Rechner– und Systemarchitekturen

Einführung in die Informatik 3
für den Studiengang Informatik
im Fachbereich MND
der Fachhochschule Frankfurt am Main

Prof. Dr. Erik Jacobson

Wintersemester 2000/2001

Übersicht

Belegnummer: 06 65 35

Inhalte

1. Hardware (wie ein Computer funktioniert)

- Schaltungstechnik: Schaltungslogik, Schaltelemente
- Beschreibungsmittel (Graphen): Petri-Netze, Zustandsdiagramme
- Datendarstellungen
- Rechnerarchitekturen:
  - Die Zentraleinheit
  - Die CPU, Maschinenbefehle und Mikroprogrammierung
  - Der Bus
  - Der Arbeitsspeicher
  - Ein- und Ausgabe, Schnittstellen
  - Periphere Geräte: Terminal, Drucker, Plotter, Massenspeicher

2. Softwareware (Die Basismaschine)

- Betriebsarten, Betriebssystemfunktionen
- Prozesse, Prozeßverwaltung
- Speicherverwaltung
- Dateiverwaltung
- Interprozeßkommunikation
- Benutzeroberflächen
- Schichtenmodell
- Anwendungsbeispiel: UNIX, MVS, VMS, BS200, RSX, RT-11, o.ä.
- Spezialthemen, z.B.: TP-Systeme, TP-Monitore

3. Prozeßdatenverarbeitung, PDV (Anwendungsgebiet)

- Prozeßelemente, Prozeßglieder (Sensoren, Motoren, ADCs, DACs)

Literatur

- Coy, W.: Aufbau und Arbeitsweise von Rechenanlagen. Vieweg 1988
- Jacobson, E.: Einführung in die Prozeßdatenverarbeitung. Hanser 1996
- Sokolowski/ Huge: Digitale Schaltungs- und Rechnertechnik.

Labor (nach Vereinbarung)

- Bussignale und -protokolle
- COM-Schnittstelle (RS 232C, V.24)
- Laufzeitmessungen

Leistungsnachweis

Prüfungsklausur am Semesterende

Kapitel 0

Einleitung

0.1 Historische Grundlagen

0.1.1 Hilfsmittel zur Datenverarbeitung

0.1.1.1 Primitive Hilfsmittel

- 6000 v Chr. 2 Hände x 5 Finger = biquinär
  = 10 Finger = dezimal
  Finger = lat.: digitus → digital
  Zählen = lat.: computare → computer
  Zählstäbchen = lat.: calculi → Kalkül
- 1000 v Chr. Abakus (Griechenland, China, Rußland, Rom)

0.1.1.2 Mechanische Hilfsmittel

1623 Schickard, W. Tübingen: 6-stellige Rechenuhr (Add, Sub)
1641 Pascal, Blaise Frankreich: 6-stellige Addiermaschine
1650 Partridge, England: Rechenschieber
1671 Leibniz, G.W. Hannover: 4-Spezies-Rechner mit Staffelwalzen (Kurbelmaschine)
1900 elektrischer Antrieb für Kurbelrechenmaschinen

0.1.1.3 Logische Hilfsmittel

- 1000 Zahldarstellung mittels Buchstaben (Alphabet = α, β)
- 0 Mathematische Grundlagen: Pythagoras, Aristoteles, ..
± 0 Römisches Zahlensystem (I, V, X, L, C, D, M)
500 Hindu-Arabisches Zahlensystem: (dezimale) Stellenschreibweise
1574 Adam Riese: Popularisierung der Rechenkünste
1614 Napier: Logarithmentafeln
1703 Leibniz: Dualsystem
1847 Boole: Schaltalgebra
1833 Babbage & Ada (Augusta Comtess of Lovelace geb. Byron): Difference Engine, Analytic Engine (Mill, Store, Control)
1940 Programmiersprachen

0.1.2 DV-Systeme

1890	Hollerith:	Lochkartenmaschine zur Volkszählung in Chikago
1934	Konrad Zuse, Berlin:	Z1: erster programmgesteuerter Rechner Z3: CPU mit 600 Relais, 22 Dualstellen, Speicher mit 64 Worten (2000 Relais), Steuerung mit Programm auf Lochstreifen
1944	Aiken @ Harvard, USA	Relais-Rechner MARK 1
1946	Ekert & Mauchly @ MIT, USA	Röhren-Rechner ENIAC (18 000 Röhren)
1946	John v. Neumann @ MIT	Rechner-Architektur: Steuerwerk + Rechenwerk ein Speicher für Daten und Programme, Ein- und Ausgabeeinheiten (-werke)
1950	Universalrechner:	Hardware und Betriebssysteme (in ROM)
1959	Siemens 2002	2000 Worte à 14 BCD-Stellen (56 Bit)

0.1.3 Rechner-Generationen

Nr	Zeit	Schaltelement	-zeit	Operationszeit	Add/Sek
0	1935 - 1945	Relais	1 ms	100 ms	30
1	1945 - 1955	Röhren	10 μs	1 ms	1000
	1951	Kernspeicher	1 μs		
2	1955 - 1965	Transistoren	1 μs	100 μs	10^4
3	1965 - 1971	Integrierte Schaltung (SSI)	10 ns	1 μs	10^4
4	1971 - 1985	LSI	1 ns	100 ns	10^6
5	1985 - 1995	VLSI, XLSI	1 ns	10 ns	10^7

0.1.4 Physikalische Grenzen

Größe	aktuell	Grenze
Breite einer Leiterbahn	$0.4\mu m$ (400 nm)	100 Atome $= 100 \cdot 10^{-10} m = 10nm$
Taktfrequenz	200 – 400 MHz	20 GHz ($\lambda = c^*/f = 1cm$) ($c^* = c/\varepsilon \approx 200\,000$ km/s)

0.2 Begriffe

DIN 44 300, 66 xxx, ISO 2382 Teile 1 bis 36

Informatik (engl.: Computer Science, CS): Wissenschaft von der Informationsverarbeitung

Anwendung: Informationstechnik (Information Technology, IT)

0.2.1 Grundbegriffe

Information (objektartig): Dargestelltes Wissen über beliebige Objekte: Fakten, Ereignisse, Dinge, Vorgänge, Ideen, Konzepte, ...

Daten: Formalisierte Darstellung von Information, die geeignet ist zur Weitergabe, Interpretation oder Verarbeitung.

Signale: Veränderung einer physikalischen Größe zur Darstellung von Daten. Zum Beispiel Strom, Spannung, Helligkeit, Farbe.

0.2.2 Basisreferenzmodell

Schichten-, Schalen-, Strukturmodell

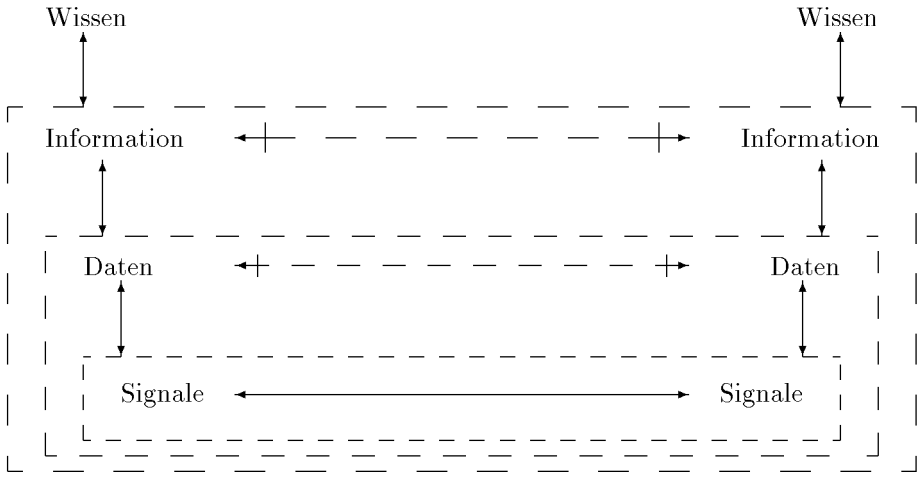
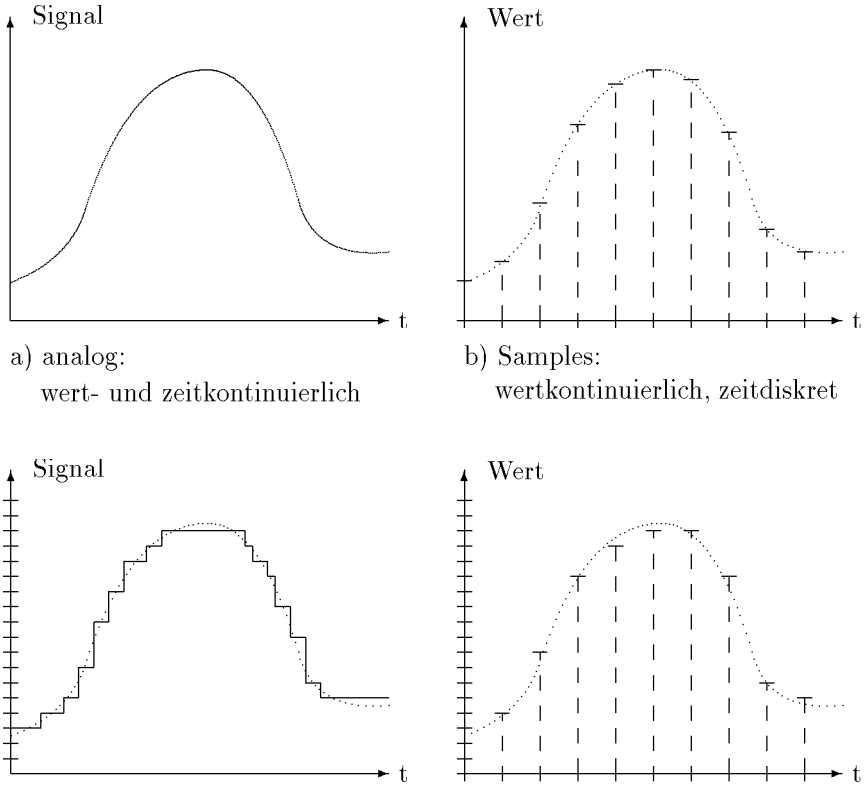


Bild b1p01

↓ : Darstellung, Repräsentation, Codierung
↑ : Erwerb, Interpretation, Decodierung
⟷ : Übertragung, Verarbeitung, Speicherung

0.2.3 Signale

Signalwert (Signalparameter): veränderlicher Wert, abhängig von Ort oder Zeit (y-Koordinate), z.B. elektrische Spannung in Volt.
Signalverlauf in der Zeit (t) oder als Funktion des Ortes (x-Koordinate)
Signalqualität:
- kontinuierlich (auf reele Zahlen abbildbar)
- diskret (auf natürliche (ganze) Zahlen abbildbar)



a) analog:
wert- und zeitkontinuierlich

b) Samples:
wertkontinuierlich, zeitdiskret

c) quantisiert:
wertdiskret, zeitkontinuierlich

d) digital:
wert- und zeitdiskret

Bild b1p02

0.2.4 Daten

analog: kontinuierlich.

diskret: diskontinuierlich, unterscheidbare Werte.

digital: durch Daten aus Zeichen dargestellt.

numerisch: Daten aus Numeralen (Zeichenfolgen für Zahlen).
Beispiel: 11, elf, XI, B_{16}

alphanumerisch: Daten aus Buchstaben und Ziffern.

Zeichen: Element einer Menge von Symbolen.

Ziffer: Zeichen zur Darstellung einer (nichtnegativen) Zahl.

Symbol: Graphische Darstellung mit vorgegebener Bedeutung (kontextabhängig).

Bit: Zeichen zur Darstellung von Ziffern im Dualsystem.

Wort: Syntaktische Einheit von Zeichen.

Byte: (Teil)Wort fester Länge im Binärsystem (meist 8 Bit) zur Darstellung von Zeichen aus anderen Zeichensätzen (z.B. ASCII).

Oktett: Folge von 8 Zeichen (Bits).

String: Folge von Zeichen unbestimmter Länge.

Satz: Folge von Worten (mit Struktur).

Datei: Folge von Sätzen (gleicher Struktur).

Text: Darstellung von Konstrukten einer Sprache durch Daten. Gewöhnlich in Form von Zeichen, Symbolen, Worten, Phrasen, Absätzen, Sätzen, Tabellen oder anderen Zusammenstellungen von Zeichen.

Datenobjekt: Daten mit vorgegebenen Datentyp.

Datentyp: Aufbau und Anordnung von Daten und die auf ihnen erlaubten Operationen.

Adresse: Wort zur Kennzeichnung (Lokalisierung) eines Datenobjekts.

Adreßraum: geordnete Menge von (zugelassenen) Adressen

Programm: Folge von Anweisungen und Vereinbarungen zur Lösung einer Aufgabe (eines Problems).

Anweisung: Arbeitsvorschrift, abgefaßt in einer (Programmier-)Sprache

Vereinbarung: Festlegung von Sprachelementen für Anweisungen.

Operation: Ein Methode, um nach vorgegebenen Regeln aus gegebenen Objekten (Operanden) ein neues Objekt (Resultat) zu erzeugen.

Instruktion: elementare Anweisung.

System: Eine Menge von Objekten und ihren Beziehungen, die von ihrer Umgebung abgegrenzt erscheint.

Struktur: Die Beziehungen zwischen den Elementen eines Systems.

Kapitel 1

Grundlagen der Digitalelektronik

1.1 Boole'sche Algebra

1.1.1 Definition

Einfachste Algebra mit
- 2 Elementen: {0, 1} {L, H} {O, L}
- 3 Grund-Operationen:
  - unäre Operation:
    Komplement, Negation, NICHT, NOT,

$x \rightarrow \bar{x}$	
0	1
1	0

  - binäre Operationen:
    Disjunktion, ODER, OR, . .Konjunktion, UND, AND,
    f(x,y) = Max(x,y) = x + y f(x,y) = Min(x, y) = $x \cdot y$

$x \vee y \rightarrow z$		
0	0	0
0	1	1
1	0	1
1	1	1

$x \wedge y \rightarrow z$		
0	0	0
0	1	0
1	0	0
1	1	1

- Zusammengesetzte binäre Operationen (2 Beispiele):
  Äquivalenz, . .Antivalenz, Exklusives Oder, XOR,
  $f(x,y) = x \cdot y + \overline{x} \cdot \overline{y}$ $f(x,y) = x \cdot \overline{y} + \overline{x} \cdot y$

$x \equiv y \rightarrow z$		
0	0	1
0	1	0
1	0	0
1	1	1

$x \oplus y \rightarrow z$		
0	0	0
0	1	1
1	0	1
1	1	0

Gesetze der Boole'schen Algebra:

Assoziativgesetz
$$(x + y) + z = x + (y + z) = x + y + z$$
$$(x \cdot y) \cdot z = x \cdot (y \cdot z) = x \cdot y \cdot z$$
$$(x \oplus y) \oplus z = x \oplus (y \oplus z) = x \oplus y \oplus z$$

Kommutativgesetz
$$x + y = y + x$$
$$x \cdot y = y \cdot x$$
$$x \oplus y = y \oplus x$$

Distributivgesetz
$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$
$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

De Morgansche Regeln
$$\overline{(x + y)} = \overline{x} \cdot \overline{y}$$
$$\overline{(x \cdot y)} = \overline{x} + \overline{y}$$

Absorptionsregeln
$$x + \overline{x} = 1 \qquad x \cdot \overline{x} = 0 \qquad x \oplus \overline{x} = 1$$
$$x + x = x \qquad x \cdot x = x \qquad x \oplus x = 0$$
$$x + x \cdot y = x \qquad x \cdot (x + y) = x$$

Neutrale Elemente

0 bezüglich OR (+): $x + 0 = x \qquad x \cdot 0 = 0 \qquad x \oplus 0 = x$

1 bezüglich AND ($\cdot$): $x \cdot 1 = x \qquad x + 1 = 1 \qquad x \oplus 1 = \overline{x}$

1.1.2 Boole'sche Funktionen

Abbildung von m logischen Werten auf n logische Werte

$B^m \rightarrow B^n$ mit $B = \{0, 1\}$

dargestellt durch Funktionen: $\left\{ \begin{array}{c} x \\ y \end{array} \right\} = \left\{ \begin{array}{c} f_1(a, b, c) \\ f_2(a, b, c) \end{array} \right\}$

oder durch Wertetabellen,

zum Beispiel:

a	b	c	x	y
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

1.1.3 Normalformen

Erstellung von Boole'schen Funktionen aus Wertetabellen.
Vollständige Normalformen enthalten in allen Funktionstermen alle Variablen (oder deren Inverse). Sie können meist unter Anwendung der Gesetze der Boole'schen Algebra oder mit Hilfe von KV-Diagrammen zu kompakten (reduzierten) Normalformen vereinfacht werden.

1.1.3.1 Disjunktive Normalform

DNF (Min-Terme)
- Tabellenzeilen mit Funktionswert '1' liefern einen Funktionsterm (Min-Term)
- Funktionsterme werden aus den Eingangsvariablen mit UND verknüpft
- alle Funktionsterme werden mit ODER verknüpft
- DNF = $\Sigma\Pi x_i$ (Summe von Produkten)

1.1.3.2 Konjunktive Normalform

KNF (Max-Terme)
- Tabellenzeilen mit Funktionswert '0' liefern einen Funktionsterm (Max-Term)
- Funktionsterme werden aus den invertierten Eingangsvariablen mit ODER verknüpft
- alle Funktionsterme werden mit UND verknüpft
- KNF = $\Pi\Sigma\overline{x_i}$ (Produkt über Summen)

1.1.3.3 Beispiel

a	b	c	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

. .Disjunktive Normalform:

$x = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c$ (vollständige DNF)

$x = \bar{a} \cdot b \cdot \bar{c} + \bar{b} \cdot c + a \cdot \bar{b} \cdot \bar{c}$ (reduzierte DNF)
(Absorption von $(a + \bar{a}) = 1$)

oder

$x = \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} + \bar{a} \cdot \bar{b} \cdot c$ (reduzierte DNF)
(Absorption von $(c + \bar{c}) = 1$)

oder (nach Verdopplung von $a \cdot \bar{b} \cdot c$)

$x = a \cdot \bar{b} \cdot c + \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c$

$x = \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b}$

Konjunktive Normalform:

$x = (a + b + c) \cdot (a + \bar{b} + \bar{c}) \cdot (\bar{a} + \bar{b} + c) \cdot (\bar{a} + \bar{b} + \bar{c})$

1.1.4 Karnaugh-Veitch-Diagramme (KV-Diagramme)

Erstellung von Minimalen Disjunktiven Normalformen (MDNF) in vier Schritten:

a) Erstellung des KV-Diagrammrahmens nach folgender Regel:
Die (n) Variablen werden paarweise.. mit ihrer Negation in Spalten und Zeilen einer Matrix angeordnet derart, daß Quadrate (falls n gerade ist) oder Rechtecke (falls n ungerade ist) entstehen, deren Felder eindeutig einer Variablenkombination zugeordnet sind.

	x_3		$\overline{x_3}$	
	x_1	$\overline{x_1}$	$\overline{x_1}$	x_1
x_2	$x_1\,x_2\,x_3$	$\overline{x_1}\,x_2\,x_3$	$\overline{x_1}\,x_2\,\overline{x_3}$	$x_1\,x_2\,\overline{x_3}$
$\overline{x_2}$	$x_1\,\overline{x_2}\,x_3$	$\overline{x_1}\,\overline{x_2}\,x_3$	$\overline{x_1}\,\overline{x_2}\,\overline{x_3}$	$x_1\,\overline{x_2}\,\overline{x_3}$

Es ist zu beachten, daß benachbarte Felder sich in der Negation von nur einer Variablen unterscheiden.

b) Eintragen der Funktionswerte in den Rahmen

c) Umschließen von benachbarten Feldern, die eine "1" enthalten, zu Rechtecken möglichst großer Seitenlängen in Schritten von 1, 2, 4 ... Einheiten (dabei sind auch Felder einzuschließen, die - einer Torus-Topologie entsprechend - an der gegenüberliegenden Seite liegen).

d) Jedes der gefundenen Rechtecke stellt den Konjuktionsterm seiner beiden Seiten dar. Die Disjunktion dieser Terme ergibt die gesuchte minimale Funktion.

Beispiel

	c		$\overline{c}$	
	a	$\overline{a}$	$\overline{a}$	a
b	0	0	(1)	0
$\overline{b}$	(1	1)	0	(1)

oder

	c		$\overline{c}$	
	a	$\overline{a}$	$\overline{a}$	a
b	0	0	(1)	0
$\overline{b}$	(1	(1)	0	(1)

Bild b1p02c

$x = \bar{a} \cdot b \cdot \bar{c} + \bar{b} \cdot c + a \cdot \bar{b} \cdot \bar{c}$ oder $x = \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} + \bar{a} \cdot \bar{b} \cdot c$

ergibt zusammen

	c		$\overline{c}$	
	a	$\overline{a}$	$\overline{a}$	a
b	0	0	(1)	0
$\overline{b}$	(1	1)	0	(1)

Bild b1p02b

$x = \bar{a} \cdot b \cdot \bar{c} + \bar{b} \cdot c + a \cdot \bar{b}$

1.2 Elektrotechnische Grundlagen

1.2.1 Zweipole

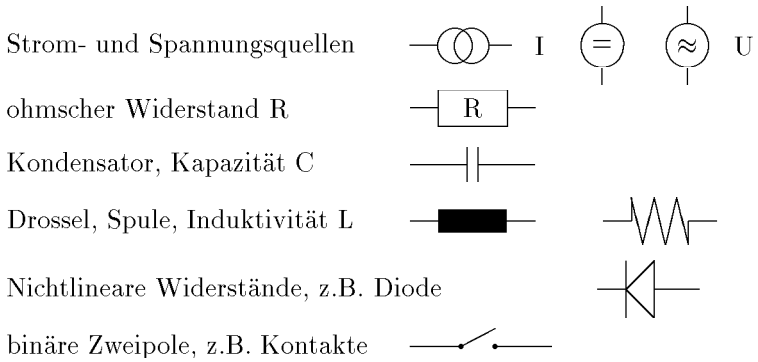


Bild b1p03a

Ohm'sches Gesetz:
$U = I \cdot R$

Komplexe Wechselstromlehre
$\tilde{U} = \tilde{I} \cdot \tilde{R}$ mit $\tilde{R} \in \{R, R_C = \frac{1}{i\omega C}, R_L = I\omega L\}$
$\tilde{U} = \hat{U} \cdot e^{i\omega t}$ und $\tilde{I} = \hat{I} \cdot e^{i\omega t}$

Multipole
Verknüpfungen von Zweipolen zu Multipolen mit Maschen (M) mit Knoten (K). (Jeder Knoten kann als Pol betrachtet werden)

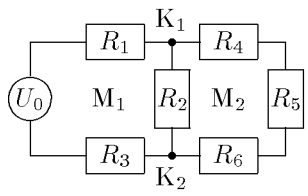


Bild b1p03

Kirchhoffsche Regeln:

Maschenregel: $\sum U_i = 0$

Knotenregel: $\sum I_j = 0$

Vierpole
Eingang: U_1, I_1
Ausgang: U_2, I_2

Übertragungsfunktion:
$\tilde{H} = \tilde{U}_2/\tilde{U}_1 = \tilde{H}(\omega)$

Darstellung durch
–**Ortskurve**
in der komplexen Zahlenebene
–**Bodediagramme**
für den Betrag $H(\omega) = \sqrt{Re\,H^2 + Im\,H^2}$
und den Phasenwinkel $tan\varphi = Im\,H/Re\,H$.

Beispiel:

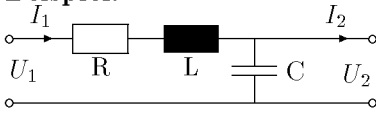
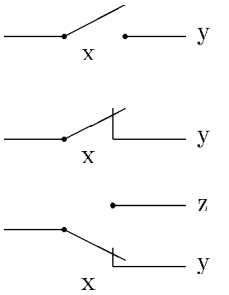


Bild b1p03b

$U_2/U_1 = 1/(1 - \omega^2 LC + i\omega RC)$

1.2.2 Schalter



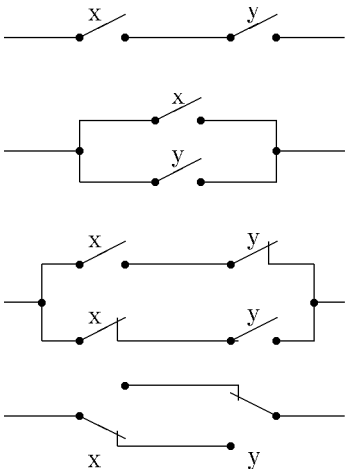
Schließer: Arbeitskontakt (Einschalter) $y = x$

Öffner: Ruhekontakt (Ausschalter) $y = \overline{x}$

Wechsler:
Arbeits- und Ruhekontakt (Umschalter) $y = \overline{x}, z = x$

Bild b1p04

1.2.3 Verknüpfungen



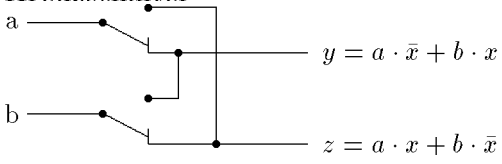
$z = x \cdot y$ UND (AND)

$z = x + y$ ODER (OR)

$z = x \oplus y$ Exklusives ODER (XOR)

Bild b1p05

Kreuzschalter



$y = a \cdot \bar{x} + b \cdot x$

$z = a \cdot x + b \cdot \bar{x}$

Bild b1p05a

1.2.4 Relais

a) Passive Relaisschaltungen:

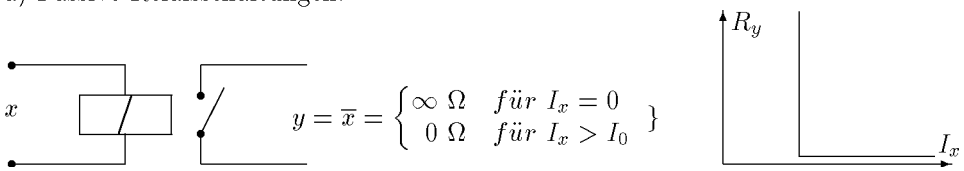
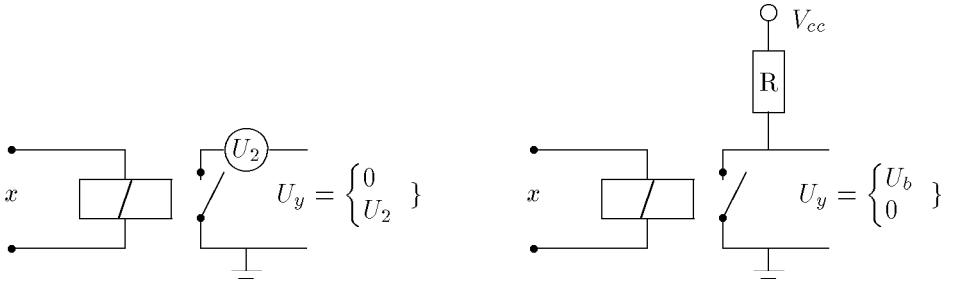


Bild b1p06

Die Ausgangsgrößen (Durchlaßwiderstände R_y) entsprechen nicht den Eingangsgrößen (Ströme I_x)

b) Aktive Schaltungen benötigen eine Hilfsspannung (V_{cc}) und haben die gleichen Eingangs- und Ausgangsgrößen (U_x, U_y).



aktive Relaisschaltung

aktive Relaisschaltung (invertierend)

Bild b1p06a

1.2.5 Transistoren

a) Transistoren als Vierpole (z.B. in Emitterschaltung):

$I_1 = Y_{11} \cdot U_1 + Y_{12} \cdot U_2$ . .Eingangswiderstand $R_{in} = 1/Y_{11}$ (groß) . .$Y_{12} \approx 0$

$I_2 = Y_{21} \cdot U_1 + Y_{22} \cdot U_2$ Steilheit $S = Y_{21}$ (Verstärkung) $Y_{22} \approx 0$

Mit Vektoren und Matrizen:

$\vec{I} = \hat{Y} \cdot \vec{U}$ oder $\vec{U} = \hat{H} \cdot \vec{I}$

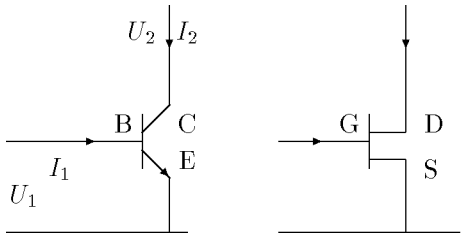


Bild b1p07

. .

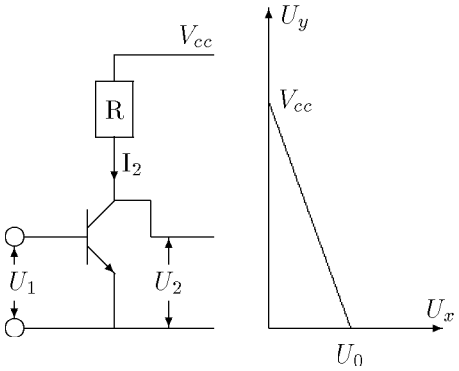
NPN-Transistor:
C = Collektor
E = Emitter
B = Basis

FET (verallgemeinert):
D = Drain
S = Source
G = Gate

b) Feld-Effekt-Transistoren (FET)
besonders kleine Eingangsströme ($I_1 \approx 0$)

c) Aktive Schaltung (Analogverstärker)

Einstufiger Verstärker (invertierend) . .Zweistufiger Verstärker (nicht invertierend)



Schaltung Kennlinie

Bild b1p08

$U_2 = V_{cc} - I_2 R$ mit $I_2 = Y_{21} U_1 + 0$

$U_2 = V_{cc} - RS\, U_1$

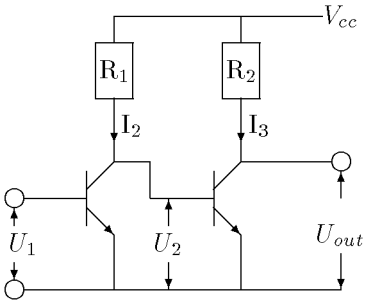


Bild b1p08a

$U_{out} = V_{cc} - I_3 R_2$ mit $I_3 = Y_{21} U_2 + 0$

$U_2 = V_{cc} - I_2 R_1$ mit $I_2 = Y_{21} U_1 + 0$

$U_{out} = V_{cc}(1 - R_1 Y_{21}) + R_1 R_2 Y_{21}^2 U_1$

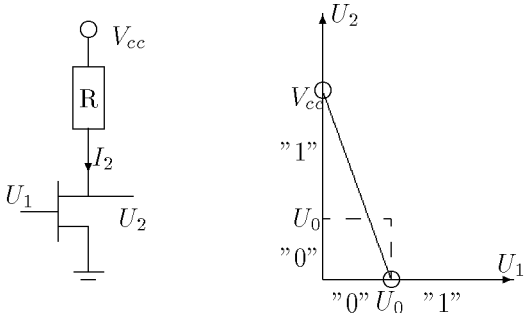
$U_{out} = V_{cc}(1 - RS) + RS^2 \cdot U_1$

(lineare Verstärkung)

1.2.6 Schalttransistoren

Sättigungsbetrieb in 2 Zuständen: ($I_2 = 0$ und $U_2 = 0$).

