Realzeitsysteme

Erik Jacobson
FH Ffm, Fb MND

Sommersemester 2001
Belegnummern:
06 67 42 (SL)
06 67 48 (PL)

Übersicht

Inhalte

1. Realzeitsysteme: Abgrenzung, Definitionen, Begriffe
2. Modellierung: statisch und dynamisch (u.a. Petri-Netze)
3. Realzeitprogrammierung:
- Software-Strukturen,
- Betriebssysteme UNIX und LynxOS

4. Prozeßdatenverarbeitung:
- Automatisierung
- Modellierung
- Realzeitbetrieb: Zeitbedingungen, Ablaufdarstellungen (A-t-Diagramme)

5. Regelungstechnik: Wirkungen, Übertragungsglieder, Regelung
6. Prozeßrechner-Hardware:
- Die Zentraleinheit: CPU, Bus, Arbeitsspeicher, Anschlüsse (Wh.)
- Standardperipherie: Terminal, Drucker, Plotter, Massenspeicher (Wh.)
- Prozeßperipherie: Digitale und Analoge Ein- und Ausgabe
- Prozeßglieder: Meß-, Stell- und Übertragungsglieder

Literatur

- Banahan/Rutter: UNIX. Hanser 1984
- Bennett; S.: Real-time Computer Control. Prentice Hall 1994
- Dannegger/Geugelin-Dannegger: Parallele Prozesse unter UNIX. Hanser 1991
- Jacobson, E.: Einführung in die Prozeßdatenverarbeitung. Hanser 1996
- Kernighan/Ritchie: Programmieren in "C". Hanser 1983
- Rembold/Levi: Realzeitsysteme zur Prozeßautomatisierung. Hanser 1994

Labor

- Laufzeitmessungen
- COM-Schnittstelle (RS 232C, V.24)
- System API: Timer, Interprozeßkommunikation, ...
- Reale und simulierte Prozesse (Eierkochen,...)
- Einfache Prozeßsteuerung und Datenerfassung

Leistungsnachweise

Prüfungsleistung (PL)und **Studienleistung** (SL)
- Klausur am Semesterende (Minimalanforderung = 50 %)
- testierte Laborberichte für Zulassung zur Klausur

Wiederholungsklausuren: (PL und SL)
- Klausur am Ende WS 2001/02, mit Anerkennung der Laborleistungen (danach verfallen die anerkannten Laborleistungen)

Kapitel 1

Realzeitsysteme

1.1 Abgrenzung und Definitionen

a) **Definition:** "Ein Realzeitsystem zeichnet sich dadurch aus, daß auf ein gegebenes Ereignis (Eingabe) innerhalb einer vorgegebenen Reaktionszeit t_R eine Antwort (Ausgabe, Aktion) erfolgt (EVA-Prinzip). Eine Überschreitung ist nicht zulässig (Realzeitbedingung)."

Beispiele:

– Flugzeug: < 5 ms

– Kernkraftwerk < 5 s

Gegenbeispiele:

– Lohnabrechnung: ca 1 Tag

– Kreditangebot: ca 2 Tage

b) **Anwendungsgebiete:**

- 1. Prozeßdatenverarbeitung, Automatisierungstechnik (hier)
- 2. Datenübertragung, Rechnerkommunikation (→ Rechnernetze)
- 3. Transaktionen in Datenbankanwendungen,
  z.B. Buchungssysteme, Termingeschäfte (→ Datenbanken)

c) **Die Zeit als Variable:**

- Die absolute Zeit
  z.B. MEZ = (Weltzeit, MGT) + 1
  Darstellung als String (Numeral): `jjjj.mmm.dd hh:mm:ss` (ISO)
  `dd.mm.(jj)jj hh:mm:ss` (DIN)
  Die (absolute) Zeit ist veränderlich, aber nicht veränderbar.
- Die relative Zeit = Zeitintervalle
  Darstellung als
  - Zeitdifferenz (eigene Algorithmen !), z.B. 11:58 bis 14:20 = 1:22 h
  - Maßzahl*Maßeinheit (Takt), z.B. 12,3s, 200 ticks, 20*10μs
- Problem der Gleichzeitigkeit (Relativitätstheorie)
  Die neue Randbedingung für die Korrektheit einer Verarbeitung ist die **Zeit**

1.2 Anforderungen an Realzeitsysteme

- Deterministisches Zeitverhalten: $t_R < t_{max}$
  (Anwender-Software und Betriebsysteme, s.a. Kapitel 3)
- Hohe Ausfallsicherheit (Hardware, s. Kapiltel 6)
- Hohe Betriebssicherheit, wenig Programmierfehler
- Gesicherter Wiederanlauf nach Störung

1.3 Realzeitbetrieb

1.3.1 Betriebssicherheit

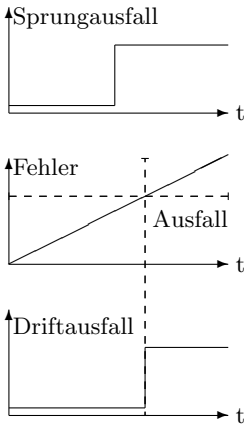


Bild r1p01

Begriffe:

- Fehler: unzulässige Abweichung vom Sollzustand
- temporärer Fehler: vorübergehender Fehler, z.B. Übertemperatur
- permanenter Fehler: bleibender Fehler, z.B. Verschleiß
- Ausfall, Störung: Fehler überschreitet Grenzwert.
- Sprungausfall: ein Fehler führt sofort und ohne Vorzeichen zum Ausfall
- Diftausfall: der Fehler kann vor einem Ausfall erkannt werden

Gefahrenquellen:

– temporäre Störungen der Energieversorgung, Netzausfall
– Verfälschung von Daten bei der Übertragung
– temporäre Hardwarefehler, z.B. Speicherfehler
– permanente Hardwarefehler, Ausfall von Funktionseinheiten
– permanente (!) Softwarefehler

Sicherheitstechnik

– Sicherheit für den Anwender (Bediener)
– Sicherheit für die Anwendung (Maschine, Computer)

Sicherheitsklassen:

- Medizintechnik (höchste Sicherheitsklasse)
- Personenbeförderung
- ...
- Haushaltsgeräte (niedrigste Sicherheitsklasse)

Sicherheitskonzepte

- aktive Sicherheit: im Falle einer Störung werden Sicherheitsmaßnahmen ergriffen.
- passive Sicherheit: im Falle einer Störung sind bedrohte Systemkomponenten geschützt.
- inhärente Sicherheit: eine Störung führt automatisch in einen sicheren Zustand (z.B. Abschaltung); engl.: "fail-save".

1.3.2 Zuverlässigkeit und Verfügbarkeit

Ein System oder Gerät befindet sich immer entweder im Zustand **"betriebsbereit"** (funktionsfähig) oder **"ausgefallen"** (gestört).

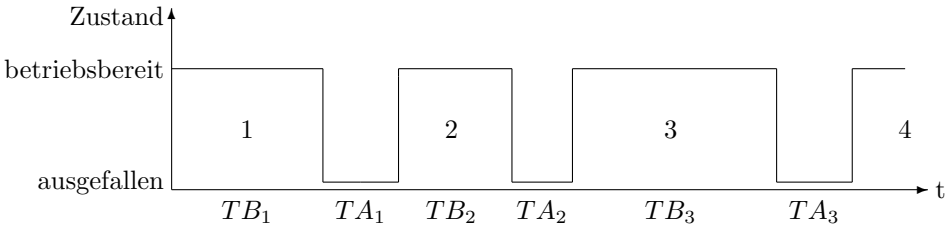


Bild r1p02

a) Bezeichnungen

mittlere Instandsetzungsdauer = **Mean Time To Repair** = **MTTR** = $\overline{TA}$
mittlere Betriebszeit = **Mean Time To Failure** = **MTTF** = $\overline{TB}$
mittlerer Ausfallabstand = **Mean Time Between Failures** = **MTBF** = $\overline{TA+TB}$

b) Begriffe:

- die **Ausfallrate** l (manchmal auch mit dem Buchstaben λ bezeichnet) $l = 1/MTTF = \lambda$
- die **Instandsetzungsrate** $r = 1/MTTR = \rho$
- die **Verfügbarkeit** $p = \frac{MTTF}{MTTF+MTTR} = \frac{\Sigma TB_i}{\Sigma TB_i+\Sigma TA_i}$
- die **Unverfügbarkeit** $q = \frac{MTTR}{MTTF+MTTR} = \frac{\Sigma TA_i}{\Sigma TB_i+\Sigma TA_i}$

Summe der Wahrscheinlichkeiten: p + q = 1.

c) Berechnung der Zeitabhängigkeit der Verfügbarkeit p:
Annahme:
Ausfallrate lund **Instandsetzungsrate** r sind voneinander unabhängig und zufallsverteilt. p(t) und q(t) werden als kontinuierlich veränderliche Größen angesehen.

Ergodentheorem: Zeitlicher Zustand einer (großen) Menge von Einzelteilen entspricht dem zeitlichen Verlauf für ein Einzelteil. p und q beschreiben die Mengen der funktionsfähigen bzw. der ausgefallenen Teile eines Systems.

Bilanzgleichungen

$$\begin{aligned} dp/dt &= -l \cdot p(t) + r \cdot q(t) \\ dq/dt &= +l \cdot p(t) - r \cdot q(t) \end{aligned}$$

$l \cdot p$ = Anteil der pro Zeiteinheit dt ausfallenden Teile, proportional zur **Ausfallrate** l und zur Anzahl der noch funktionsfähigen Teile p.
$r \cdot q$ = Menge der pro Zeiteinheit dt reparierten defekten Teile.

Lösung für den Startwert p(0) = 1, (am Anfang ist alles in Ordnung):

$$\boxed{p(t) = \frac{r}{r+l} + \frac{l}{r+l} \cdot e^{-(l+r)\cdot t}} \qquad q(t) = 1 - p(t)$$

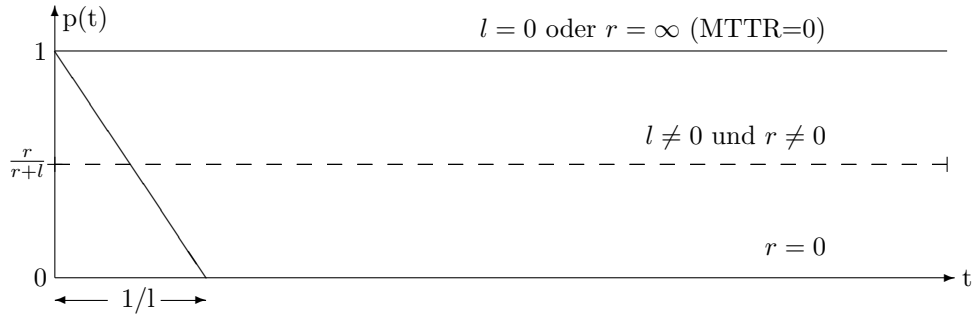
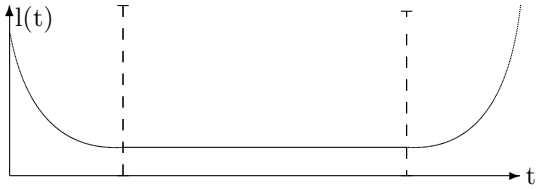


Bild r1p03

Die Zeitskala für solche Vorgänge liegt im Bereich von Monaten oder Jahren.

Die Zeitabhängigkeit der **Betriebsparameter** l und r.

Die **Ausfallrate l** zeigt
„Badewannenkurve“
– zu Beginn **Frühausfälle** (engl.: **burn in**),
– zum Schluß **Verschleißerscheinungen**



Die **Reparaturrate r** zeigt entgegengesetztes Verhalten;
– zu Beginn ist r klein, die Reparaturen dauern länger.
– zum Schluß gibt es weniger Fachkräfte für Reparaturen

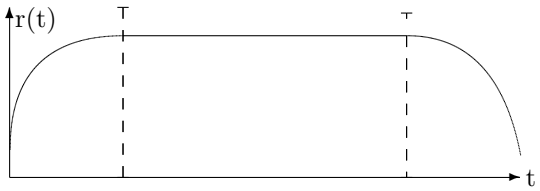
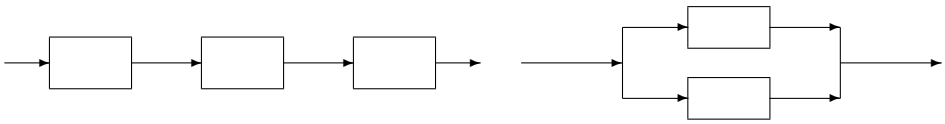


Bild r1p04

Bauteile (z.B. Widerstände, Kondensatoren, Integrierte Schaltungen) werden nicht mehr repariert, $r = 0 \rightarrow p(t \rightarrow \infty) = 0$
MTTF = mittlere **Lebensdauer** $L = 1/l$.
Betriebsbedingungen bestimmen die **Lebensdauer**
(Betriebstemperatur, -spannung, -strom)

1.3.3 Systemverfügbarkeit und -struktur



a) Serienanordnung b) Parallelanordnung

Bild r1p05

Serienschaltung (a) führt zu einer logische UND-Verknüpfung, dann legt ein einziges ausgefallenes Teil das Gesamtsystem lahm.
Parallelschaltung (b) führt zu einem logischen ODER, hier kann der Betrieb auch nach Ausfall von mehreren Teilen weitergehen.

