelt es sich um Kontrollmechanismen, mit denen der Schutz der Urheberrechte auf PCs durchgesetzt werden kann. Urheberrechte beziehen sich dabei auf die Nutzung von Musikstücken, Filmen, Bildern, Dokumenten etc. Im wesentlichen soll ein solches DRM das unerlaubte Abspielen von solchen Inhalten verhindern.

Technisch ist das jedoch nicht einfach. Der Schutz von Musikstücken muss sich z.B. von der Quelle (CD, Internet) bis auf die Abspielgeräte erstrecken (d.h. im PC: Soundkarte). Dabei geht es vor allem um die digitalen Daten; analog lässt sich natürlich jedes Musikstück kopieren. Abspielgerät im PC ist die Soundkarte; die Daten müssen also auf dem gesamten Weg bis zur Soundkarte geschützt werden:

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

- Das Sound-Aufzeichnungsprogramm totalrecorder (www.totalrecorder.com) installiert sich z.B. in Form eines Soundtreibers im System. Spielt man die Musik auf diesem "Soundtreiber" ab, so wird sie auf Festplatte in ein Datei geschrieben.
  Um den PC DRM-fähig zu machen, dürfen also nur zertifizierte Sound-Treiber geladen werden. Dass OS muss alle anderen Treiber verbieten oder zumindest keine DRM-geschützten Inhalte diesen Treibern überlassen.
- Auch wenn nur geschützte Treiber verwendet werden: während des Abspielvorganges werden die Daten natürlich immer wieder im Speicher gehalten und z.B. von einem Puffer in den nächsten kopiert. Gelingt es nun einem Programm, diese Speicherbereiche auszuspionieren, so kann es die Musikdaten kopieren.
  Die gesamte Verarbeitungskette von DRM-Material muss also geschützt werden.
- Dies betrifft auch die Hardware: sonst könnte ein findiger Bastler eine Soundkarte bauen, die alle eintreffenden Daten auf einen Datenträger, z.B. Festplatte oder Speicher, schreibt. Das OS bzw. die Rechner-Hardware muss also alle Hardware-Geräte kontrollieren und sicherstellen, dass diese nicht manipuliert wurden.

Trotz diesen Problemen ist die Industrie stark an der Entwicklung von DRM-Systemen interessiert. Microsoft entwickelt entsprechende Konzepte für die nächste Generation des Windows-Betriebssystems (Codename Longhorn), wie NGSCB (Next Generation Secure Computing Base). Auch die benötigte Hardware befindet sich in der Entwicklung.

Weitere Aspekte von DRM sind neben der Technik (Hardware und Betriebssystem) die rechtlichen und wirtschaftlichen Hintergründe und die sozialen Folgen dieser Technik. Eine ausführliche Behandlung dieses Themas würde den Rahmen der Vorlesung sprengen. Weitere Informationen dazu gibts im Internet (z.B www.heise.de/security, nach "ngscb" suchen); eine aktuelle Beschreibung befand sich in Zeitschrift ct 12/04.

8.2.3 Geräte-unabhängige Software

Viele Teile eines Gerätetreibers sind geräte-unabhängig; typische geräteunabhängige Aufgaben werden in den nächsten Abschnitten dargestellt.

8.2.3.1 Einheitliches Interface für alle Treiber

Wichtig, wenn neue Hardware zum System hinzukommt (dann will man nicht das Ganze Betriebssystem ändern müssen). Deshalb verwendet man ein einheitliches Interface, dass alle Treiber implementieren müssen.