Vorzugsweise mit FET wegen höherer Schaltgeschwindigkeit.



Transistor Kennlinie

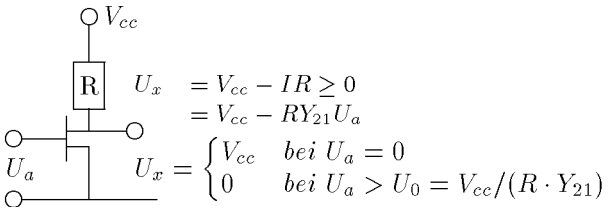
Bild b1p07a

1.3 Logische Schaltungen

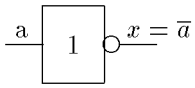
1.3.1 Elementare logische Schaltungen

Aus FETs

1.3.1.1 Inverter (NOT)



Schaltung



Schaltzeichen

Bild b1p09

Kennlinie:

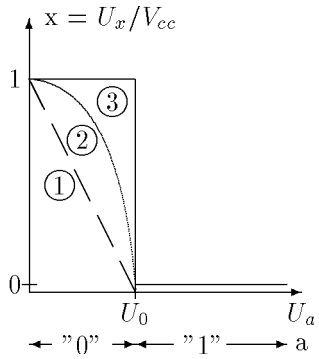


Bild b1p10

Ausgangswert (x) in Abhängigkeit vom Eingangswert (a)
1: bei linearer Verstärkung
2: bei Berücksichtigung von Nichtlinearitäten und Schwellspannungen
3: idealisierter Verlauf für binäre Ein- und Ausgangsvariablen (a, x)

1.3.1.2 Zweistellige Gatter

NAND-Gatter

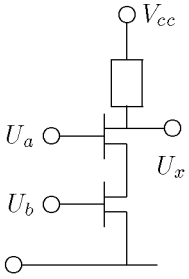


Bild b1p11

$$U_x = \begin{cases} V_{cc} \; falls \; I = 0 \\ 0 \quad falls \text{ beide FETs leiten, } U_a \wedge U_b \geq U_0 \end{cases}$$

Schaltzeichen:

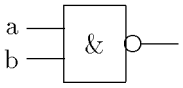


Bild b1p11a

NOR-Gatter

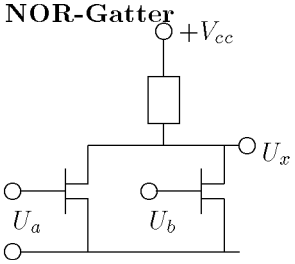


Bild b1p12

$$U_x = \begin{cases} V_{cc} \; falls \; I_a \; und \; I_b = 0 \\ 0 \quad falls \text{ ein FET leitet, } U_a \vee U_b \geq U_0 \end{cases}$$

Schaltzeichen:

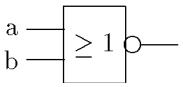
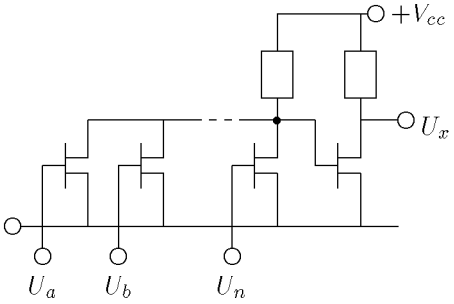


Bild b1p12a

1.3.1.3 Mehrstellige Gatter

– Multi-OR (Vielfach-ODER)



Schaltzeichen

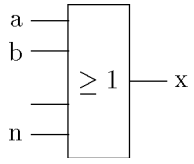
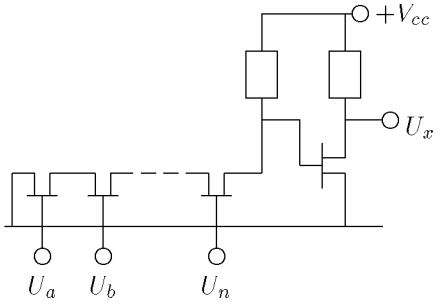


Bild b1p13

– Multi-AND (Vielfach-UND)



Schaltzeichen

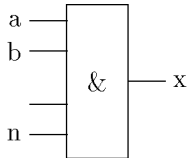


Bild b1p14

1.3.1.4 Schaltnetze für Boole'sche Funktionen

Boole'sche Funktionen können in Normalformen dargestellt werden, welche sich direkt in Schaltnetze umsetzen lassen: jeder Funktionsterm läßt sich durch ein Mehrfach-Gatter darstellen, und ebenso deren Verküpfungen.

- DNF = $\Sigma\Pi x_i$ (ODER-Verküpfung über UND-Gatter)
- KNF = $\Pi\Sigma\overline{x_i}$ (UND-Verknüpfung über ODER-Gatter)

Bei Anwendung der De Morganschen Regeln können aus Min-Termen oder Max-Termen Schaltungen mit gleichen Typen von invertierenden mehrstelligen Gattern verwendet werden.

Min-Terme der DNF = $\Sigma\Pi x_i = \overline{\Pi\overline{\Pi x_i}}$ (nur NAND-Gatter)

Max-Terme der KNF = $\Pi\Sigma\overline{x_i} = \overline{\Sigma\overline{\Sigma\overline{x_i}}}$ (nur NOR-Gatter)

Beispiel 1

Das XOR $x = a \oplus b = a \cdot \overline{b} + \overline{a} \cdot b = \overline{(a\overline{b})} \cdot \overline{(\overline{a}b)}$

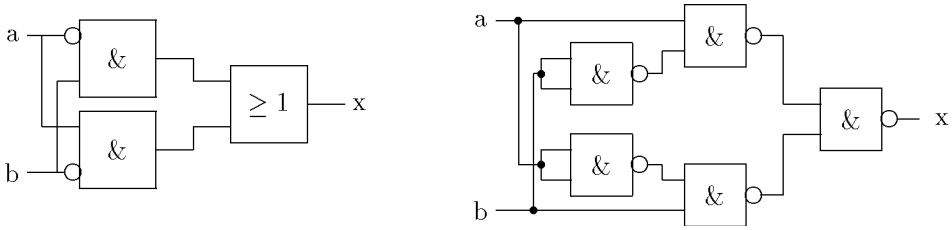


Bild b1p15

Beispiel 2

$x = \bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b} \cdot \bar{c} + a \cdot \bar{b} \cdot c$

$x = \overline{\bar{a} \cdot \bar{b} \cdot c + \bar{a} \cdot b \cdot \bar{c} + a \cdot \bar{b}}$

$x = \overline{\bar{a} \cdot \bar{b} \cdot c} \cdot \overline{\bar{a} \cdot b \cdot \bar{c}} \cdot \overline{a \cdot \bar{b}}$

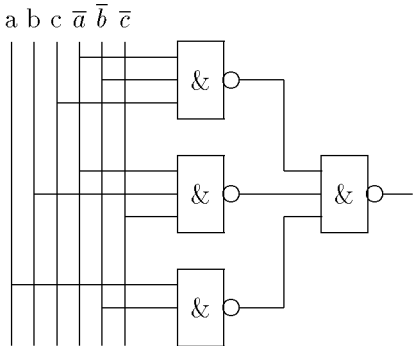


Bild b1p16

1.3.2 Rechnerbausteine

1.3.2.1 Addierer

Halbaddierer a + b = Ergebnis e und Übertrag c

a	b	e	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$e = a \oplus b$

$c = a \cdot b$

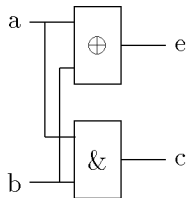


Bild b1p17

Volladdierer $a_i + b_i + c_i$ = Ergebnis e_i und Übertrag c_{i+1}

a_i	b_i	c_i	e_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$e_i = (a_i \oplus b_i) \oplus c_i$

$c_{i+1} = a_i \cdot b_i + c_i \cdot (a_i \oplus b_i)$

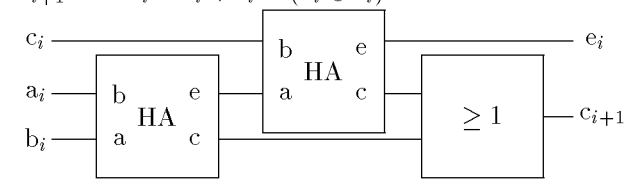
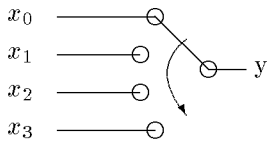


Bild b1p18

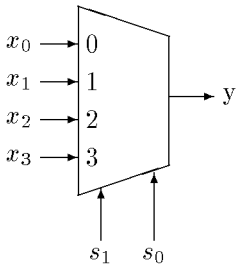
Subtrahierer x = a + (-b) mit -b = $\overline{b} + 1$ (2-er Komplement)

1.3.2.2 Multiplexer



mehrstufiger Umschalter

Bild b1p20



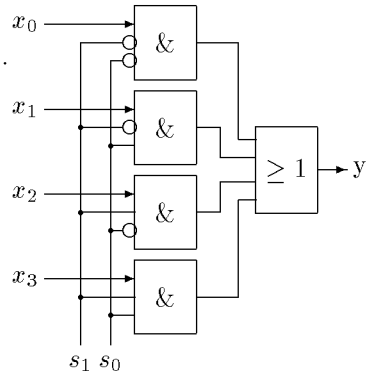
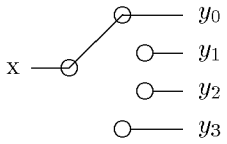


Bild b1p21

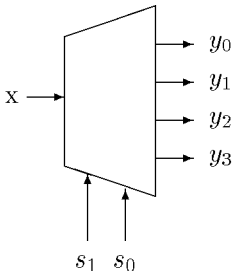
s_1	s_0	y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

Statische Anwendung:
Boole'sche Funktion y = f(s) durch Belegung der Eingänge eines Multiplexers mit festen Werten x_i

Demultiplexer



mehrstufiger Umschalter



s_1	s_0	y_0	y_1	y_2	y_3
0	0	x	0	0	0
0	1	0	x	0	0
1	0	0	0	x	0
1	1	0	0	0	x

Bild b1p20a

1.3.2.3 PROM

Programmable Read-only Memory
Funktionswerte { d_i } = f(a) werden über Adreßwerte { a_i } aus einem Speicher gelesen, der wahlfrei (RAM) adressierbar ist. Die Zuordnung der Funktionswerte zu den Adressen erfolgt durch "Brennen" der zugehörigen Verbindungen.

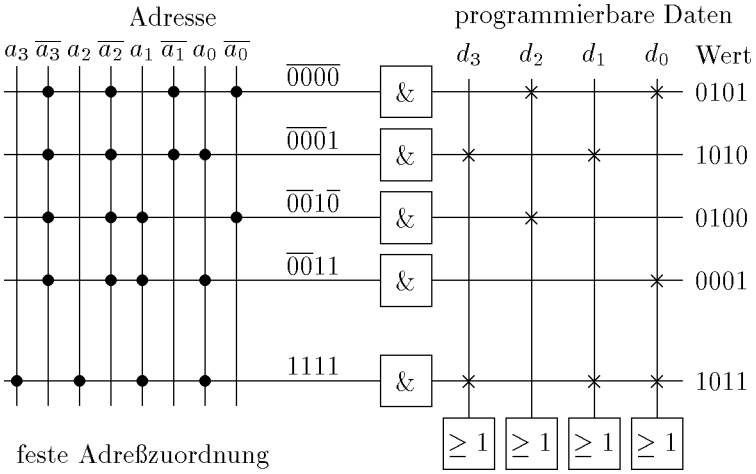


Bild b1p22

1.3.2.4 PAL

Programmable Array Logic
Zum PROM inverse Festlegung von Funktionswerten zu (programmierbaren) Adressen.

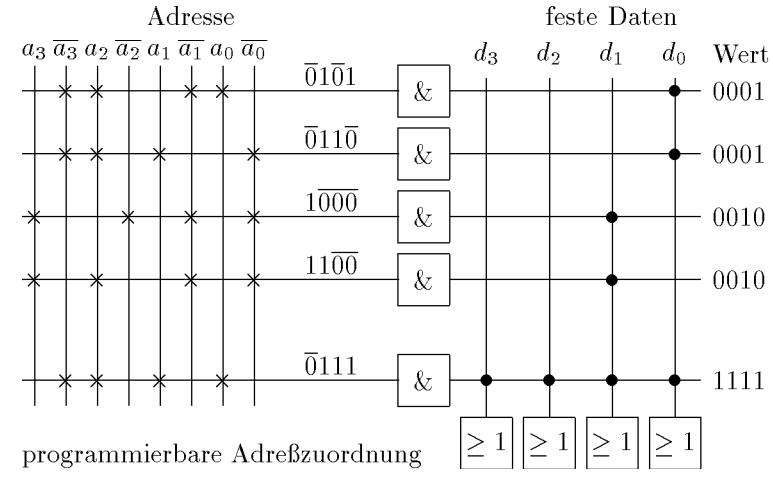


Bild b1p23

1.4 Sequentielle Schaltungen

1.4.1 Zeitverhalten

Gatterlaufzeiten
Schaltverzögerung (delay time t_d)

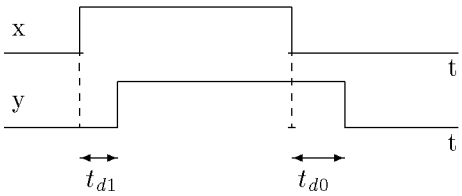
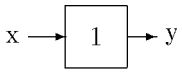
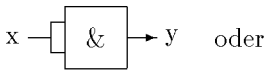


Bild b1p24

Identität



z.B.



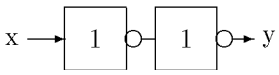


Bild b1p24a

Flankensteilheit Anstiegs- und Abfallzeit (rise time t_r, fall time t_f)

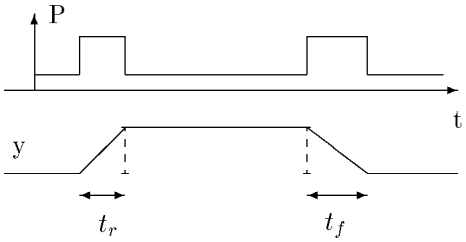


Bild b1p25

in der Regel ist $t_{d0} \approx t_{d0} \approx t_r \approx t_r$
und hängt von der verwendeten Halbleitertechnologie ab.

Technologie	t_d
TTL	5 ns
ECL	1 ns
CMOS	15 ns

Ansprechschwellen (threshold)

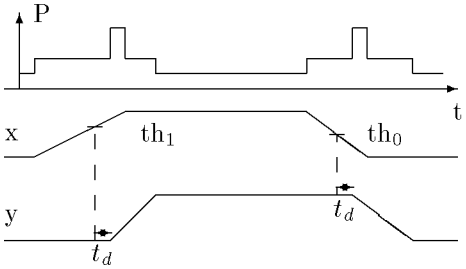


Bild b1p26

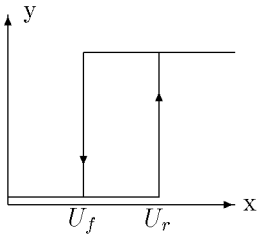


Bild b1p26a

Kennlinie (Schmitt-Trigger: $U_f < U_r$)

1.4.2 Bistabile Schaltungen

Elementare Schaltwerke, Automaten

a) Rückgekoppeltes NAND instabile Kippstufe, Multivibrator

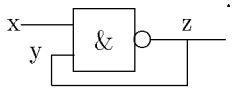


Bild b1p27

x	y	z = $\overline{xy}$
0	0	1
0	1	1
1	0	1 = $\overline{y}$
1	1	0 = $\overline{y}$

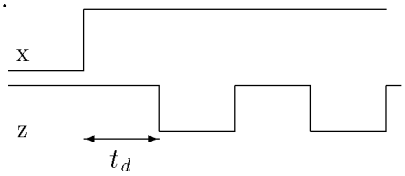


Bild b1p27a

x	y(t)	y(t+t_d)	y(t+2t_d)	
0	0	1	1	y = 1 = const
1	1	0	1	$f_y = 1/2t_d \approx 100MHZ$

b) bistabile Kippstufe

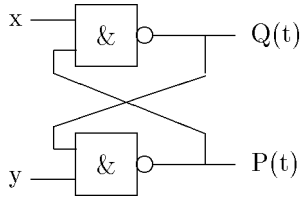
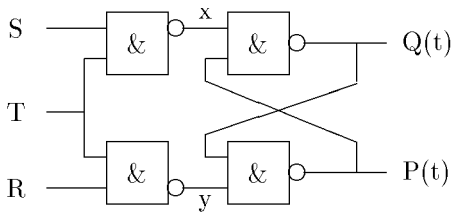
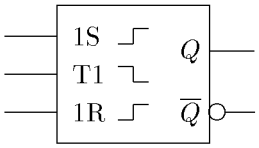


Bild b1p28

x	y	Q(t)	P(t)	Q	P(t+t_d)	Q	P(t+2t_d)	
0	0	1	1	1	1	1	1 = const	$\overline{clear}$
0	1	1	P_t	1	0 = $\overline{Q_t}$	1	0 = const	(set)
1	0	Q_t	1	0	1	0	1 = const	(reset)
1	1	1	1	0	0	1	1 ≠ const	(instabil)
1	1	1	0	1	0	1	0 = const	(bi)stabil
1	1	0	1	0	1	0	1 = const	(bi)stabil
1	1	0	0	1	1	0	0 ≠ const	instabil

c) RS–Flipflop



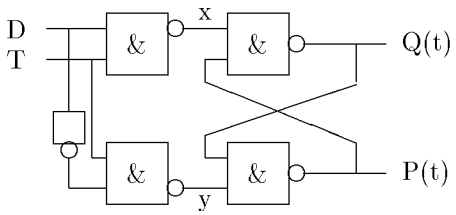


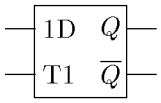
Schaltsymbol

Bild b1p29

R	S	T	x	y	Q	P
0	0	0	1	1	Q_0	$P_0 = \overline{Q_0}$ = const
1	0	0	1	1	Q_0	$P_0 = \overline{Q_0}$
0	1	0	1	1	Q_0	$P_0 = \overline{Q_0}$
1	1	0	1	1	Q_0	$P_0 = \overline{Q_0}$
0	0	1	1	1	Q_0	$P_0 = \overline{Q_0}$ = const
1	0	1	1	0	0	1 (reset)
0	1	1	0	1	1	0 (set)
1	1	1	0	0	1	1 instabil

d) D-Flipflop Speicherelement für 1 Bit
Ausschließen des instabilen Zustands durch D = S = $\bar{R}$





Schaltsymbol

Bild b1p30

D	T	x	y	Q
0	0	1	1	Q_0
1	0	1	1	Q_0
0	1	1	0	0 = D (set)
1	1	0	1	1 = D (reset)

e) JK-Flipflop

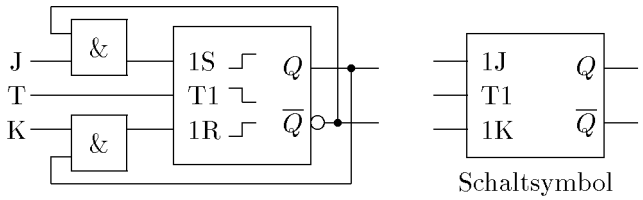


Bild b1p31

J	K	Q_0	S	R	Q(T) für T = 1	
0	0	0	0	0	Q_0	Store
0	0	1	0	0	Q_0	Store
0	1	0	0	0	$0=Q_0$	Reset
0	1	1	0	1	0	Reset
1	0	0	1	0	1	Set
1	0	1	0	0	$1=Q_0$	Set
1	1	0	1	0	$1=\overline{Q_0}$	Kippen
1	1	1	0	1	$0=\overline{Q_0}$	Kippen

f) Master-Slave-Flipflop

Serienschaltung von einem JK– und einem RS–Flipflop.
Ein- und Ausgang werden entkoppelt. Das neue Signal erscheint erst nach Beendigung des Taktimpulses (verzögert), an dessen Endflanke (flankengesteuertes Flipflop).

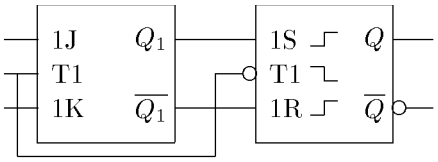


Bild b1p32

J	K	S(T)	Q(T)	Q($\overline{T}$)
0	0	Q_0	Q_0	Q_0
0	1	0	Q_0	0
1	0	1	Q_0	1
1	1	$\overline{Q_0}$	Q_0	$\overline{Q_0}$

g) T-Flipflop

Anwendung eines Master-Slave-Flipflops zum Zählen

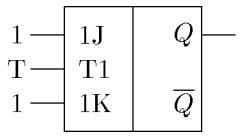


Bild b1p32a

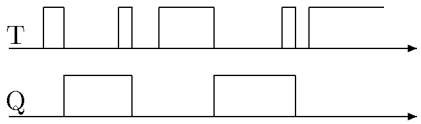


Bild b1p32b

1.4.3 Register

a) Speicherregister (latch) .
Parallelschaltung von n D-Fliflops mit gemeinsamen Ladevorgang (Load)

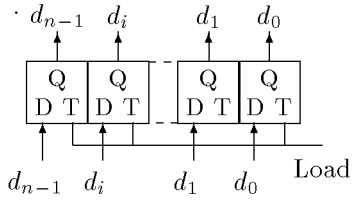


Bild b1p33

b) Zähler (counter) Sequentielle Addition
Serienschaltung von T-Flipflops.
Rücksetzen (reset) durch J = K = 0 beim nächsten Taktimpuls (synchrones Reset).
Die Steuerung für Vorwärts/Rückwärtszählen (F/R) erfolgt durch Nutzung des Q- bzw des $\overline{Q}$-Ausgangs (Die Umschaltung kann durch ein XOR simuliert werden).

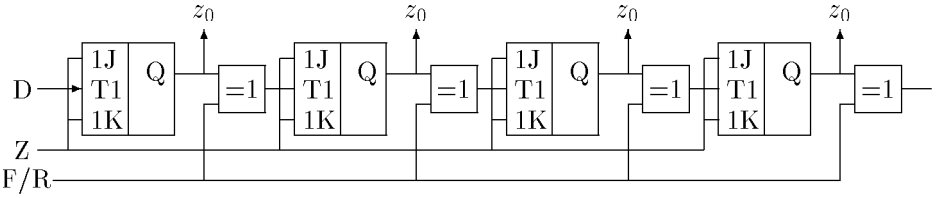


Bild b1p34

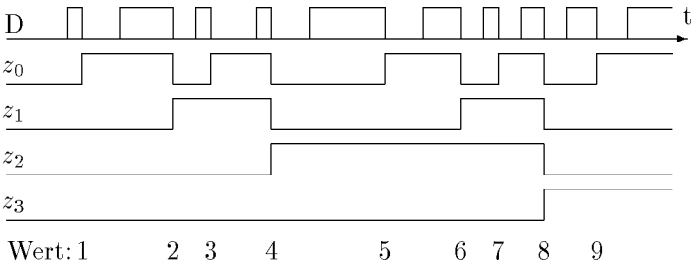


Bild b1p34a

c) Teiler

Serienschaltung von Master-Slave-Flipflops und Rücksetzen nach Erreichen bestimmter Werte (binär, dezimal, beliebig), die durch eine logische Verküpfung der Zählerstandswerte bestimmt wird (R = f(z)). Dieser Übertrag (Carry) kann als Eingangstakt für eine nächste Stufe (Stelle) dienen.

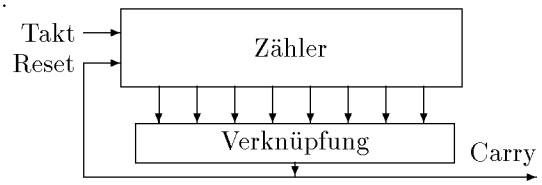


Bild b1p35

Z.B. z = (1001) = 9 ergibt einen Dezimalzähler (0 ... 9).

d) Schieberregister

Serienschaltung von JK-Flipflops mit seriellem Eingang und parallelem Ausgang (Seriell-Parallel-Wandler)

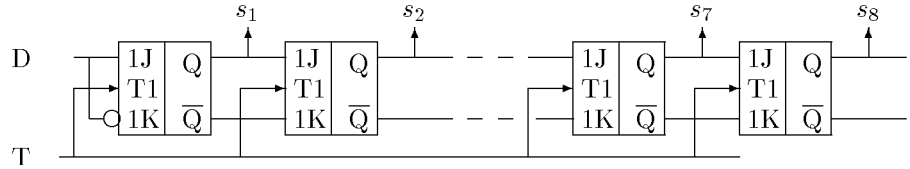


Bild b1p36

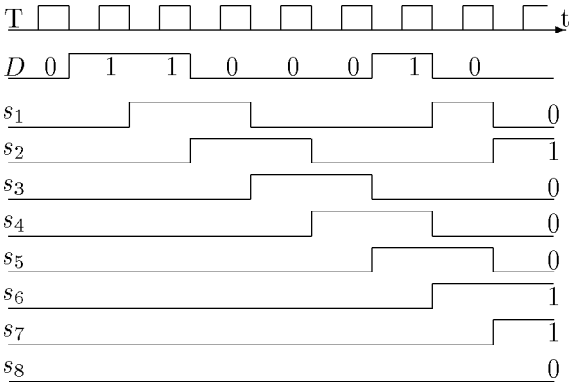


Bild b1p36a

Nach 8 Takten stehen die 8 seriell eingegangenen Bits "0 1 1 0 0 0 1 0" an den Ausgängen $s_8..s_1$ zur Verfügung

e) Parallel-Seriell-Wandler
Schieberegister mit parallem Laden (Load) und seriellem Ausgang

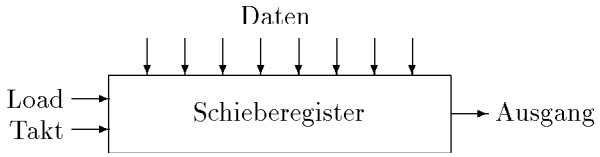


Bild b1p37

f) FIFO (First In First Out)
Registersatz, der von oben geladen und nach unten entleert wird.
Die oben mit einem Ladesignal eingegebenen Daten werden in das unterste noch freie Register gebracht. Wenn das FIFO voll ist, wird ein entsprechendes Signal gegeben.
Mit dem Signal "Hole" werden Daten aus dem untersten Register abgeholt. Die übrigen Daten rücken im FIFO nach. Wenn das FIFO keine Daten mehr enthält, wird das Signal "Leer" abgegeben.
Das FIFO ist ein Speicher mit sequentiellem Zugriff (SAM, sequential access memory).

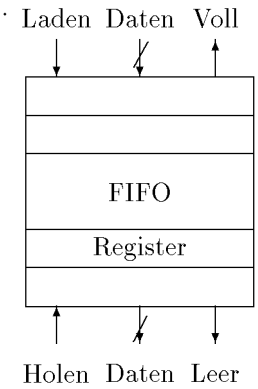


Bild b1p38

Kapitel 2

Systeme, Automaten und Modelle

Kleine Einführung in die Automatentheorie

2.0 Grundbegriffe

a) System (DIN 44 300 T1, DIN 66 201)
Eine Gesamtheit von Objekten, die sich aufgrund ihrer gegenseitigen Beziehungen
– von ihrer Umwelt abhebt,
– davon abgegrenzt erscheint und
– daher ein als Einheit anzusehendes und gegliedertes Ganzes bildet
Die Objekte können sein:
– reale oder abstrakte Gegenstände
– Methoden oder Vorgänge
Aus Systemen können durch
– Unterteilen oder
– Zusammenfügen neue Systeme gebildet werden.
Das bedeutet: Systeme können Objekte in (d.h. Bestandteile von) Systemen sein.

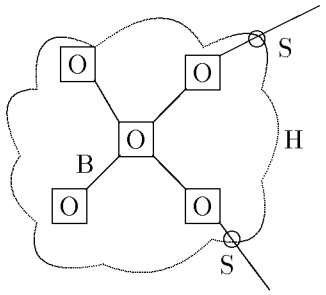


Bild n2p01

..O: Objekte des Systems (objects)
B: Beziehungen zwischen den Objekten (relations, links)
H: Hüllfläche (surface)
S: Schnittstellen zur Umwelt (interfaces)

Beispiele:
1. Haus: Räume (O), Anordnung, Zugänge (B), Dach, Mauern (H), Türen, Versorgung, Entsorgung, Kommunikation (S).
2. Computer: Hardware, Software (O), Signale (B), Gehäuse (H), Netzleitung, Bildschirm, Tastatur (S).

Als Abgrenzung eines Systems gegenüber seiner Umwelt läßt sich eine Hüllfläche denken, die von allen Beziehungen zur Umwelt durchdrungen wird. Die Orte der Durchdringung lassen sich als Schnittstellen auffassen; die dort geltenden Bedingungen beschreiben das Systemverhalten nach außen.

Das Wort "System" sollte stets durch einen Präfix näher bestimmt werden, z.B. Rechnersystem, Versorgungssystem.

Im **ITV** (ISO/IEC 2382-01.01.04A) und im **IEV** (351-01-01) ist "system":
A set of interrelated elements considered in a defined context as a whole and separated from their environment.
NOTE 1: Such elements may be both material objects and modes of thinking as well as the results thereof (e.g. forms of organisation, mathematical methods, and programming languages).
NOTE 2: The system is considered to be separated from the environment and other external systems by an imaginary surface, which can cut the links between them and the considered system.

System (in der deutschen Übersetzung):
System: Gesamtheit untereinander zusammenhängender Elemente, die als Ganzes und von ihrer Umgebung abgegrenzt betrachtet werden, um ein bestimmtes Ziel zu erreichen.
ANMERKUNG 1: Solche Elemente können Gegenstände oder auch Denkweisen und deren Ergebnisse (z. B. Organisationsformen, mathematische Verfahren und Programmiersprachen) sein.
ANMERKUNG 2: Das System wird als von der Umgebung und anderen äußeren Systemen durch eine imaginäre Hüllfläche abgegrenzt gedacht, welche die Verbindungen zwischen diesen Systemen und dem betrachteten System durchschneiden kann.

b) Struktur
Die Menge der Beziehungen zwischen den Teilen eine Ganzen, zwischen den Objekten eines Systems (abstrakt).
Einfachere Definition in ISO/IEC 2382-01.01.3B und IEV 351-01-02 "Struktur":
Gesamtheit der Beziehungen zwischen den Elementen eines Systems.
(engl. structure: The relations among the elements of a system.)
Beispiele:
1. Hierarchie, Baumstruktur
2. Folge, Sequenz
3. Maschennetz
4. Fraktale
5. Abhngigkeit, Versorgung (Erzeuger - Verbraucher)

c) Automat (DIN 19 223)
Ein Automat ist ein künstliches System, das selbstätig ein Programm befolgt. Auf Grund des Programms trifft das System, Entscheidungen, die auf der Verknüpfung von Eingaben mit dem jeweiligen Zustand des Systems beruhen, und Ausgaben zur Folge haben.