Serienstruktur

Ein **(Rechner)system** ist nur dann voll verfügbar, wenn alle Komponenten funktionsfähig sind. Die **Gesamtverfügbarkeit** p ist

$$\boxed{p = p_1 \cdot p_2 \cdot p_3 \cdot .... \cdot p_n = \prod p_i < 1} \qquad \text{oder} \qquad \boxed{p = p_0^n} \text{ wenn alle } p_i = p_0$$

Die **Unverfügbarkeit** q oder die **Ausfallwahrscheinlichkeit** wird:

$$\boxed{\begin{aligned} q = 1 - p &= 1 - p_1 \cdot p_2 \cdot p_3 \cdot ... \cdot p_n \\ &\approx q_1 + q_2 + q_3 + ... + q_n \text{ da alle } q_i < 1 \\ &= n \cdot q_0 \end{aligned}}$$

Die **Ausfallrate** des Systems wird $\boxed{l = l_1 + l_2 + l_3 + ... + l_n = n \cdot l_0}$
falls die **Ausfallraten** l_i der einzelnen **Komponenten** voneinander unabhängig sind.

Bei sehr unterschiedlichen **Ausfallraten** wird $\boxed{l = l_{max}}$
und das System wird durch das schwächste Glied in der Kette (**Serienschaltung**) bestimmt.

> Ein Beispiel:
> Für elektronische Bauteile (Widerstände, Kondensatoren, o.ä.) werden größenordnungsmäßig Ausfallraten von $5 \cdot 10^{-9} 1/h$ angegeben, für Integrierte Schaltungen (ICs) eine Lebensdauer von ca. 1000 Jahren (10^{-7} 1/h). Eine Baugruppe mit 100 Widerständen, 100 Kondensatoren und 100 ICs hat dann eine Ausfallrate von $(2 \cdot 100 \cdot 5 \cdot 10^{-9} + 100 \cdot 10^{-7}) 1/h = 1,1 \cdot 10^{-5} 1/h$ und eine mittlere Lebensdauer von ca. 10 Jahren (bei optimalen Betriebsbedingungen).

Parallelstruktur:

Die **Systemverfügbarkeit** bleibt so lange erhalten, solange noch mindestens ein Teil funktionsfähig ist;

> z.B. die Ausgabe eines Prozeßrechner mit 3 gleichwertigen Terminals

Die **Ausfallwahrscheinlichkeit q** wird

$$\boxed{q = q_1 \cdot q_2 \cdot q_3 \cdot .... \cdot q_n = \prod q_i \qquad \text{oder} \qquad q = q_0^n}$$

und da alle q_i klein gegen 1 sind, wird q sehr klein.
Die Verfügbarkeit p wird dann fast 1:

$$\boxed{p = 1 - q = 1 - q_1 \cdot q_2 \cdot q_3 \cdot .... \cdot q_n \qquad \text{oder} \qquad p = 1 - q_0^n}$$

Zum Beispiel
wenn die Ausfallwahrscheinlichkeit eines Terminals bei 10% liegt, dann wird die Verfügbarkeit des Systems mit 3 Terminals $p = 1 - 0,001 = 99.9\%$.

m-aus-n-Systeme:
Systeme, bei denen m von n gleichen Einheiten mit der gleichen **Ausfallwahrscheinlichkeit** q_0 zur **Funktionsfähigkeit** erforderlich sind, haben eine **Gesamtverfügbarkeit** von

$$\boxed{p = 1 - q \quad = \quad q_0 \cdot \sum_{k=0}^{n-m} \frac{n!}{k!(n-k)!} (\frac{1}{q_0} - 1)^{n-k}}$$

Für den Fall $m = n$, d.h. wenn alle n Teile funktionieren müssen (Serienschaltung), wird

$$\boxed{p = (1 - q_0)^n = p_0^n}$$

Für den Fall m = 1, d.h. von allen muß eines funktionieren (Parallelschaltung), gilt

$$\boxed{p = 1 - q_0^n}$$

Folgefehler Die Unabhängigkeit der Fehler und Ausfälle von einzelnen Komponenten ist nicht immer gegeben, denn es treten immer wieder **Folgefehler** und **Folgeausfälle** auf. Unter Umständen können durch eine Ursache mehrere Fehler gleichzeitig entstehen, und so eine **Fehlerkaskade** bilden („Ein Unglück kommt selten allein !“).

Schutz hiergegen kann durch inhärente Sicherheit erreicht werden.

1.3.4 Fehlertolerante Systeme

Erhöhung der Verfügbarkeit durch **Redundanz**, d.h. durch eine **Reserve** von funktionsfähigen Geräten, auf die bei Bedarf zurückgegriffen wird. Hierbei hat man mehrere Strategien zur Auswahl:

- **Kalte Reserve** (engl.: **cold standby**): dabei ist das **Reservegerät** oder -system nicht in Betrieb oder wird für andere Aufgaben benutzt. Die Inbetriebnahme erfordert hier u.U. erheblichen Aufwand und Zeit.

MTTR ist groß

- **Warme Reserve** (engl.: **warm standby**): das **Reservegerät** ist betriebsbereit und in Wartestellung, oder es wird für andere Aufgaben niedrigerer Priorität genutzt. Das Reservegerät muß unter Umständen immer wieder mit aktuellen Daten versorgt werden, um im Fehlerfall den Betrieb mit den neuesten Daten weiterführen zu können. Sonst muß der Prozeß auf einen **Synchronisationspunkt** oder **Wiederaufsetzpunkt** (engl.: **restart point**) zurückgeführt werden.

MTTR ist klein

- **Heiße Reserve** (engl.: **hot standby**): für kritische Prozesse, bei denen keine Unterbrechung geduldet werden kann,

MTTR = 0

Entscheidungssysteme, Verfügbarkeitsverbund

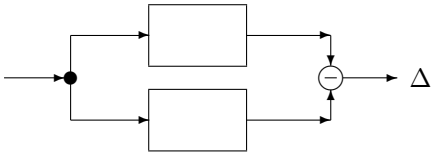


Bild r1p06

Problem der **Fehlererkennung** und **Fehlerlokalisierung**

Bei 2 Systemen, 2 Rechnern, wird ihre Funktion laufend verglichen (-) und bei Ungleichheit ein Fehler angenommen werden. Die Gleichheit bedeutet nicht unbedingt Fehlerfreiheit, beide Systeme könnten ja auch den gleichen Fehler machen. Damit ist aber noch nicht festgestellt, welches System fehlerhaft arbeitet und welches korrekt. Auch der Vergleicher kann fehlerhaft arbeiten.

Mehr als 2 Systeme ermöglichen **Mehrheitsentscheidungen**. Typisch sind **2-aus-3-Systeme**.

1.3.5 Erhöhung von Zuverlässigkeit und Verfügbarkeit

passive Maßnahmen:
- Einsatz von möglichst ausgewählter, zuverlässiger Einzelteile,
- Einheiten mit genügender **Reserve**,
- Geräte in stabilen Ausführungen (Industrie- oder Miltärausführung)

aktive Maßnahmen:
- **Kühlung**
- **Vorbeugende Wartung**

Vorbeugende Wartung (in regelmäßige Abständen t_w) bringt ein System jedesmal in den Neuzustand ($p = 1$). Danach sinkt die Verfügbarkeit p wieder auf einen Wert $p_x = exp(-t_w/L)$ (L = Lebensdauer)

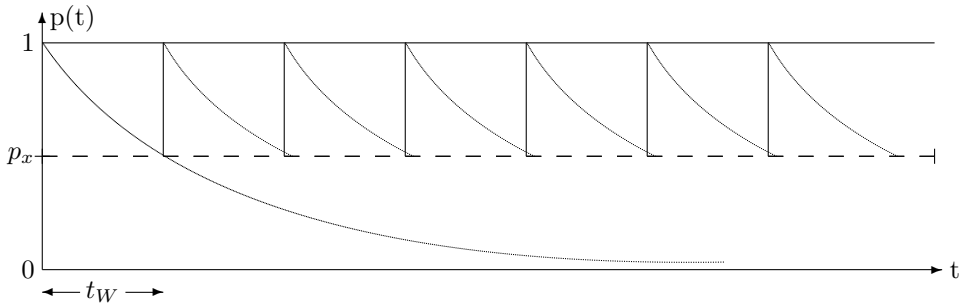


Bild r1p07

1.3.6 Management

Konzeption: (nach SWE-Methoden)
- **1. Voruntersuchung:** Wirtschaftlichkeit (Kosten-Nutzen-Analyse), Machbarkeit (feasibility study), Automatisierungsgrad (Personaleinsparung)
- **2. Istaufnahme:** Prozeßanalyse, Prozeßmodell, Prozeßaufgaben: Meßaufgaben, Steuerungsaufgaben, Bedienaufgaben, Sicherheitsfragen (s. DIN 66 216)
- **3. Sollkonzept:** Prozeßmodell, Prozeßaufgaben; Ausschreibung mit Pflichtenheft

Installation:
Planung von
- **Raumbedarf**
  - **Stellfläche**, **Zugangswege**, **Verkehrsflächen**, **Serviceflächen**.
  - **Anschlußverkabelung** und **Verteileranlagen**
  - Raum für ungehinderte **Wärmeabfuhr**
- **Energiebedarf**
- **Netzstörungen**
  - kurzzeitige, impulsförmige Störungen z.B. beim Schalten von großen Leistungen an schweren Maschinen, Tiefpaßschaltungen (**Netzfilter**)
- **Netzausfälle**
  Unterbrechungsfreie Stromversorgungen (USV), **Notstromaggregate**
- **Klimatisierung:** Leistungsaufnahme wird 100-prozentig in Wärme umsetzt,
- **Schallschutzmaßnahmen**:
  Schallemission (Schallabgabe), **Schallimmission** (Schalleinwirkung)
- Fernhalten von **Staub** und korrosiven **Dämpfen**, evtl. separater Raum.

Betrieb
Sicherheitsvorkehrungen sollten zur Routine werden:
- **Sicherung** (engl.: **backup**) der Datenbestände in regelmäßigen Abständen
  Gesamtsicherung in größeren Abständen (ca. 1 Woche)
  Differenzsicherung in kürzeren Abständen (ca. 1 Tag)
- **Wartungsplan**
- **Logbuch** über alle aufgetretenen Störungen und alle durchgeführten Wartungs- und Reparaturarbeiten, auch mit Rechnerunterstützung **(ERROR-LOGGER)**.

Weitere Schutzmaßnahmen:

gegen **Fehlbedienungen**, Spionage oder gar **Sabotage**
- kritische Programme im **ROM**,
- **Plausibilitätsprüfung** der Dateneingaben,
- **Paßwortschutz** für kritische Eingaben
- **Zugangsschutz** über ein Paßwort beim Einloggen.
- **Zugriffsschutz** auf fremde Daten, z.B. durch eine **MMU**
- **Programmentwicklung** auf separaten Rechnern
  keine **Entwicklungswerkzeuge** auf Prozeßrechnern.
- **Datenschutzgesetz** ist auf Prozeßdaten nicht anzuwenden, sondern nur auf personenbezogene Daten etwa in der Lohnbuchhaltung.

Kapitel 2

Modellierung

2.1 Modellierung von Prozessen

Prozeßmodelle

Modell = abstrakte Darstellung der Realität.
Prozeß = Vorgang zur Umforung, Speicherung und Transport (Verarbeitungsart) von Materie, Energie oder Information (Verarbeitungsgut).

Prozeßbeschreibung

Beschreibung und Gliederung nach unterschiedlichen Gesichtspunkten (Betrachtungseinheiten) als auch nach unterschiedlichem Detaillierungsgrad (Betrachtungsebenen).

a) Statische Beschreibung von Bau- und Funktionseinheiten in Bau- bzw. Funktionsplänen. (Beschreibung der (statischen) Prozeßkennwerte.)

Beschreibungsmittel:

- Text (informell)
- Listen (formell), z.B. Stücklisten
- Diagramme

Graphen, z.B. Bäume, Netze

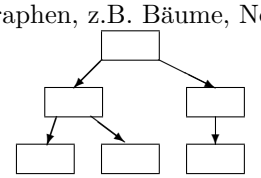


Bild r2p01

. .Schichtdiagramme

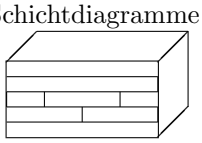


Bild r2p02

- Blockschaltbilder

Wechselwirkung von Funktionseinheiten,
"Materialfluß" steht hier stellvertretend für jeglichen Fluß von Verarbeitungsgut.