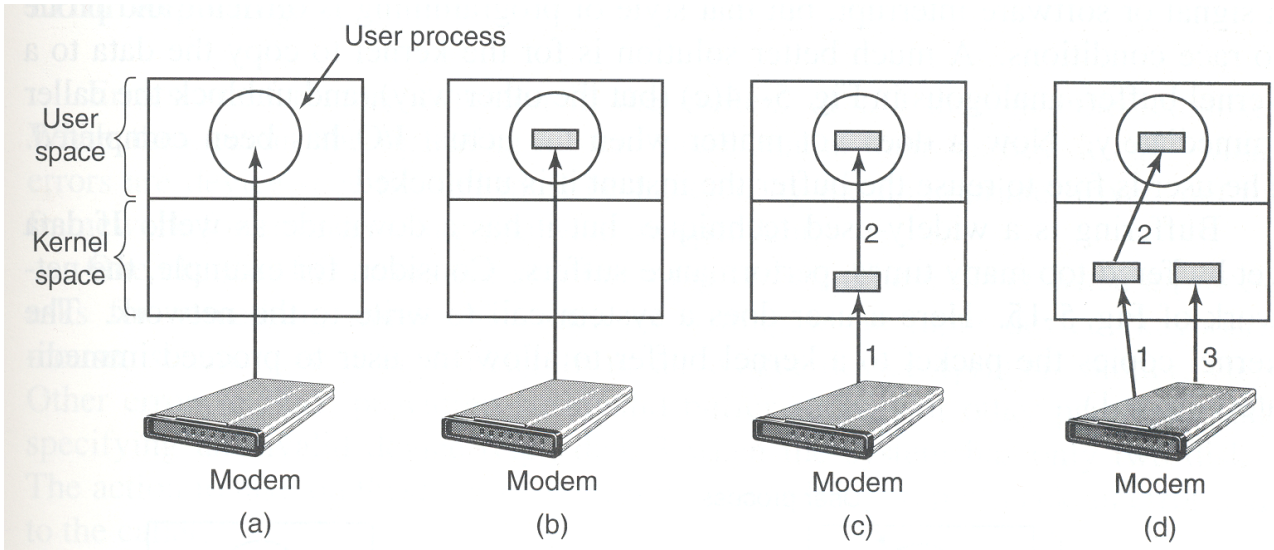
8.2.3.2 Datenpufferung

Kopieren der Daten von und zu den User-Prozessen

Problem:
Typischerweise hat ein User-Prozess Daten von einem Gerät angefordert, z.B. von einer Datei von der Festplatte. Dazu hat der User-Prozess einen Puffer angegeben, in dem diese Daten geschrieben werden können. Dorthin soll der Gerätetreiber nun die Daten schreiben; da er jedoch im Kernel-Mode läuft, hat er gar keinen Zugriff auf den Speicherbereich des User-Prozesses!

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

Lösung: Das Betriebssystem stellt einen eigenen I/O Manager bereit, der Daten zwischen User- und Kernelspace hin- und herkopieren kann, und der vom Gerätetreiber genutzt wird.



8.2.3.3 Fehlermeldungen

Standardisierte Fehlermeldungen und -behandlung, z.B. Gerät als inaktiv markieren, Fehlermeldung ausgeben, System anhalten & rebooten

8.2.3.4 Belegung und Freigabe von Geräten (Resource-Handling)

Das OS stellt Funktionen zur Resourcen-Verwaltung zur Verfügung

8.2.3.5 Geräte-unabhängige Block-Größe

Die unterschiedlichen Block-Größen verschiedener Geräte wird versteckt, und nur eine System-Blockgröße zurückgegeben. Ggfs. werden mehrere Blöcke zusammengefasst.

8.2.4 User-Space I/O Software

Stellt ein einheitliches Interface zur Hardware zur Verfügung:
- printf() Funktion zur Ausgabe von Zeichen
- Spooling, zur Verhinderung von Resourcen-Blockaden

8.3 Plattenlaufwerke

Es gibt verschiedene Typen von Plattenlaufwerken:
- Magnetische Disks (Festplatte, Floppies)
- CD-ROMs
- CD-R, CD-RW
- DVD

Im folgenden betrachten wir die Festplatte etwas näher.