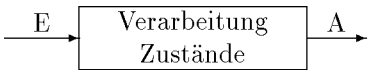


Bild n2p02

EVA-Prinzip:
E: Eingabe
V: Verabeitung
A: Ausgabe

Beispiele:
– inhärente Automaten, z.B. Sicherungsautomat (kein endlicher Automat !)
– festverdrahtete Automaten, z.B. Kaffee-, Zigarettenautomat
– programmierbare Automaten, z.B. SPS (speicherprogrammierbare Steuerung), Computer.
Endliche Automaten haben nur endlich viele Zustände.

d) Zustand (eines Systems):
Die Gesamtheit der aktuellen Werte der veränderlichen Objekte oder Beziehungen eines Systems (Zustandsvektor).
Beispiel: Zigarettenautomat
Schacht: leer/gefüllt (Anzahl der Packungen {0 ... Max})
Geldeinwurf: nichts, zu wenig, ausreichend, zu viel.
Geldvorrat: 0, ...
Bereitschaft: bereit, defekt

2.1 Automaten

Endliche Automaten, Finite State Machines
Sequentielle Maschinen M = { E, Z, A, δ, λ, Z_0 } (eine Menge von Mengen)
E: eine (endliche) Menge von Eingabewerten = Eingabealphabet, -vektor
Z: eine (endliche) Zustandsmenge = Zustandsalphabet, -vektor
A: eine (endliche) Menge von Ausgabewerten = Ausgabealphabet, -vektor
Z_0: Anfangszustand
δ: Zustandsübergangsfunktion, $Z' = Z(T + \delta t) = \delta(Z(t), E(t))$
λ: Ausgabefunktion $A = \lambda(Z, E)$

Schematischer Aufbau:

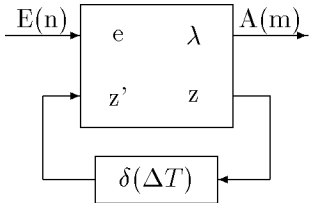


Bild n2p03

Eingabevektor E mit n Komponenten
Ausgabevektor A mit m Komponenten

Rückkopplung mit Verzögerung (ΔT)
– getaktet, synchron: ΔT = 1/ f (Taktfrequenz)
– ungetaktet, asynchron: $\Delta T = \Sigma t_i$ (Laufzeiten)

2.1.1 Moore-Automat

$Z' = \delta(Z, E)$ (immer) . .
$A = \lambda(Z)$
die Ausgabe A hängt nur vom jeweiligen Zustand (Z) ab,
die Ausgabe ist ein stationäres Signal.

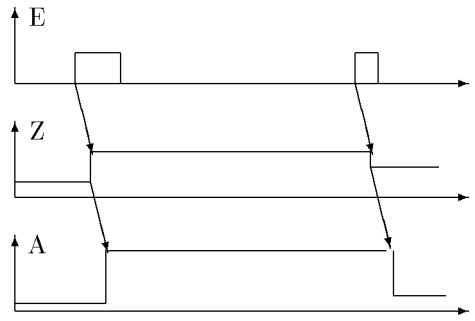


Bild n2p04

Beispiel: Belegung eines Parkhauses
E = Ereignis, ein- oder ausfahrendes Auto
Z = Anzahl der geparkten Autos { 0 .. Max}, δ: Z' = Z $\pm$ 1
A = Anzeige der Anzahl der freien Plätze. { Max .. 0 }. λ: A = Max - Z
Sobald sich die Zahl Z ändert, wird auch A geändert.

2.1.2 Mealy-Automat

$Z' = \delta(Z, E)$ . .
$A = \lambda(Z, E)$
die Ausgabe A hängt von der Eingabe und vom aktuellen Zustand ab.
Die Ausgabe ist ein (transientes) Ereignis, ein Impuls.

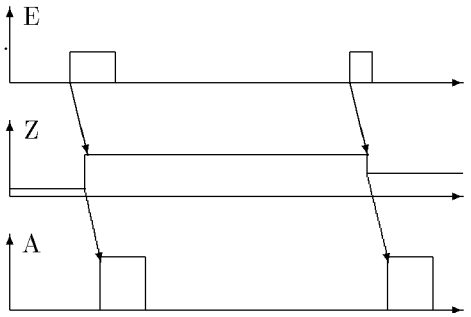


Bild n2p05

Beispiel: Zigarettenautomat
E = Eingabe von Geld,
Z = gespeicherte Zigarettenschachteln Z und eingegebenes Geld G
A = Ausgabe einer Zigarettenschachtel und/oder Geldrückgabe. Nach der Eingabe von ausreichend Geld (λ: 5 DM/Schachtel) erfolgt die Ausgabe als Ereignis, und das System geht in einen neuen Zustand über (δ: Z' = Z - 1, G' = G + x).

2.2 Zustandsdiagramme und -tafeln

Beschreibung von Automaten

2.2.1 Moore-Automat

Zustandstafel

	Eingaben			
Zustand	E_1	E_2	E_n	Ausgabe λ
Z_0	Z_a	Z_b	Z_c	A_1
Z_i		δ: Folgezustände		A_i
Z_m	Z_x	Z_y	Z_z	A_n

Zustandsdiagramm
Knoten (Kreise): Zustände und Ausgaben
Kanten (Pfeile): Eingaben, Ereignisse

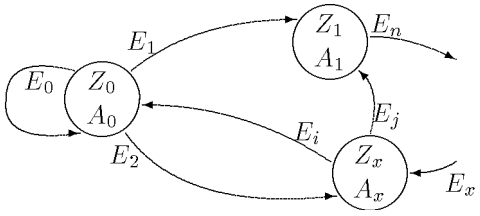


Bild n2p06

Strukturierung
Verfeinerung durch Aufgliedern und Vergröberung durch Zusammenfassen

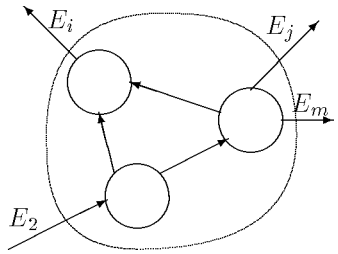


Bild n2p07

Beispiel 1 Das JK-Flipflop

Z = {0, 1} = Q
E={J, K, T}
λ: A=Z
δ: Z'=δ(Z,E) gemäß Wertetabelle

J	K	Q_0	Q(T) für T = 1	
0	0	0	0	} Q_0
0	0	1	1	
0	1	0	0	} Reset
0	1	1	0	
1	0	0	1	} Set
1	0	1	1	
1	1	0	1	} $\overline{Q_0}$
1	1	1	0	

Zustandstafel

	E = (J, K)				
Zustand	(0,0)	(0,1)	(1,0)	(1,1)	Ausgabe
Z_0	Z_0	Z_0	Z_1	Z_1	Q=0
Z_1	Z_1	Z_0	Z_1	Z_0	Q=1

Zustandsdiagramm

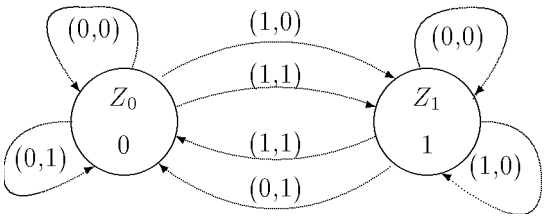


Bild n2p09

2.2.2 Mealy-Automat

Zustandstafel

	Eingaben			
Zustand	E_1	E_2	E_3	E_n
Z_0	Z_a/A_{01}	Z_b/A_{02}	Z_c/A_{03}	Z_d/A_{0n}
Z_i	Z_j/A_{i1}	Z_k/A_{i2}	Z_l/A_{i3}	Z_p/A_{in}
	Folgezustände/Ausgaben			
Z_m	Z_r/A_{m1}	Z_s/A_{m2}	Z_u/A_{m3}	Z_v/A_{mn}

Zustandsdiagramm

Knoten (Kreise): Zustände Z
Kanten (Pfeile): Eingaben und Ausgaben (E/A),
auch die leere Ausgabe (-) ist möglich.

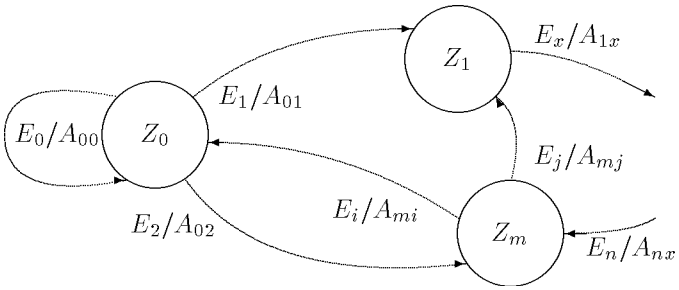


Bild n2p08

Strukturierung

Verfeinerung durch Aufgliedern und Vergröberung durch Zusammenfassen

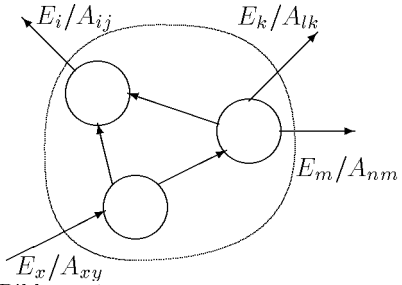


Bild n2p07a

Beispiel 2 Der Kaffee-Automat

Zustände: bereit (Z_0), bereit mit n DPf Guthaben (Z_{nn})
Eingaben: Münzen á 10, 50 Dpf = E(10), E(50) (nicht berücksichtigt ist eine Auswahl)
Ausgabe: nichts (-) oder Kaffee (+ Rückgeld)
λ: Kaffeeausgabe falls mindestens 70 DPf eingeworfen wurden
δ: Z'=δ(Z,E(x)) gemäß Wertetabelle

Zustandsdiagramm

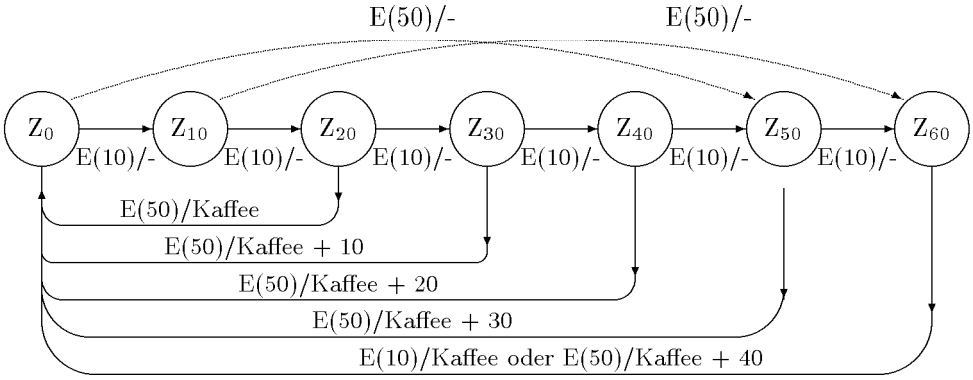


Bild n2p10

Zustandstafel

Zustand	E(10)	E(50)
Z_0	Z_{10}/-	Z_{50}/-
Z_{10}	Z_{20}/-	Z_{60}/-
Z_{20}	Z_{30}/-	Z_0/Kaffee
Z_{30}	Z_{40}/-	Z_0/Kaffee + 10
Z_{40}	Z_{50}/-	Z_0/Kaffee + 20
Z_{50}	Z_{60}/-	Z_0/Kaffee + 30
Z_{60}	Z_0/Kaffee	Z_0/Kaffee + 40

2.3 Netze

Netze aus Instanzen und Kanälen, NIK

Beschreibung statischer und stationärer Strukturen und Systeme (aus DIN 66 200)

Grundelemente

Knoten: 1. Instanzen = Verarbeitungseinheiten mit Entscheidungsbefugnis (Rechtecke)
2. Kanäle = Objekte, die ver- oder bearbeitet werden, z.B. Daten (Kreise)
Kanten: Beziehungen mit Flußrichtung, Verarbeitungsfluß (Pfeile):

Grundstruktur

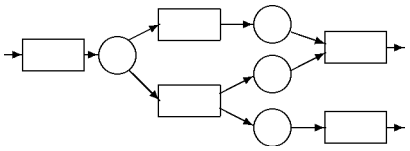


Bild n2p11

Verknüpfungsregel:
Nur unterschiedliche Knoten dürfen durch Pfeile verknüpft werden (EVA-Prinzip).

Verfeinerung / Zusammenfassung
Nur Knoten gleichen Typs dürfen an den Randstellen einer neuen Einheit sitzen. Sie legen damit den Typ des neuen Knotens fest.

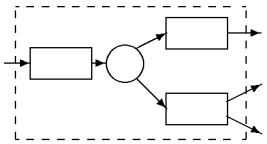


Bild n2p12

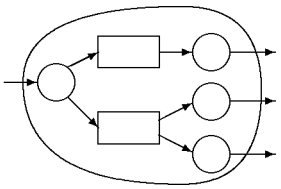


Bild n2p12a

Beispiel 1: Programmerstellung

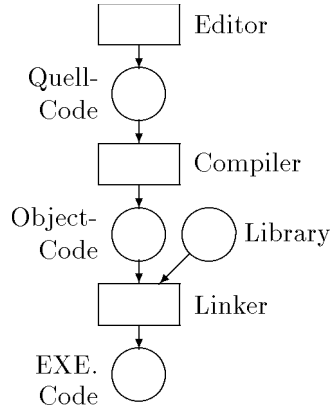


Bild n2p13

Beispiel 2: Sequentieller Addierer

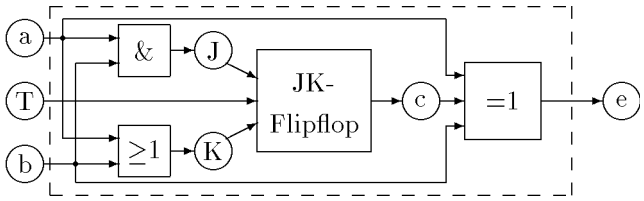


Bild n2p13a

Man überlege sich anhand eines Impulsdiagramms die Wirkungsweise dieses Addierers.

2.4 Petri-Netze

2.4.1 Statische Petri-Netze

Variante der Netze aus Instanzen und Kanäle (NIK)

Anwendung: Beschreibung von Beziehungen in (abgeschlossenen) Systemen, d.h. zur Strukturbeschreibung von Systemen.

Strukturelemente:

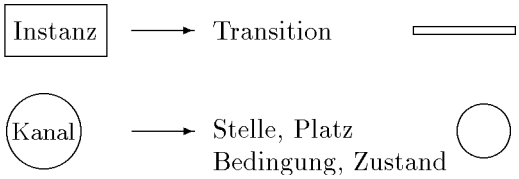


Bild n2p14

Grundstruktur:

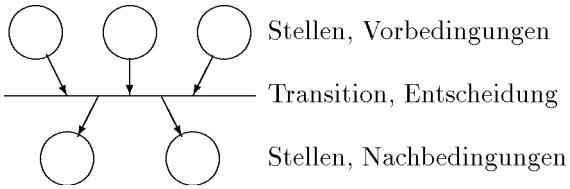


Bild n2p14a

Strukturvarianten:

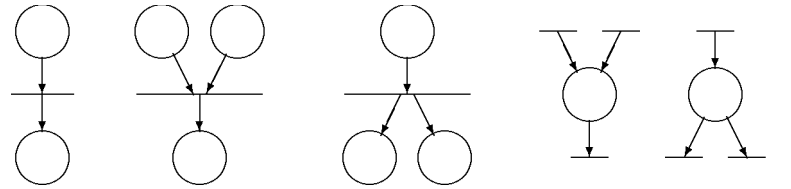


Bild n2p15

Strukturierung (Verfeinerung / Zusammenfassung):

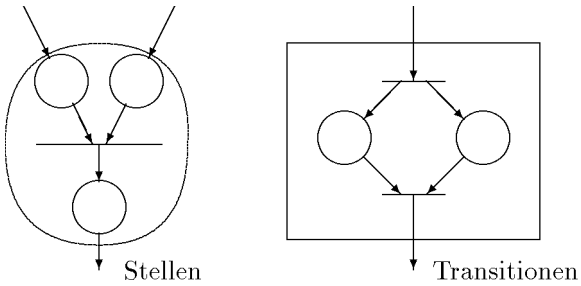


Bild n2p16

2.4.2 markierte Petri-Netze

Anwendung: Beschreibung der Dynamik, d.h. der zeitlichen Kopplungen in Systemen.

a) Beschreibung von Zuständen

Zustände von Stellen werden durch Marken (gefüllte Kreise) beschrieben.

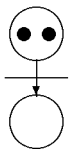


Bild n2p17

b) Beschreibung von Zustandsänderungen

(transiente Vorgänge, deren Zeitdauer unbeobachtbar kurz ist)

Regeln:

- Alle Vorbedingungen einer Transition müssen erfüllt sein, d.h. alle Eingangsstellen müssen jeweils mindestens eine Marke enthalten (Anfangszustand).
- Ereignis (Verarbeitung): Die Transition schaltet und
  - verbraucht je eine Marke aus jeder Eingangsstelle,
  - erzeugt je eine Marke in jeder Ausgangsstelle (Nachbedingung)

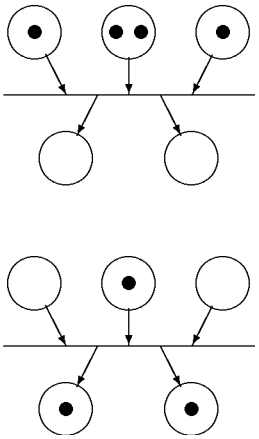


Bild n2p18

c) Beschreibung von Systemverhalten

Lebendigkeit (Liveness): jede Transition wird mindestens einmal aktiviert.

Eindeutigkeit (Determiniertheit): Jeder Gesamtzustand (Verteilung der Marken auf die Stellen) hat einen eindeutig definierten Folgezustand.

Gegenbeispiele:

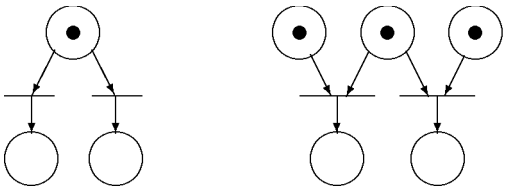


Bild n2p19

Die Marke (in der mittleren Stelle) kann nur für eine der beiden Transitionen genutzt (verbraucht) werden. Eine Vorschrift, für welche, liegt hier nicht vor. Ein Lösung kann in unterschiedlicher Bewichtung der einzelnen Kanten (Pfeile) erfolgen (bewichtete Petrie-Netze), durch die unterschiedliche Prioritäten vergeben werden.

Verklemmung (deadlock): tritt ein, falls im Lauf der Zeit auf, wenn die Anzahl der Marken so abnimmt, daß keine Transition mehr schalten kann.

Beispiel:

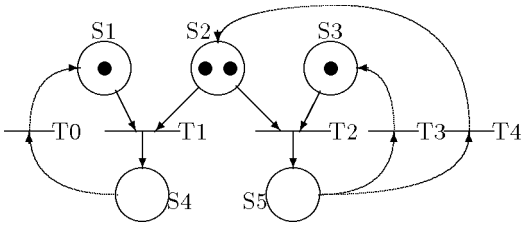


Bild n2p20

Nach dem ersten Durchlauf (Zünden der Transitionen T1 und T2) sind nur noch 2 Marken übrig. Die linke wandert aus S4 wieder in ihre Startposition in S1, für die rechte (aus S5) kann entweder nach S2 oder nach S3 wandern. Im ersten Fall kann die Transition T1 noch einmal zünden, die T2 nie wieder. Im anderen Fall wird S3 besetzt und keine der Transitionen kann zünden, da S2 leer bleibt.

Überlauf (overflow): wenn sich im Lauf der Zeit die Zahl der Marken in einer Stelle stark vermehrt, kann die Kapazität dieser Stelle überschritten werden (wie der Überlauf in einem Datenspeicher - "Festplatte voll!!") und das System geht in eine Verklemmung, die sich aber unter Umständen nach einiger Zeit wieder auflösen kann.

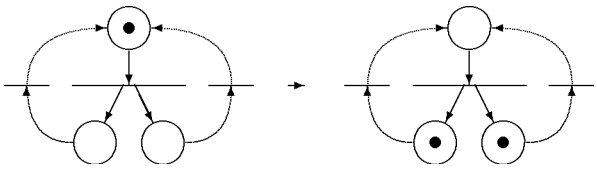


Bild n2p21

2.4.3 Formale Beschreibung von Petri-Netzen

Statische Beschreibung

Alle Transitionen können durch die eingehenden und die ausgehenden Stellen in Form einer Tabelle beschrieben werden. Z.B. (p20)

Transition	Eingangsstellen		Ausgangsstellen
T0	S4		S1
T1	S1	S2	S4
T2	S2	S3	S5
T3	S5		S3
T4	S5		S2

Eine äquivalente Beschreibung der Stellen, mit anliefernden und abnehmenden Transitionen ist ebenso möglich.

Dynamische Beschreibung

Die dynamische Beschreibung soll hier für getaktete Systeme gezeigt werden, wo zu jedem Zeitintervall ein bestimmter Gesamtzustand, d.h. eine bestimmte Verteilung der Marken auf die Stellen gegeben ist. Die dann schaltenden Transitionen sind im rechten Teil durch 'x' markiert. Bei Mehrdeutigkeiten müssen ab dem entsprechenden Zustand 2 oder mehr Folgetabellen geführt werden.

Als Beispiel wird die Verklemmung des vorhergehenden Abschnitts (Bild 20) beschrieben.

Zustand	Stellen					Transitionen					
Z	S1	S2	S3	S4	S5	T0	T1	T2	T3	T4	
Z0	1	2	1	0	0	-	x	x	-	-	
Z1	0	0	0	1	1	x	-	-	x	-	1. Alternative
Z2	1	0	1	0	0	-	-	-	-	-	deadlock
Z1a	0	0	0	1	1	x	-	-	-	x	2. Alternative
Z2a	1	1	0	0	0	-	x	-	-	-	
Z3a	0	0	0	0	0	-	-	-	-	-	deadlock

2.4.4 Anwendungen von Petri-Netzen

Semaphore Nachrichtenstelle zum gegenseitigen. Ausschluß (mutual exclusion) bei der Belegung von Betriebsmitteln, z.B. Drucker in einem Netzwerk (Sperrsynchronisation).
Konkurrierende Prozesse P1 und P2 können quasi gleichzeitig eine Druckanforderung erzeugen (S1 und S2). Da aber nur ein Drucker vorhanden (nur eine Marke in S0) ist, kann nur ein Auftrag bearbeitet werden, nur eine Transition (T1) wird aktiv. Nach Durchlaufen des Druckprozesses (S3) fährt der Prozeß P1 mit der Transition T3 fort, geht in einen neuen Abschnitt S5 und gibt den Drucker frei, indem eine neue Marke in S0 erzeugt wird, mit der Prozeß P2 weitermachen kann.

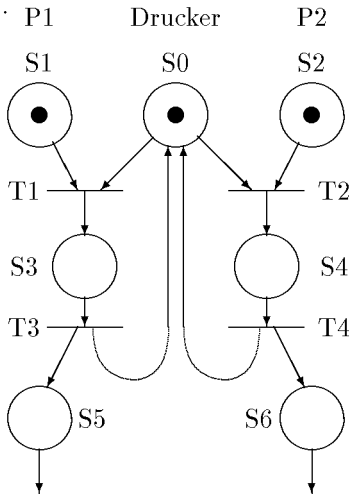


Bild n2p22

Z	S0	S1	S2	S3	S4	S5	S6	T1	T2	T3	T4	
Z0	1	1	1	0	0	0	0	x	-	-	-	1. Prozess druckt
Z1	0	0	1	1	0	0	0	-	-	x	-	Drucken beendet
Z2	1	0	1	0	0	0	0	-	x	-	-	2. Prozess druckt
Z3	0	0	0	0	1	0	0	-	-	-	x	Drucken beendet
Z4	1	0	0	0	0	0	1	-	-	-	-	Drucker bereit

FIFO

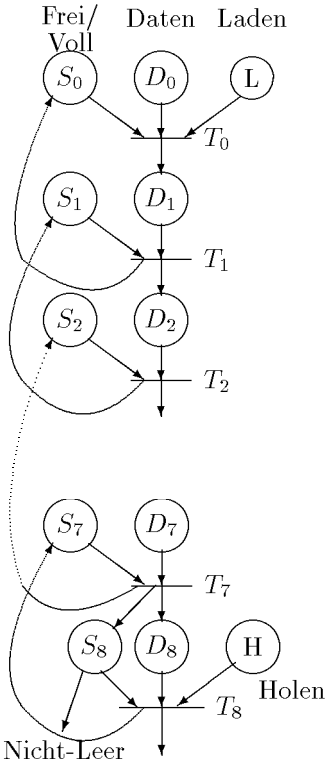


Bild n2p23

. .Die Erstellung der statischen und der dynamischen Tabellen soll dem Leser überlassen bleiben.

2.4.5 Varianten von Petri-Netzen

a) gewichtetet Petri-Netze
Bei dieser Variante werden den einzelnen Zuführungen von Transitionen zu Stellen unterschiedliche Gewichte zugeordnet, damit können Konkurrenzsituationen besser aufgelöst werden. (vgl. w.o.)

b) farbige Petri-Netze
Bei dieser Variante können den Marken unterschiedliche Farben zugeordnet werden, derart daß für die Aktivierung einer Transition stets Marken gleicher Farbe notwendig sind.

c) Petri-Netze mit Nachbedingungen
Bei dieser Variante müssen für die von einer bestimmtem Transition erzeugten Marken bestimmte Bedingungen erfüllt sein, so z.B. daß die Ausgangsstellen leer sind.

Kapitel 3

Rechnerarchitekturen

3.0 Der Rechner als System

Grobstrukturen:
- Hardware: Zentraleinheit und Peripherie
- Software: Programme und Daten

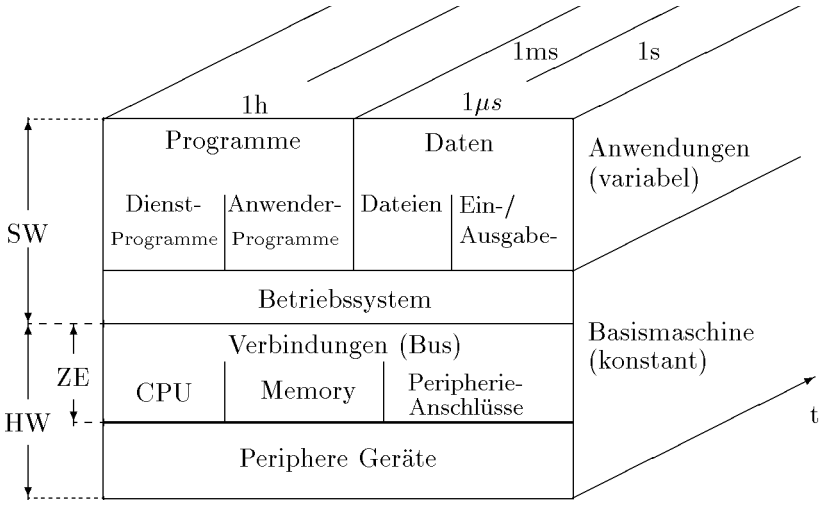


Bild n3p00

Das Bindeglied zwischen Hard- und Software ist die CPU (Central Processing Unit, der Zentralprozessor). Deren Leitwerk liest die Instruktionen aus dem Arbeitsspeicher, interpretiert sie und führt sie aus. Dabei steuert sie das interne Rechenwerk, den CPU-internen Datenfluß und den gesamten Datenverkehrüber über das Verbindungsnetzwerk (meist einen Bus).

Betriebssysteme und Betriebsarten:

1.	Single User z.B. MS-DOS ohne Datenschutz	Multi User Unix mit Datenschutz: (Benutzerkennung, Passwort)
2.	Single Tasking ein einziges (komplexes) Programm in Betrieb nur Spezialrechner	Multi Tasking Mehrprogrammbetrieb mehrere Programme in Konkurrenz um 1 oder n CPU (concurrency) Zuteilungsstrategien (sheduling): – time-sharing – resource-sharing (MS-DOS), Unix
3.	Single Processing Ein Prozessor (1 CPU) z.B. Prozeßrechner in Geräten	Multiprocessing Mehrere Prozessoren ($n \leq 2^{16} = 65\,536$) Massive Parallel Computing z.B. Hypercube, Suprenum

Der Rechner als endlicher Automat:

- Eingabealphabet: Binärwerte
- Ausgabealphabet: Binärwerte
- Zustandsalphabet: $Z = \{Z_0 ... Z_z\}$

Bei einem Binärrechner ist $z = 2^x$, wobei x die Zahl aller Binärvariablen (Bits) ist.
Abschätzung: x = c + m + p

c: Bits innerhalb der CPU ca. 20 Register à 16 bit:

m: Kapazität des Arbeitsspeichers, typ. 1 MB = 1 000 000 * 8 bit

p: Kapazität der Peripherie, einschließlich Bildschirm und Tastatur.

Einheit	Menge	Zugriffszeit
CPU	c = 1024 (32 Register à 32 Bit)	1 - 20 ns
Arbeitsspeicher	m = $16 * 10^6$ (20 MB)	1 μs
Peripherie:		
Tastatur	4 * 102 Tasten	1 s
Bildschirm	1 Mio Pixel	10 ms (100 Hz)
Festplatte	$p > 8 * 10^9$ (1 GB)	10 ms

Damit wird x im wesentlichen durch p (die Speicherkapazität der Peripherie) bestimmt; im folgenden wird mit $x \approx p \approx 10 \cdot 10^9$ gerechnet.

Abschätzung: $z = 2^x = 2^{10 \cdot 10^9} = (2^{10})^{10^9} \approx (10^3)^{10^9} = 10^{3 \cdot 10^9}$

Das ist eine Zahl mit 3 Milliarden Nullen. Es ist also eine sehr große Zahl, die aber endlich ist !

Man überlege sich die Zeit für das Ausdrucken einer so großen Zahl.

3.1 Rechnerkonfiguration

Rechner := Zentraleinheit + Peripherie

Zentraleinheit: Rechnerkern

Peripherie:
- Datenendgeräte (Terminals): Datensichtgerät (Monitor + Tastatur), Drucker, Plotter,
- Massenspeicher (Hintergrundspeicher): Magnetplatte (auch Floppydisk), Magnetband (auch Cassetten),
- Kommunikationsperipherie: Modem, Ethernetanschluß o.ä.
- Prozeßperipherie, insbesondere A/D- und D/A-Wandler

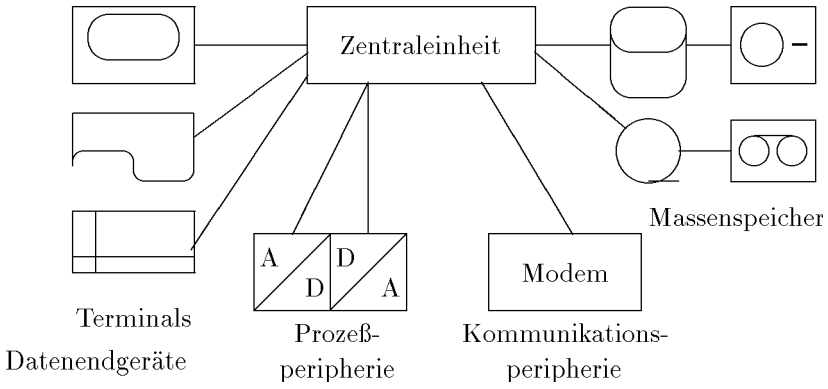


Bild n3p01

Begriffsbildungen:
Betrachtungseinheiten

Baueinheiten /	Beispiel	Funktionseinheiten /	Beispiel
Baueinheit:	Haus	Funktionseinheit:	Wohnhaus
Baugruppe:	Dach, Raum, Tür	Funktionsgruppen:	Küche, Zugänge Sitzgruppe
Bauelement:	Ziegel, Balken, Nagel	Funktionselement:	Möbel, Haken

Computer:

Der sog. Rechnerkern umfaßt als Baueinheit oft mehrere Funktionseinheiten: Die Zentraleinheit, Massenspeicher, Stromversorgung.