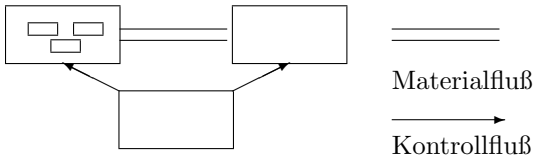


Bild r2p03

Beschreibungsmittel sollen eine (top-down) Strukturierung nach Betrachtungsebenen ermöglichen, d.h. Verfeinerung oder Vergröberung zulassen.
Gegenbeispiel: Text.

b) Dynamische Beschreibung (Wirkungen)

- stationär: kontinuierliche Veränderungen (Vorgänge, Abläufe)
- transient: diskontinuierliche abrupte Veränderungen (Ereignisse)

– **Stationäre Beschreibung** von Vorgängen und Abläufen:
(Beschreibung der Prozeßzustandsvariablen und -parameter)
Beschreibungsmittel:

- Texte
- Tabellen
- Formeln
- Diagramme
  - Funktionsgraphen (Kurven)
  - Flußpläne, z.B. nach DIN 66001, SADT
  - Struktogramme, z.B. Nassi-Shneiderman (DIN 66 262), Jackson

– **Transiente Beschreibung** von Ereignissen
Beschreibungsmittel:

- Tabellen, z.B. Zustandsübergangstabellen
- Diagramme
  - Zustandsübergangsdiagramme
  - Petri-Netze

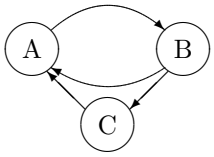


Bild r2p04

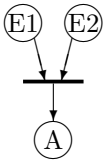


Bild r2p04b

- Impulsdiagramme

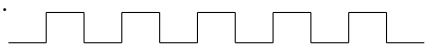


Bild r2p04a

c) schrittweise Verfeinerung (top-down-design)

2.2 Programmbeschreibung

2.2.1 Statische Beschreibung

2.2.1.1 Strukturelemente

Deterministische Abläufe
aus den **Elementartypen**
– **Folge**,
– **Verzweigung**,
– **Schleife**

gemäß
– **Flußdiagramme** (DIN 66 001-PA),
– **Struktogramme** nach
– **Nassi-Shneiderman** (DIN 66 261)
oder
– **Jackson**.

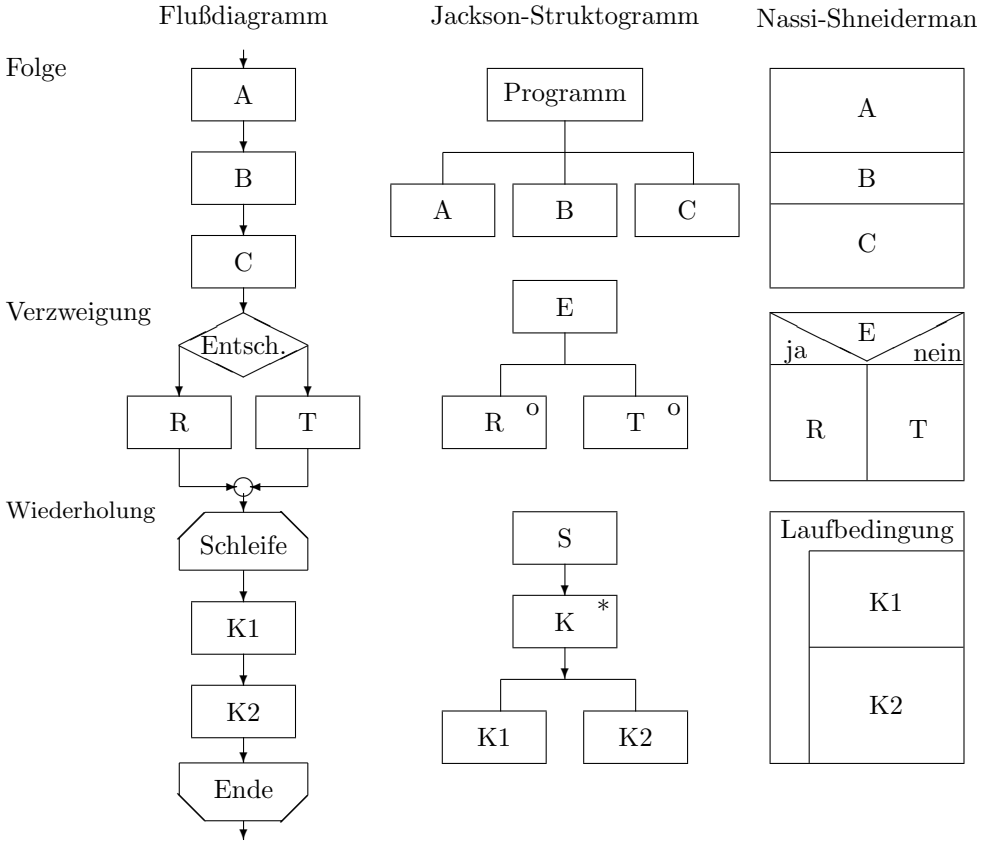


Bild r2p06

Programmstrukturen sind Folge (a), Verzweigung (b) und Wiederholung (c),

Polling und Interrupts

Abrufbetrieb (Polling) einfach und vollständig beschrieben.
Anforderungsbetrieb mit Interrupts ist damit jedoch nicht darstellbar.

2.2.1.2 Module

Statische Verarbeitungsstrukturen: Programmen und ihre gegenseitigen Beziehungen und Abhängigkeiten.
Schichtenmodelle
Aufrufschemata, als hierarchische Bäume aufgebaut (DIN 66 001-PH), oder vermaschte Netze (DIN 66 001-PN).

2.2.1.3 Dualismus von Daten und Verarbeitung

Datenstrukturen und **Verarbeitungsmodule** zeigen **Dualismus**.
- **Datenflußpläne** (DIN 66 001-DF), mit denen Dateien und Verarbeitungsmodule als wechselseitige Verkettung dargestellt werden; d.h. die Dateien als **Datenquellen** und **Datensenken** von Programmen, und Programme als **Erzeuger** (**Produzenten**) oder **Benutzer** (**Konsumenten**) von Dateien.
Statische Datenstrukturen in Programmen und bei der Organisation auf Massenspeichern **(Dateiverwaltung)**: **Strukturbäume** (DIN 66 001-DH)

Jackson-Struktogramme Verknüpfungen von Dateien oder deren gegenseitige Abhängigkeiten durch **Datennetze** (DIN 66 001-DN).

2.2.2 Entscheidungstabellen

(DIN 66 241) keine Verarbeitungsabläufe sondern nur **Verarbeitungsregeln** für die Auswahl von **Aktionen** beim Vorliegen von bestimmten **Bedingungen**.

Tabelle, die horizontal und vertikal in insgesamt 4 Hauptfelder eingeteilt ist, in welche die Bedingungen, Aktionen und Regeln eingetragen werden. In der oberen Hälfte werden Bedingungen zeilenweise aufgeführt, die im **Textteil** (links) beschrieben und im **Anzeigerteil** (rechts) markiert werden.

Entscheidungs-	Regeln					
tabelle	1	2	3	.....	k	
Bedingung 1	Y	Y	Y		N	Bedingungsteil
.....						(wenn ...)
Bedingung n	Y	N	Y	.....	N	
Aktion 1	X	-	X			Aktionsteil
.....						(dann ...)
Aktion m	-	X	X			

Beschreibung (Textteil) — Markierungen (Anzeigerteil)

Eine Entscheidungstabelle enthält 4 Hauptfelder: die beiden linken Felder enthalten Beschreibungen in Form von Texten, die beiden rechten enthalten Markierungen. In vertikaler Richtung gliedert sich die Tabelle in den Bedingungsteil und den Aktionsteil. Diese beiden Felder werden durch die Markierungen mit „wenn.. dann..“-Beziehungen verknüpft.

Binäre Angaben (Ja/Nein) als "begrenzte Anzeiger", aber auch komplexere Beschreibungen.

Bei der binären Darstellung ergeben sich aus den n Bedingungen dann 2^n verschiedene Fälle.

Über die vertikalen Spalten werden als Regeln bestimmte Aktionen zugeordnet, die im unteren Tabellenteil zeilenweise beschrieben sind, indem in deren Anzeigerteil für jede durchzuführende Aktion eine Marke (X) gesetzt wird. Bei m verschiedenen Aktionen ergibt das 2^m Möglichkeiten, die nur dann ausgeschöpft werden, wenn ebensoviel verschiedene (2^n) Fälle vorliegen. Tatsächlich treten hier erheblich weniger verschiedene Regeln auf, und ihre Anzahl kann verdichtet werden, indem manche Bedingungen für manche Fälle als bedeutungslos (–) oder unzulässig (#) markiert werden (s. Tabelle 4.1).

In der **PDV** ordnet man die Bedingungen zusätzlich noch nach **Zuständen** und **Ereignissen**, um statische und dynamische Bedingungen zu unterscheiden.

Besondere Bedeutung:
man kann die **Vollständigkeit** einer Beschreibung feststellen, indem man von rein binären **Bedingungen** ausgeht, die sich gegenseitig ausschließen, und **Regeln** konstruiert, die man dann unter Umständen anschließend verdichten kann.

Entscheidungstabellen Können in einen Verbund gestellt werden, in dem Konstruktionen wie Folge, Verzweigung und Schleife, aber auch Verfeinerungen (Schachtelungen) angewendet werden können (DIN 66 241).

2.2.3 Zustandsdiagramme und -tafeln

Aus der **Automatentheorie**

Beschreibung von Automaten (s. Automatentheorie)

$$Automat = \{\{A\}, \{E\}, \{Z\}, Z_0, \delta, \lambda\}$$

Eingaben Verarbeitung Ausgaben

Bild r2p11

$\{A\}$= Eingabealphabet
$\{E\}$= Ausgabealphabet
$\{Z\}$= Zustandsalphabet
$Z_0 \in \{Z\}$= Anfangszustand
δ= Zustandsübergangsfunktion: $Z' = \delta(E, Z)$
λ= Ausgabefunktion: $A = \lambda(E, Z)$

Zu jedem Zustand Z_i und jeder Eingabe E_j gehört ein Folgezustand Z_f und eine Ausgabe A_{ij}; beim Moore-Automaten genügt es, die Ausgabe, die jedem Zustand zugeordnet ist, anzugeben.

Zustand Z_i	Eingabe E_1	E_2	E_3	E_4	.....	(Ausgabe) A_i
Z_1	Z_x/A_{11}	Z_k/A_{12}	.....			(A_1)
Z_2	Z_y/A_{21}	Z_l/A_{22}	.....			(A_2)
Z_n	Z_z/A_{n1}	Z_m/A_{n2}	.....	.....	.....	(A_n)

Eingaben, Zustände und Ausgaben (s. Tabelle).
- **Eingaben** (E) wirken auf einen Automaten (und auf einen Prozeß) in Form von **Ereignissen** (vgl. **Interrupt**), oder sie werden als Daten eingelesen (vgl. **Polling**).
- **Zustände** (Z) repräsentieren in ihrer Gesamtheit einen Automaten; hier sollen nur Automaten mit endlich vielen, diskreten Zuständen betrachtet werden. Dann befindet sich der Automat immer in einem bestimmten Zustand oder er geht von einem Zustand in einen anderen über, wenn eine Eingabe erfolgt.
- **Ausgaben** (A) können auf zweierlei Art erzeugt werden:

Beim **Moore-Automaten** ist jedem Zustand eine bestimmte, konstante Ausgabe zugeordnet, unabhängig davon, wie dieser Zustand erreicht wurde, d.h. unabhängig von der Eingabe.

Beim **Mealy-Automaten** hängt die Ausgabe außer vom erreichten Zustand auch noch von der Eingabe ab, also vom Weg, auf dem der Zustand erreicht wurde.

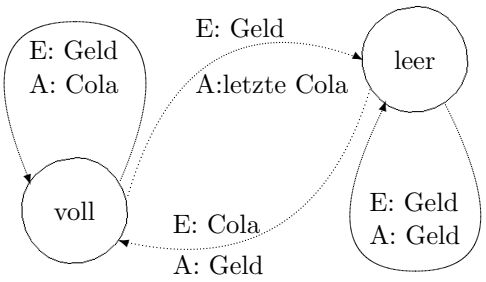
Beim **Moore-Automaten** reicht es aus, für jede Eingabe nur den **Folgezustand** anzugeben und für jeden Zustand die zugehörige Ausgabe in einer zusätzlichen Spalte einzutragen.

Im **Zustandsdiagramm** eines **Automaten** (oder eines Prozesses) werden die **Zustände** als Kreise (**Knoten**) eingetragen, die miteinander durch **Pfeile** (**gerichtete Kanten**) verbunden sind, an denen die **Eingabewerte** angegeben sind. Die

Ausgabewerte werden ebenfalls an diesen Pfeilen eingetragen, wenn es sich um **Mealy-Automaten** handelt, bei denen ja die Ausgabe von der Eingabe abhängt, sonst können sie den Zuständen zugeordnet werden.

In manchen Fällen sind die Ausgaben bei Zwischenzuständen gar nicht interessant, sondern nur der erreichte **Endzustand** und dessen Ausgabe.

Beispiel: Cola-Automat



Eingaben: Geldmünzen, Cola
Ausgaben: Cola, Geldmünzen
Zustände: leer / voll Cola
bereit / defekt
mit / ohne Wechselgeld
Verarbeitung: Geld prüfen, Cola freigeben, Geld ausgeben

Bild r2p12

In diesem Schema können Zustände zusammengefaßt oder verfeinert dargestellt werden, wie es für eine stufenweise **Strukturierung** notwendig ist.