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

8.3.1 Hardware

Physikalisches Prinzip:
Die Daten werden auf einer magnetischen Schicht gespeichert. Beim Schreiben wird über einen Kopf (Head) ein Magnetfeld erzeugt, dass in der magnetischen Schicht gespeichert wird. Beim Lesen wird das Magnetfeld mit demselben Kopf wieder ausgelesen.

Grundlagen der Hardware einer Festplatte (Zylinder, Tracks, Sektoren, etc.) werden wir nicht behandeln (ggfs. im Tanenbaum [1] nachlesen).

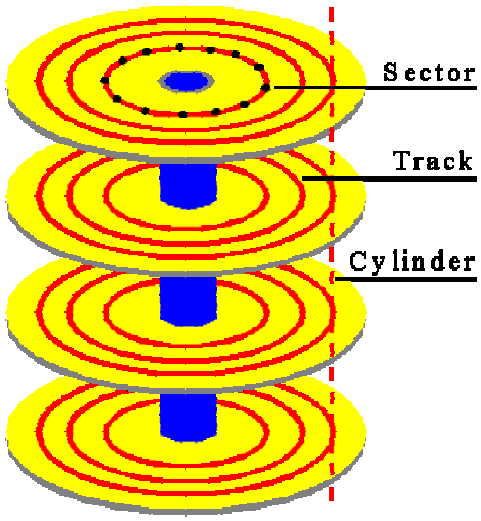


Abbildung 94: Festplatten-Geometrie und Bezeichnungen

Früher wurden die Festplatten vom Betriebssystem in den Koordinaten Kopf, Zylinder und Sektor angesprochen. Moderne Festplatten führen die Adressierung selbst durch:

- Die Anzahl der Sektoren in einem Track ist heutzutage typischerweise aussen höher als innen. Bei konstanter Plattengeschwindigkeit ist die Bitraten innen und aussen unterschiedlich (Constant Angular Velocity, CAV). Alternativ wird auch die Plattengeschwindigkeit variiert (Constant Linear Velocity, CLV).
- Defekte Sektoren werden erkannt und vom Festplattencontroller durch Reservesektoren ersetzt.

Das Betriebssystem adressiert die Festplatte in *logischen Blöcken* adressiert, d.h. alle Blöcke sind durchnummeriert, beginnend mit Null. Die typische Blockgröße beträgt 512 Bytes, obwohl auch andere Größen möglich sind.

8.3.2 Zugriffsstategien

Begrenzende Faktoren für die Geschwindigkeit von Festplatten sind:

1. **Zugriffszeit:** die Zeit, die der Magnetkopf braucht, um auf den richtigen Zylinder bewegt zu werden; typische Zugriffszeit heutzutage: 5-10 ms
2. **Rotationsbedingte Latenz**: Ist der Magnetkopf auf der richtigen Spur positioniert, muss gewartet werden, bis der gesuchte Sektor vorkommt. Bei einer Drehzahl von 7200/min = 120/s beträgt die durchschnittliche Verzögerung etwa 4 ms. Teilweise werden höhere Geschwindigkeiten eingesetzt, z.B. 10000/min bei Serverplatten.
3. **Datentransferzeit:** Einlesen der Daten, Aufbereitung und Übertragung in den Hauptspeicher; Datentransferrate heutzutage etwa 40 MB/s

Ein entscheidender Faktor für die Geschwindigkeit von Festplatten ist also die Zugriffszeit. Um die Plattenperformance zu steigern, wird der Zugriff auf die Platten optimiert. Dabei sind folgende Kriterien zu beachten:

- Geringe Zugriffszeit: die Zugriffszeit soll minimiert werden (Verbesserung der Performance)
- Fairness: alle Anforderungen sollen möglichst gleich berücksichtigt werden; keine Anforderung soll benachteiligt werden.