Eine Funktionseinheit, wie z.B. der Arbeitsspeicher, kann auf mehrere Baueinheiten (Speicherwerke) verteilt sein.

3.2 Die Zentraleinheit

Definition: Alle Funktionseinheiten, die vom Zentralprozessor (CPU) direkt erreicht (adressiert) werden können.

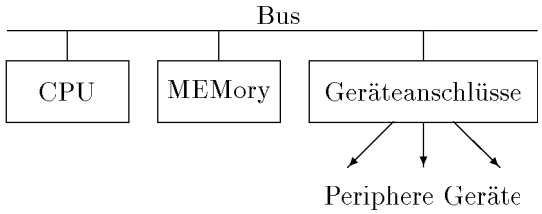


Bild n3p02

Grundfunktionen

- Die **CPU** ist der eigentliche (einzige) aktive Kern des Rechners
  - das **Steuerwerk** liest Instruktionen aus dem MEMory
  - das **Leitwerk** interpretiert sie und steuert Rechenwerk
  - das **Steuerwerk** liest Daten aus dem MEMory oder vom GA
  - das **Rechenwerk** verarbeitet die Daten
  - das **Steuerwerk** schreibt Ergebnisse ins MEMory oder zum GA
  - das **Unterbrechungswerk** verwaltet Interrupts
- Der **Arbeitsspeicher** (MEMory) enthält **Daten** und **Instruktionen** als **Bitmuster** in **adressierbaren Speicherworten** (der Breite n Bit).
- Die Geräteanschlüsse sorgen für die Kommunikation mit den peripheren Geräten; sie transferieren Daten von und zu Geräten
  sie steuern den Transfer und die Geräte (oft enthalten sie eigene Prozessoren).
- Der Bus verbindet alle Funktionseinheiten, er überträgt **Daten**, **Adressen** und **Steuersignale**. (es können auch andere Topologien eingesetzt werde, z.B. Ring oder Stern)

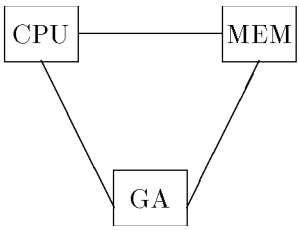
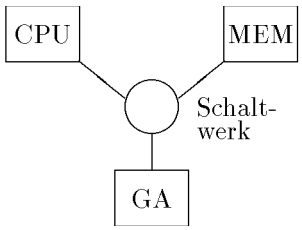


Bild n3p02a



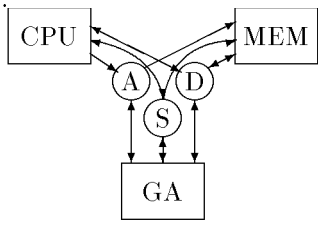


Bild n3p02b

Die Zentraleinheit als Netz aus Instanzen und Kanälen

3.2.1 Die CPU

- Steuerwerk: koordiniert die Zugriffe auf den Bus und den Datentransfer.
- Leitwerk: interpretiert Instruktionen anhand eines Mikroprogramms, das in einem ROM gespeichert ist.
- Rechenwerk: führt Operationen aus, meist nur auf ganze Zahlen (Integer)
  Der Kern ist die ALU (Arithmetic and Logic Unit)
- Coprozessoren für Gleitpunktoperationen (Floating Point Processor), Textoperationen (Byte Processor)
- Register enthalten die wichtigsten Daten
- Der Zugang zum Bus ist meist gepuffert, d.h. hier werden Zwischenspeicher für Adressen und Daten eingesetzt, die einen größeren Umfang annehmen können und dann als Cache bezeichnet werden.

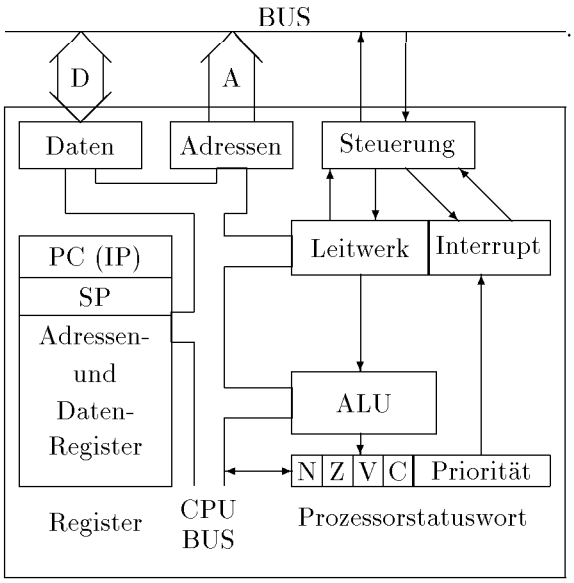


Bild n3p03

Die wichtigsten Register der CPU

PC (IP): Der Program Counter (Instruction Pointer) zeigt auf die nächste zu bearbeitende Instruktion

SP: Der Stack Pointer (Kellerzeiger) zeigt auf die zuletzt auf dem Stack benutzte Adresse (LIFO, Last In First Out)

Die Flags im Prozessorstatuswort

N (negativ): Resultat war negativ
Z (zero): Resultat war Null
V (oVerflow): Arithmetischer Überlauf
C (carry): Übertrag

- Das Unterbrechungswerk (Interrupt Arbitrator, IA) bearbeitet externe Unterbrechungsanforderungen, indem es geeignet Hilfsprogramme (Interrupt Service Routinen) aufruft, die vom Betriebssystem zur Verfügungs gestellt werden müssen.

Leistungsmerkmale

- Verarbeitungsbreite, Wortlänge (n Bit, n = 4, 8, 16, 32, 48, 64)
- Taktfrequenz (1 ... 500 MHz)
- Instruktionssatz und dessen Mächtigkeit
  - CISC: Complex Instruction Set Computer
  - RISC: Reduced Instruction Set Computer
- Rechenleistung wird meist in MIPS (Millionen Operationen pro Sekunde), oder MegaFLOPS (Millionen Floating Point Operationen pro Sekunde) angegeben. Realistischere Angaben erhält man über Instruktionsmischungen, wie sie in realen Programmen vorkommen.

3.2.1.1 Das Mikroprogramm

Jede Instruktion, die aus dem Arbeitsspeicher gelesen wird, wird im Leitwerk anhand eines Mikroprogramms interpretiert. Das Mikroprogramm ist festverdrahtet, bzw. in einem ROM gespeichert. (Darstellung als Jackson-Struktogramme).

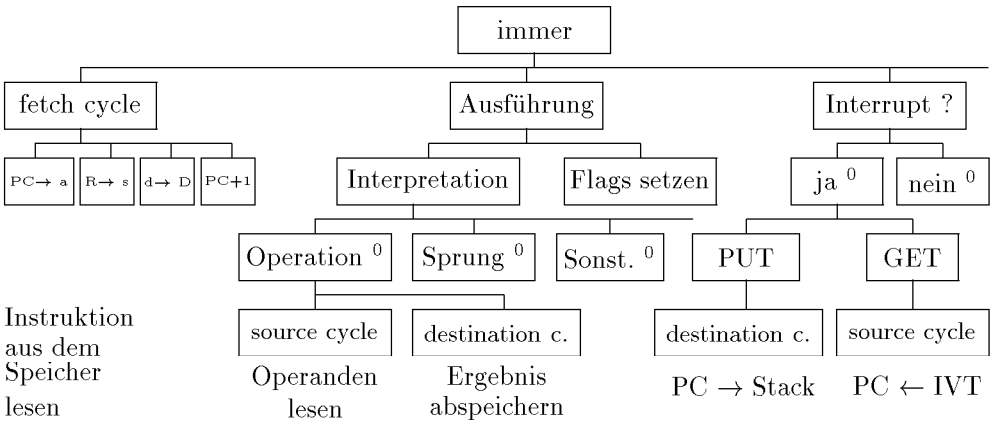


Bild n3p04

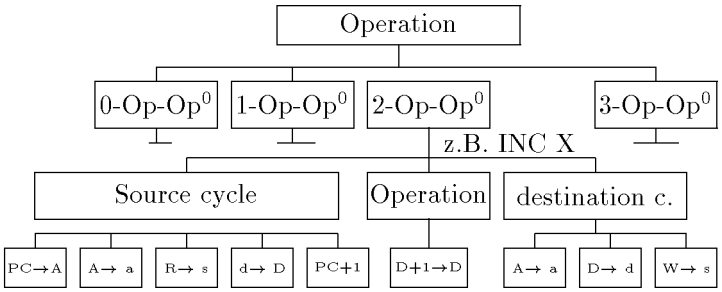


Bild n3p04b

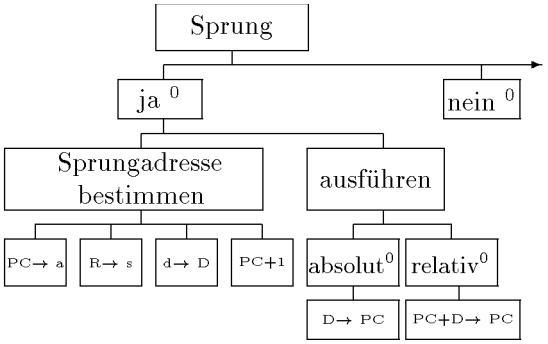


Bild n3p04a

3.2.1.2 Instruktionen und Adressierungen

Instruktionssatz oder **Befehlsvorrat:**
Menge der zulässigen Instruktionen (engl.: **instruction set**).
Jede Instruktion besteht im Prinzip aus zwei Teilen,

Op-code	Adreßteil

- Der **Instruktionsteil, Operationscode, (Op-code)**
legt die Art der Instruktion fest und damit auch die Zahl der Operanden,
- Der **Adreßteil** beschreibt den Zugriff auf Operanden oder Adressen.
In der Regel steht hier nur der **Adressierungsmodus**, ein Verweis dafür, wie Operand oder Adresse gefunden werden kann.

Operationscode und Adreßteil können auch bitweise gemischt sein; das Steuerwerk der CPU muß nur wissen, was wo steht, und dementsprechend handeln.

In manchen Fällen werden Operanden oder Adressen indirekt durch den **Operationscode** festgelegt, z.B. INC X (erhöhe X um 1); LDA X (lade den AKU mit Operand X).

Instruktionsgruppen
- **Steuerungsinstruktionen** wie HALT, RESET, WAIT, NOP (0 Operanden)
- **Verarbeitungsoperationen** wirken auf einen oder mehrere Operanden.
  - 1-Operanden - Operationen, wie CLEAR X, TEST X
  - 2-Operanden - Operationen, wie COMP A,B (compare), MOV A,B (move)
    (auf 1 Operand) reduzierbare wie NEG X, ABS X, DEC X, INC X, ROTR/L X
  - 3-Operanden - Operationen, wie ADD, SUB, MUL A, B, C (A + B = C)
  - 4-Operanden - Operationen, wie DIV A, B, C, D (A/B = C rest D)
- **Verzweigungen:**
  - unbedingte und bedingte **Sprünge** mit **Sprungbedingung** über die **Status Flags** wie BEQ a (branch on equal) nach a, wenn das Z-bit gesetzt ist,
  - **Unterprogrammsprünge** (meistens ohne Bedingung) (CALL subroutine). **Rücksprungadresse** auf dem **Stack**. Beim **Rücksprung** (RETURN) wird die Rücksprungadresse dort wieder abgeholt.

Komplikation bei der **Multiplikation**: Das Produkt aus 2 Zahlen mit je n bit ergibt ein Ergebnis mit 2n bit, und kann nur in 2 Speicherworten untergebracht werden; z.B. in doppelt langen AKUs, oder im Speicher in zwei aufeinanderfolgenden Worte (Double). Entsprechende Komplikationen treten bei der **Division** auf.
Das **Rechenwerk**

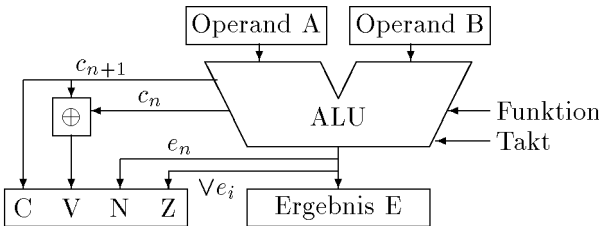


Bild n3p04d

Adressierung
Ein Operand kann in einem internen **Daten-Register** der CPU stehen, oder im Speicher, aus dem er erst noch gelesen werden muß.

Bei der **Speicheradressierung** wird stets ein internes Register der CPU verwendet, entweder der **Program Counter**, ein **Adreßregister** oder der **Stack Pointer**

Adressierungsmodi (bestimmte Bitkombinationen) bestimmen, wie diese Register zu verwenden sind:

reg	mode

Dabei benötigt man r Bit (3...5) für die Auswahl des Registers und m Bit (3...5) für den Modus.

Folgende Adreßmodi (mode = 0 ... $2^m - 1$) sind denkbar:

a) Alle Register (reg = 0 ... $2^r - 1$)
0: direkt: die Daten befinden sich im Register.
1: absolut: das **Register (Adreß-** oder **Indexregister)** enthält die absolute Adresse des Operanden,
2: indirekt: das Adreßregister enthält die Adresse der Adresse des Operanden,
3: indiziert: der Inhalt des Adreßregisters addiert zum Inhalt des unmittelbar folgenden Wortes ergibt die Adresse des Operanden,
4: indirekt indiziert: ergibt wie oben die Adresse der Adresse des Operanden.
5: auto-increment (absolut oder indirekt): mit einer automatischen Erhöhung des Adreßregisters nach dessen Benutzung,
6: auto-decrement (absolut oder indirekt): mit einer automatischen Erniedrigung des Adreßregisters vor dessen Benutzung.
7: relativ: der aktuelle Wert des PC zuzüglich einer Verschiebung (offset) aus einem Register oder aus der nachfolgenden Speicherzelle.

b) Adressierungen, die den PC benutzen (reg = 7) müssen dafür sorgen, daß der PC nach Benutzung erhöht wird, um auf das nächstfolgende Speicherwort zu zeigen. Manche CPU macht dies automatisch nach jeder PC-Benutzung, dann machen die Modi 5 und 6 keinen Sinn. Oder es kann praktisch nur der Modus 5 (auto- increment) benutzt werden.

Beim Modus 0 wird in jedem Fall durch eine Veränderung des PC ein Sprung durchgeführt.

c) Die Adressierung mit dem **Stack Pointer** (reg = 6) erfolgt meist implizit, bei bestimmten Instruktionen.

Bei einem **Unterprogrammsprung** wird der aktuelle **PC** auf dem **Stack** abgelegt. Dieser PC-Wert zeigt schon auf die nächste Instruktion, die aber erst nach Rückkehr bearbeitet werden soll; er stellt also die **Rücksprungadresse** dar, die beim **Rücksprung** dort wieder abgeholt wird.

Der **Stack Pointer** wird nach Ablegen eines Wertes (PUSH) jeweils anschließend automatisch erhöht (auto-increment), um auf einen freien Platz zu zeigen; beim **Abholen** (POP) wird er erniedrigt, um den Platz wieder als frei zu kennzeichnen. Da der **Stack** bei den meisten Rechnern ein beliebiger Bereich im Arbeitsspeicher sein kann, muß der Stack Pointer von der Software vor der ersten Benutzung einen Wert erhalten, und während der Arbeit sollte er immer überprüft werden, ob er den vorgesehen Bereich nicht überschreitet (engl.: **stack overflow**) und andere Daten zerstört.

Instruktionsgröße
die Anzahl der Speicherworte, die eine Instruktion belegt, hängt von der Instruktion, den Operanden und deren Adressierungsmodi ab.
So findet man bei einem 2-Adreßrechner folgende Konstellationen:

1 Wort bei **Steuerungsinstruktionen** (0-Operanden) und bei allen Operationen, bei denen alle Operanden oder Adressen schon in der CPU vorhanden sind:

Steuerung

oder

Op-code	Adr.

2 Worte bei **Einadreßinstruktionen** mit dem **Operanden** oder dessen (absoluter, relativer, indirekter oder indizierter) **Adresse** im unmittelbar darauf folgenden Wort:

Op-code	Adr.
Operand/Adresse	

3 Worte bei **2-Adreßinstruktionen** wenn weder Operanden noch Adressen in der CPU vorliegen.

Op-code	Adr.	Adr.
1. Operand/Adresse		
2. Operand/Adresse		

1-Adreß-Rechnern können alle 0- und 1-Operanden-Operationen auch auf Operanden im Speicher ausführen. Für 2-Operanden - Operationen (und mehr) benutzen sie meist ein internes Arbeitsregister, den AKU.

Bei einigen 2-Operanden - Operationen wie NEG, ABS, etc. wird hier der Ausgabeoperand, das Ergebnis, auf den Platz des Eingabeoperanden, des Quelloperanden geschrieben und dieser geht dabei verloren.

Bei den anderen 2-Operanden - Operationen wie MOV, COMP, muß einer der Operanden zuerst in ein besonderes Register der CPU, in der Regel in den AKU gebracht werden.
Dann wird aus einer Instruktion, wie . .eine Reihe von einfachen Instruktionen, wie

MOV A, B

LDA A (lade den AKU von A) und
STA B (speichere AKU nach B)

Bei 3-Operanden - Operationen (ADD, SUB, MULT) muß dann einer der Quelloperanden schon im AKU stehen, während der zweite vom Speicher eingelesen werden kann. Das Ergebnis wird dann auf einen der Quelloperanden (meist in den AKU) geschrieben, so daß dieser Wert verloren geht.

Bei 4-Operanden-Operationen (z.B. **DIV**) könnten dann im Prinzip beide Quelloperanden durch die Ergebnisse überschrieben werden; tatsächlich verwendet man aber hier meist 2 verschiedene 3-Operanden-Operationen, z.B. DIV und MOD, die jeweils nur den Quotienten oder den Rest liefern.

2-Adreß-Rechnern müssen solche Tricks eigentlich erst ab 3 Operanden anwenden, jedoch werden auch hier meistens solche Operationen wie NEG, ABS, etc. wie bei 1-Adreß-Rechnern als 1-Operanden - Operationen behandelt, bei denen der Originalwert verloren geht.

Bei den 3-Operanden - Operationen wird in der Regel so verfahren, daß der 2. Operand vom Ergebnis überschrieben wird

ADD A, B $\rightarrow (A + B \Rightarrow B)$

3-Adreß-Rechner findet man selten

3.2.2 RISC

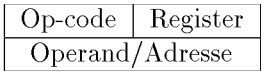
RISC (Reduced Instruction Set Computing) unterscheidet sich vom bisher betrachteten **CISC** (Complex Instruction Set Computing) insbesondere in zwei Punkten:
1. der **Load-and-Store-Architektur**,
2. dem **Pipelining**.

Der „reduzierte Instruktionssatz", der in diesem Begriff so betont wird, ist eigentlich von untergeordneter Bedeutung. Betrachtet man die Menge der unterschiedlichen **Op-Codes**, so stehen die RISC- den CISC-Prozessoren nur wenig nach (ca. 50 : 60). Der große Unterschied ergibt sich erst, wenn man gleichzeitig den **Adressierungsumfang**, d.h. die Menge der verschiedenen Adressierungsarten betrachtet (s.w.u.). Und auch das Prinzip des Pipelining ist in Ansätzen bei CISC-Prozessoren zu finden, kann dort aber nicht konsequent durchgehalten werden.

3.2.2.1 Load-and-Store-Architektur

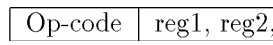
Die Load-and-Store-Architektur eines RISC-Prozessors besteht darin, daß der **Instruktionssatz** in mehrere Klassen aufgeteilt wird:
- Instruktionen zum **Speicherzugriff**, die im wesentlichen aus den Instruktionen

LOAD register from memory
STORE register to memory



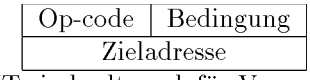
bestehen, und den gesamten Adressierungsumfang besitzen wie CISC-Prozessoren;
- **Verarbeitungsinstruktionen**, die nur auf die Inhalte von Registern wirken, und so einen sehr eingeschränkten Adressierungsumfang haben, jedoch meisten **3-Adreßinstruktionen** darstellen: z.B.

ADD reg1,reg2,reg3 (reg1 = reg2 + reg3)



Dadurch bleiben die Quelloperanden erhalten und müssen nicht aus dem Speicher nachgeladen werden. Damit möglichst viele Operanden in den Registern gehalten und nicht immer wieder nachgeladen werden müssen, wird die Zahl der CPU-Register möglichst groß gemacht, typisch sind 32 Datenregister.
- **Verzweigungsinstruktionen**, wie

BRAnch(BEDingung) Ziel



- **Steuerungsinstruktionen**, wie HALT und WAIT sind selten, dafür Verwaltungsintruktionen für das Pipelining, insbesondere bei Sprüngen und Unterbrechungen.

3.2.2.2 Pipelining

Pipelining (Rohrleitung) bedeutet, daß das Verarbeitungsgut (Information) von einer Bearbeitungsstation zur nächsten weitergeleitet wird wie bei einem technischen Prozeß.

Beim CISC-Prozessor findet man auch mehr oder minder abgeschlossene Zweige, wie das Lesen einer Instruktion (Fetch Cycle). Dieses kann unabhängig von allen anderen Abläufen parallel erfolgen (**instruction prefetch**). Bei allen anderen CISC-Instruktionen sind die verschiedenen Möglichkeiten zu unterschiedlich.

Beim RISC-Prozessor wird eine Klasseneinteilung vorgenommen in
- **Verarbeitungsinstruktionen**, bei denen die Durchführung der Operation Zeit beansprucht, und nur mit einfachsten Adressierungen für Quell- und Zieloperanden durchgeführt werden,
- **Load- und Store-Instruktionen**, bei denen praktisch keine Verarbeitung erfolgt, dagegen Zeit für ausgiebige Adreßermittlung beim Speicherzugriff bleibt. Daß hierbei nicht auf den Hauptspeicher, sondern auf ein **Cache Memory** (s. 2.5.5) zugegriffen wird, ist fast selbstverständlich; meist wird sogar mit 2 Cache Memories gearbeitet, eines für die Daten, ein zweites für die Instruktionen.

In der Regel werden die RISC-Instruktionen in folgenden Schritten abgearbeitet:
- IF: Instruktion lesen (Fetch Cycle)
- RD: Operanden (aus Registern) lesen (read)
- OP: Operation durchführen
- MM: Zeit für einen zusätzlichen Speicherzugriff (bei LOAD oder STORE)
- WT: Abspeichern des Resultats (write)

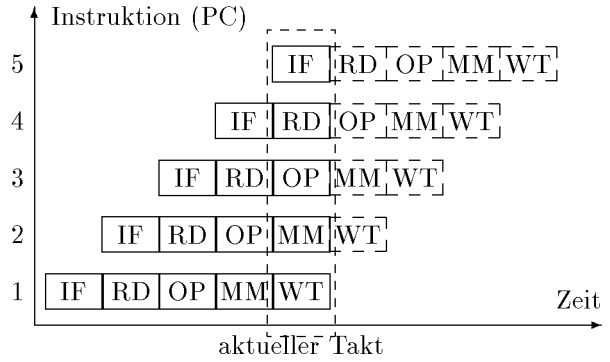


Bild b3p35

RISC-Pipelining:
Ein RISC-Prozessor teilt die Bearbeitung einer Instruktion in mehrere (5) Phasen auf, die auf entsprechende Bearbeitungsstationen verteilt und im Pipelining weitergegeben werden, so daß mehrere Instruktionen gleichzeitig in Bearbeitung sind und nach außen die Arbeitsgeschwindigkeit um das 5-fache erhöht erscheint.

Besondere Probleme ergeben sich, wenn der sequentielle Bearbeitungsfluß unterbrochen wird: wenn innerhalb des Programms (synchron) verzweigt wird, oder wenn eine (asynchrone) Unterbrechung von außen erfolgt. Dann müssen solche Instruktionen, die bereits begonnen wurden, abgebrochen werden; bei manchen, die z.B. schon ein Ergebnis ins Memory geschrieben haben, geht das aber nicht, dann muß die Verarbeitungskette angehalten werden (**pipe stalling**) und diese Instruktion bis zum Ende durch geführt werden. Dabei muß auch der Program Counter in der CPU entsprechend zurück gesetzt werden, damit bei der Wiederaufnahme an dieser Stelle korrekt fortgesetzt wird.

3.2.3 Der Bus

Bustypen:
- parallele Busse, Daten- Adreß- und Steuerbus
- serielle Busse (in der Zentraleinheit selten)
- synchrone (mit Taktsignal) und asynchrone Busse
- Adressen und Daten im Raummultiplex (eigene Leitungen)
- Adressen und Daten im Zeitmultiplex (nacheinander auf den gleichen Leitungen)

a) Das Busprotokoll für einen synchronen Bus (im Raummultiplex). Adressen und Daten werden als Worte parallel übertragen.

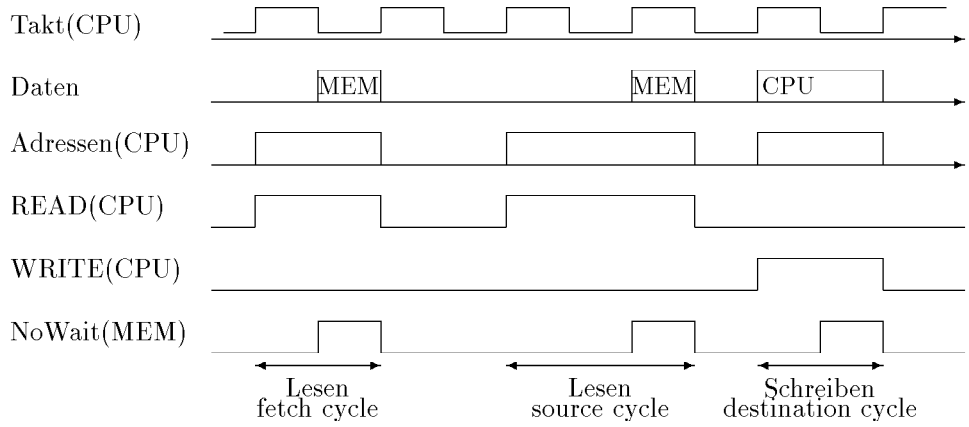


Bild n3p05a

b) Das Busprotokoll für einen asynchronen Multiplex-Bus (im Zeitmultiplex). Adressen und Daten werden als Worte auf denselben Leitungen nacheinander übertragen. Die Steuersignale sorgen im Handshaking-Verfahren für eine gesicherte Datenübertragung.