2.2.4 Instanzennetze

„Netze aus Instanzen und Kanälen“(**NIK**) (s. DIN 66 200 T1).

Bild r2p13

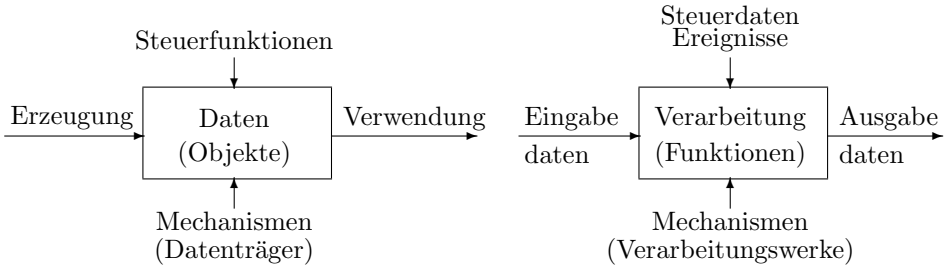
Instanzen repräsentieren Verarbeitungsfunktionen und werden durch Rechtecke dargestellt.
Kanäle repräsentieren Daten und werden durch Kreise dargestellt.
Verbindungen werden durch **Pfeile**, gerichtete Kanten dargestellt, die keine weitere Kennzeichnung tragen.
Ein **bipartiter Di-Graph**: nur unterschiedliche Knotentypen dürfen verbunden werden; die Kanten sind gerichtet.

Beide Darstellungselemente können weiter verfeinert werden; dabei muß darauf geachtet werden, daß am Rand nur gleichartige Elemente auftreten.

Diese Netze sind besonders für den Entwurf geeignet, da sie sich leicht verfeinern lassen.

Sie beschreiben vorwiegend nur **statische Strukturen**

Bei **SADT** (**Structured Analysis and Design Technique)** werden ebenfalls Netze verwendet, um Datenfluß- und Verarbeitungsstrukturen zu beschreiben. Hier werden aber Daten und Verarbeitungen als **komplementär** betrachtet, und alle Vorgänge können jeweils durch ein **Datenmodell** und ein **Funktionsmodell** beschrieben werden. Beide werden jeweils mit denselben, analogen Grundelementen aufgebaut, die als Knoten im Netz stehen und in ihrer Struktur wieder Prozesse darstellen.



a) Datenmodell b) Funktionsmodell

Bild r2p14

Im **Datenmodell** werden die Daten als zentrale Elemente betrachtet, auf welche die **Verarbeitungsfunktionen** wirken, die die Daten erzeugen oder verwenden, und die als **Wirkungspfeile** eingezeichnet werden. Zusätzlich können auch **Steuerungsfunktionen** über Pfeile von oben auf die Daten wirken, z.B. um sie zu verwalten oder zu beschränken. Schließlich werden mit Pfeilen von unten auch Mechanismen zur Realisierung der Daten angegeben, in der Regel die zugehörigen **Datenträger** oder -speicher.

Im **Funktionsmodell** stehen die Verarbeitungsvorgänge im Zentrum und bilden die **Netzknoten**. Verbunden werden sie über die **Datenströme** (in horizontaler Richtung), und gesteuert durch **Steuerdaten** oder **Ereignisse** (von oben). Auch hier können **Realisierungsmechanismen** angegeben werden, z.B. Steuer- oder Rechenwerk.

Diese Darstellungsform eignet sich gut für die Beschreibung von **stationären Abläufen**. Die **komplementäre** Darstellung mit den zwei Modellen zwingt zu besonders exakter Analyse und Beschreibung, wenn beide Formen ohne Widersprüche einander entsprechen sollen, und verhindert dadurch Fehler gerade in der Entwurfsphase.

2.3 Petrinetze

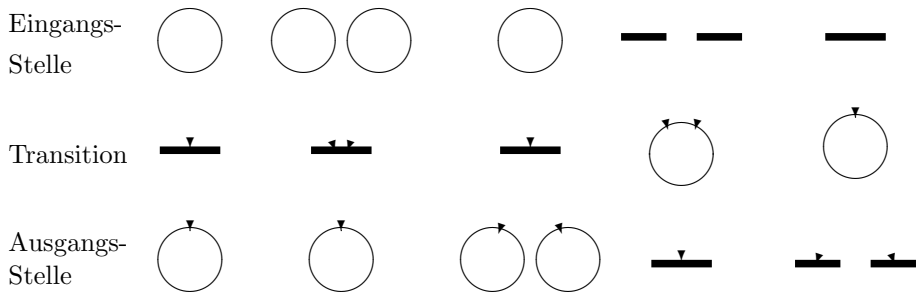


Bild r2p15

Stellen (Kreise) repräsentieren Daten
Transitionen (Balken) repräsentieren Funktionen.
Pfeile, gerichtete Kanten, repräsentieren Datenfluß.

Ein **bipartiter Di-Graph**: nur unterschiedliche Knotentypen dürfen verbunden werden; die Kanten sind gerichtet.

Mathematische Definition: $PN = \{S, T, R\}$
mit

$S = \{S_1, S_2, ...\}$	Menge der Stellen (Kreise)
$T = \{T_1, T_2, ...\}$	Menge der Transitionen (Balken)
$R \subseteq \{(SxT) \cup (TxS)\}$	Menge der Relationen (Pfeile)

2.3.1 Statische Petrinetze

Jede Transition hat mindestens eine Eingangs- und eine Ausgangsstelle. Umgekehrt hat jede Stelle mindestens einen Transition, von der sie bedient wird, und eine oder mehr Transitionen, die sich daraus bedienen.

Beide Darstellungselemente können weiter verfeinert werden; dabei muß darauf geachtet werden, daß am Rand nur gleichartige Elemente auftreten.

Transitionen verbinden dann **Eingangsstellen** (-plätze, oder -kanäle) und **Ausgangsstellen**. Mehrere Eingangsstellen wirken auf eine **Transition** und bilden eine logische **Konjunktion** (UND). Mehrere **Ausgangsstellen** können an einer **Transition** angeschlossen sein, die gleichwertig bedient werden.

In der **komplementären** Betrachtungsweise werden hier **Transitionen** werden über die **Stellen** miteinander verbunden. Wenn mehrere Pfeile auf eine Stelle weisen, bilden sie eine **Disjunktion**, ein ODER.

Stellen und **Transitionen** können weiter zerlegt werden, wie bei den **Instanzennetzen**.

2.3.2 Dynamische, markierte Petrinetze

Dazu werden **Marken** eingeführt, die den **Zustand** von **Plätzen** beschreiben und in der Graphik als Punkte eingezeichnet werden. Sie beschreiben damit die Aktivierung einer Stelle, z.B. als dem Vorhandensein von neuen Daten, oder den Eintritt eines Ereignisses. Je nach Art einer Stelle können auch mehrere Marken vorhanden sein (z.B. in einem Puffer), und ihre Maximalzahl wird als **Kapazität** dieser Stelle bezeichnet.

Schaltregel:

Wenn alle **Eingangsstellen** einer **Transition** jeweils mindestens eine **Marke** tragen, dann wird die **Transition** aktiviert und schaltet. Beim **Schalten** wird aus jeder beteiligten **Eingangsstellen** genau eine **Marke** entfernt und in jeder angeschlossenen **Ausgangsstelle** eine neue Marke gesetzt.

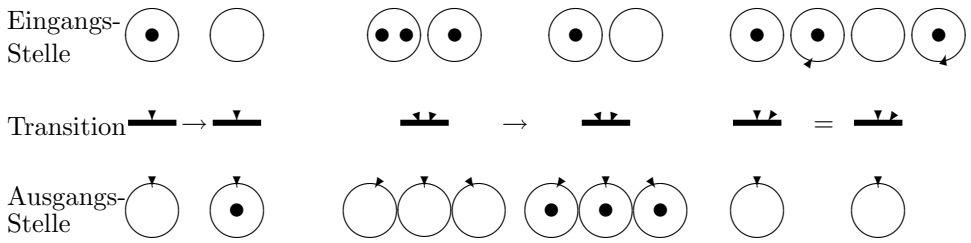


Bild r2p16

Dieses „Markenspiel“ ist schriftlich etwas umständlich darzustellen, aber durch geeignete Programme läßt sich das zeitliche Verhalten eines solchen Netzes, seine Dynamik, hervorragend darstellen und untersuchen.

Benötigt wird:

1. eine Beschreibung des Systems:

$S = \{S_1, S_2, ...\}$	Menge der Stellen (Kreise) in einem Array
$T = \{T_1, T_2, ...\}$	Menge der Transitionen (Balken) in einem Array
$R \subseteq \{(SxT) \cup (TxS)\}$	2 Mengen der Relationen (Pfeile)
$\{f_i\} : \{S_{ij}\} \rightarrow T_i$	Eingangsstellen für die Transitionen
$\{g_i\} : \{T_{jk}\} \rightarrow S_k$	Ausgangsstellen von den Transitionen

2. ein Arbeitsfeld:

S_1	S_2	S_3	S_4	S_5	T_1	T_2	T_3	T_4
0	1	0	0	0	0	0	x	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Zur Vereinfachung wird oft ein zusätzlicher Wirkungstyp eingeführt, als eine **Freigabe** oder **Verriegelung** (dann mit einem Minuszeichen versehen), die als eine unterbrochene Linie eingezeichnet wird und in einer vollständigen Darstellung durch 2 gegenläufige Pfeile wiedergegeben werden kann. Hier wird die Marke nicht 'verbraucht', sondern nach Verwendung wieder hergestellt. Manchmal wird das auch als **Triggern** oder als **Kommunikationskopplung** bezeichnet.

Mit einem solchen markierten Netz lassen sich dann auch Untersuchung wichtiger globaler Eigenschaften eines Netzes:
- ob aus einer bestimmten Anfangsmarkierung jede Transition mindestens einmal aktiviert wurde; ein solches Netz heißt dann „**lebendig**“;
- ob sich die Zahl der Marken im Verlauf der **Aktivierungen** verringert, bis keine **Transition** mehr schalten kann, dann liegt eine **Verklemmung** (engl.: **dead lock**) vor; oder
- ob sich die Zahl der **Marken** so vermehrt, daß die **Kapazitäten** von Stellen überschritten werden;
- ob alle Abläufe, alle **Schaltvorgänge eindeutig** sind oder nicht.

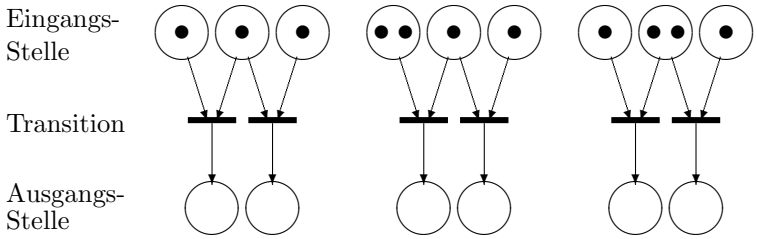


Bild r2p17

Petrinetze mit Konflikten In Petrinetzen können sich Konflikte andeuten, wenn sich Anfangssituationen einstellen, die nicht zum eindeutigen Schalten von Transitionen führen (links); dafür ist nicht die Gesamtzahl der Marken in den Eingangsstellen ausschlaggebend (mitte), sondern ihre richtige Verteilung (rechts).

2.3.3 Petri-Netz Varianten

Gewichtete Kanten:

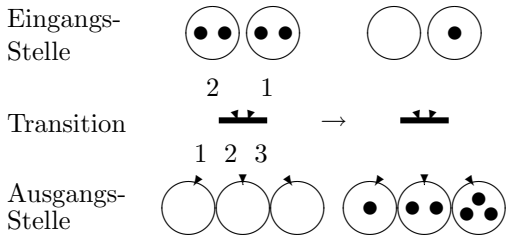


Bild r2p16a

Es werden mehrere Marken zum Schalten benötigt.

2.3.4 Anwendungen

Multitasking
Koordination von 2 Tasks durch eine gemeinsame Datenstelle, ein **Semaphor** (**Signalträger**). Z.B. 2 Programme in einem **Time-sharing-System**, die beide einen **kritischen Abschnitt** (S11 und S21) enthalten, die voneinander ausgeschlossen werden müssen (mutual exclusion), z.B. Ausgabe auf einen gemeinsamen Drucker. Das heißt, daß eine Task (z.B. 2) in diesen Abschnitt (S21) erst eintreten darf, wenn die andere Task (1) die entsprechende Stelle (S11) verlassen hat.

In der Ausgangsposition (volle Marken) ist die Task 2 in **Wartestellung** (S23) und kann über die Transition T21 erst weitermachen, wenn das Semaphor (S) eine Marke erhält. Dies erfolgt aber erst, wenn die Task 1 den kritischen Abschnitt (S11) beendet und die Transition T12 geschaltet hat (gekreuzte Marken). Erst danach kann die Task 2 in den kritischen Abschnitt eintreten, und man erhält eine Situation (offene Marken), die bezüglich der Tasks 1 und 2 gerade vertauscht ist. Und das Spiel beginnt von vorne.