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

8.3.2.1 FCFS Scheduling

Die Anforderungen werden in der Reihenfolge abgearbeitet, in der sie ankommen. Der Algorithmus ist an sich fair, da keine Anforderung benachteiligt wird. Allerdings ist die Performance nicht besonders gut, was in folgendem Beispiel sichtbar wird:
Angenommen, die Anforderungen kommen in der folgenden Reihenfolge an:

98, 183, 37, 122, 14, 124, 65, 67.

Zu Beginn steht der Kopf auf 53. in der folgenden Graphik sind die Kopf-Bewegungen dargestellt: problematisch ist z.B. der Wechsel von 122 zu 14 und wieder zurück zu 124, der sehr zeitaufwändig ist; dabei hätten auch zugleich die Spuren 65 und 67 eingelesen werden können.

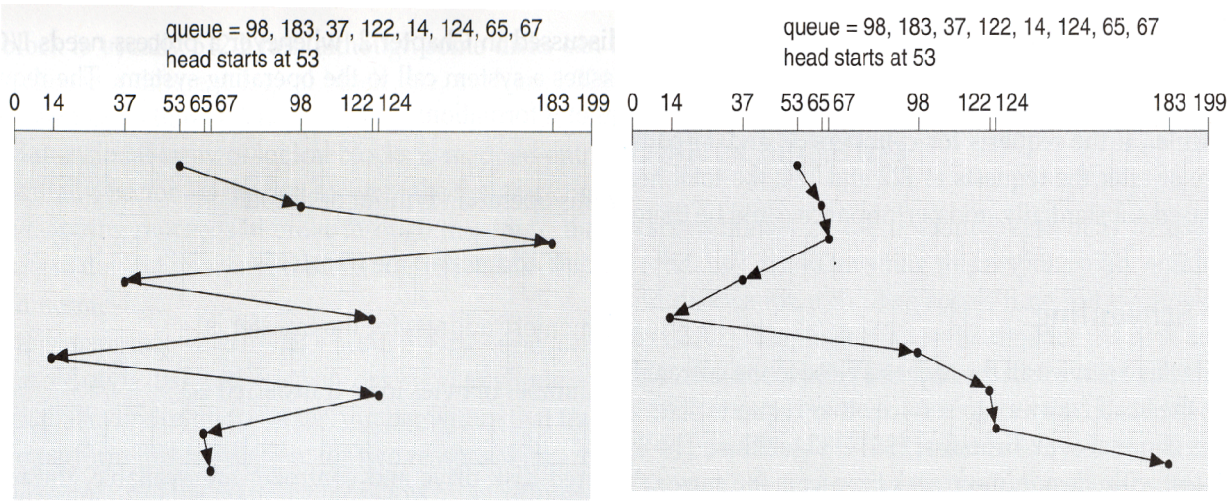


Abbildung 95: Festplattenzugriffe FCFS (links) und SSTF (rechts), nach [2]

8.3.2.2 SSTF Scheduling (auch SST)

Die Performance wird gesteigt, indem immer die Spur angefahren wird, die sich am nächsten an der augenblicklichen Kopfposition befindet (shortest-seek-time-first, SSTF). Dies ist am Beispiel an der obigen Graphik gezeigt.
Der Algorithmus ist jedoch nicht fair: bei hoher Last, d.h. wenn permanent neue Anforderungen ankommen, bewegt sich der Kopf immer weniger aus der Mittelposition heraus. Anforderungen, die Randsektoren betreffen, haben dann eine sehr lange Wartezeit.

8.3.2.3 Aufzug-Algorithmus (SCAN)

Beim SCAN Algorithmus startet der Kopf an einem Ende der Platte, und fährt dann in Richtung des anderen Endes, und arbeitet dabei alle Anforderungen ab. Dort angekommen, ändert er seine Richtung, und setzt die Abarbeitung fort. Der Algorithmus wird manchmal auch Aufzug-(Elevator)-Algorithmus, da derselbe Algorithmus auch bei Aufzugssteuerungen eingesetzt wird.
Das Verhalten in in der folgenden Graphik dargestellt.