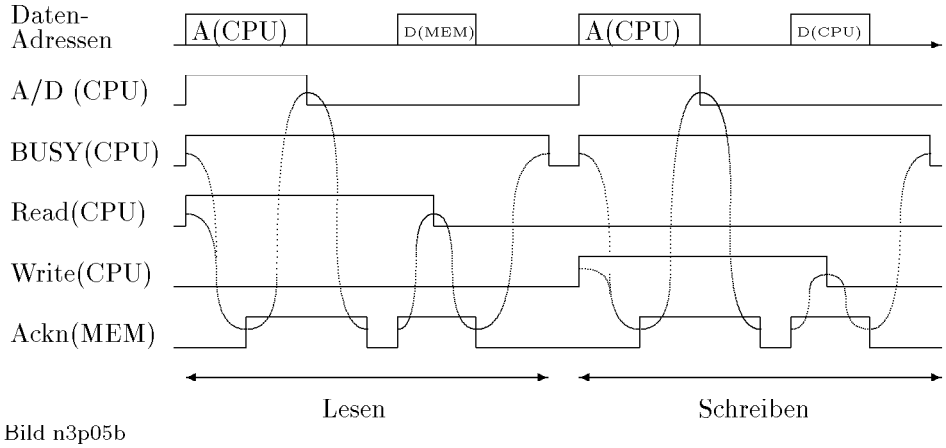


Bild n3p05b

c) Interrupts

Die Bussignale bei einer Interrupt-Anforderung (READ- bzw. WRITE-Signale sind hier weggelassen)

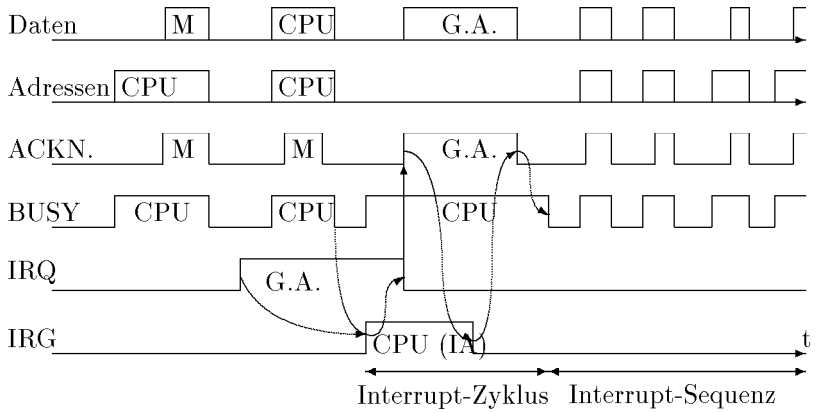


Bild n3p06

Ein Interrupt Request (IRQ) wird erst nach Abarbeitung einer Instruktion von der CPU zugelassen (IRG, Interrupt Grant). Dann schickt der Geräteanschluß auf den Datenleitungen einen Wert, der als Interrupt Vektor von der CPU (bzw dem Umterbrechungswerk) interpretiert wird und dazu dient, aus der Interrupt Vektor Tabelle (IVT) eine Adresse für eine Interrupt Service Routine in den Programm Counter (PC) zu lesen (und auch eine neues Prozessor Status Wort (PSW). Der alte PC (und auch der Prozessor Status, PSW) werden für einen Rücksprung auf dem Stack gesichert. Diese 4 Speicherzugriffe stellen die Interrupt Sequenz dar.

d) einige Bussysteme

Bussystem	Übertragungs-Protokoll	Daten-	Adreß-	Steuer-	Gesamt-	Interrupt-Ebenen	Buszyklus (nsec)	Standards
		Leitungen						
S-100 Bus	asynchron	8+8	16(24)	44	100	2	$\approx$ 500	IEEE 696
STD-Bus	synchron (+WAIT)	8	16	22	56	2	$\approx$ 500	diverse Hersteller
MULTIBUS	asynchron	8/16	20(24)	23	86+60	8	$\approx$ 100	IEEE 796
VME-BUS	asynchron	16(64)	24(32)	38	96+96	7	$\geq$ 50	IEC 821
Q - BUS	asynchron multiplex	16	18(22)	22	72	4	$\approx$ 500	DEC
EISA-Bus	synchron	32	32			16+8	120	diverse Hersteller
MCA-Bus	synchron	32	32			16+8	100	IBM

e) Kombinierte Bussysteme

– Die Bussystem der VAX-11 (DEC)

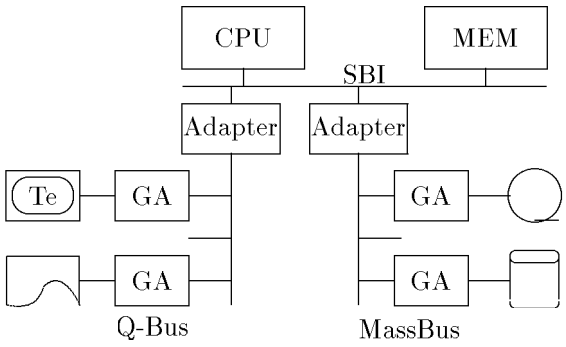


Bild n3p06a (interne Busse der ZE)

– Die Bussysteme eines Prozeßrechners (Motorola)

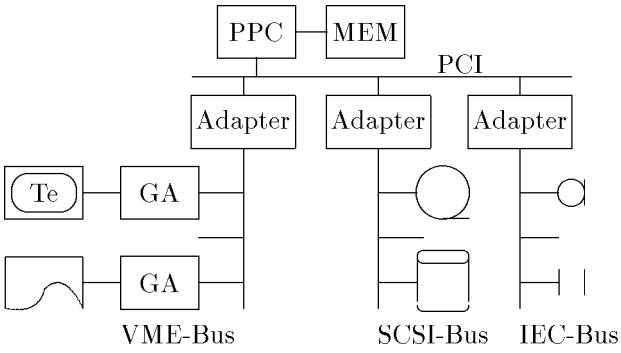


Bild n3p06b (externe Busse für periphere Geräte)

3.2.4 Der Arbeitsspeicher

Speicherhierarchie

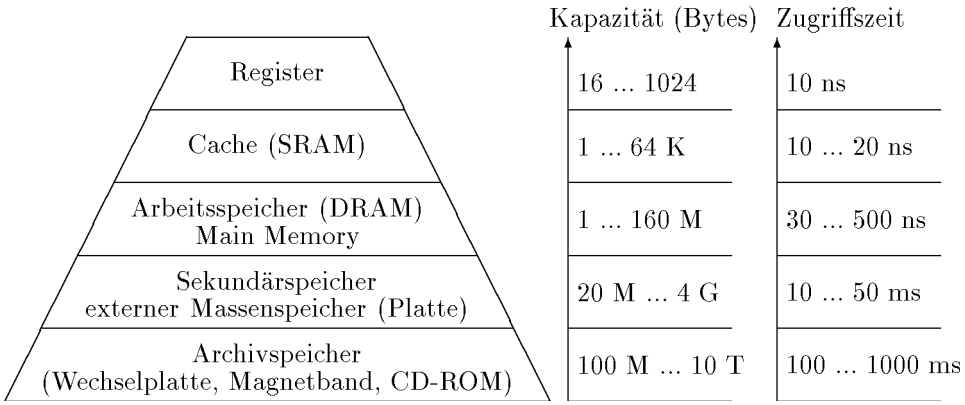


Bild b4p07

Begriffe

- RAM = Random Access Memory (Method): die Adresse kann eine beliebige zufällige Zahl sein (wahlfreier Speicherzugriff)
- SAM = Sequential Access Memory (Method): es gibt eine Startadresse, die folgenden Werte müssen dann sequentiell gelesen werden, z.B. FIFO, Stack, Lochstreife, Magnetband
- ROM = Read Only Memory (Nur-Lese-Speicher): z.B. Lochstreifen
- RWM = "RAM" = Read-Write Memory: Schreib-Lese-Speicher

a) Grundstruktur

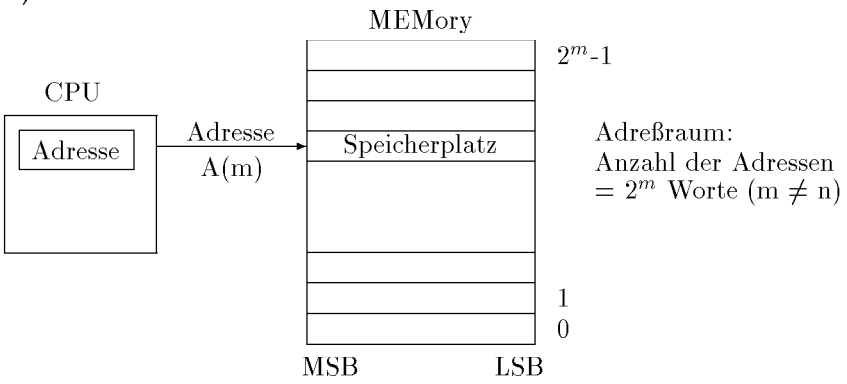


Bild b4p07a

Kenngrößen

- Speichergröße, in Bit, Byte oder Worten (1 K = 2^{10} = 1024, 1 M = 1 Mebi = 2^{20})
- Zugriffszeit (typ: t_a = 30 ... 500 ns)
- Zykluszeit, bei wiederholtem Zugriff ($t_c \leq t_a$)
- Aufbau und Struktur

b) Bytestruktur

Ein Byte ist ein Teilwort, meist 8 bit breit, zur Speicherung von Zeichen.

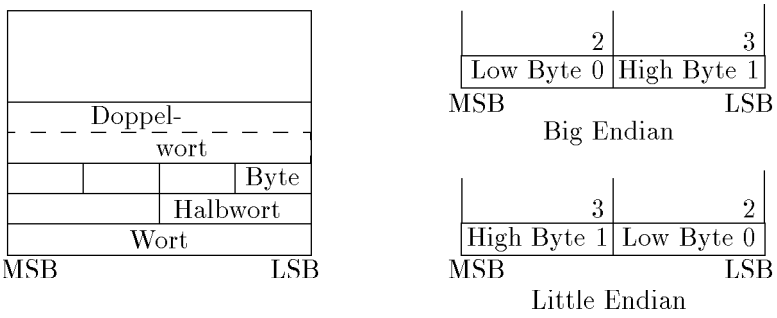


Bild b4p07b

c) Adreßräume

Flacher Adreßraum: Die CPU kann logische Adressen (mit n bit Wortbreite) erstellen, welche in ihrer Breite (m bit) ausreichen, um alle Speicherplätze des realen Adreßraums direkt zu adressieren (d.h. $n \geq m$).

Hierarchischer Adreßraum: Die CPU liefert Adressen, welche in einem Speicherverwaltungswerk (Memory Management Unit, MMU) zu realen Adressen umgewandelt werden. Hier ist in der Regel $n < m$. Durch Hinzufügen von weiteren Adreßbits aus Registern der MMU (Memory Management Register, MMR) wird die reale Adresse erstellt.

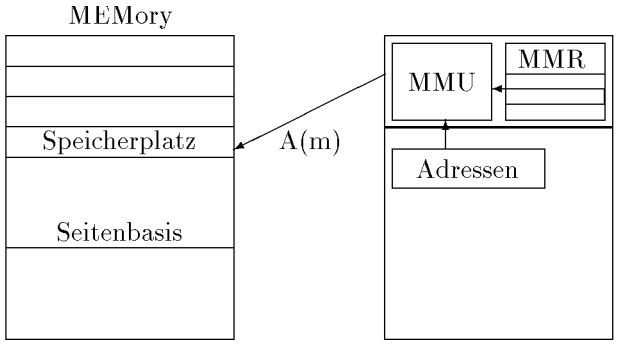


Bild b4p09

Die Bildung der realen Adressen kann nach unterschiedlichsten Regeln erfolgen:
a) durch einfaches Anfügen weiterer Bits aus dem MMR (nur ein MMR möglich)
b) durch Addieren des MMR zur logischen Adresse.
c) durch Auswahl eines MMR durch die obersten Bits der logischen Adresse und anschließender Verknüpfung wie in a) oder b).
In jedem Fall bestimmt das MMR die unterste verfügbare reale Speicheradresse, die Seitenbasis. Die Seitengröße wird durch die von der CPU gelieferte Adresse (oder deren Rest im Falle c)) bestimmt, also höchstens 2^n Speicherplätze.

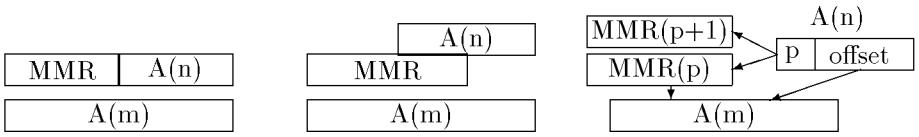


Bild b4p09a

3.2.5 Speicherwerke (Memory Units)

a) Grundstruktur
Ein Speicherwerk (memory unit) enthält eine Speichermatrix, welche die Daten und Instruktionen in Speicherelementen enthält. Diese werden über die (decodierte) Adresse aktiviert. Der Adreßdecoder kann auch integrierter Bestandteil eines Speicherchips sein. Die Basisadresse wird nur benötigt, wenn mehrere Speicherwerke (Speicherkarten) vorhanden sind. Die Auswahl der Karte erfolgt meist über die höchstwertigen Adreßbits. Über die Steuersignale (READ/WRITE) wird die Richtung der Datenübertragung bestimmt (R/W).

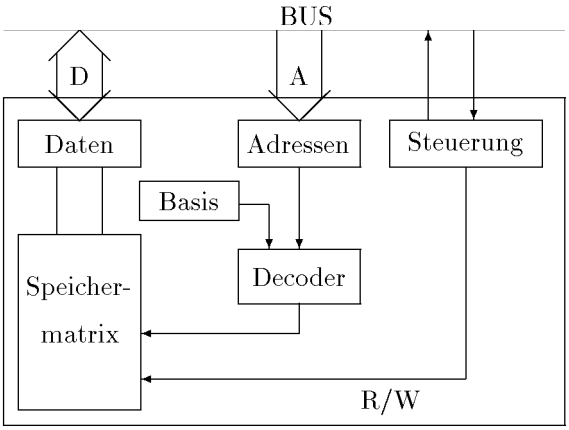


Bild b4p08

b) Adreßdecoder

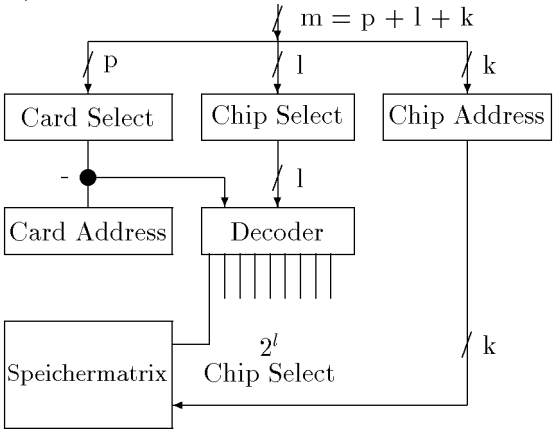


Bild b4p08a

c) Speicherbausteine (Chips)

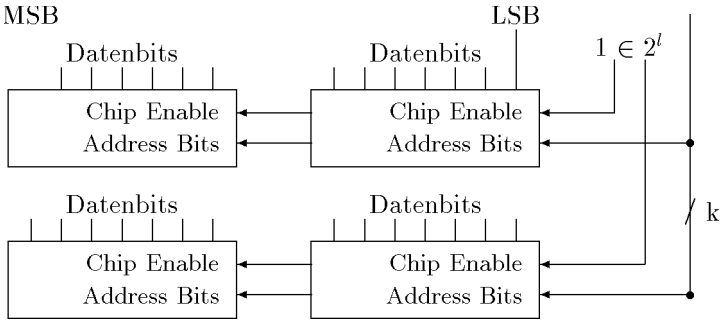


Bild b4p08b

d) Chipstruktur

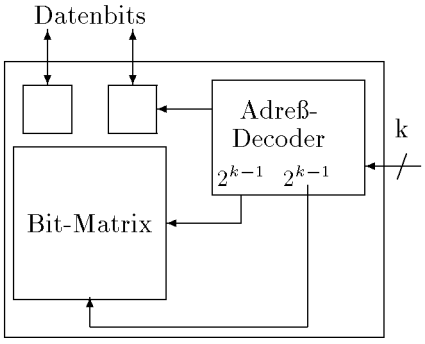


Bild b4p08c

. .Die Bits im Chip sind in der Regel in einer quadratischen Matrix angeordnet; z.B. 1 M(ebi) Bits in 1K Reihen und 1K Spalten. Der Zugang kann unterschiedlich sein; z.B. kann bei einem 1M Bit Chip meist nur auf ein Bit (von 2^{20}) mit 20 Adreßleitungen zugegriffen werden, oder auf 2 Bit bei 19 Adreßbits.

3.2.6 Speicherelemente (Memory Elements)

a) Grundstruktur

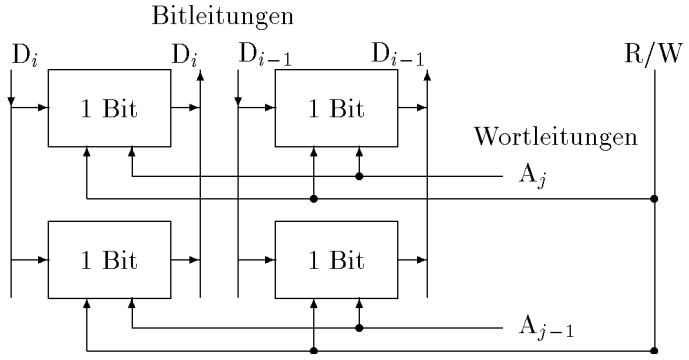


Bild b4p10

b) Ladungsspeicher (dynamisches RAM, DRAM)

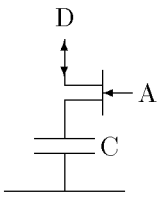


Bild b4p10a

. .Vorteile:
- sehr klein, hohe Speicherdichte: ca 1 Mbit / Chip
- billig: ca 10 DM/Mbit
- schnell: $t_a \approx 10 - 200$ ns
- sparsam: ca 1 W / Chip

Nachteile
- flüchtig (volatil) bei Stromausfall
- destruktives Lesen: $t_c \approx 2\ t_a$
- selbstentladend: $\tau = 1/RC \approx 1$ ms (refresh erforderlich)
- störanfällig: Entladung (Soft-Error) durch ionisierende Strahlung (Röntgen, Höhenstrahlung, o.ä.)

Anwendung: Arbeitsspeicher in sicherheitsunkritischen Anwendungen (PC)

c) Flipflop-Speicher (statisches RAM, SRAM)

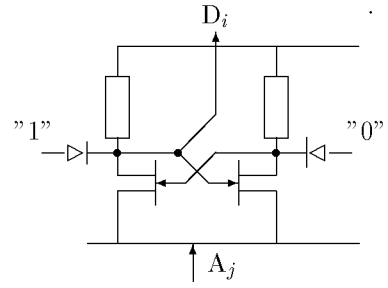


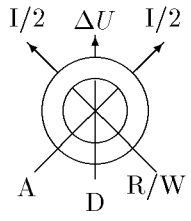
Bild b4p10b

. .Vorteile:
- sehr schnell: $t_a \approx 2 - 20$ ns
- nicht destruktiv beim Lesen: $t_c = t_a$
- nicht selbstentladend
- (klein) ca 8 Kbit / Chip

Nachteile
- flüchtig (volatil)
- konsumptiv (viel Strom verbrauchend, ein Transistor leitet immer) ca 3 W / Chip

Anwendung: Register und Puffer

d) Ringkernspeicher (Ferritkerne)



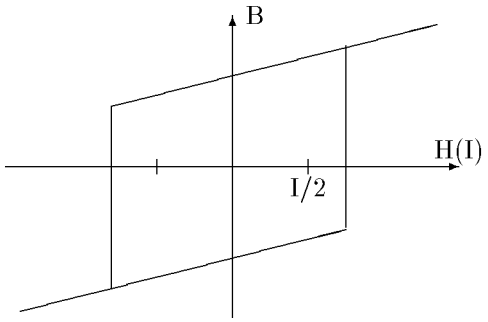


Bild b4p10c

Nachteile:
- groß: ca 100 bit / 1 mm^3
- langsam: $t_a \approx 500$ ns
- destruktives Lesen
- komplexe Ansteuerung
- teuer (ca 500 DM/ Kbyte)

. .Vorteile:
- nicht flüchtig (nonvolatil)
- störsicher gegen praktisch alle Umwelteinflüsse (Strahlung, Felder, Temperatur, Schock)

Anwendung: in sicherheitsrelevanten Anwendungen, Raumfahrt, Medizintechnik

e) Festwertspeicher (ROM)

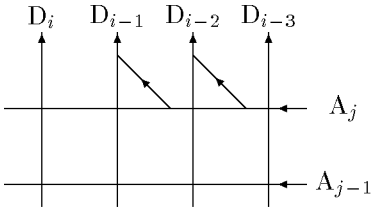


Bild b4p10d

. .Vorteile:
- billig: ab 10 DM/Mbit
- schnell: $t_a \approx 10 - 200$ ns
- sehr sparsam: ca 0.01 W / Chip
- nicht flüchtig (nonvolatil)
- störsicher gegen die meisten Umwelteinflüsse (Strahlung, Felder, Schock)

Nachteile
- einmaliges oder aufwendiges Schreiben

Anwendung: Betriebssystem-Komponenten, sicherheitsrelevante Programme

Varianten
- Diodenmatrix
- PROM, Programmable ROM: die Dioden werden vom Anwender (für "0"en) durchgebrannt (WORM = Write Once, Read Multiple)
- EPROM, Erasable PROM, wiederlöschbares PROM: Statt der Dioden werden Transitoren eingesetzt, die durch UV-Licht oder Röntgenstrahlung wieder "geheilt' werden können.
- EEPROM, Electrically Erasable PROM: die Löschung erfolgt elektrisch.

3.3 Geräteanschlüsse

Aufgaben:
- Steuerung von Geräten und Zustandsabfrage
- Datenübertragung von und zu Geräten
- Adressierung von Geräten
- Interfaces: byteweise Übertragung zu und und von Datenendgeräten
- Controller: blockweise Übertragung von und zu Massenspeichern

3.3.1 Grundstruktur

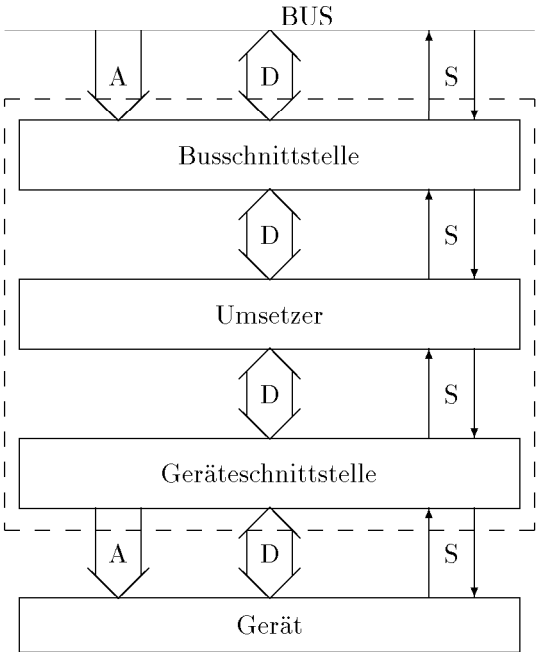


Bild n3p11

- Busschnittstelle: rechnerspezifisch, Kommunikation erfolgt über Register, die wie Speicherworte angesprochen werden können.
- Geräteschnittstelle: gerätespezifisch, byte- oder blockorientiert.
- Umsetzer: für Daten und Steuersignale. Adressen für Geräte müssen aus Daten von der CPU gebildet werden.

3.3.2 Busschnittstelle

a) Grundstruktur
Für alle Geräteanschlüsse einheitlich.

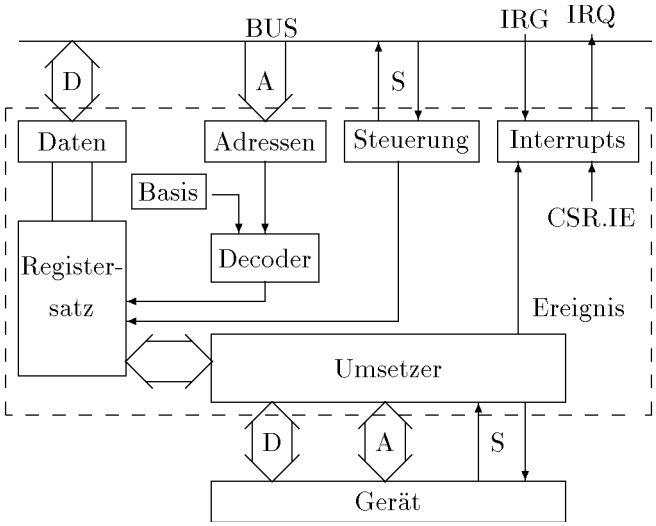


Bild n3p11a

b) Zugriff der CPU auf Register in einem speziellen Adreßraum (IO-page)
Basisadresse für jedes Gerät konfigurierbar,
z.B. COM-1 Port = 3F8h
Registeradressen mit festem Offset
DBR: Data Buffer Register, Datenpuffer
CSR: Control and Status Register, Abfrage und Steuerung von Gerät und Umsetzer.
z.B. beim COM-1 Port (UART 8250/16450):

- 00h | MSB LSB |
  Empfangs-Datenregister (Receive Data Buffer Register,R-DBR) read only
  Sende-Datenregister (Transmit Data Buffer Register,T-DBR) write only
- 01h Interrupt-Aktivierungsregister (Interrupt Enable)

0	0	0	0	SINP	ERBK	TBE	RxRd

Interrupt-Aktivierung wenn Bitwert = 1
  - SINP = Serial INPut (Zustandsänderung einer Empfangsleitung)
  - ERBK = Error or Break: Parity-, Overflow- oder Framing-Fehler oder BREAK
  - TBE = Transmitter Buffer Empty (Sendepuffer leer)
  - RxRd = Received Data Ready (ein Zeichen im Empfangspuffer)

3.3.3 Polling, Interrupts und DMA

- Geräteschnittstelle: byteorientiert oder blockorientiert
- Busschnittstelle: CPU-Kontrolle oder DMA-Betrieb

3.3.3.1 Abfragebetrieb, Polling:

Übertragung unter CPU-Kontrolle

- Einzelzeichenübertragung:
  MEM ↔ CPU ↔ DBR ↔ Gerät (Datenübertragung)
  MEM ↔ CPU ↔ CSR ↔ Gerät (Steuerung)
  unter Programmkontrolle z.B.

```
TSTB CSR        ; Abfrage ob READY
BPL  .-1        ; Rücksprung falls nicht
MOV  DBR, MEM   ; Daten holen und speichern
```

- Blockpufferung: Die zu übertragenden Zeichen werden in einem FIFO zwischengespeichert und erst bei Bedarf ohne CPU Kontrolle übertragen.
  (Das FIFO kann auch im Gerät angesiedelt sein.)
  Datenübertragung: MEM ↔ CPU ↔ DBR ↔ FIFO ↔ Gerät

3.3.3.2 Interrupts

Interrupt-Behandlung in mehreren Ebenen

a) Interrupt-Erzeugung

Quelle für (fast) jede Interrupt-Anforderung ist ein Geräte-Anschluß
Aufagben der Interrupt-Logik

- außere Ereignisse erfassen und eventuell speichern,
- Interrupt-Anforderung (Interrupt Request, IRQ) an die CPU senden und auf deren Bestätigung (Interrupt Grant, IRG) warten,
- eventuell weitere Daten über den geforderten Interrupt senden, (Priorität des Interrupts (HW-P), Zeiger (IV) auf einen Eintrag in der IVT (vektorisierte Interrupts).)

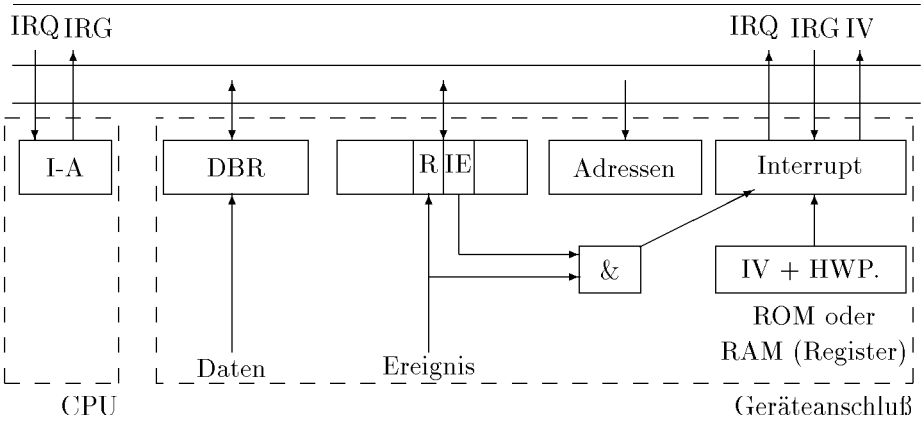


Bild b3p15

b) Filterung der Interrupt-Anforderungen
in der CPU im Interrupt-Verarbeitungswerk (Interrupt Arbitrator, I-A) nach verschiedenen Verfahren

b1) Maskierung, Verriegelung
Bei der Interrupt-Maskierung wird unter Programmkontrolle ein Register geladen, dessen Bits einzelnen Geräte-Anschlüssen zugeordnet sind und deren Interruptanforderungen weitergeben oder blockieren. Beim gleichzeitigen Auftreten mehrerer IRQ könnte die Bitposition für eine weitere Auswahl (Gewichtung) herangezogen werden. Nachteilig bei dieser Interrupt-Behandlung ist die geringe Anzahl von unterschiedlichen IRQs.

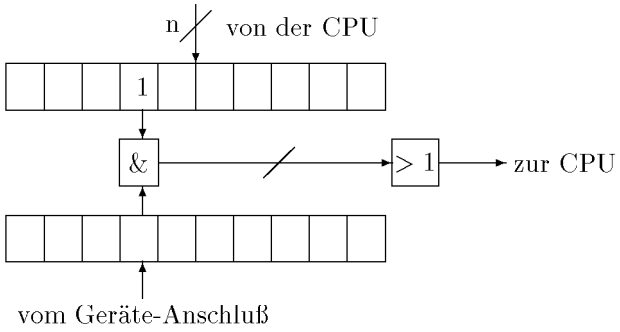


Bild b3p16

b2) Interrupt-Gewichtung
Bei der Interrupt-Gewichtung werden die von den Geräte-Anschlüssen ankommenden IRQs über einen Decoder in einen Zahlenwert umgewandelt, die Hardware-Priorität (HWP), der mit einem arithmetischen Vergleicher mit dem in einem Register stehenden Zahlenwert, der Software-Priorität (SWP), verglichen wird. An die CPU wird die Interrupt-Anforderung nur dann weitergegeben, wenn $HWP > SWP$ ist.

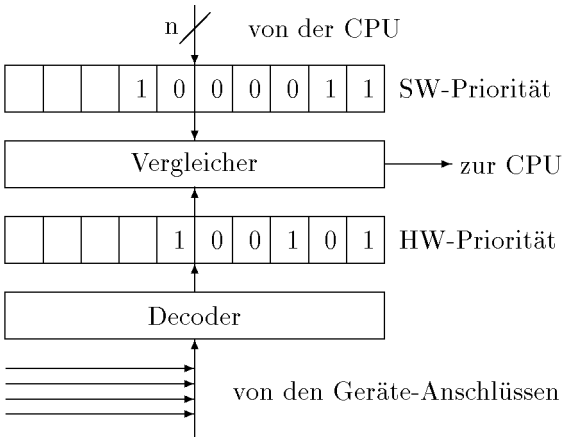


Bild b3p17

c) Bearbeitung des Interrupts
durch das Mikroprogramm der CPU

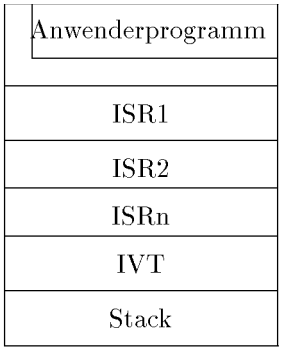


Bild b3p09

- Ein Anwenderprogramm P läuft (in einer Schleife).
- Ein Ereignis löst eine Interruptanforderung aus.
- Die CPU bestätigt die Anforderung am Ende einer Instruktion
- Der aktuelle PC (und eventuell auch andere Werte, wie das Prozessor-Status-Wort PSW) wird auf dem Stack gesichert
- Aus der Interrupt Vektor Tabelle IVT wird ein neuer PC, die Einsprungadresse der ISR geholt.
- Nach Abarbeitung der ISR wird beim Rücksprung der alte PC (und die übrigen Werte) vom Stack zurückgeholt und das unterbrochene Programm fortgesetzt.

Darstellung der Abläufe durch A-t-Diagramme

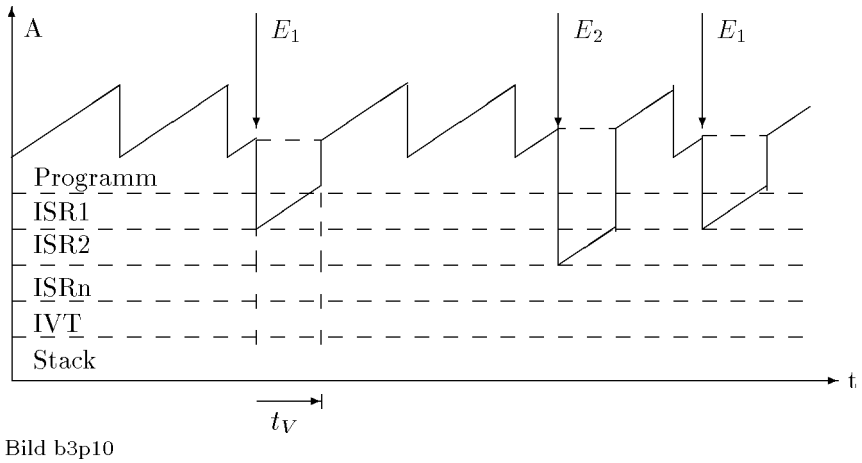


Bild b3p10

Abläufe auf dem Bus beim Interrupt

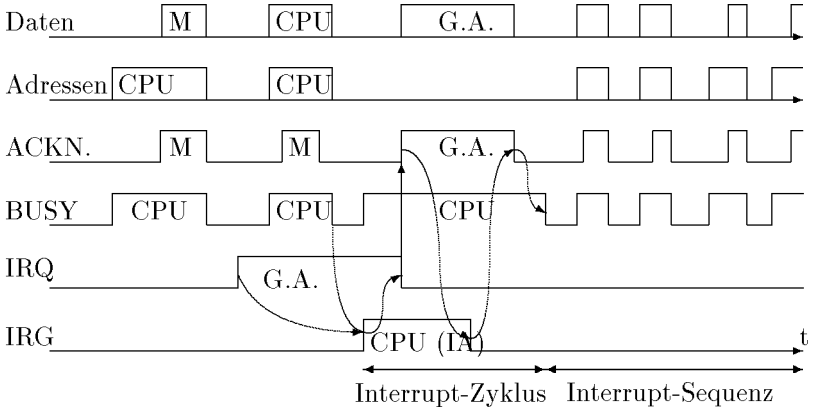


Bild n3p06

d) Darstellung der Prioritätensteuerung durch P-t-Diagramme
Zur Darstellung der wechselnden Prioritäten und der gegenseitigen Abhängigkeit von Hard- und Software-Prioritäten dienen P-t-Diagramme. Hier wird in Abhängigkeit von der Zeit t die Software-Priorität P (der CPU) als Kurvenzug aufgetragen, der nur (wenige) diskrete Werte annehmen kann. Als Ereignispfeile (E_i) werden die auftretenden Interrupts mit ihren Hardware-Prioritäten eingetragen. Falls ein solcher Pfeil eine höhere Priorität anzeigt, wird ein Interrupt ausgelöst, der in einem parallel verlaufenden A-t-Diagramm dargestellt werden kann. Falls die Hardware-Priorität niedriger ist und der Interrupt nicht sofort durchgeführt wird, kann die dabei auftretende Wartezeit (Reaktionszeit) dargestellt und bestimmt werden.