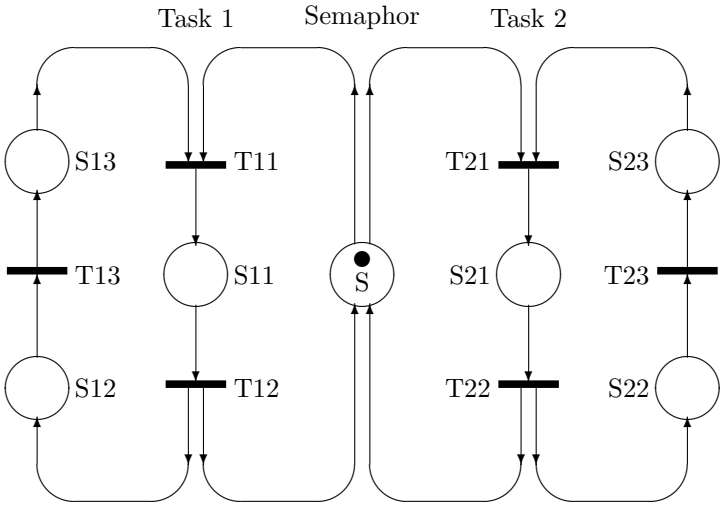


Bild r2p18 **Petrinetz mit Semaphor**

Die Kopplung von 2 Vorgängen, Task 1 und Task 2, kann über einen Signalträger, ein Semaphor erfolgen. Eine Anwendung wird im Text beschrieben.

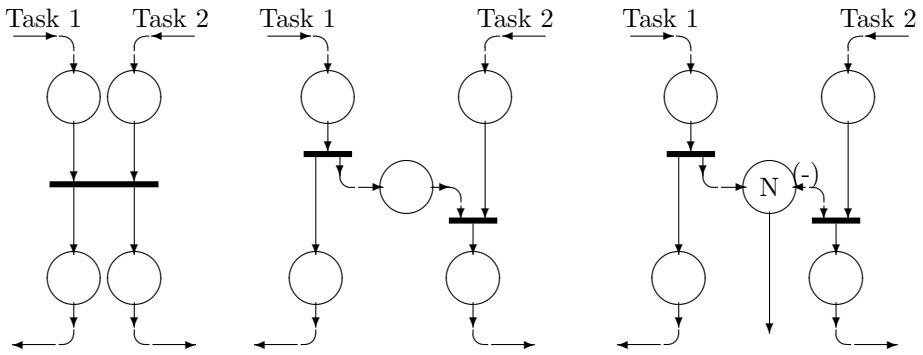
Dieses Beispiel soll zum einen die praktische Anwendbarkeit der Petrinetze für die **Modellierung** von technischen Prozessen zeigen. Zum anderen soll es auf das gedankliche Problem hinweisen, daß nämlich **Stellen** und **Transitionen** im Netz als unterschiedliche Einheiten auftreten, wenn sie verschieden genutzt werden, in der Realität aber dieselben sein können. Und schließlich sollte es zeigen, daß Transitionen und Stellen als technische **Betriebsmittel** anzusehen sind, denen aktive und passive Eigenschaften zugeschrieben werden können. Diese Betriebsmittel können von verschiedenen (Teil-)Prozessen gemeinsam benutzt werden, müssen dann aber jeweils als eigenständige Einheiten getrennt dargestellt werden.

Petrinetz mit Synchronisation

Man spricht dann auch von **Nebenläufigkeit**. Damit in diesem Sinne 2 Tasks gleichzeitig starten oder sonst in ihrem Ablauf bestimmte Abschnitte gleichzeitig durchlaufen, müssen sie durch eine gemeinsame Instanz synchronisiert werden. Das geschieht in der Regel durch eine gemeinsame **Transition**, die genau dann weiterschaltet, wenn beide Eingangsstellen besetzt sind, d.h. beide Tasks die entsprechenden Zustände erreicht haben. Dann werden beide Tasks gleichzeitig fortgesetzt und laufen parallel weiter.

In diesem Fall hat man eine **symmetrische Synchronisation**, d.h. beide Tasks sind gleichwertig und müssen jeweils aufeinander warten. Im weiteren Verlauf können dann solche **Synchronisationen** noch beliebig oft durchgeführt werden, falls es nötig ist. Bei der Verfeinerung der synchronisierenden Transition kann man noch verschiedene Konzepte unterscheiden, wie das **Monitorkonzept** oder das **Rendezvouskonzept**, auf die hier aber nicht mehr eingegangen, sondern auf die Spezialliteratur verwiesen werden soll.

Bei anderen Anwendungen jedoch kann eine **unsymmetrische Kopplung** erforderlich werden, bei der z.B. eine Task 2 auf eine Task 1 warten muß, aber nicht umgekehrt. Das kann man wieder über eine **Semaphorstelle** erreichen, die eine **Nachrichtenübermittlung** bewirkt. Wenn diese Stelle nur zur **Freigabe** oder **Verriegelung** dient, muß sie durch eine andere (zentrale) Instanz irgendwann einmal wieder geleert werden.



a) symmetrische Synchronisation b) asymmetrische Synchronisation c) Nachrichtenstelle

Bild r2p19

Die Synchronisation von 2 Tasks ist symmetrisch, wenn beide gleichberechtigt daran teilnehmen (a). Eine asymmetrische Synchronisation liegt vor, wenn eine Task einer anderen untergeordnet ist und auf sie warten muß (b). Die Synchronisation kann auch durch eine Nachrichtenstelle erfolgen (c), die von einer oder von beiden Tasks bedient wird.

Kapitel 3

Realzeitprogrammierung

3.1 Rechnersysteme

<table>
<tr><td colspan="2">Programme</td><td colspan="2">Daten</td></tr>
<tr><td>Dienst programme</td><td>Anwender Programme</td><td>Dateien</td><td>Ein-/Ausgabe daten</td></tr>
<tr><td colspan="3" rowspan="2">Betriebssystem</td><td></td></tr>
<tr><td>Treiber</td></tr>
<tr><td colspan="4">Hardware</td></tr>
<tr><td colspan="2">CPU</td><td>Memory</td><td>Peripherie</td></tr>
</table>

- Basismaschine: Hardware + Betriebssystem
- Dienstprogramme: Shell, Editoren, Compiler, Linker
- Anwenderprogramme: Textverarbeitung, Graphikerstellung, DTP
- Dateien: interne (im Arbeitspeicher, externe (auf Datenträgern)

3.2 Realzeit-Software

3.2.1 Anforderungen

- Fehlerfreiheit, Korrektheit
- Realzeitfähigkeit: Einhalten der Realzeitbedingungen
- Sicherheit muß manchmal vor Schnelligkeit gehen, z.B. deas Öffnen und Schließen von Dateien, in die Meßdaten geschrieben werden.
- Robustheit: insbesondere gegen unerwartete Ereignisse und Fehlbedienung
- Ergonomische Benutzeroberfläche (hilft Fehlbedienungen zu vermeiden)
- Plausibilitätsprüfungen: Eingaben müssen in einem vorgegebenen Bereich liegen
- Paßwort-Schutz bei kritischen Eingaben
- Bewertung von Ereignissen, nach Prioritäten
- Selbstlernende Programme

3.2.2 Entwicklungsstrategien

- Top-Down-Design
- Modularisierung
- Strukturierung
- Modellierung (statisch, dynamisch)
  - statisch
    - Blockschaltbilder, Funktionspläne (DIN 66001)
    - Graphen, Strukturbäume, -netze
  - dynamisch
    - Petri-Netze
    - Impulsdiagramme
    - A-t-Diagramme
    - P-t-Diagramme
    - Entscheidungstabellen (DIN 66241)
    - Nassi-Sheiderman-Struktogramme (DIN 66261)
    - Flußdiagramme (DIN 66262)
    - Zustandsdiagramme (Endliche Automaten, FSM = Finite States Machine)
- Dokumentation (begleitend, abschließend)

3.2.3 Realzeitumgebungen

Zielsysteme:

- Zielsysteme mit Realzeit-Betriebssystem
- Entwicklung auf Zielsystem oder identischem Entwicklungssystem
- Entwicklung auf einem unterschiedlichen Entwicklungssystem; hier müssen Cross-Compiler oder -Assembler vorhanden sein, die den Maschinencode für das Zielsystem generieren. Das Testen der Software kann meist nur auf dem Zielsystem vorgenommen werden.
- Zielsysteme ohne Betriebssystem (stand-alone)
- Entwicklung auf einem Entwicklungssystem und anschließende Übertragung (down load).

Ungewöhnliche Zielsysteme:
Firmware für periphere Geräte, wie z.B. Maus und Plotter,
Steuermodulen in der Raumfahrt
Ungewöhnliche Randbedingungen z.B.
sehr geringe Speicherkapazität, geringe Prozessorleistung (4 bit Wortbreite) oder starke Störeinwirkungen (EMV).

3.2.4 Ablaufstrukturen

In allen Phasen müssen Fehlerzustände erkannt und behandelt werden können:

- Initialisierungsphase: Fehler gezielt aufspüren. Alle Systemkomponenten auf Funktionsfähigkeit abprüfen (Funktionstest ähnlich wie beim Booten eines Rechners). Während dieser Phase dürfen keine kritischen (technische) Prozesse ablaufen. Gegenseitigen Abhängigkeiten von Prozeßkomponenten hierbei beachten (z.B. Testen einer Heizung beeinflußt benachbarte Temperaturfühler).

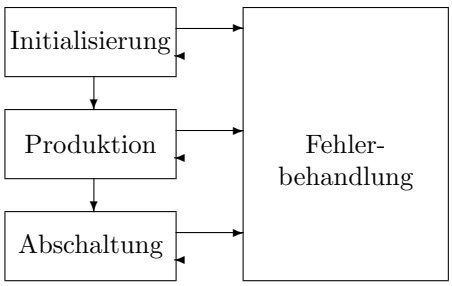


Bild r3p01

- Produktionsphase: Mit Fehlern ist zu rechnen. Ein Maß für die Güte einer Software ist es, wie viele der möglichen Fehlerzustände sie berücksichtigt. Keine Frage von Betriebssystemen oder Programmiersprachen, sondern reine Entwurfsdisziplin.
- Abschaltphase: Es sollten keine Fehlerzustände auftreten. Diese Phase wird unter Umständen bei der Fehlerbehandlung anderer Phasen durchlaufen, z.B. eine Notabschaltung, muß hier auf Robustheit großer Wert gelegt werden.

3.2.5 Kommunikation zwischen Modulen

Realzeit-Software ist modular aufgebaut.
Die Module entsprechen festgelegten Prozeßschritten.

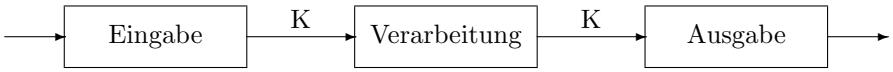


Bild r3p02

Module:
- atomare Aktion (aA), die nicht unterbrochen werden darf
- Thread, Abschnitt (leichtgewichtiger Prozeß), nebenläufig, aus mehereren aA
- Task (Programm, Rechenprozeß) aus mehreren Threads
- Job, kooperierende Tasks

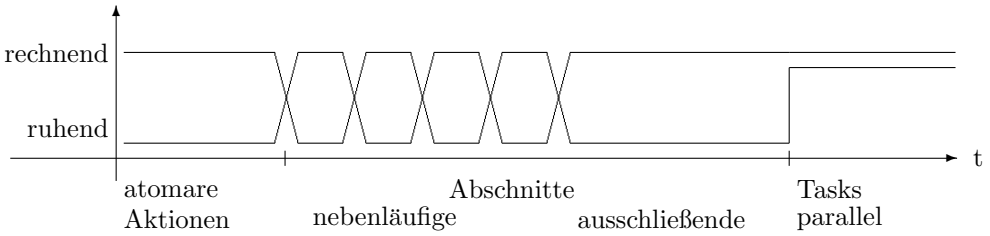


Bild r3p02a

Kommunikation (Inter Process Communication, IPC).
- Synchronisation,
- Erzeuger-Verbraucher-Beziehung,
- gegenseitiger Ausschluß
- Forkprozesse

3.3 Realzeitbetriebssysteme

3.3.1 Anforderungen

Anforderungen an das Betriebssystem auf dem Zielrechner

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

3.3.2 Aufgaben

Bereitstellung von Modulen und Schnittstellen
- Abstrahierung von der konkreten Hardware (Hardware-Software-Interface)
- Module für den Zugriff auf die Hardware (Gerätetreiber)
- Module für komplexe Rechenoperationen (sin x) (SW-Treiber)
- Module zur Systemsteuerung (Interrupt-, Task-, Zeit-Management)

3.3.3 Struktur

Hierarchie von abstrakten Maschinen:

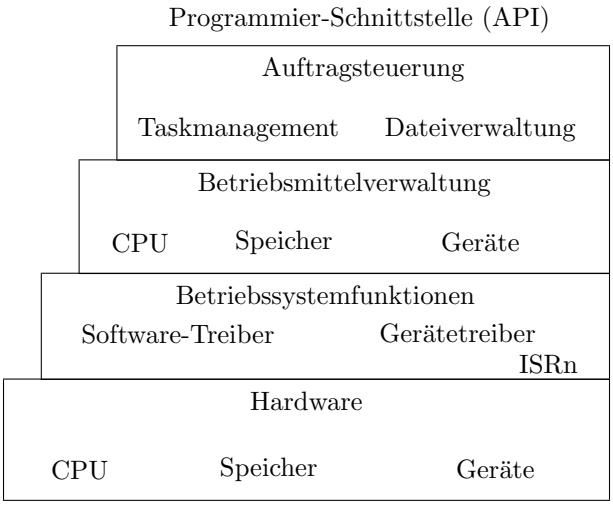


Bild r3p03

Der Betriebssystemkern
Der Betriebssystemkern (kernel, nucleus) umfaßt die Betriebssystemkomponenten:
- Betriebsmittelverwaltung mit Tabellen für die Speicherbelegung, Gerätebelegung, laufende Tasks, Interrupt-Vektoren und Algorithmen zu deren Verwaltung.
- Prozeßverwaltung (im Sinne von Taskverwaltung) mit Tabellen und Algorithmen zu deren Verwaltung, insbesondere für die Taskumschaltung bei Multitasking
- Dateiverwaltung mit Routinen für Dateizugriff und Tabellen für Dateistrukturen.
- Oft benötigte Systemfunktionen, z.B. Datenkonvertierung.