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

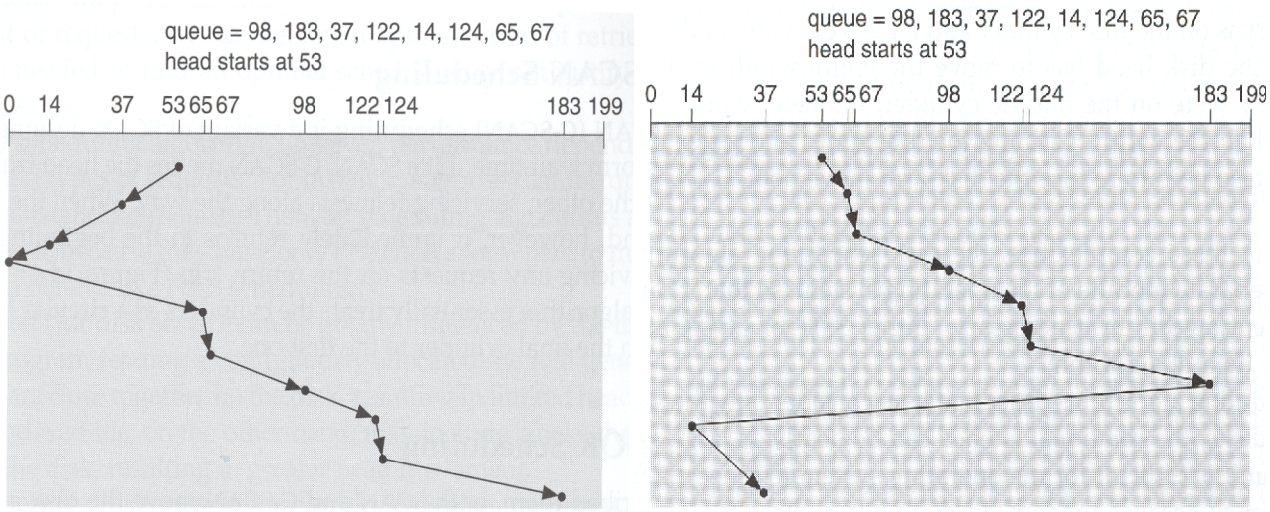


Abbildung 96: Festplattenzugriffe SCAN (links) und C-SCAN (rechts), nach [2]
Verbesserungen vom SCAN Algorithmus sind der C-(Circular)-SCAN Algorithmus, in dem der Kopf nur in eine Richtung fährt; ist er am Ende angekommen, fährt er schnellstmöglichst zum Start zurück, ohne Anforderungen zu bearbeiten. Dort kehrt er seine Richtung um und beginnt wieder von vorne.

8.3.3 Fehlerbehandlung

Schlechte Sektoren werden durch Ersatzsektoren ausgetauscht.Dazu sind in jeder Spur einige Ersatzsektoren vorgesehen (siehe folgende Abbildung, links). Ist ein Sektor fehlerhaft, so wird er durch einen Ersatzsektor ausgetauscht (b). Allerdings ist dann die Zugriffsfolge 6-7-8 sehr ungünstig, da die Blöcke nicht kontinuierlich eingelesen werden können. Eine andere Möglichkeit besteht darin, die folgenden Blöcke zu verschieben, um die Reihenfolge zu erhalten (c).