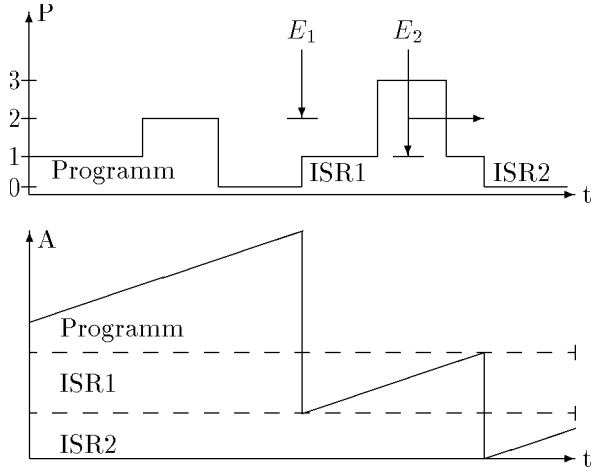


Bild b3p18

e) Vergleich der Betriebsarten:

	Polling	Interrupt
Vorteile	einfache Hard- und Software	CPU-Auslastung $< 100\%$
Nachteile	CPU-Auslastung $= 100\%$	komplexe Hard- und Software

3.3.3.3 DMA-Betrieb

Direct Memory Access (Speicherdirektzugriff)

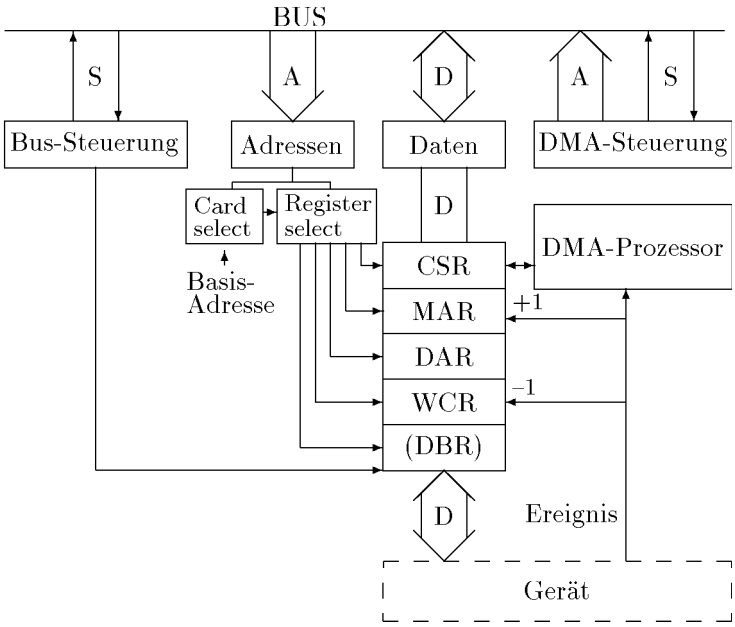


Bild n3p12

- MAR: Memory Address Register: Adresse im Arbeitsspeicher, von/zu der das nächste Wort übertragen wird.
- DAR: Device Address Register: Adresse des Gerätes bzw. Gerätebereichs (Plattenfläche, Sektor, Spur)
- WCR: Word Count Register: Zähler für die zu übertragenden Daten.

DMA-Vorbereitung unter Programm-Kontrolle:

```
MOV X, MAR      ; MAR laden
MOV Y, DAR      ; DAR laden
MOV Z, WCR      ; WCR laden
MOV F, CSR      ; Funktion laden
INC  CSR        ; Go-Bit setzen
```

DMA-Übertragung ohne CPU-Mitwirkung
(internes Programm des DMA-Controllers in ROM):

- DMA-Request, sobald Daten übetragen werden (Ereignis)
- DMA-Grant von der CPU nach dem nächsten Buszugriff $\overline{BUSY}$
- Buszugriff durch Controller "BUSY" (DMA-Zyklus)
- Controller bedient A- und S-Leitungen
- Daten kommen vom Memory (bei READ)
- Controller gibt den Bus wieder frei
- MAR + 1
- WCR - 1

DMA-Zugriffsmethoden:
- Cycle-Stealing: CPU und Controller teilen sich den Bus 1:1
- Burst-Mode: Controller macht n Zugriffe nacheinander (1:n)
- Prozessor-Halt: Controller macht beliebig viele Zugriffe.

DMA Bus-Protokoll

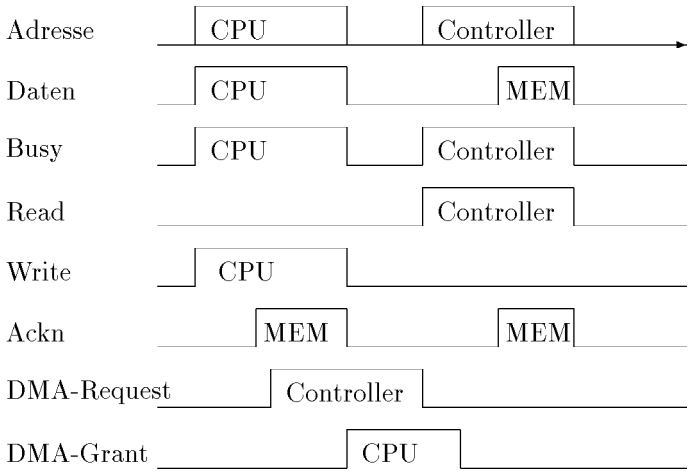


Bild n3p12a

DMA-Abschluß unter Programm-Kontrolle
- Wenn WCR = 0, wird das READY-Bit im CSR setzen
- Falls ein Fehler auftritt, wird das ERROR-Bit im CSR gesetzt
- Falls Interrupt Enable gesetzt, erfolgt Aufruf einer ISR
  Andernfalls muß die CPU immer wieder mal nachfragen (pollen)

3.3.4 Geräteschnittstellen

zur Ankopplung von peripheren Geräten, für jeden Gerätetyp unterschiedlich:

a) Parallel-Schnittstelle (für Drucker von Centronics)
Die Daten werden aus dem DBR (über Leitungsverstärker) parallel ausgegeben. Steuersignale werden aus dem CSR gebildet.

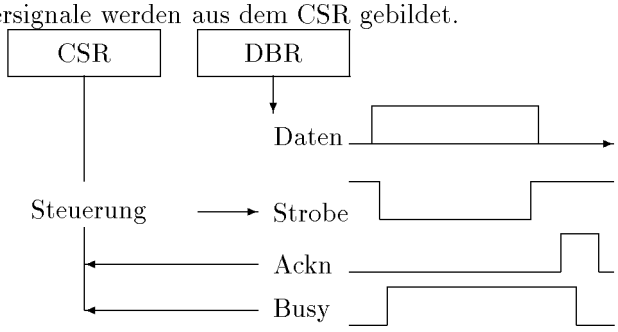


Bild n3p13

b) Terminalschnittstelle seriell, asynchron (V.24, RS 232-C)
Der Zugriff auf diese Schnittstelle erfolgt gewöhnlich über je ein CSR und ein DBR in Sende- und in Empfangsrichtung.

Zur Umsetzung wird ein Parallel-Seriell-Wandler eingesetzt, der die Datenbits eines Zeichens in eine Folge von (gleich langen) Signalen umsetzt.

Die V.24-Schnittstelle ist also sowohl für synchrone als auch für asynchrone Datenübertragung vorgesehen. Also solche, bei der die Zeichen in beliebigen Abständen oder unterbrechungslos (mit Pausenzeichen) übertragen werden.

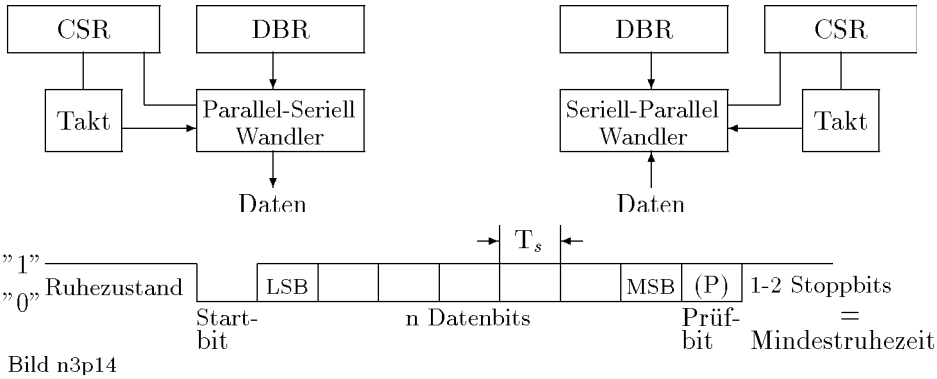


Bild n3p14

V.4 (DIN 66 022) legt die Zeichendarstellung auf den Datenleitungen für die asynchrone Übertragung, das **Start-Stop-Verfahren** fest:
- Der **Ruhezustand** der Leitung ist '1'.
- Jedes Bit wird in einem **Taktschritt** fester Länge T_s übertragen.
- Der Beginn eines Zeichens wird durch ein **Startbit** mit dem Zustand '0' markiert.

- Die **Datenbits** werden mit dem niederwertigsten Bit (**LSB** = Least Significant Bit) zuerst und dem höchstwertigen Bit (**MSB** = Most Significant Bit) zuletzt übertragen. In der Regel werden 7 Datenbits des ASCII-Zeichensatzes übertragen.
- Den Datenbits kann ein **Prüfbit** gerader oder ungerader **Parität** folgen (even/ odd **parity**). Das Prüfbit wird so gesetzt, daß hiermit die Summe aller '1'-en gerade bzw. ungerade, d.h. die gesamte Quersumme modulo 2 eine 0 oder eine 1 ergibt.
- Das Ende eines Zeichens wird signalisiert durch 1, 1.5 oder 2 **Stoppbits**, bei denen die Leitung im Ruhezustand '1' bleibt, bevor das nächste Zeichen übertragen wird. Die Stoppbits beinhalten keine Informationsübertragung, und die Angabe ihrer Anzahl bedeutet nur die **Mindestruhezeit** in Einheiten von Taktschritten T_s.
- Das nächste Zeichen kann unmittelbar darauf folgen, oder zu einem beliebigen späteren Zeitpunkt.

Freie Parametern, die am Sender und am Empfänger übereinstimmend eingestellt sein müssen:
- **Baudrate** oder **Schrittgeschwindigkeit** $f_s = 1/T_s$, für die folgende Werte gewählt werden können: 75, 110, 150, 200, 300, 600, 1200, 2400, 4800, 9600, und 19200 Baud. Wegen der binären Codierung ist hier die Schrittgeschwindigkeit (in Baud) mit der **Bitübertragungsrate** (in **bps = bits per second**) identisch.
- Zahl der Datenbits (n = 5,6,7,8). Für die Übertragung von beliebigen, uncodierten Dualwerten in Form von Bytes ist n = 8 zu wählen.
- Anwendung eines Prüfbits (p = 0 oder 1). Falls ein Prüfbit verwendet werden soll (p=1), muß noch die **Parität** auf **'gerade'** oder **'ungerade'** eingestellt werden. In manchen Fällen werden auch Prüfbits mit festen Werten verwendet:
'0' = **space parity** oder '1' = **mark parity**, die vom Sender erzeugt, vom Empfänger nicht ausgewertet, aber mitgezählt werden (vgl. 2.6.6).
- Anzahl der **Stoppbits** (z = 1, 1.5, 2).

Die **Übertragungszeit** für ein Zeichen ergibt sich dann zu

$$T_z = (1 + n + p + z) * T_s$$

und die **Zeichenübertragungsrate** zu

$$f_z = 1/T_z \qquad \text{(in } \mathbf{cps} \text{ = characters per second)}$$

Typische Einstellungen sind:
- Für die amerikanischen Fernschreiber (**Teletype**):
n = 7 (ASCII-Zeichen), p = 1, odd parity, z = 2, f_s = 110 Baud;
das ergibt eine Zeichen(druck)geschwindigkeit f_z = 110/(1+7+1+2) = 10 cps
- Für Binärdaten bei hoher Geschwindigkeit:
n = 8 (Byte), p = 0, z = 1, f_s = 9600 Baud;
und einer resultierenden Übertragungsrate f_z = 9600/(1+8+1) = 960 Byte/sec.

Die V.24 Schnittstelle ist ursprünglich für die Kopplung von 2 gleichwertigen Datenstationen über eine Telefonleitung konzipiert worden (a). Dazu werden auf beiden Seiten Modems eingesetzt, welche die Aufgabe haben, die digitalen Signale von Terminal und Rechner in Wechselspannungen umzusetzen, wie sie auch bei der Sprachübertragung auf Telefonleitungen übermittelt werden. Wenn das Terminal nahe beim Rechner steht (b), können die Modems entfallen; dann muß aber ein Nullmodemkabel

eingesetzt werden, welches eine Sende- auf die zugehörige Empfangsleitung umsetzt, und umgekehrt (c).

Dazu sind an jedem Stecker jeweils folgende Verbindungen herzustellen:
- RTS, CTS (Stifte 4 und 5) und
- DSR, DTR, DCD (Stifte 6 und 20 und 8).

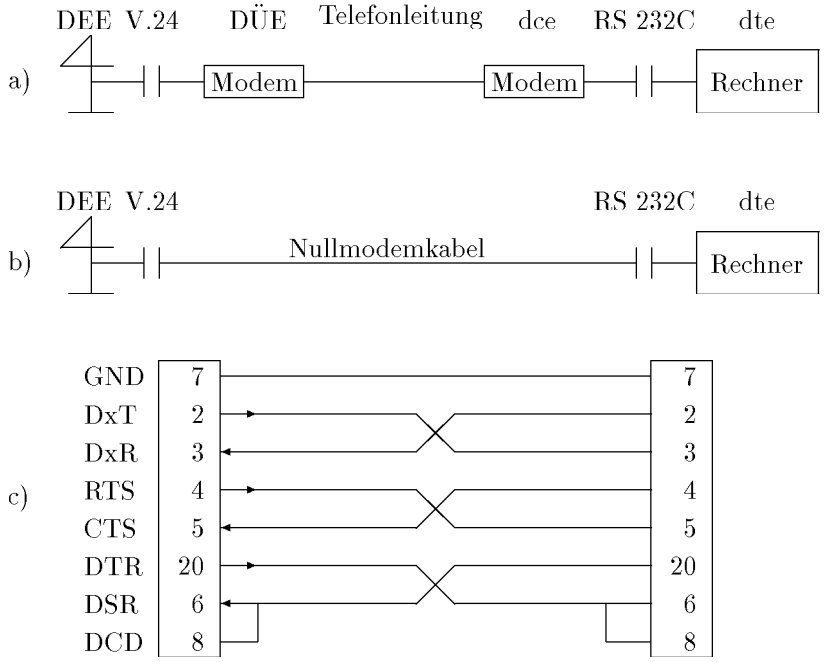


Bild n3p14a

V.24 (DIN 66 020, 66 021) legt die physische und die logische **Schnittstelle** fest, d.h. die Stiftbelegung eines genormten, 25-poligen Anschlußsteckers (nach ISO 2110) und die Bedeutung der auf den Leitungen übertragenen Signale. Sie beziehen sich immer auf die Verbindung zwischen der **Datenendeinrichtung** (**DEE**, engl.: **data terminal equipment**, **dte**), welche sowohl ein Terminal als auch ein Rechner sein kann, und dem **Modem** (Modulator/Demodulator), der auch als **Datenübertragungseinrichtung** (**DÜE**, engl.: **data communication equipment** oder **data circuit terminating equipment**, **dce**) bezeichnet wird.

V.28 (DIN 66 259; T1) legt die elektrischen Werte für die Darstellung der Daten und der Steuersignale fest:
- Die Datenwerte werden durch **Signalpegel** dargestellt, die über einen **Taktschritt**, eine **Bitdauer** (engl.: **bit time**) konstant bleiben. Dieses Verfahren wird als **NRZ (Non Return to Zero)** bezeichnet. Für die Pegel wird ein **Doppelstromverfahren** benutzt, also Pegel entgegengesetzter Polarität, die bestimmte Grenzwerte einhalten müssen (s. Tabelle 2.8).
- Die Steuersignale können beliebige Längen haben, die Pegel für ihre EIN- bzw. AUS-Zustände sind in Tabelle 2.8 angegeben.

Schnittstellenleitungen nach V.24

Kurzzeichen			Stift	Beschreibung		Richtung
CCITT V.24	EIA RS232	DIN 66020	ISO 2110	deutsch	englisch	Terminal Modem
101	P-GND	E 1	1	Schutzerde	protective ground	o——o
102	S-GND	E 2	7	Signalerde	signal ground	o——o
103	TD	D 1	2	Sendedaten	Transmitted Data	o——►
104	RD	D 2	3	Empfangsdaten	Received Data	◄——o
105	RTS	S 2	4	Sendeteil einschalten	Request to Send	o——►
106	CTS	M 2	5	Sendebereitschaft	Clear to Send	◄——o
107	DSR	M 1	6	Betriebsbereitschaft	Data Set Ready	◄——o
108.1		S 1.1	20	Übertragung. anschalten		o——►
108.2	DTR	S 1.2	20	Terminal betriebsbereit	Data Terminal Ready	o——►
109	DCD	M 5	8	Empfangssignalpegel	Carrier Detected	◄——o
110	SQ	M 6	21	Empfangsgüte	Signal Quality	◄——o
125	RI	M 3	22	Ankommender Ruf	Ring Indicator	◄——o
111	DRS	S 4	23	Übertragungsgeschw.	Rate selctor (dte)	o——►
112		M 4	23	dto	Rate selctor (dce)	◄——o
126	STF	S 5	11	Sendefrequenz(200 Baud)	Select frequency	o——►
113	TC	T 1	24	Sendeschrittakt	Transmit Clock (dte)	o——►
114	TC	T 2	15	Sendeschrittakt	Transmit Clock (dce)	◄——o
115	RC	T 3	17	Empfangschrittakt	Receive Clock	◄——o
118	TD.2	HD 1	14	Hilfskanal Sendedaten	Second. Trans. Data	o——►
119	RD.2	HD 2	16	Hilfskanal Empfangsdaten	Second. Receiv. Data	◄——o

Pegel der V.24-Schnittstelle (V.28)

Pegel	Daten	Signale
-15 V bis -3 V	'1' (mark)	AUS
+15 V bis +3 V	'0' (space)	EIN
-3 V bis +3 V	(no carrier)	undefiniert

Varianten für die (elektrische) Darstellung der Bits gegenüber der V.28 Norm sind:
– die historische **20-mA-Stromschleife** (current **loop** vgl. DIN 66 258; T1), ein **Einfachstromverfahren** mit folgenden Werten:

'0' = 0 mA
'1' = 20 mA

– **V.10** bzw. **X.26** (DIN 66 259; T2), eine unsymmetrische **Doppelstromschnittstelle**, die auch für Koaxialleiter gedacht ist, und **Übertragungsraten** bis 100 kBaud ermöglichen soll. Sie arbeitet mit **Signalpegeln**, die mit integrierten Schaltungen leicht realisiert werden können:

'0' = +0.3V bis +5.0 V
'1' = –0.3V bis –5.0 V

– **V.11** bzw. **X.27** (DIN 66 259; T3), eine symmetrische **Doppelstromschnittstelle** bis 10 MBaud mit denselben Signalpegeln wie V.10, jedoch für verdrillte Doppeladern gedacht.

Neue **Schnittstellennormen**, welche die **RS-232-C** Schnittstelle ablösen, sind:
– **RS-449** bis 10 MBaud und elektrischen Werten von V.11 (RS-422A) oder V.10 (RS-423A). Neue Signalleitungen und ein 37-poliger Stecker.
– In den neuen Datennetzen (ISDN) sollen **X.21** und **X.24** die V.24-Norm ablösen.
Rechnerbausteine:
UART, Universal Asynchronous receiver and Transmitter oder
PUSART, Programmable Universal Synchroneous and Asynchroneous Receiver and Transmitter

c) Massenspeicheranschlüsse
– **SCSI** (Small Computer System Interface) mit DMA-Controller (ANSI X.3.131)

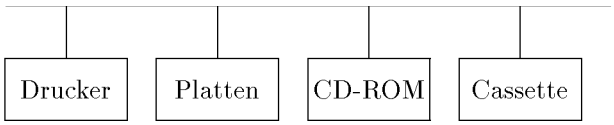


Bild n3p14b

SCSI-Typ	Anz. Geräte	max. Länge	max. Datenrate	Bitbreite Datenbus	Terminierung
Asynchronous SCSI	8	6 m	5 MByte/s	8 Bit	passiv
Fast-SCSI	8	3 m	10 MByte/s	8 Bit	passiv
Wide-SCSI	16	3 m	20 MByte/s	16 Bit	passiv
Ultra-SCSI	4	3 m	20 MByte/s	8 Bit	aktiv
Ultra-SCSI	8	1,5 m	20 MByte/s	8 Bit	aktiv
Ultra-Wide-SCSI	4	3 m	40 MByte/s	16 Bit	aktiv
Ultra-Wide-SCSI	8	1,5 m	40 MByte/s	16 Bit	aktiv
Ultra-Wide diff.-SCSI	16	25 m	40 MByte/s	16 Bit	aktiv
Ultra2-SCSI	8	12 m	40 MByte/s	8 Bit	aktiv
Wide Ultra2-SCSI	16	12 m	80 MByte/s	16 Bit	aktiv

Tabelle 1: Verschiedene SCSI-Varianten (auch mit unterschiedlichen Steckern)

SCSI-Signalübertragung	Kürzel	Spannungpegel
Single-Ended	SE	+5 V
High Voltage Differential	HVD	+1.5 V -1.5 V
Low Voltage Differential	LVD	+60 mV -60 mV

Tabelle 2: Signalübertragung bei SCSI

– **IEC-Bus** (IEEE 488) für diverse Geräte, auch Prozeßperipherie

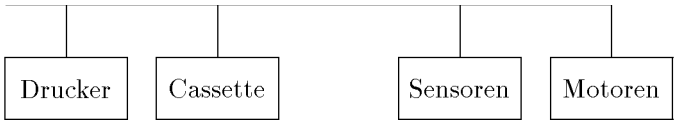


Bild n3p14c

3.4 Standardperipherie

- Ein/Ausgabegeräte für den Benutzer:
  - Terminal, Datenstation, Datensichtgerät (Bildschirm und Tastatur)
  - Drucker
  - Plotter
  - Maus
  - Scanner
- Kommunikation
  - Modem
  - Netzkarten
- Massenspeicher
  - Magnetplatten, Floppydisk
  - CD-ROM
  - Magnetband, Cassetten
  -
- Prozeßperipherie
  - Sensoren
  - Motoren
  - AD-Wandler
  - DA-Wandler

3.4.1 Terminal

Das Terminal besteht aus 2 Einheiten, dem Bildschirm und der Tastatur. Für jede wird ein eigener Satz von Registern (DBR, CSR) zum Senden (transmit, T) und Empfangen (receive, R) benötigt.

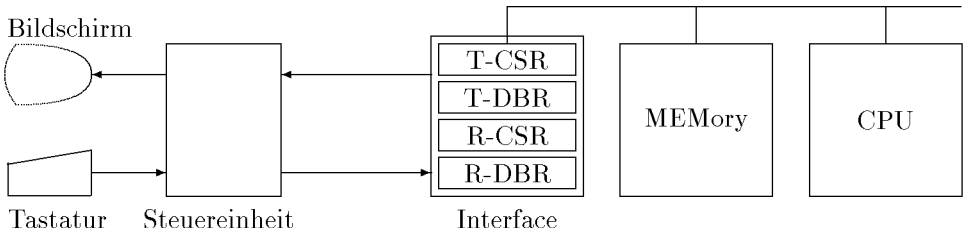


Bild n3p15

a) Der Bildschirm (Monitor)
– Kathodenstrahlröhre (CRT, Cathode Ray Tube)

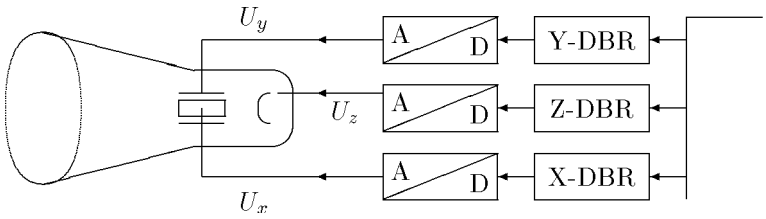


Bild n3p16

Die Ablenkung in X- und in Y-Richtung erfolgt durch Analogspannungen (U_x und U_y), die aus Digitalwerten durch A/D-Wandler erzeugt werden. Die Helligkeit wird durch eine weitere Spannung (U_z) gesteuert. Bei Farbbildröhren werden 3 solcher Spannungen benötigt für die Grundfarben Rot, Grün, Blau, RGB).
In vielen CRT-Geräten wird die X- und Y-Ablenkung mit Schaltungen der Fernsehtechnik realisiert, die anstelle von Analogsignalen nur noch Synchronisationsimpulse (horizontal und vertikal) benötigen.

- Vektorbildschirm (selten und teuer): Jeder Bildpunkt P wird durch geeignete Ablenkspannungen U_x(P) und U_y(P) im RAM-Verfahren erstellt.
- Rasterbildschirm: periodische Ablenkung durch U_x(t) und U_y(t), Helligkeit unfd Farbe werden zum "richtigen" Zeitpunkt durch die Steuerspannung(en) U_z(RGB) erzeugt (SAM).

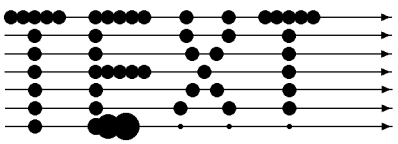


Bild n3p16a

.Kennwerte von (Raster-)Bildschirmen
- horizontale Auflösung: Pixel/Zeile ≈ 1000
- vertikale Auflösung: Zeilen/Bild ≈ 1000
- zeitliche Auflösung: Bilder/Sekunde ≈ 100 (Bildfrequenz)

Zeilenfrequenz: ≈ 100 kHz
Übertragungsrate: $\approx 100 \cdot 10^6$ Hz = 100 MHz
Ablenkgeschwindigkeit (in x-Richtung): $v = \frac{\Delta s}{\Delta t} = \frac{0.5m}{100 \cdot 10^3 s^{-1}} = 50 km/s$
Röntgenstrahlung durch Abbremsen der Elektronen beim Aufprall auf den Bildschirm (U_b = 35 kV), Elektronenaufprallgeschwindigkeit $v_e = 111 \cdot 10^3$ km/s (ca. 1/3 Lichtgeschwindigkeit).
Nachleuchten der Bildschirmpunkte:
– lange Nachleuchtdauer → Schmieren – kurze Nachleuchtdauer → Flimmern

– **LCD-Bildschirme** (Liquid Crystal Display, LCD)
Flüssigkristalle drehen die Polarisation von Licht und lassen sich durch elektrische Felder verändern.
TFT (Thin Film Transistor) Dünnschicht-Transistoren im Kreuzungspunkt von Leiterbahnen legen die elektrischen Felder an die Elektroden und lassen Licht hindurch.