Der Betriebssystemkern ist der Teil der Betriebssystem-Software, der der Hardware am nächsten ist und ihr speziell angepaßt sein muß.

Der Betriebssystemkern wird oft in einem ROM gespeichert, um größtmögliche Sicherheit gegen ein Überschreiben dieser Software zu haben. Variablen, Tabellen und ladbare Treiber müssen dann natürlich in einem "RAM" abgespeichert werden.

3.3.4 Taskmanagement

Daten

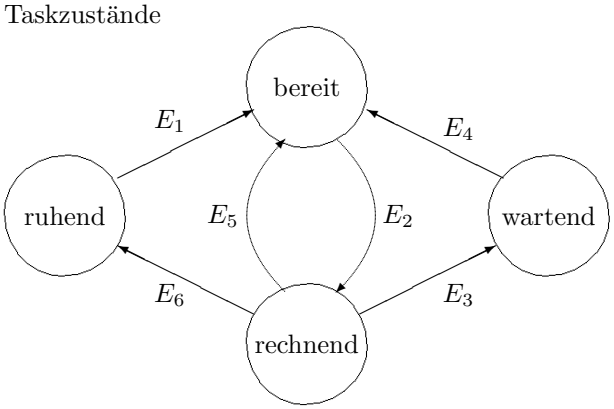
Soft-ware-Kontext | Programm | Hard-ware-Kontext

Speicherbedarf

. .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.
- Der Hardware-Kontext bestimmt die benötigten Geräte
- Der Software-Kontext bestimmt die die benötigten Systemfunktionen
- Die Datenstrukturen werden im Speicher angelegt (Stack, Register)
- Daraus ergibt sich der gesamte Speicherbedarf

Bild r3p04

Taskzustände



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

Prozeßbaum: Kindprozesse

Prozeßliste: LINUX-Kommando "ps"

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

3.3.5 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)

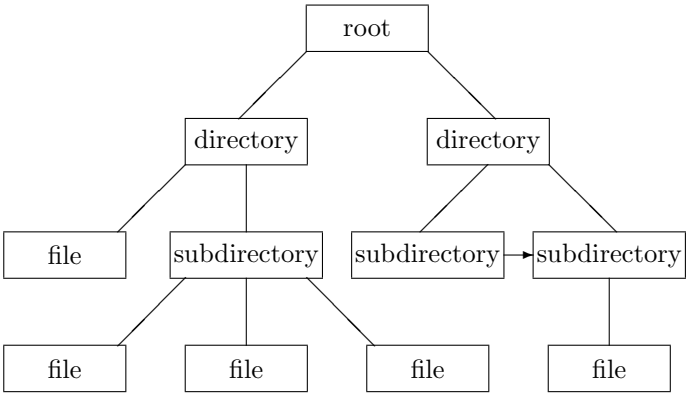


Bild r3p06

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

3.3.6 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)

3.3.7 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 (vgl.)

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.

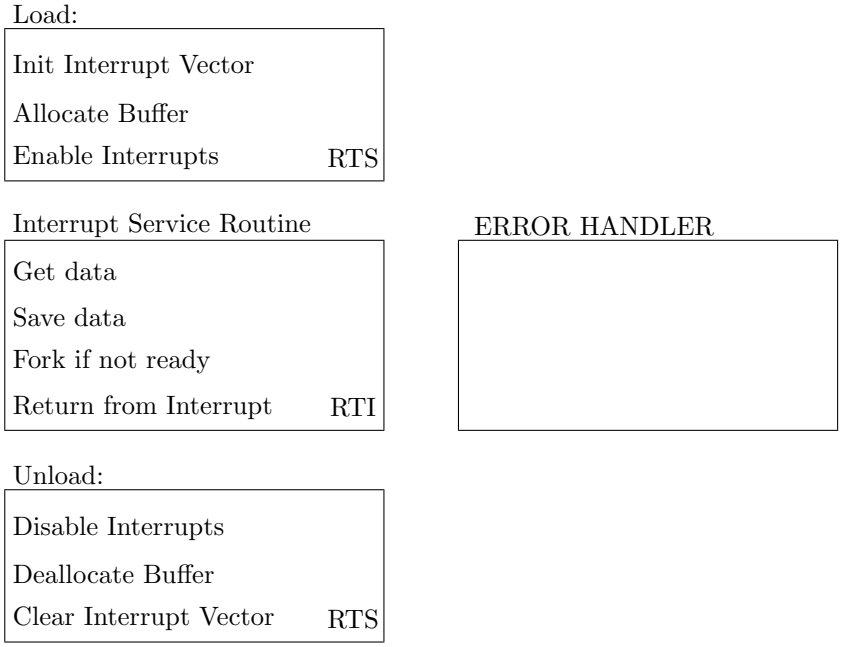


Bild r3p07

3.3.8 Systemkonfigurierung, Systemmanagement

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

3.3.9 Beispiele für Realzeitbetriebssysteme

Betriebssystem	Prozessor	Hersteller	Anmerkungen
Versados	M 68 000	Motorola	
OS-9000	M 68 000	Microware	
OS-9	viele		
LynxOS	diverse		
RTX	M 68 000		
RT-11, RSX-11	PDP-11	DIGITAL (DEC)	historisch
Betriebssysteme, die auch für Realzeitprgramme benutzt werden, aber wenig dafür geeignet sind:			
MS-DOS	Intel 80x86	MicroSoft	
Windows	Intel 80x86	MicroSoft	
OS/2	Intel 80x86	IBM	
UNIX u.a.	diverse	AT&T; u.a.	
VAX-VMS	VAX	DEC	

3.4 Die UNIX – Systemschnittstelle

3.4.1 Systemaufrufe

Unterste Schnittstelle zum Betriebssystem. Ihre Mächtigkeit spielt eine entscheidende Rolle bei der Realzeitprogrammierung.
Typische Systemaufrufe in der Realzeitprogrammierung sind:

Aufruf	Funktion
get-time	Uhrzeit abfragen
get-date	Datum abfragen
set-timer	Zeitgeber setzen
fork	parallele Task erzeugen
exec	anderes Programm aufrufen
exit	Programm beenden
open(file)	Datei eröffnen, Gerät initialisieren
status	Gerät abfragen
read	Daten einlesen
write	Daten ausgeben
close(file)	Datei schließen, Gerät freigeben
malloc, free	Speicherplatz belegen, freigeben
sleep(time)	Programm(modul) für die Zeit 'time' in den Ruhezustand versetzen
wait, pause	Programm(modul) in den Ruhezustand versetzen, bis ein bestimmtes Ereignis eintritt
send	Daten an ein anderes Programm(modul) senden
receive	Daten von einem anderen Programm(modul) empfangen
pipe, queue	Daten in eine Warteschlange einfügen
random	Erzeugen von (Pseudo)Zufallszahlen
system	Aufruf eines shell-Kommandos

LINUX-Funktionen:

- int rand(void); /* in <stdlib.h>
  void srand(int seed); setzt einen Startwert (default = ?)
  n = rand(); liefert einen Wert im Intervall {0, RAND_MAX}
- int system(const char *s);/* in <stdlib.h>
  n = system('command'); führt ein shell-Kommando aus und liefert einen Rückgabewert über den Erfolg
- clock_t clock(void); /* in <time.h>
  ct = clock(); zeigt die verbrauchte Prozessor-Zeit in CLOCKS_PER_SEC
- long time(time_t *tod); /* in <time.h>
  ct = time(NULL); zeigt Zeit in Sekunden seit dem 1.1.19970 0:00:00
- long times(time_t *tim); /* in <sys/times.h>
  ct = times(NULL); zeigt die System-Zeit seit Systemstart in CLOCKS_PER_SEC
- void sleep(int zeit); /* in <sys/time.h>
  Ruhezeit 'zeit' in Sekunden

3.4.2 Parallele Prozessse

Der FORK-Prozeß:

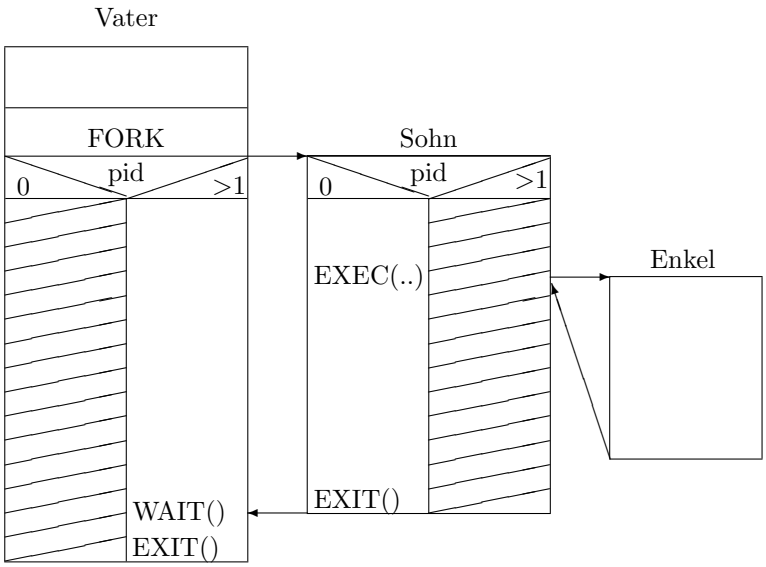


Bild r3p10

Erzeugen paralleler Prozesse:
int fork();
pid = fork()
-1 = ERROR
0 = Sohn
>1 = Sohn-PID für Vater

. .Aufruf:
int execXX(param);
int execlp(file, arg0, arg1, ...argN, NULL);
char *arg0, ... *argN;
-1 = ERROR;

Warten:
int wait(&status);
int status;
n = wait(NULL);/* Sohn-PID;

. .Beenden:
void exit(status);
int status;/* an Vater-Prozeß (shell)

Funktionen:
int getpid()
int getppid()

3.4.3 Interprocess Communication

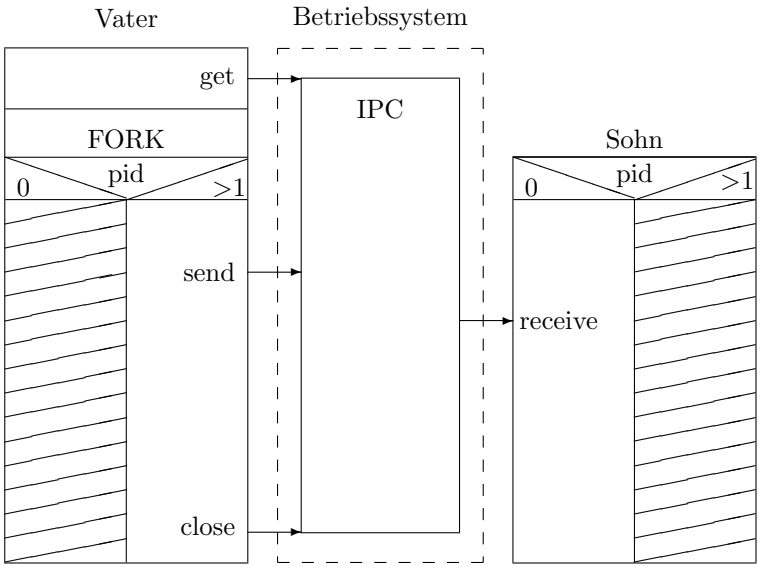


Bild r3p11

IPC	Header	get	control	operation
Pipes		fdopen(pdk[0]		putc, getc
Messages	sys/msg.h	msgget	msgctl	msgsnd, msgrcv
Semaphor	sys/sem.h	semget	semctl	semop()
Shared Memory	sys/shm.h	shmget	shmctl	shmat, shmdt

3.4.4 Signale

Kommunikation mit dem Betriebssystem (definiert in <signal.h>)
a) Signalbehandlung
Aufruf bei Programmstart (vereinfacht):

signal(int sig, void *funct);

ruft die Funktion 'funct' auf, falls vom BS das Signal 'sig' eintrifft.
Signale (auszugsweise):
- SIGABRT; ABoRT
- SIGFPE; Floating Point Error
- SIGILL; ILLegal instruction
- SIGINT; INTeraction
- SIGSEGV; SEGmentation Violation
- SIGTERM; TERMination request
- SIGKILL; Abbruch durch CTRL/C, nicht umleitbar
- SIGBUS; BUS Fehler, z.B. Zugriff auf nicht existenten Speicher oder Register
- SIGPIPE; End of PIPE
- SIGALARM; ALARM eingetroffen, s.w.u.
- SIGUSR1; Benutzer-Signal 1
- SIGUSR2;
- SIGCLD; Death of Child
- SIGPWR; PoWeR fail Restart

Standard-Funktionen
- SIG_DFL; Default, Standardbehandlung durch BS
- SIG_ERR; ERRoneous call
- SIG_IGN; IGNore signal

Benutzerfunktionen müssen vor Aufruf von 'signal' definiert oder deklariert sein.

b) Signalisierung

int kill(int pid, int sig)

sendet das Signal 'sig' and den Prozeß 'pid'

int pause()

wartet auf (irgendein) Signal, vorausgesetzt es ist nicht (mit SIG_IGN) abgstellt.

unsigned alarm(sec)

erzeugt nach 'sec' Sekunden ein Signal 'SIGALARM' an sich selber.
Rückgabewert ist die Restzeit eines früheren Aufrufs von 'alarm'.