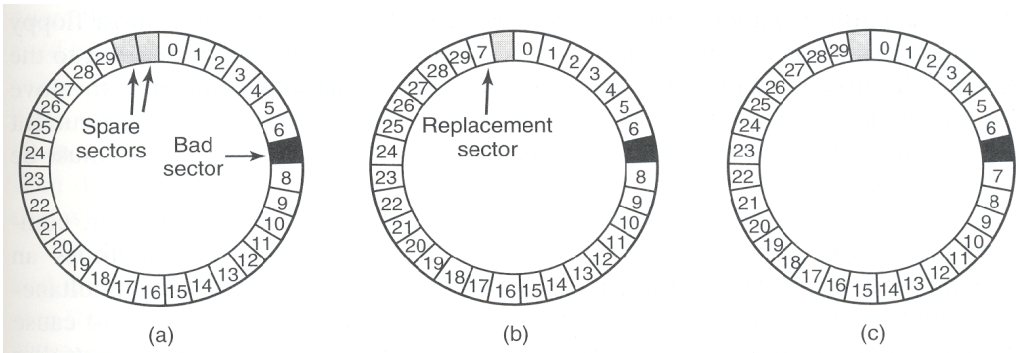


Abbildung 97: Fehlerhafte Sektoren

8.3.4 Stabilität

Was passiert, wenn ein Fehler während eines Schreibvorgangs auftritt?
-> Stabile Schreibvorgänge nötig

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

8.4 Serielle Schnittstelle

8.4.1 Grundlagen der Asynchronen Seriellen Kommunikation (RS232)

- Datenlänge 7 oder 8 Bit
- 1 Start-Bit, 1 oder 2 Stop-Bits
- Optional 1 Parity Bit, wobei Odd oder Even Parity möglich ist
- TTL-Pegel 0 V (logisch 0) und 5 V (logisch 1); RS232 Pegel logisch 0 entspricht +12V (+3V bis +15V), logisch 1 – 12V (-3V bis -15V)
- unterschiedliche Baud-Raten möglich; da asynchrone Kommunikation (keine Clock-Leitung), müssen Empfänger und Sender vorher die Baudrate kennen (manche Geräte haben jedoch eine automatische Baudratenerkennung)

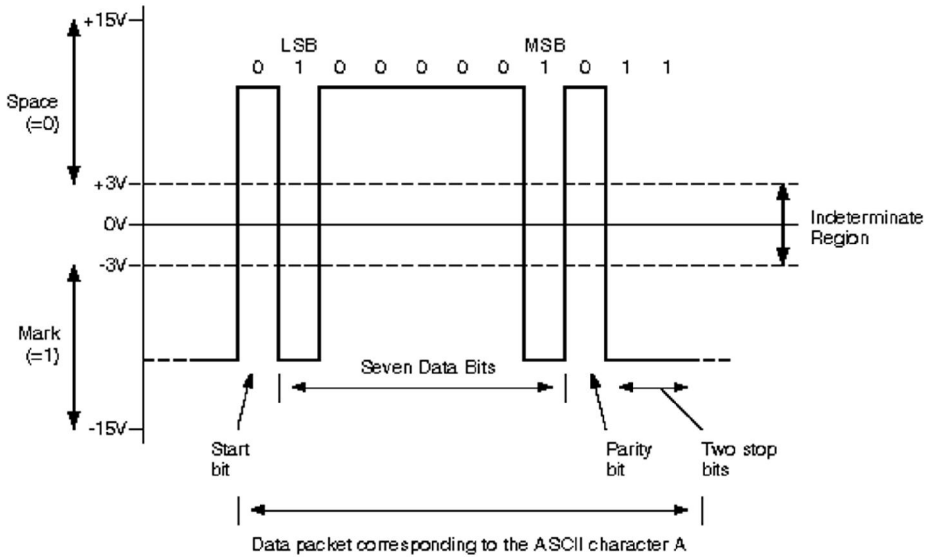


Abbildung 98: RS232-Übertragung (7-Bit)

8.4.2 Controller-Register

Ein typischer Controller ist z.B. der UART 16550 (Universal Asynchronous Receiver Transmitter). Das Prinzip ist immer dasselbe:

- Senderegister (1 Byte oder mehr Byte)
- Empfangsregister (1 Byte oder mehrere Bytes, z.B. 16)
- Kontroll-Register, mit denen die wesentlichen Parameter eingestellt werden können:
    - Baudrate, Datenlänge, Parity, Stop-Bits, etc.
    - Ob und wann ein Interrupt ausgelöst werden soll, z.B. nach dem Empfang eines Zeichens
- Fehlerbehandlung
    - Erkennen von Datenübertragungsfehlern
    - Erkennen von Leitungsbruch