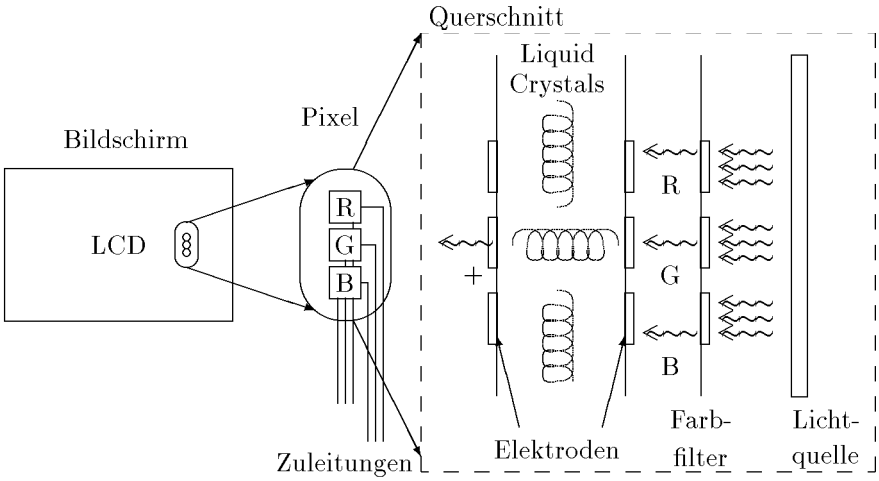


Bild n3p55

Eigenschaften:

- sehr flache Bildschirme
- geringe Leistungsaufnahme
- keine Röntgenstrahlung
- geringer Sichtwinkel (Farbverzerrungen)
- relativ langsam: ca. 20 ms = 50 Hz
- jedes Pixel wird einzeln angesteuert (RAM), vgl. Speichermatrix

b) Die Tastatur Das Drücken einer Taste adressiert ein Wort in einem ROM. Sondertasten (Alt, Shift, Ctrl/Strg) wählen verschiedene Zeichensätze (ROMs) aus

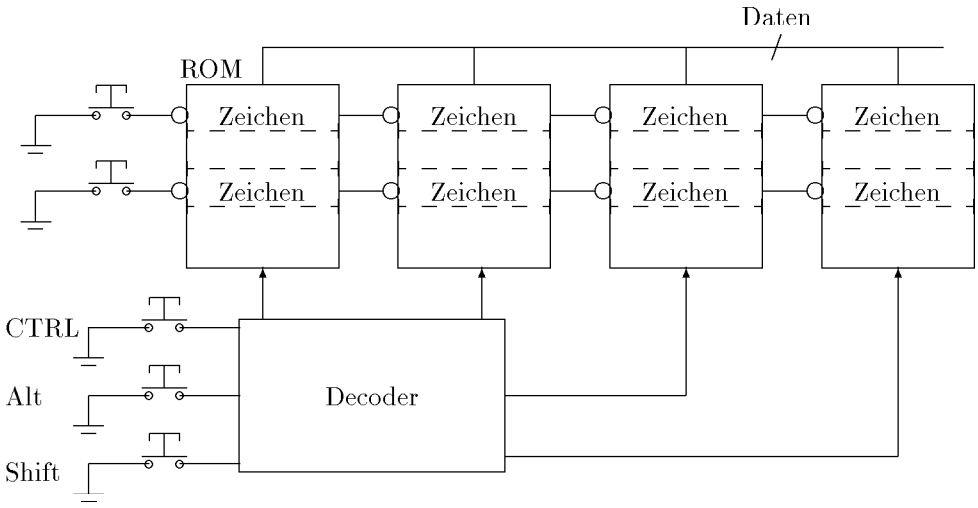


Bild n3p17

3.4.2 Drucker

- Typendrucker (Fernschreiber, Trommel-, Banddrucker)
- Matrixdrucker Pixel → Zeichen → Zeile → Blatt (-Drucker)

Schwärzung, Farbe

- Farbband (anschlagende (impact) Drucker)
- Tinte (nichtanschlagende (non-impact) Drucker)
- Toner (Laserdrucker)
- Wärme (Thermodrucker)

Druckertyp	Druckeinheit	Druckfolge	Druckgeschwindigkeit
Trommeldrucker Ketten-, Banddrucker	Typen	zeichen- parallel	200 - 2400 Zeile/min 3 - 5000 Zeichen/sec
Typenhebel-, -rad-, Typenkorb-, Kugelkopfdrucker		zeichen- seriell	10 - 30 Zeichen/sec
Nadeldrucker Tintenstrahl- Thermodrucker	Punkte	spalten- seriell	30 - 300 Zeichen/sec
Elektrostatische, Laserdrucker		zeilen- parallel	10 - 60 Seiten/min 1 - 200 Seiten/min

3.4.3 Plotter

- Transport von Papier oder Stift in X- und Y-Richtung
- Heben und Senken des Stifts in Z-Richtung
- Auswahl von farben (4. Dimension)

Die Basismaschine ist ein Inkrementalplotter, bei dem ein Zeichenstift (Z) abgehoben und gesenkt werden kann (Z-Richtung). Die Bewegung in X- und Y-Richtung erfolgt Schrittmotoren, die 2-phasig angesteuert werden müssen, und so ein Vorwärts- und Rückwärtsbewegung ermöglichen. Zur Ansteuerung müssen Impulse in einer der beiden Drehrichtungen erzeugt werden:

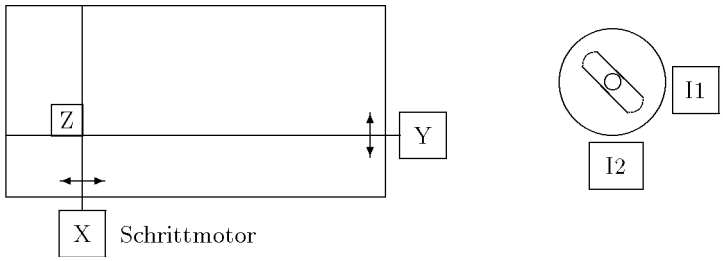


Bild n3p27

Die Impuls erzeugen über Elektromagnete Kraftwir-. kungen auf den (magnetischen) Rotor. In der Praxis werden hier meist 7 Paare verwendet, und die Ansteuerung wird von speziellen Chips übernommen.

. I2	I1	Bewegung
0	0	Ruhe
0	1	+1
1	0	–1
1	1	Ruhe

Die Benutzerschnittstelle wird durch ein DBR realisiert, in welches geeignet Bitkombinationen zu schreiben sind, die über eine Parallelschnittstelle an den Plotter übertragen werden. Zum Erzeugen der Impulse kann ein zusätzliches Startbit (Go) notwendig sein.

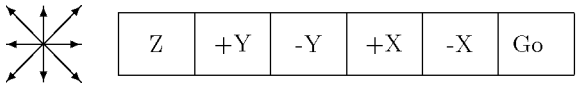


Bild n3p28

Mit der dargestellten Anordnung können von jedem Punkt aus 8 verschiedene Linienelemente benutzt werden. Aus diesen setzt sich dann jede Linie und Kurve zusammen.

Moderne Plotter enthalten in der Regel die notwendige Logik in Form von Firmware, die als Interpreter mehr oder minder abstrakte Plottbefehle ausführen kann (z.B. HPGL). Die Übertragung zum Plotter erfolgt hier meist über serielle Schnittstellen als ASCII-Text.

3.4.4 Maus

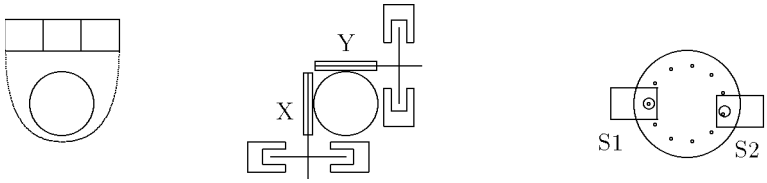


Bild n3p24

Die Bewgung der Maus auf einem Untergrund wird durch die Rollkugel auf 2 orthogonale Achsen übertragen, an denen Lochscheiben mit je 2 Lichtschranken sitzen. Die Lichtschranken müssen um 1/4 Lochabstand versetzt sein, um eine Rückwärts- von einer Vorwärtsbewegung zu unterscheiden (orthogonale Impulsfolgen). Die Taster ergeben Z-Signale (setzen Flags im CSR).

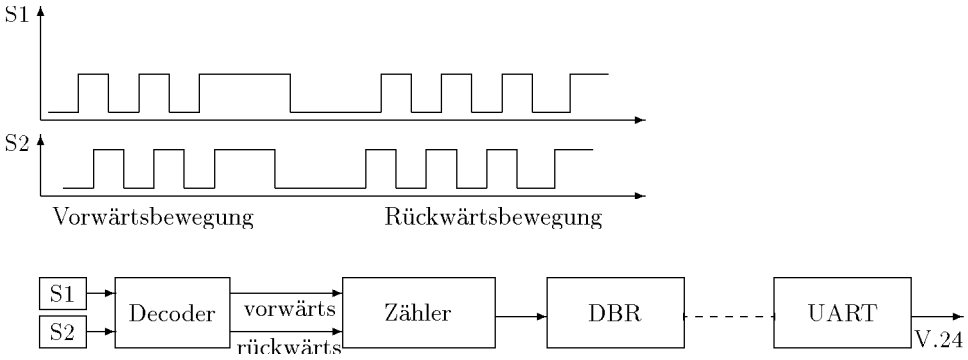


Bild n3p25

3.4.5 Scanner

Optische Abtastung einer Fläche (Vorlage) durch einen oder viele. . Lichtstrahlen und Erfassung der reflektierten Intensität durch Sensoren (Photozellen, -dioden, -transistoren)
- einen Lichtstrahl mit X-Y-Ablenkung (vgl. Plotter)
- eine Zeile von Sensoren oder Leuchtpunkten
- Auswahl von Farben durch Lichtfilter

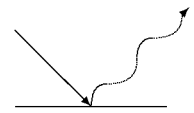


Bild n3p26

3.5 Massenspeicher

3.5.1 Magnetplatten

- Floppydisk (Diskette)
- Festplatte (Plattenstapel)

Magnetplatten Ein Magnetplatte besteht in der Regel aus mehreren Plattenflächen mit je 2 Seiten. Jeder Seite ist ein Schreib/Lese-Kopf zugeordnet.

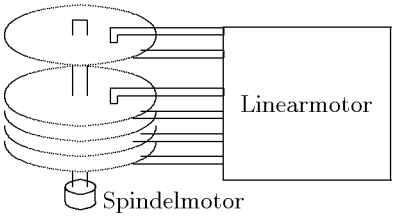


Bild n3p18

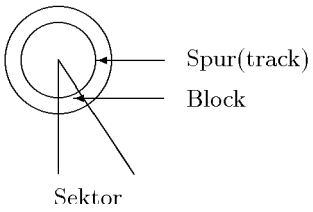


Bild n3p19

Einteilung in Spuren, Sektoren und Blöcke
Ein Datenblock liegt im Schnitt einer Spur mit einem Sektor. Übereinander liegende Spuren, die bei fester Stellung des Linearmotors erreicht werden, bilden einen Zylinder. Jeder Block kann wahlfrei adressiert werden (RAM).

Zugriff:
- auf Sektoren durch die Rotation ($\phi = \omega \cdot t$ mit $\omega = 2\pi n$)
- auf Spuren durch Radialbewegung r(t)

Platten-Kapazität C = B * p * t * s
- s = Anzahl der Sektoren = 10 ... 100)
- t = Anzahl der Spuren (tracks) = 100 ... 1000
- p = Anzahl der Plattenflächen = 1 ... 20
- B = Blockkapazität = 4096 Bit (= 512 Byte = 1/2 KB)

Typisch sind heute: 512 Byte · 20 Seiten · 800 Spuren · 50 Sektoren · = 400 Mbyte

Ein Datenblock wird aus einer (sequentiellen) Folge von Bits gebildet (SAM). Zusätzlich zu den Nutzdaten enthält ein Block weitere Felder: eine Präambel zur Synchronisation, ein Adreßfeld (Seite, Spur, Sektor), eine Prüfsumme und sonstige Daten am Ende (Trailer). Zwischen 2 Blocks liegt immer ein Zwischenraum (Gap), der nicht beschrieben wird.

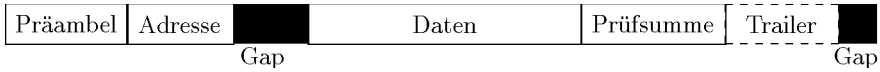


Bild n3p20

Magnetische Aufzeichnung

Beim Schreiben wird durch den Schreibstrom (I) ein Magnetfeld erzeugt, welches auf der Magnetschicht eine Magnetisierung (Φ) hinterläßt, deren Richtung die binäre Information repräsentiert.
Beim Lesen wird an den Stellen, wo ein Flußwechsel in der Magnetschicht erreicht wird, eine Induktionsspannung erzeugt ($U_{ind} = d\Phi/dt$).

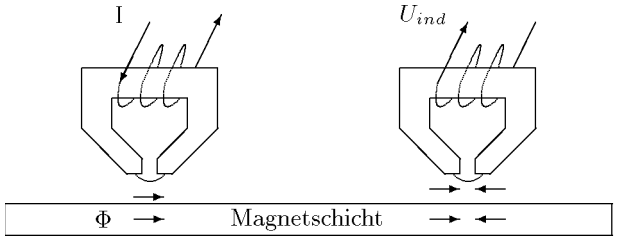


Bild n3p21

Für die Darstellung von Daten können unterschiedliche Verfahren eingesetzt werden (vgl. DIN 66010):

- Richtungsschrift, NRZ (non return to Zero): Flußwechsel bei Datenwechsel:
- Richtungstaktschrift (phase encoding): positiver Flußwechsel bei "1", negativer bei "0" (u.U. werden zusätzliche Flußwechsel eingefügt).
- Wechselschrift (NRZ 1): Flußwechsel bei "1", kein Flußwechsel bei "0".
- Wechseltaktschrift: Flußwechsel für jedes Bit, zusätzlicher Flußwechsel bei "1".

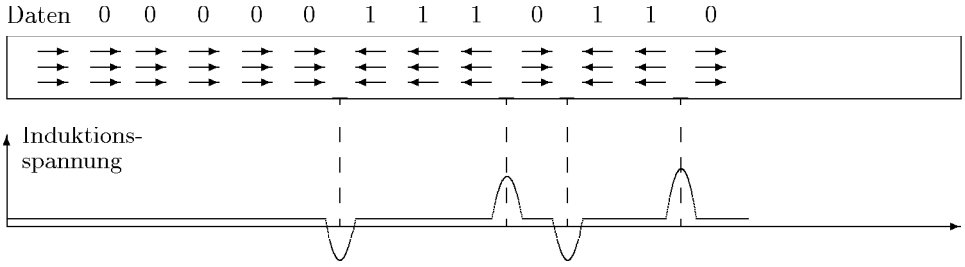


Bild n3p22

Die Darstellung in Richtungsschrift erweist als recht ungünstig, da beim Lesen langer Folgen von gleichartigen Bits kein Lesesignal erzeugt wird.

Aufzeichnungsdichten werden in Bits/mm, bits/inch (bpi) oder Bits/rad gemessen. In Deutschland werden Flußwechsel pro cm bzw. pro Zoll (1- 2.54 cm) oder pro rad ($1rad = 55.6°$, bzw. 1 Umdrehung = $2\pi rad$) angegeben. Die Werte liegen zwischen 800 und 10000 bpi bzw. 5 bis 65 Kbit/rad.
Die (radiale) Dichte der Spuren wird in Spuren/cm bzw tracks/inch (tpi) gemessen.

Bei konstanter Aufzeichnungsrate (bits/s) erhält man eine konstante Aufzeichnungsdichte in bit/rad. Wegen der nach außen zunehmenden Radien auf einer Platte wird die lineare Aufzeichnungsdichte (bpi) nach Außen ab nehmen. d.h. die einzelnen Bits sind weniger dicht gepackt und damit auch weniger störanfällig.

Die Übertragungsrate von der Platte in den Arbeitsspeicher wird im wesentlichen von der Spurkapazität und der Drehzahl bestimmt. Typische Drehzahlen liegen bei 3600 min^{-1}, d.h. 60 /sec. Bei einer Spurkapazität von 100 Sektoren a 4096 bit erhält man ca 24 Mbit/s oder 3 MByte/s.

3.5.2 Magnetbänder

Die Aufzeichnung auf Magnetbänder erfolgt im Prinzip genau so wie bei Magnetplatten, allerdings werden auf einem Band oft mehrere Schreibspuren parallel aufgezeichnet; meist sind es 8 Daten- und eine Prüfspur (z.B. DIN 66015).

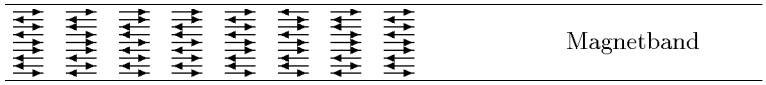


Bild n3p23

3.5.3 Optische Speicherplatten, CD-ROM

Optische Abtastung einer Scheibe durch einen (Laser)Lichtstrahl und Erfassung der reflektierten Intensität durch eine Sensor (Photozelle, -diode, -transistor) in einer Spiralbewegung. Die lineare Schreibdichte bleibt konstant. Der Zugriff kann nicht wahlfrei (RAM) erfolgen, sondern sequentiell von der aktuellen Position aus.

3.6 Abläufe in der ZE

Darstellung mit A-t-Diagrammen

Darstellung des Programmablaufs anhand der Adressen der abgearbeiteten Instruktionen, d.h. dem Inhalt des Program Counters PC (Instruction Pointers IP)

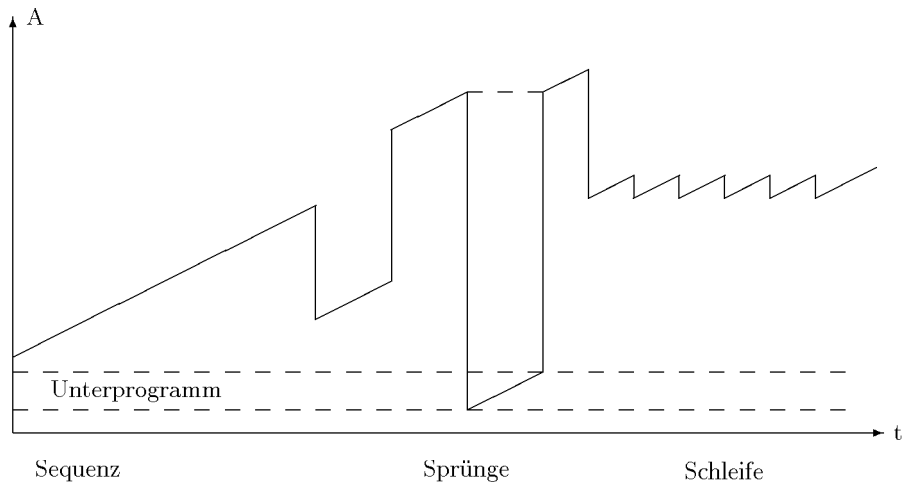
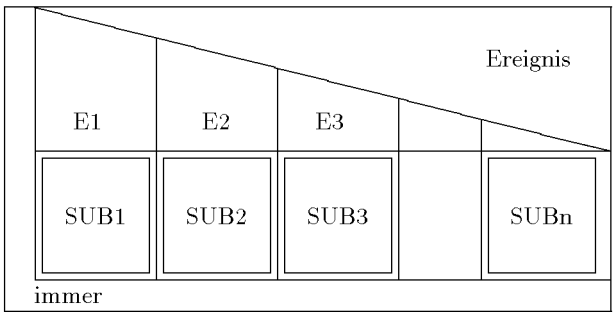


Bild r4p44

- Sequenz (Folge von Instruktionen) ergibt eine Gerade, deren Steigung die Verarbeitungsgeschwindigkeit des Prozessors wiedergibt.
- Sprünge zeigen sich als senkrechte Übergänge
- Unterprogrammsprünge kehren in die vorhergehende Sequenz zurück,
- Schleifen ergeben einen Sägezahnverlauf.

3.6.1 Das Zeitverhalten im Polling-Betrieb

Polling-, Abfrage- bzw. Abrufbetrieb
- eine endlos laufende Abfrageschleife
- bei Eintreten eines Ereignisses wird in ein Bearbeitungsmodul verzweigt, das man sich als Unterprogramm vorstellen kann.




```
100   IF(E1) CALL SUB1
      IF(E2) CALL SUB2
      ..............
      IF(En) CALL SUBn
      GO TO 100
```

Bild r4p45

Als A-t-Diagramm:

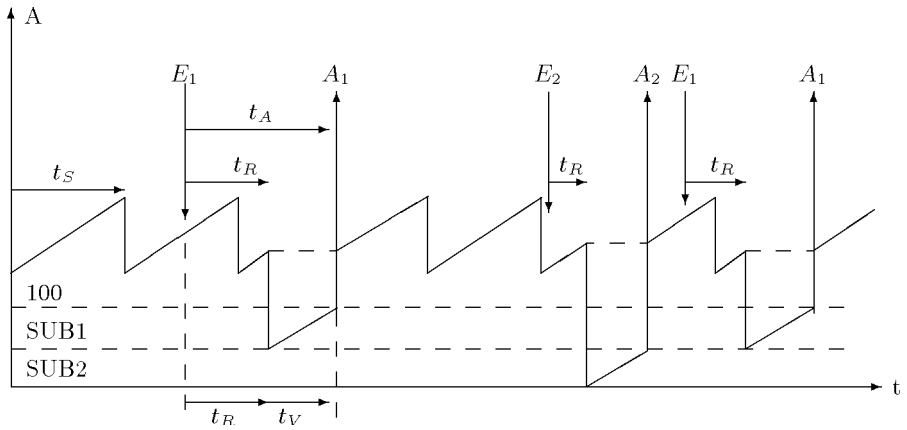


Bild r4p46

Mittlere Reaktionszeit, wenn nur eine Ereignisquelle vorhanden ist (n=1), $\bar{t}_R = t_s/2$.
Die Antwortzeit wird $t_A = t_R + t_V$.
Im allgemeinen Fall ist die maximale Anwortzeit

$$Max(t_A) = Max(t_R) + t_V = t_s + \sum_i t_{Vi}$$

im Mittel wird

$\bar{t}_A \leq t_s + n \cdot \bar{t}_V$ (wobei $\bar{t}_V$ die mittlere Verarbeitungszeit ist)

Da t_s linear mit der Anzahl n der Abfragen (der Ereignisquellen) ansteigt, wird auch t_A linear mit n ansteigen

Problemfälle

a) Falls während eines Durchlaufs der Abfrageschleife 2 oder mehr Ereignisse auftreten, wird ihre Bearbeitung durch die Reihenfolge der Abfragen im Programm bestimmt und nicht durch die der Ereignisse. Ihre Reihenfolge kann also dann verlorengehen, wenn ihr zeitlicher Abstand $\Delta t = t(E_i) - t(E_j) \leq t_s/2$ wird; dieser Wert kann als Zeitauflösungsgrenze verstanden werden.

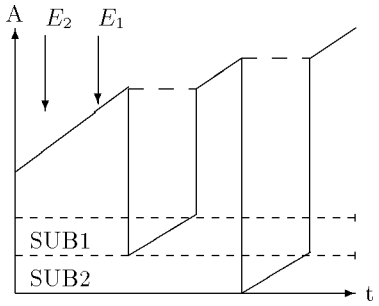


Bild r4p47

b) Falls während eines Durchlaufs der Abfrageschleife 2 oder mehr Ereignisse (E1) aus derselben Quelle auftreten, kann eines davon verlorengehen. Es muß also stets $t_P \leq t_s$ sein.

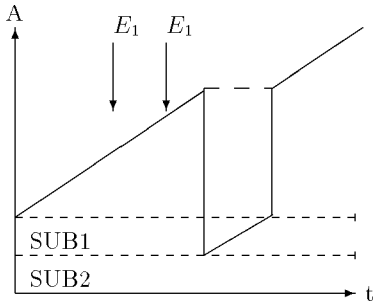


Bild r4p48

3.6.2 Zeitverhalten im Interrupt-Betrieb

Interrupt- bzw. Anforderungsbetrieb
Interrupt-Erzeugung durch Geräte-Anschluß.
Filterung der Interrupt-Anforderungen durch CPU im Interrupt-Verarbeitungswerk (Interrupt Arbitrator, I-A)
– **Maskierung, Verriegelung**
– **Interrupt-Gewichtung** $HWP > SWP$
Bearbeitung des Interrupts durch das Mikroprogramm der CPU
– **Darstellung der Prioritätensteuerung durch P-t-Diagramme**
– **Darstellung der Abläufe durch A-t-Diagramme**

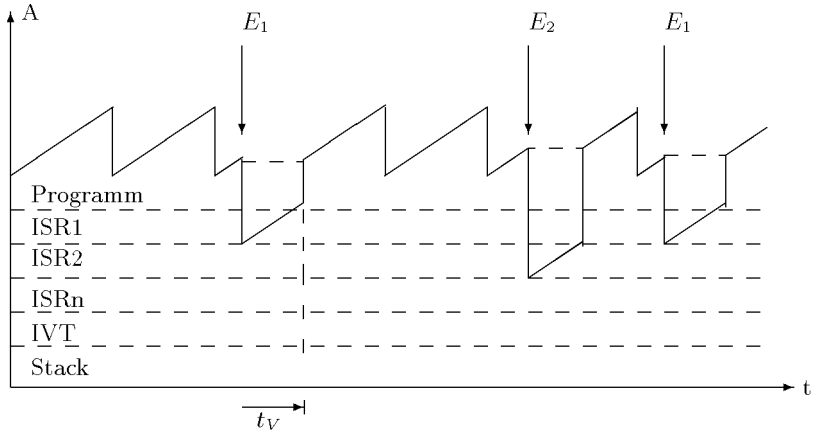


Bild r4p50

Reaktionszeiten
Die maximale Reaktionszeit wird zunächst nur durch die Interrupt-Behandlung bestimmt. Hier ist die längste Instruktion zu betrachten und die Zeit für das Abspeichern des aktuellen PC (und des PSW) auf dem Stack sowie die Zeit für das Lesen des neuen PC (und PSW) aus der IVT. Ingesamt sind dafür 2 (oder mehr) Buszyklen notwendig. Dieser Wert wird im folgenden als vernachlässigbar klein angesehen und $t_R = 0$ angenommen.

Konkurrenzsituationen
Falls während der Bearbeitung eines Ereignisses in einer ISR ein weiteres Ereignis eintritt, das eine Interruptanforderung stellt, werden unterschiedliche Strategien eingesetzt:

a) Die laufende ISR wird grundsätzlich nicht unterbrochen.
Das bedeutet, daß die Reaktionszeit für ein neues Ereignis (E2) einen endlichen Wert annimmt ($t_R > 0$). Im schlimmsten Fall müssen alle anderen Interrupts vorher abgehandelt werden, dann wird die maximale Reaktionszeit wieder
$Max(t_R) = \sum t_{Bi}$
und die maximale Antwortzeit
$Max(t_A) = \sum t_{Bi}$
Die Reihenfolge der Bearbeitung entspricht aber in jedem Fall der Reihenfolge der Ereignisse.

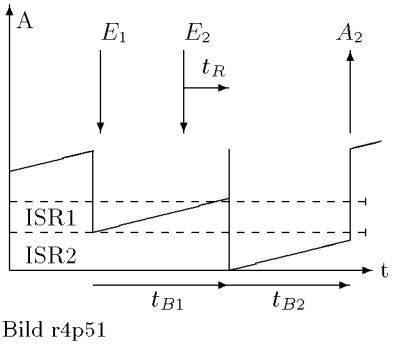


Bild r4p51

b) Die laufende ISR wird in jedem Fall unterbrochen. Hier wird $t_R = 0$. Die Antwortzeit t_A, auf die es aber ankommt, wird im schlimmsten Fall wieder $Max(t_A) = \sum t_{Bi}$
Die Antworten auf die Ereignisse, die Ausgaben A_i können jetzt in der umgekehrten Reihenfolge auftreten, wenn die Ausgaben ganz am Ende der Interrupt Service Routinen erfolgen. Es ist also sehr wichtig, die Ausgaben möglichst schnell zu erstellen und andere Arbeiten hinten an zustellen.
Ein weiteres Problem ergibt sich in diesem Fall daraus, daß mit jeder Unterbrechung neue Werte auf dem Stack gesichert werden müssen, der aber nur eine endliche Größe haben kann. Irgendwann kann ein Überlauf eintreten, der den ganzen Prozeß gefährdet.

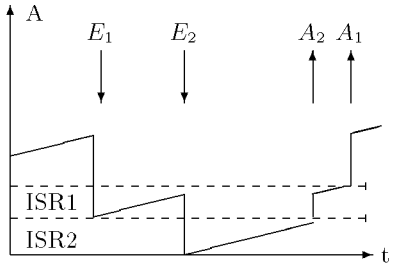


Bild r4p52

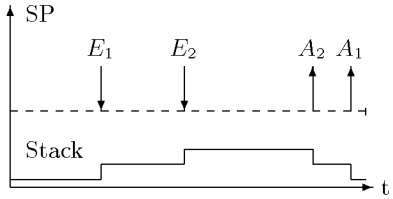


Bild r4p53

c) Die laufende ISR wird nicht in jedem Fall unterbrochen, sondern nur, wenn sie es selber (programmgesteuert) zuläßt. Hier wird im günstigsten Fall die Reaktionszeit $t_R = 0$ und die Antwortzeit $t_A = t_B$ (falls es sich in bevorzugtes Ereignis handelt); im schlimmsten Fall wird die Antwortzeit beliebig groß ($t_A \to \infty$), wenn es sich um ein Ereignis untergeordneter Bedeutung handelt. Hier müssen Prioritäten gesetzt werden, die der Bedeutung der Ereignisse bzw der Prozesse entsprechen.

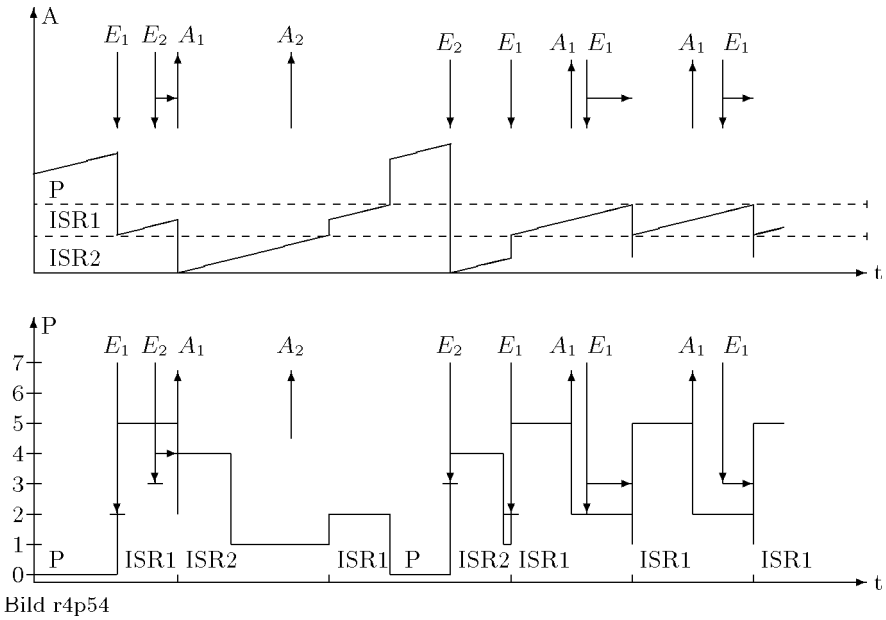


Bild r4p54

Kapitel 4

Betriebssysteme

4.0 Der Rechner als System

Grobstrukturen:
- Hardware: Zentraleinheit und Peripherie
- Software: Programme und Daten

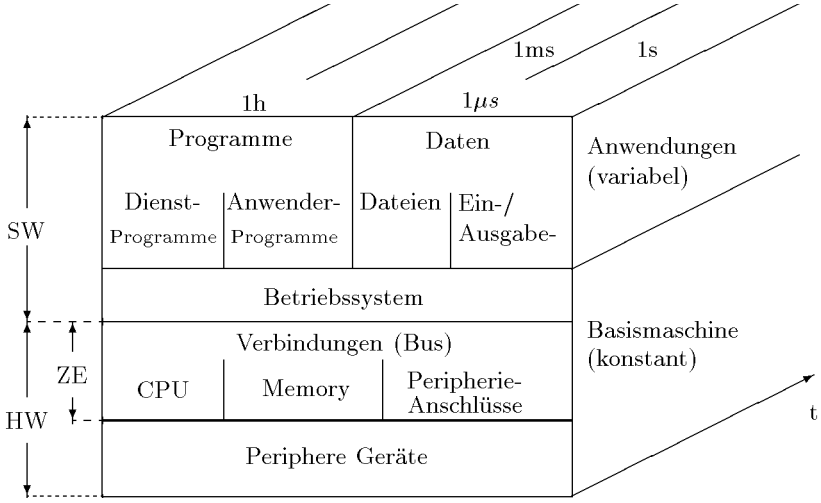


Bild n3p00

- **Dienstprogramme**
  - zur Verwaltung des Rechners (Dateiverwaltung, Systemsteuerung).
  - zur Programmentwicklung (Editor, Compiler, Binder, etc),
- **Anwenderprogramme** zur Produktion:
  Programmentwicklung, Textverarbeitung, Betriebsdatenverarbeitung, Prozeßdatenverarbeitung (CAM = Computer Aided Manufactoring), Betriebslenkung (CIM = Computer Integrated Manufactoring), Entwicklung (CAD = Computer Aided Design), etc.
- **Dateien** auf den Massenspeichern (Magnetband oder -platte) umfassen sowohl Programme, als auch Daten, die von den Produktionsprozessen stammen oder für sie bestimmt sind. Oft sind diese in **Datenbanken** organisiert.