3.5 Realzeitprogrammiersprachen

3.5.1 Anforderungen

Eine Realzeitprogrammiersprache sollte folgende Angebote machen:
Es sollte Sprachelemente für Funktionen zur Verfügung stellen, die für die Realzeitverarbeitung notwendig oder typisch sind (s.w.u.).
Es sollte auch Strukturelemente (Programmstrukturen) unterstützen, die bei konventioneller Programmierung nicht üblich sind, insbesondere für die Programmierung von Teibern (ISR), direkten Zugriff auf die Hardware,

Für die gewählte Programmiersprache müssen effiziente Compiler verfügbar sein. Die größte Effizienz bietet natürlich ein Assembler; Nachteile sind jedoch:
– keine Portierbarkeit auf andere Rechnersysteme,
– umfangreicher, unübersichtlicher Quellcode,
– wenige verfügbare Werkzeuge zur Unterstützung während der Entwicklung und Wartung.

Notwendige Fähigkeiten Sprachelemente für

- Systemaufrufe zur
- Intertaskkommunikation
- Speicherverwaltung
- Ein/Ausgabe auf beliebige Geräte
- Tasksynchronisation
- Taskmanagement: Erzeugung, Suspension, Hibernation, Wait(event), Sleep(time), Exit
- Bitmanipulation
- Zeitein/ausgabe
- Fehlerbehandlung
- Warteschlangen

Erwünschte Fähigkeiten:

- Verarbeitung spezieller Datentypen, wie Uhrzeit und Datum.
- Overlay-Technik, d.h. das gezielte Überlagern von Programmodulen, um Speicherplatz optimal auszunutzen.

3.5.2 Beispiele

Einige wichtige Realzeitprogrammiersprachen
Als Ursprung ist jeweils die Sprache angegeben, aus der eine neue entstanden ist (in Klammern die beteiligten Institutionen oder der Vorgänger), das Entstehungsdatum ist nur als Anhaltspunkt gedacht.

Sprache	Ursprung	Jahr	Anwendung	Referenzen
ADA	(PASCAL)	1979	Prozeß/Systemprogramme	ANSI 1815A
APT	(MIT)	1958	NC-Steuerungen	VDI 3552, 3255
Assembler	maschinenabhängig	-	Systemprogramme	keine Normen
ATLAS	(USA)	1974	Produktprüfung	ATLAS Inc.US
BASEX	BASIC	1974	Prozeßautomation	
C	(UNIX, BELL)	1972	Systemprogramme	ANSI X3J11
C-PASCAL	ALGOL (ETH Zü)	1970	Realzeitprogramme	
CORAL-66	ALGOL,FORTRAN	1966	Systemprogramme	BSI 76/63605
CHILL	(CCITT)		Datenübertragung	CCITT Z.200
EXAPT	APT	1965	NC-Steuerungen	DIN 66 025
FORTH				
MODULA-2	PASCAL	1978	Prozeß/Systemprogramme	
PAS-1	PL/1	1971	Prozeßprogramme	BBC.DE
PASCAL	ALGOL (ETH Zü)	1970	Strukturierte Programme	DIN 66 256
PEARL	ALGOL, PL/1	1968	Prozeßprogramme	DIN 66 253
PROCOL	FORTRAN	1970	Prozeßprogramme	PROCOL.FR
OCCAM-2	(Inmos)	1982	Multi-Processing	INMOS.GB
RT-FORTRAN	FORTRAN	1975	Prozeßprogramme	VDI/VDE 3556
RTL/2	ALGOL,FORTRAN	1967	Prozeßprogramme	BSI 76/64581
SL3	ALGOL (AEG)	1974	Systemprogramme	AEG.DE

Kapitel 4

Prozeßdatenverarbeitung, PDV

Realzeitanwendungen
- 1. Prozeßdatenverarbeitung, Automatisierungstechnik (hier)
- 2. Datenübertragung, Rechnerkommunikation (→ Rechnernetze)
- 3. Transaktionen in Datenbankanwendungen,
  z.B. Buchungssysteme, Termingeschäfte (→ Datenbanken)

4.1 Automatisierung, Begriffe und Modelle

4.1.1 Begriffe

Automat (DIN 19233): ein selbsttätiges künstliches System. Sein Verhalten wird dadurch bestimmt, daß Ausgangsgrößen nach festgelegten Regeln aus Eingangs- und Zustandsgrößen gebildet werden. Die Zustände des Systems ergeben sich aus vorangegangenen Eingaben

Anmerkung: Die Ausführung der Regeln kann durch eine Programmsteuerung oder durch anderweitig festgelegte Beziehungen erfolgen.

Automatisierung: die Einführung von Automaten.

Automatisierungsgrad:

$$\begin{aligned}\text{Umfang der Automatisierung} &= \frac{\text{(Zahl der automatisierten Bearbeitungsvorgänge)}}{\text{(Zahl aller Bearbeitungsvorgänge)}} \\ &= 1 - \frac{\text{(Zahl der manuellen Bearbeitungsvorgänge)}}{\text{(Zahl aller Bearbeitungsvorgänge)}} < 1\end{aligned}$$

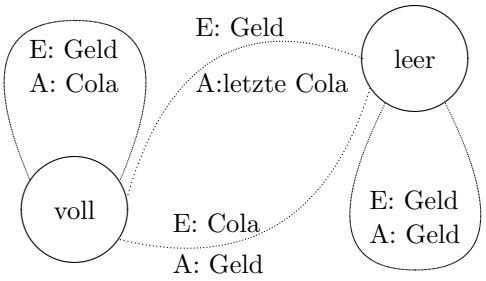
Beschreibung von Automaten (s. Automatentheorie)

$$Automat = \{\{A\}, \{E\}, \{Z\}, Z_0, \delta, \lambda\}$$

Bild r4p01

$\{A\}$= Eingabealphabet
$\{E\}$= Ausgabealphabet
$\{Z\}$= Zustandsalphabet
$Z_0 \in \{Z\}$= Anfangszustand
δ= Zustandsübergangsfunktion:
$Z' = \delta(E, Z)$
λ= Ausgabefunktion: $A = \lambda(E, Z)$

Beispiel: Cola-Automat



Eingaben: Geldmünzen, Cola
Ausgaben: Cola, Geldmünzen

Zustände: leer / voll Cola
bereit / defekt
mit / ohne Wechselgeld

Verarbeitung: Geld prüfen, Cola freigeben, Geld ausgeben

Abweichende Definitionen in ISO 2382 part 1 – Fundamental terms:
- automatic: pertaining to a process or equipment that, under specified conditions, functions without human interaction.
- to automate: to make a process or equipment automatic
- automation: the conversion of processes or equipment to automatic operation, or the result of the conversion
- computerization: automation by means of computers
- process: a predetermined course of events defined by its purpose or by its effect, achieved under given conditions.

4.1.2 Automatisierungsziele

Zielfunktionen
- Produktivität, Durchsatz (Menge / Zeit)
- Stückkosten (Preis/Menge)
- Qualität
- Flexibilität (Anpassung an wechselnde Aufgaben)
- Sicherheit:
  Sicherheit im Produktionsprozeß (Investitionsschutz)
  Störungsfreiheit (Produktivität)
  Sicherheit der Bediener (Personenschutz

Diese Zielfunktionen sind nicht gleichzeitig maximierbar.

Schlagworte
- CAM (Computer Aided Manufacturing)
- Effizienz
- Flexibilisierung
- Lean Production (Verringerung der Lagerhaltungskosten)

4.1.3 Grenzen der Automatisierung

a) Grenzen der Datenverarbeitung

- Hardware
- Verarbeitungsgeschwindigkeit
- Speichergröße
- Datenübertragung

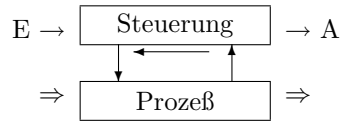
- Software
- Vollständigkeit (alle Möglichkeiten geplant)
- Korrektheit (Fehlerfreiheit)
- Zuverlässigkeit := Vollständigkeit + Korrektheit
- Fähigkeiten:
  z.B. begrenzte Erkennung von Bildern (s. Computer Vision)
  und Tönen (Spracherkennung)

b) technische Grenzen
- Materialbelastung
- begrentzte Ausbreitungsgeschwindigkeit von Signalen ($\leq$ Lichtgeschwindigkeit)

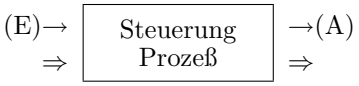
4.1.4 Automaten und Steuerungen

Automat := Prozeß + Steuerung

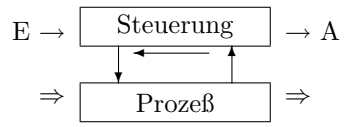
Die hier betrachteten Automaten bestehen aus zwei Funktionseinheiten:
- der Steuerung, die Eingaben (E) erhält und Ausgaben (A) erzeugt, Prozeßzustände erfaßt und auf den Prozeß einwirkt (Kontrollfluß →),
- dem Prozeß, in den ein Verarbeitungsgut hinein fließt und aus dem ein Verarbeitungsgut heraus fließt (Materialfluß ⇒).



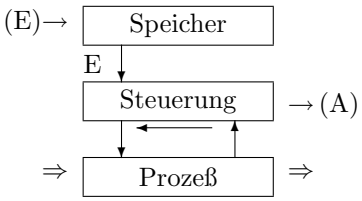
1. Inhärente Steuerung:
Prozeß und Steuerung sind untrennbar miteinander verbunden, der Materialfluß kann auch auf die Steuerung einwirken;
z.B. Sicherungsautomat



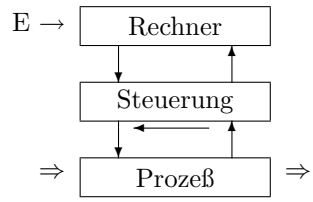
2. Festverdrahtete Steuerung:
die Steuerung ist festverdrahtet, ihre Funktion kann aber durch Eingaben geändert werden;
z.B. Waschmaschine, Kopierer, Fahrstuhl.



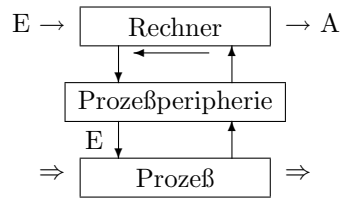
3. Speicherprogrammierte Steuerung (SPS):
die Eingaben erfolgen nicht mehr direkt, sondern werden aus einem Speicher abgerufen;
NC-Maschinen (Numerical Control) mit einer Werkzeugmaschine (z.B. Drehbank) als technischen Prozeß.



4. Freiprogrammierte Steuerung:
die Eingaben werden von einem Rechner bereitgestellt; die Verknüpfung von Eingaben und Prozeßzuständen erfolgt in der Steuerung; z.B. Roboter, CNC-Maschine (Computerized Numerical Control)



5. Prozeßdatenverarbeitung (PDV):
die Eingaben werden aufgrund von Prozeßzuständen von einem Prozeßrechner berechnet;



4.1.5 Prozesse

Prozeß (DIN 19226 bzw DIN 66201): ein Vorgang zur Umformung, zum Transport oder zur Speicherung von Materie, Energie oder Information.

- Verarbeitungsart: Umformung, Transport, Speicherung
- Verarbeitungsgut: Materie, Energie, Information.

- **Technischer Prozeß:** ein Prozeß zur Verarbeitung von Materie oder Energie.
  Ein Prozeß, dessen Zustandsgrößen mit technischen Mitteln erfaßt und beeinflußt werden können.