8.4.3 Ansteuerung auf Interrupt-Ebene

Der Controller wird wie folgt angesteuert:

Dr. Ulrich Margull
Kirchweg 6 * 93055 Regensburg
Tel. (+49) 941 78956-8 Fax –9
www.margull.de

- Initialisierung und Setzen der Kontrollregister
- Senden:
    - Zu sendende Daten befinden sich in einem Puffer im Speicher.
    - 1 oder mehrere Zeichen (je nach Fähigkeiten des UART) werden in den Sendepuffer des UART kopiert. Dann wird die Sendung gestartet.
    - Der UART sendet nun Bit für Bit auf der Leitung. Wenn er mit dem Zeichen fertig ist, nimmt er das nächste Zeichen aus seinem Sendepuffer und sendet dieses. Ist der Sendepuffer leer, wird ein Interrupt ausgelöst.
    - Im Interrupt werden weitere Zeichen aus dem Puffer (im Speicher) in den Controller kopiert und der vorige Schritt wird wiederholt.
    - Wurden alle Zeichen gesendet, wird in der Interrupt-Routine ein entsprechendes Flag gesetzt, damit das Hauptprogramm über das Ende der Bearbeitung benachrichtigt wird.
- Empfangen analog. Beim Empfangen ist zu beachten, dass die Zeichen rechtzeitig aus dem Controller ausgelesen werden müssen. Beträgt die Baudrate z.B. 115kBit, so werden etwa 11 Zeichen / Millisekunde empfangen. Beträgt der Empfangspuffer im Controller 16 Bytes, so müssen er alle 1,5 Millisekunden ausgelesen werden. Wird das einmal versäumt, so läuft der Empfangspuffer über und Zeichen gehen verloren.

8.4.4 Ansteuerung auf User-Ebene (Win32)

Windows (Win32) bietet folgende Funktionen zur Ansteuerung der seriellen Schnittstelle:

- Initialisieren der RS232 Schnittstelle mittels **SetCommState()** / **GetCommState**; dabei wird eine Datenstruktur vom Typ DCB verwendet, die alle Einstellungen der Schnittstelle enthält
- Öffnen einer Kommunikation mittels **CreateFile()**
- Schreiben von Daten: **WriteFile()**
- Lesen von Daten: **ReadFile()**
- **Asynchroner Datentransfer:** Durch ein ReadFile() wird der Datentransfer initialisiert, und die Funktion kehrt sofort zurück (ohne Daten!). Wurden die Daten eingelesen, so benachrichtigt das OS das Programm, das die Daten nun vorliegen.
    - Vorteil: bessere Performance, da das Programm nicht auf die Daten warten muss
    - Nachteil: Sehr viel aufwändiger zu implementieren, da mehrere Threads nötig sind und diese entsprechend synchronisiert werden müssen.

Mit all diesen Funktionen (CreateFile/ReadFile/WriteFile) können die unterschiedlichsten Geräte angesprochen werden (Datei, Console, seriell, parallel, u.a.). Durch ein einheitliches Interface wird der Umgang und die Programmierung vereinfacht. Zudem enthält das Win32 API dadurch weniger Funktionen. Funktionalität die allen Geräten gemeinsam ist – wie z.B. asynchrone Funktionsaufrufe zur Ein/Ausgabe – kann mit denselben Funktionen erfüllt werden. Ebenso eine einheitliche Fehlerbehandlung für alle Geräte

8.5 Weitere Geräte

Viele weitere Geräte: Uhren, Zeichen-Orientierter Tastaturen, Graphische User-Interfaces, Netzwerk, Power-Management
behandlen wir leider nicht! -> Für Interessierte: einschlägige Literatur, z.B.[1]