- **Ein- und Ausgaben** vom/zum **Anwender** oder Prozeß über EA-Geräte (Eingaben von der Tastatur, der Maus oder von externen Quellen, Ausgaben auf Bildschirm, Drucker oder Plotter).

Verwaltungsaufgaben:

Verwaltung von Programmen und Daten . .Systemsteuerung: z.B. Booten
Programme laden, starten, koordinieren (→ BS) - LOAD program
Daten erstellen (durch Anwenderprogramme) - INSTALL "
Daten verwalten (durch Dienstprogramme, Betriebssystemfunktionen, Kommandos): - BUFFERS reservieren
- STACKS "

- DIR (ls): Dateien auflisten
- COPY (cp). Dateien kopieren
- RENAME (mv), Dateien umbenennen
- DELETE (rm), Dateien löschen
- TYPE (cat), Dateien anzeigen
- PRINT (lpr), Dateien ausdrucken
- FORMAT (fmt), Formatieren oder Initialisieren von Datenträgern.

Programmentwicklung . .

- **Editoren** zur Erstellung von Quellprogrammen;
- **Compiler** und **Assembler** zum Übersetzen der Quellprogramme in Maschinensprache.
- **Linker (Binder)** zum Anbinden von Bibliotheksroutinen, die nicht im Betriebssystem eingebettet sind, sondern zur Unterstützung einer bestimmten Programmiersprache als Laufzeitroutinen benötigt werden;
- **Loader (Lader)** zum Laden der gebundenen Programme in den Arbeitsspeicher und ihrer Eingliederung in das System; meist ist der Lader in das Betriebssystem vollständig integriert und kein Dienstprogramm mehr.

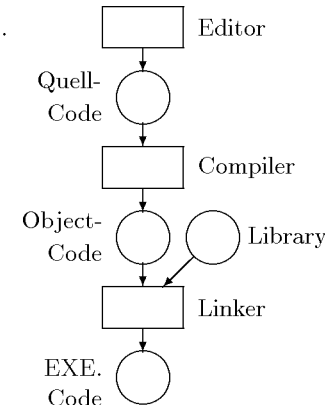


Bild n2p13

- **Testprogramme** zur Kontrolle des Funktionsablaufs und

- **Debugger** zum Auffinden von Programmfehlern (engl.: **bugs**)

- **Dateiverwaltungsprogramme**, die bis zu **Datenbanksystemen** anwachsen können (Workflow-Systeme).

Die Aufteilung der Aufgaben und Funktionen auf Dienstprogramme oder Systemroutinen ist nicht exakt festlegbar und wird in jedem **Betriebssystem** anders gehandhabt.
Zwei Grundfunktionen (Module) müssen im BS verankert sein:

- zur Kommunikation mit dem Anwender (Benutzeroberfläche)

- zum Laden und Nachladen (overlays) des BS von der Systemplatte (engl.: disk, **DOS** = disk operating system) oder von anderen Ressourcen (z.B. von einer CD-ROM oder von einem Server übers Netz).

4.1 Die Benutzeroberfläche

Zur Bedienung des Rechners müssen **Kommandos** an das Betriebssystems gegeben werden. Dazu dient ein Dienstprogramm, das Benutzereingaben annimmt, sie interpretiert und einen entsprechenden Teil des Betriebssystems aktiviert.

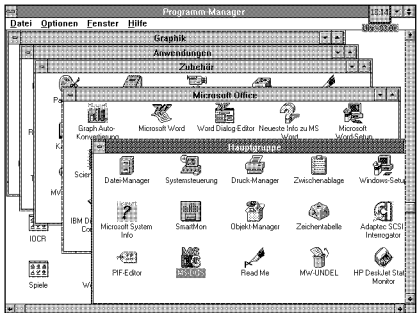
Textorientierte Kommandointerpreter, **Monitor** oder **Schale** (engl.: **shell**)
z.B. unter MS-DOS oder UNIX.

```
H:\FH_FFH\INFO3AI\SKRIPT>dir a:\

 Datentrager in Laufwerk A ist DOS-DISKEDI
 Datentragernummer: 3D5E-14DF
 Verzeichnis von A:\

COMHADD  COM        57.351 30.09.93    6:20
CMDSKEDT EXE        65.677 25.09.98   21:49
CMHDS    EXE        44.124 30.09.98   20:52
CMRCONF  EXE        25.655 30.09.98   18:43
CMDSKEDT HLP         3.792 25.09.98   21:56
LIESHICH TXT         4.474 01.10.98   21:40
EDIT     COM           429 30.09.93    6:20
DOS          <DIR>          30.12.00   13:21
CONFIG   SYS           381 30.12.00   19:25
AUTOEXEC BAT           209 30.12.00   18:11
THP          <DIR>          30.12.00   13:32
CONFIG   SY1         1.557 30.12.00   19:11
       12 Datei(en)        203.649 Byte
                            71.168 Byte frei
H:\FH_FFH\INFO3AI\SKRIPT>
```

Graphische Benutzeroberflächen (GUI Graphical User Interface), auf denen Objekte (Dateien oder Programme) als bildhaften Symbole (Icons) dargestellt und per Mausklick ausgewählt und aktiviert werden. Im Multitaskingbetrieb, bei dem viele Prozesse nebeneinander (engl.: concurrent) ablaufen, werden dem Benutzer mehrere Fenster (engl.: Windows) zur Verfügung gestellt.



In allen Fällen werden mit den Benutzereingaben Dienstprogramme oder -funktionen aktiviert, die die gewünschten Operationen durchführen.
Beispiel: Das Löschen einer Datei

- unter DOS mit dem Kommando `DELETE datei`
- unter UNIX mit dem Kommando `rm datei`
- unter GEM oder WINDOWS durch das "Ziehen" eines Dateisymbols mit der Maus in einen "Papierkorb"(tatsächlich wird damit das Dateiobjekt zunächst nur in einen anderen Ordner verschoben und kann aus diesem - wie aus einem realen Papierkorb - wieder herausgeholt werden. Erst das Leeren des Papierkorbs entfernt die Datei aus dem Dateisystem.)

4.2 Betriebsarten

- **Einzel–Betrieb** (stand-alone)
Im einfachsten Fall besteht die **Software** eines Rechners aus einem einzigen, festinstallierten Programm, das direkt auf die Hardware zugreift. Man findet sie in Kompaktsystemen (engl.: embedded systems), in Speicherprogrammierten Steuerungen (SPS) oder als Firmware in einem „intelligenten“ (μ-Prozessor gesteuerten) Gerät. Allgemein verwendbare Module werden aus einer Bilbliothek hinzugebunden und bilden ein Basis(betriebs)system. Manchmal wird eine reduzierte (und ROM-residente) Variante eines konventionellen BS verwendet (z.B. Windows CE).

- **Stapelbetrieb** (engl.: **batch mode**):
der Benutzer stellt alle Kommandos und alle Eingabedaten zu einem Auftrag zusammen (engl.: **batch job**). Erst nach dessen Beendigung stehen die Ergebnisse zur Verfügung. Dabei ist (bewußt) keine Möglichkeit vorgesehen, in den Bearbeitungsablauf einzugreifen. Lediglich einen Abbruch (engl.: **abort**) kann man erzwingen; dann sind in der Regel aber alle Zwischenergebnisse verloren.
Beispiele: Lohnabrechnungen, Steuerbescheide .
- **Dialogbetrieb** (engl.: **interactive mode**):
der Benutzer macht Eingaben, wenn das Programm läuft und auf Eingaben wartet. Unter Umständen kann er sie sogar kurz vorher eingeben (engl.: **type ahead**), wenn ein Puffer dafür zur Verfügung steht. Die Ausgaben erscheinen, sobald sie berechnet sind, sodaß der Benutzer seine folgenden Eingaben danach ausrichten kann. Beim Multitasking können hierbei Verzögerungen eintreten, wenn mehrere Benutzer auf dieselben Betriebsmittel zugreifen, z.B. auf den Drucker.
- **Realzeitbetrieb** (engl.: **real time processing**):
zeichnet sich dadurch aus, daß bestimmte Antwortzeiten eingehalten werden müssen, d.h. daß auf eine Eingabe innerhalb einer bestimmten Zeit eine Ausgabe erfolgt.

Betriebssysteme und Betriebsarten:

1. Single User	Multi User (Mehrbenutzerbetrieb)
z.B. MS-DOS	Unix
ohne Datenschutz	mit Datenschutz: (Benutzerkennung, Passwort)
2. Single Tasking	Multi Tasking (Mehrprogrammbetrieb)
ein einziges (komplexes)	mehrere Programme in Konkurrenz
Programm in Betrieb	um 1 oder n CPU (concurrency)
	Zuteilungsstrategien (scheduling):
nur Spezialrechner z.B. für	– time-sharing
"Intelligente" Geräte	– resource-sharing
Wetterkarte-Super-Computer	Windows, Unix, MVS(IBM), VMS(DEC)
3. Single Processing	Multiprocessing
Ein Prozessor (1 CPU / ZE)	Mehrere Prozessoren ($n \leq 2^{16} = 65\,536$)
	Massive Parallel Computing
z.B. Prozeßrechner in Geräten	z.B. Hypercube, Suprenum

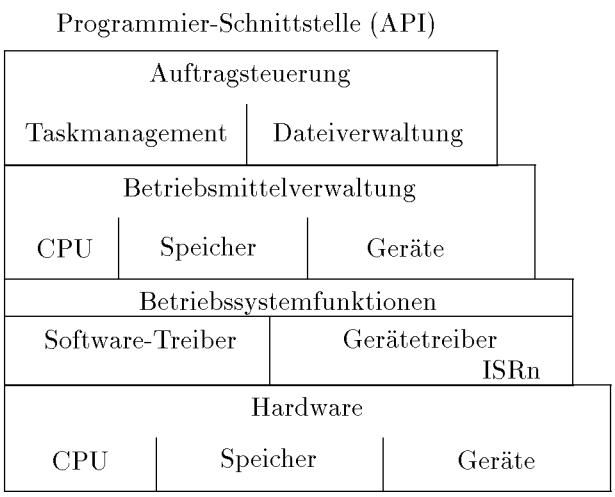
Mehrbenutzerbetrieb (engl.: multi user) erlaubt die Nutzung eines Rechnersystems durch mehrere Benutzer, die gegeneinander abgeschirmt werden müssen, damit keiner einen anderen fahrlässig oder absichtlich stören kann. Diese Problematik wird als "Zugangsschutz" und als "Speicherschutz" im Bundesdatenschutzgesetz (BDGS) behandelt.

Multitasking oder **Multiprogramming** ist der nächste Schritt zu einer effektiveren Nutzung des Rechners. Hier werden die Programme nicht mehr nacheinander abgearbeitet, sondern abwechselnd, so daß alle Benutzer den Eindruck erhalten, gleichzeitig bedient zu werden, oder ein einzelner Benutzer nutzt mehrere Programme gleichzeitig. Tatsächlich wird jedes Programm immer nur für eine kurze Zeit bearbeitet und dann zum nächsten weitergeschaltet. Dazu ist ein ausgefeiltes Taskmanagement notwendig.

4.3 Aufgabe und Struktur eines Betriebssystems

4.3.1 Struktur

Hierarchie von abstrakten Maschinen:



Aufgaben:
Bereitstellung von Funktionen durch Module und Schnittstellen

- Module für den Zugriff auf die Hardware (Gerätetreiber)
- Module für komplexe Rechenoperationen (sin x oder ASCII–HEX) (SW-Treiber)
- Module zur Systemsteuerung (Interrupt-, Task-, Zeit- Management)
- Abstrahierung von der konkreten Hardware (Hardware-Software-Interface)

Bild r3p03

Der Betriebssystemkern (kernel, nucleus) umfaßt

- Taskmanagement (Prozeßverwaltung) mit Tabellen und Algorithmen zu deren Verwaltung, insbesondere für die Taskumschaltung bei Multitasking
- Dateiverwaltung mit Routinen für Dateizugriff und Tabellen für Dateistrukturen.
- Betriebsmittelverwaltung mit Tabellen für die Speicherbelegung, Gerätebelegung, laufende Tasks, Interrupt-Vektoren und Algorithmen zu deren Verwaltung.
- Betriebssystemfunktionen, z.B. Datenkonvertierung, Gerätetreiber, Interrupt Service Routinen (ISR)

Der Betriebssystemkern ist der Hardware am nächsten und muß ihr speziell angepaßt sein (BIOS, Basic Input/Output System).

Der Betriebssystemkern wird oft in einem ROM gespeichert, um größtmögliche Sicherheit gegen ein Überschreiben dieser Software zu haben. Variablen, Tabellen und ladbare Treiber müssen im "RAM" abgespeichert werden. Weitere Teile des Betriebssystems werden beim Starten (Booten) von der Festplatte (disk) geladen (DOS = Disk Operating System).

4.3.2 Taskmanagement

Beim **Multitasking** stehen in der Regel die Programme schon ablaufbereit im Speicher. Dann wird zu jedem Programm ein Speicherbereich zur Ablage von Zwischenergebnissen und Zustandsparametern benötigt, die zum Zeitpunkt der Weiterschaltung vorliegen und 'gerettet' werden müssen; dazu dient zum einen der **Stack** zum anderen der Task Control Block (**TCB**), die den Task-Kontext enthalten. Das Betriebssystem bildet daraus eine Tabelle in Form einer verketteten Liste.

Der Mechanismus zur **Weiterschaltung** von einer Task zur anderen stellt eine besondere Aufgabe dar. Im einfachsten Fall wird die Weiterschaltung zu festen Zeiten vorgenommen (engl.: **time sharing**) und die Programme in fester Reihenfolge abgearbeitet (engl.: **round robbin**).

Eine verbesserte Ausführung bearbeitet die Programme nach ihren Anforderungen und Prioritäten, so daß alle Betriebsmittel möglichst gut verteilt und ausgenutzt werden (engl.: **resource sharing**). Dabei erfolgt die Weiterschaltung auf Grund von Ereignissen (events). Eine wichtige Aufgabe des Betriebssystems besteht nun darin, die **Kommunikation** zwischen den einzelnen Programmen (engl.: **inter task communcation**) zu ermöglichen. Dabei entsteht eine ganze Reihe von Problemen, zu deren Lösung geeignete Strategien erforderlich sind. Da die Zahl der möglichen Beziehungen zwischen mehreren Tasks mit deren Anzahl dramatisch (etwa mit n!) anwächst, kann ein solches System bald unüberschaubar werden.

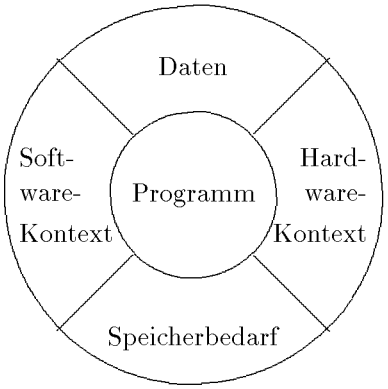


Bild r3p04

. .Eine **Task** besteht aus dem **Programm** und dem **Kontext**, der im **Taskheader** eingetragen ist.
Beim Laden einer Task werden die benötigten und im Taskheader angegebenen Betriebsmittel zugeteilt und im Task Control Block (TCB) abgelegt.

- Der Hardware-Kontext bestimmt die benötigten Geräte (exclusive oder sharable)
- Der Software-Kontext bestimmt die benötigten Systemfunktionen, Prioritäten, Zugriffsrechte, Umgebungsvariablen, Prozeßnummer (pid).
- Die Datenstrukturen werden im Speicher angelegt (Stack, Register)
- Daraus ergibt sich der gesamte Speicherbedarf

Die Aktivierung einer Task bei der Weiterschaltung durch das BS erfolgt als Unterprogrammaufruf, bei dem ein neuer Instruction Pointer (Program Counter) in der CPU geladen wird.

Taskzustände und -übergänge

E_1: Laden und Zuteilen von Betriebsmitteln

E_2: Zuteilung der CPU

E_3: Anfordern von Betriebsmitteln

E_4: Zuteilen von Betriebsmitteln

E_5: Unterbrechung

E_6: Programmende

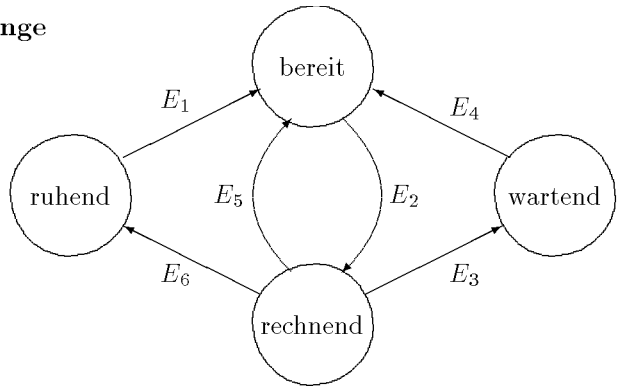


Bild r3p05

Task Scheduling, Algorithmen zur CPU-Zuteilung . .
- preemptiv
- Run-to-Completion
- Round-Robin
- Prioritäts-Scheduling
- Klassen-Scheduling

- zeitgesteuert
- ereignisgesteuert

Prozeßliste: LINUX-Kommando "ps"

pid	ppid	status	name
1234	1	Running	ps
1235	1	Hibernating	xyz
1236	1234	Stopped	rst
1237	1236	Waiting	uvw

. .**Prozeßbaum:** Kindprozesse

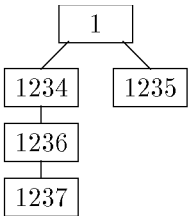


Bild r3p05a

4.3.3 Dateiverwaltung

a) Dateien auf Massenspeichern (files)
- auf blockorientierten Massenspeichern (oder im Arbeitsspeicher)
- Verzeichnisstruktur baumförmig (vernetzt bei UNIX durch links)
- Dateinamen und Pfade
- Dateiattribute ugo(a) * rwx (dlc)

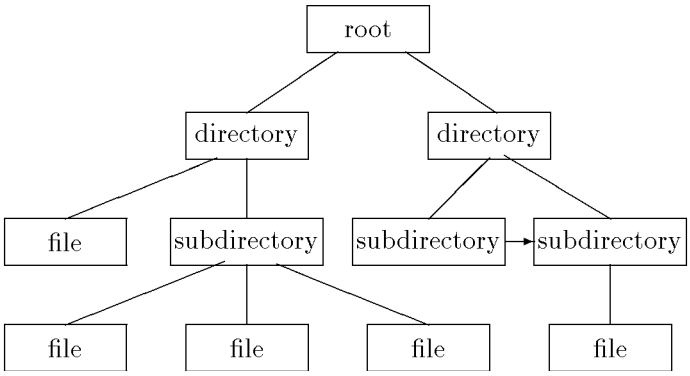
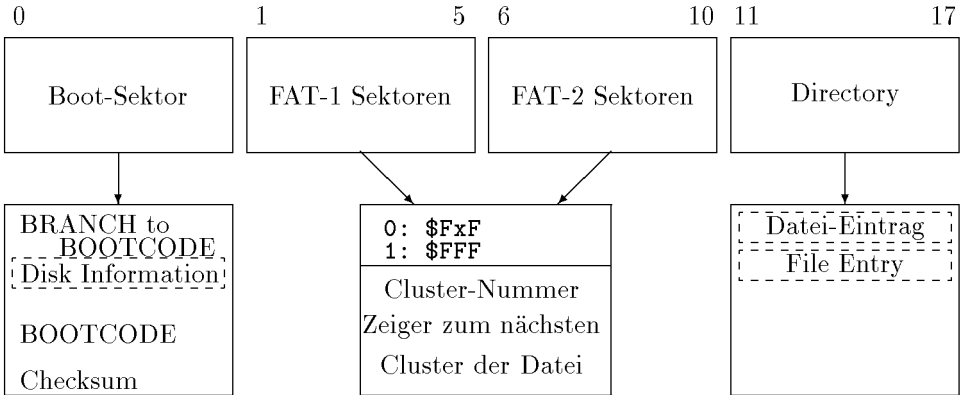


Bild r3p06

b) Ein-/Ausgabedaten
- Gerät = Datei ohne Namen (special file), Pseudo-Datei
- zeichenorientiert
- Standardein- und -ausgabe (Terminal = Tastatur und Bildschirm)

c) Interne Dateien
- files, arrays
- Puffer, Cache für Ein-/ausgabe
- Kommunikation: Pipe, Shared Memory, Semaphore, Signale

d) Beispiel: Das Diskettenformat unter MS-DOS



$ 2	OEM-Code
$ B	Anzahl Bytes pro Sektor
$ D	Sektoren pro Cluster
$ E	Reservierte Sektoren
$10	Zahl der FATs
$11	Anzahl Directory Einträge
$13	Anzahl Sektoren der Disk
$15	Media Deskriptor ($Fx)
$16	Anzahl Sektoren pro FAT
$18	Anzahl Sektoren pro Track
$1A	Anzahl Seiten der Disk
$1C	Anzahl verborgener Sektoren
$1D	Gesamtzahl Sektoren
$23	Gerätenummer
$25	Signatur
$27	Seriennummer
$2A	Disk Name
$32	FAT-ID (12/16)

$xxx	: Next
$FFF	: <EOF>
$000	: Free

0	Dateiname
10	(8+3 Byte)
11	Attribute [1]
12	reserviert (10 Byte)
22	Zeit (2 Byte)[2]
24	Datum (2 Byte)[3]
26	Startcluster (2 Byte*)
28	Länge (4 Byte*)

1: A/O/D/S/H/R
2: h(15..11)/m(10..5)/2s(4..0)
3: J-80(15..9)/M(8..5)/T(4..0)
*: Big Endian

Bild r3p06a

4.3.4 Betriebsmittelverwaltung

- Prozessor (CPU) (auch mehrere bei Multiprocessing)
  Zuteilung durch Task-Scheduler
- Arbeitsspeicher (Mmemory), Puffer (Cache)
  - Segmentation (Speicherschutz durch MMU)
  - Paging (Auslagern von Seiten auf Massenspeicher)
  - Swapping (Auslagern von Tasks auf Massenspeicher)
- Geräte (devices)
  Geräte-Treiber (driver)

4.3.5 Gerätetreiber

Treiber (Gerätetreiber, device driver) sind Software-Module innerhalb des Betriebssystems, die zu Datenübertragung und Kommunikation zwischen Zentraleinheit und peripheren Geräten dienen. Für jedes Gerät bzw. jede Geräteart muß ein Treiber vorhanden sein.

Ein Treiber enthält die Interrupt-Service-Routine (ISR) und Module zur Vorbereitung, Inbetriebnahme und zur Abschaltung des Geräts.

In Realzeitsystemen müssen oft spezielle Treiber für spezielle Geräte (z.B. ADC) installiert werden. Das muß vom Betriebssystem unterstützt werden.

Treiber sollen resident gehalten werden können, d.h. sie sollen dann nicht, um z.B. Speicherplatz für andere Programme zu schaffen, auf Massenspeicher ausgelagert werden. Das kann zu unvertretbar langen Reaktionszeiten führen.

Jeder vorhandene Treiber belegt Betriebsmittel (Speicherplatz, CPU-Zeit). Daher ist es vorteilhaft, wenn Treiber temporär (z.B. beim Booten) ge- oder entladen werden können, um maximale Systemleistung zu erhalten.

Load:	Interrupt Service Routine	Unload:
Init Interrupt Vector Allocate Buffer Enable Interrupts	Get data Save data Fork if not ready Return from Interrupt (RTI)	Disable Interrupts Deallocate Buffer Clear Interrupt Vector
ERROR Handler (Fehlerbehandlung)		

Bild r3p07

4.3.6 Anforderungen

Systemeigenschaften
- Interrupt-Verarbeitung
- Prioritätensteuerung: priorisierte Interrupts, priorisierte Tasks
- residente oder ladbare Teiber
- Resource sharing
- Multitasking
- Multiprozessorfähigkeit
- ROM-fähig
- skalierbare Prozessorleistung
- Skalierbarkeit der Hardware und der Peripherie

Systemfunktionen
- Erfassung von gleichzeitig auftretenden externen Prozeßereignissen
- IPC, Inter Process Communikation: Semaphore (switches, signals, flags), shared memory, pipes, mailboxes
- Dateisystem mit schnellem Zugriff, gesichertem Zugriff und Pufferung (Caching)
- Zeitschrankenüberwachung
- Schutzmechanismen: Speicherschutz, Zugriffsschutz
- Fenstertechnik (Benutzeroberflächen)

Systemmanagement
- Konfigurierbarkeit (Installation von Tasks und Treibern)
- Power-fail Vorkehrungen (bei Netzspannungsausfall)
- Stapelverarbeitung, Batchbetrieb
- Dienstprogramme für Dateiverwaltung und Kommunikation
- Spooling
- Error-Logger
- Backup
- Dokumentation

Unterstützung für die Programmentwicklung:
- definierte/normierte Anwenderschnittstellen (User Interface)
  API (Application Program Interface)
- Entwicklungsumgebung
- Datenbanksystem als Entwicklungstool
- Testumgebung: interaktiver Debugger

4.3.7 Beispiele für Betriebssysteme

Betriebssystem	Prozessor	Hersteller	Anmerkungen
MS-DOS	Intel	MicroSoft	für PCs
Windows	Intel 80x86	MicroSoft	dito
OS/2	Intel 80x86	IBM	für Workstations
UNIX	diverse	AT&T; u.a.	dito
VAX-VMS	VAX	DEC	für "Minicomputer"
OS-9	viele		zur Realzeitverarbeitung
LynxOS	diverse		dito
RTX	M 68 000		dito
RSX-11	PDP-11	DIGITAL (DEC)	dito, historisch
RT-11	PDP-11	DIGITAL (DEC)	dito, historisch

4.4 Systembetrieb

4.4.1 Systemstart (Booten)

Das Einschalten des Rechners erfolgt in mehreren Arbeitsschritten:
- Erstellung eines definierten Anfangszustands (Z_0), insbesondere der Instruction Pointer (IP oder PC, Programm Conter) muß auf einen bestimmten Wert gesetzt werden, der auf einen "Urlader" (Bootstrap) in einem ROM (im BIOS) verweist. Auch der Stack Pointer (SP) muß einen bestimmten Wert erhalten und alle Flags der CPU (im PSW).
  Durch ein geeignetes Signal (INIT) werden alle Geräteanschlüsse über den Bus vom Bootvorgang informiert und haben sich selber in einen definierten Anfangszustand zu bringen.
- Der Urlader führt eine Reihe von Systemtests (auf den Arbeitsspeicher und die Geräteanschlußregister) durch, bevor er das Betriebssystem von der Systemplatte in den Arbeitsspeicher lädt. Dabei kann er Konfigurationsparameter aus einer vom Administrator verwalteten Datei (z.B. config.sys) lesen und verwenden.
- Nach dem Laden des Betriebsystems führt dieses wiederum weitere Tests an Hard- und Software aus, bevor es weitere Systemkomponenten und Dienst- und Anwenderprogramme lädt, wie sie in einer oder mehreren weiteren Konfigurationsdateien (z.B. autoexec.bat) festgelegt sind.
- In den meisten Fällen kann der Bootvorgang durch Benutzereingriff unterbrochen und in interaktive Module verzweigt werden, um neue Systemparameter einzustellen oder Alternativen auszuwählen.

4.4.2 Systempflege

Systemkonfigurierung, Systemmanagement
- Installation von Treibern,
- Entfernen von Treibern,
- Einrichten von Hintergrund-Tasks (Daemons)
- Einstellen von Systemparametern wie Prioritäten von Tasks, Zeitschranken

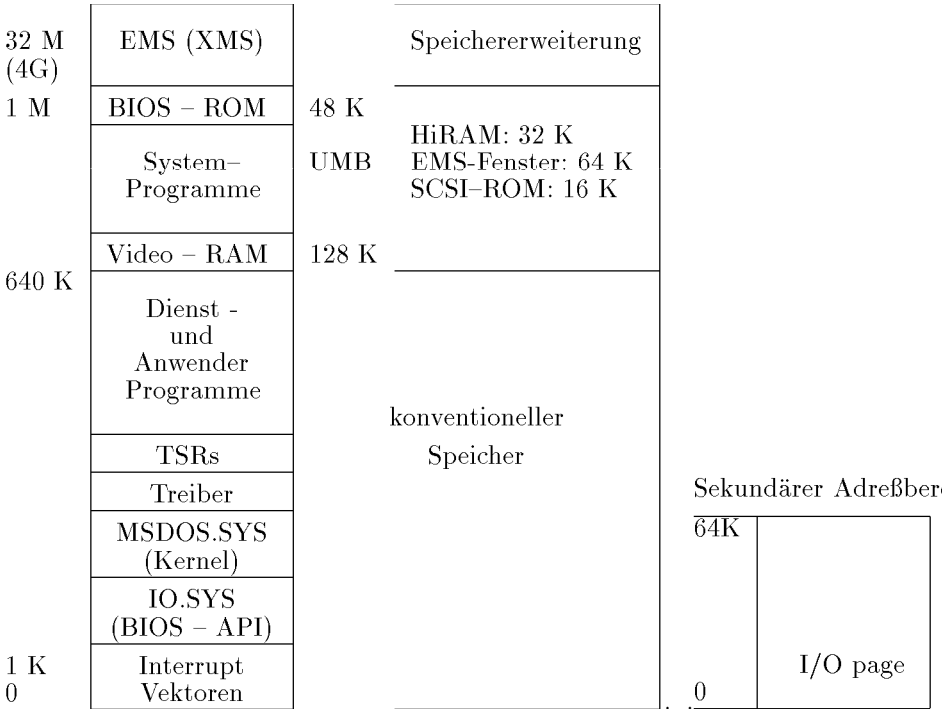
4.5 Aufbau eines Betriebssystems

Konstruktive, statische Aspekte
Typische Bestandteile sind:
- Tabellen, z.B. Interrupt Vektor Tabelle (IVT)
- Datenpuffer, z.B. Video-RAM
- BS-Routinen, wiedereintrittsfähig (reentrant), verschiebbar (position independent)

Die Programme und Daten werden unterschiedlich gespeichert:
- im ROM residente Basisfunktionen, z.B. BIOS (Basic Input/Output System)
- im RAM residente Tabellen, z.B. die Interrupt Vektor Tabelle (IVT), und Hintergrundprogramme, z.B. Daemon-Programme oder TSR-Programme (Terminate and Stay Resident)
- im RAM und auf der Systemplatte austauschbare Routinen (Swapp oder Overlay)

Im Adreßraum liegt das BS meist am unteren oder am oberen Ende. Manche BS (z.B. MS-DOS) machen beides.
Die Geräteregister liegen in einem eigenen Adreßbereich, (Input/Output Page), der entweder ein (gemappter) Bereich im obersten Abschnitt des Adreßraums ist, oder ein eigener, skundärer Adreßraum, der durch ein zusätzliches Adreßbit zu erreichen ist. (Beim Intel 80x86 wird dieses nur durch bestimmte Instruktionen wie IN und OUT aktiviert.)

Beispiel:
Belegung des Speichers unter MS-DOS (V 6.20) (nicht maßstäblich)



Inhaltsverzeichnis