  MSR = Messen, Stellen, Regeln
  Messen = Erfassen
  Stellen = Beeinflussen
  Regeln = Stellgrößen aus Meßgrößen bestimmen (Rückkopplung)
- **Rechenprozeß:** ein Prozeß zur Verarbeitung von Daten, bzw. Information.
  Verarbeitungsgut = Information
  Materialfluß = Information
  Kontrollfluß = Information
  (vgl. Sicherungsautomat: Verarbeitungsgut = Strom = Kontrollfluß)

Die begriffliche Trennung von Prozessen und den Einrichtungen, mit denen sie realisiert werden, fällt manchmal schwer.

- Prozesse sind abstrakte Begriffe,
- Prozeßeinrichtungen (-anlagen) sind real und anschaulich, sie können auch kaputt gehen.

Oft werden Bezeichnungen für konkrete Prozeßeinrichtungen synonym für Prozeßabläufe verwendet.

Abweichende Definitionen
in ISO 2382 part 21 – Interfaces between computer and technical processes:

- **technical process:** a set of operations performed by equipment in which physical variables are monitored or controlled.
  Examples: destillation and condensation in a refinery, autopiloting and automatic landing in an aircraft.

4.1.6 Systematisierung

a) Betrachtungseinheit: Nach Aufgabe und Umfang abgegrenzter Gegenstand einer Betrachtung (DIN 40150). Z.B.
- Funktionseinheit: .. nach Aufgabe und Wirkung ..
- Baueinheit: .... nach Aufbau und Zusammensetzung ...
- Betriebseinheit:
- Fertigungseinheit:
- Instandhaltungseinheit:

b) Betrachtungsebenen: Hierarchische Strukturierung des Umfangs
- Betrieb (System)
- Anlage, Einrichtung
- Gerät, Maschine, Apparat
- Gruppe
- Element

c) Beispiele:

System	Funktionseinheiten (Prozesse)	Baueinheiten (Konstruktionen)
Betrieb:	Klimatisierung	Gebäude
Anlage:	Belüftung, Klimaanlage	Räume, Fahrstuhl
Gerät:	Gebläse, Heizkessel	Toilette, Kabine
Gruppe:	Motor, Heizkörper	Tür, Fenster (Baugruppe)
Element:	Achse, Ventil	Brett, Schraube (Bauelement)

4.1.7 Hierarchie von (technischen) Prozessen

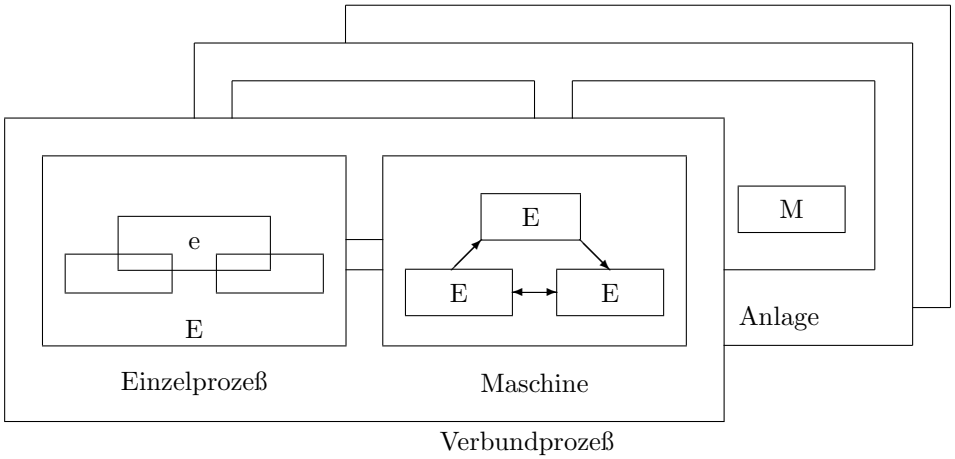
a) Betrachtungsebenen der Funktionseinheiten (nach Aufgabe und Wirkung abgrenzbare Systeme):
- **Elementarprozeß** (ideal): genau ein Verarbeitungsgut und eine Verarbeitungsart.
  z.B.: Heizung = Umformung von Energie
  Lager = Speicherung von Materie
  Leitung = Transport von Materie oder Energie.
- **Einzelprozeß** (real): ein dominanter Elementarprozeß.
  Ein Einzelprozeß repräsentiert einen oder wenige untrennbare Elementarprozesse.
  Ein Einzelprozeß kann technisch nicht weiter zerlegt werden.
  z.B.: Heizung = Umformung von Energie, aber auch der Energietransport muß berücksichtigt werden.
  Heizkessel = Umformung von Energie und Speicherung von Materie.
- **Maschine** (Gerät): Einzelprozesse, die eng mit einander verknüpft sind:
  z.B.: Staubsauger = Motor: Umformung von Energie
  + Verdichter: Umformung von Materie
  + Staubbeutel: Speicherung von Materie
- **Anlage** (Einrichtung): Zusammenwirken von Maschinen, die auch einzeln funktionsfähig wären.
  z.B.: Autowaschanlage, Arbeitszelle
- **Verbundprozeß**: Zusammenwirken von Anlagen.
  z.B.: Fertigungsstraßen, Transportsysteme.

- **Betriebsprozeß**: Zusammenwirken verschiedener Verbundprozesse
  z.B.: Betrieb = Fertigung(CAM)
  + Konstruktion (CAD)
  + Qualitätskontrolle (CAQ)
  + Planung (CAP)
  + Lager
  + Versand
  + Einkauf
  + Buchhaltung

Die übergänge zwischen diesen Kategorien sind oft fließend, eine Zuordnung muß dann nach Bedarf erfolgen:

> z.B. kann eine Autowaschanlage auch als Maschine (wenn sie konstruktiv eine Einheit ist) oder als Verbundprozeß betrachtet werden (wenn sie als eine betriebliche Einheit betrachtet wird).

CIM (Computer Integrated Manufacturing):
Zusammenwirken aller Betriebsprozesse mit Hilfe von Computern.



b) Sammelbegriffe, abgeleitete oder synonyme Begriffe:
(überwiegend ein Elementarprozeß)
- Transportprozeß: Transport von Materie oder Energie
- Speicherprozeß: Speicherung von Materie oder Energie
- Erzeugungsprozeß: Umformung von Materie oder Energie
  z.B. Stahlerzeugung, Energieerzeugung
- Trenn- oder Vereinigungsprozeß: Umformung und Transport von Materie
  z.B. Sägen, Bohren, Stanzen; Schweißen, Löten, Kleben

4.1.8 Verarbeitungsablauf

a) Verarbeitungsstrukturen durch die Prozeßkonstruktion bedingt
(überwiegend bei Elementarprozessen)
- **kontinuierliche Verarbeitung = Fließprozeß**
  z.B. Leitung: Transport von (kontinuierlicher) Material (Gas, Wasser) oder Energie (Elektrizität)
  Heizung: Umformung von Energie
- **diskontinuierliche, diskrete Verarbeitung = Stückprozeß**
  z.B. Transport von Stückgut (Flaschen)
- **zyklische Verarbeitung = Chargenprozeß**
  z.B. Bierbrauen

b) Ablaufstrukturen durch die Natur des Prozesses bedingt
- **deterministische Prozesse** (kausal) sind vorhersagbar (kontinuierliche Zeitstruktur)
  z.B. Heizen, Eisenbahnverkehr
- **stochastische Prozesse** (statistisch) sind im Einzelfall nicht vorhersagbar, haben aber statistische Merkmale wie Häufigkeit, Maximal- und Minimalwerte
  z.B. Fischfang, Straßenverkehr

4.1.9 Prozeßrechner

- Rechner: Hard- und Software, (nicht)technischer Prozeß
- Prozeßrechner: Rechner, der einen technischen Prozeß steuern kann.
- Zuordnung

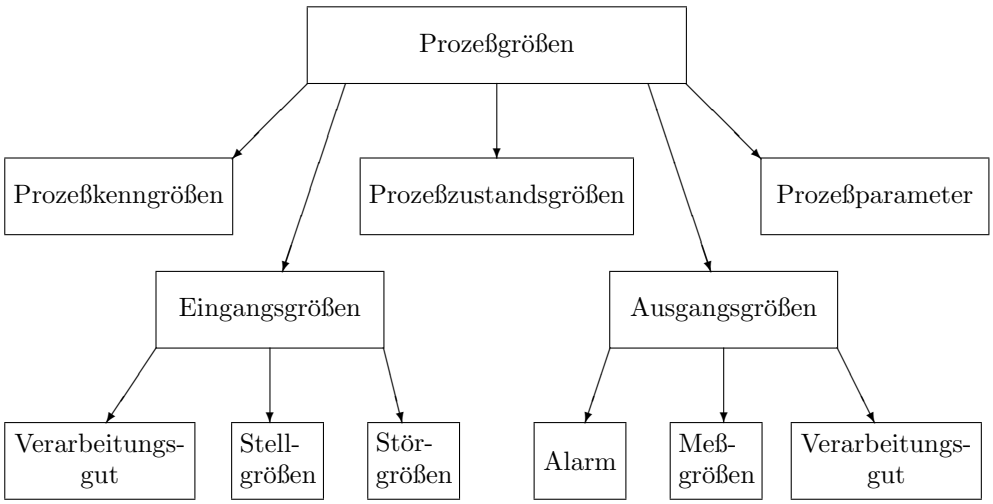
Prozeß	Rechner
Einzelprozeß	Ein-Chip-Computer (4 - 8 bit), single chip Computer
Maschine, Gerät	Ein-Platinen-Computer (8 - 16 bit), single bord computer
Anlage	Micro-Computer, PC, Workstation (16 - 32 bit)
Verbund	(Super)Mini-Computer, Mainframes (32 - 64 bit)
Betrieb	Rechnernetz

Auch hier sind die Zuordnungen nur ganz grob, eine Anlage könnte auch von einem Rechnernetz gesteuert werden oder ein Betrieb durch eine einzelne Workstation, es hängt jeweils ganz von dessen Größe und Komplexität ab.

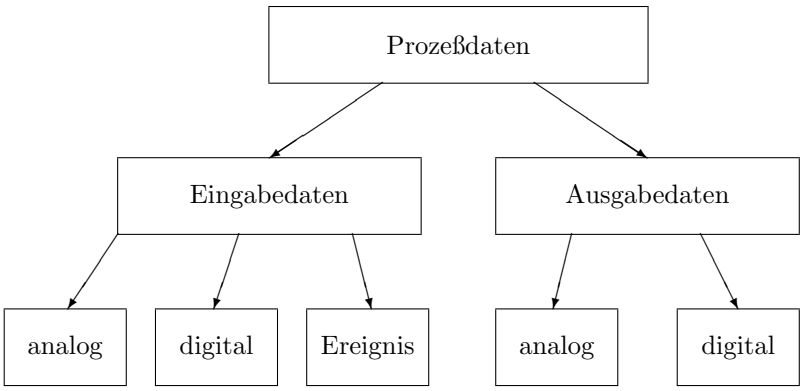
4.1.10 Prozeßdaten und Prozeßgrößen

a) Prozeßgrößen des technischen Prozesses:

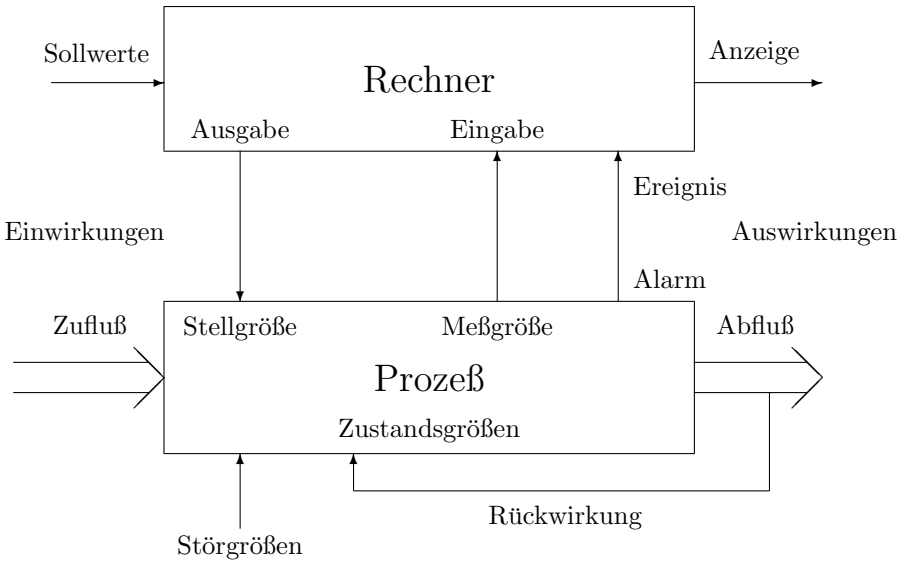
- durch die Konstruktion des Prozesses bedingte (interne) Größen sind:
  - Prozeßkenngrößen (Anlagendaten), Konstanten, Grenzwerte
    z.B. Fassungsvermögen einer Waschmaschine
  - Prozeßzustandsgrößen, Variable
    z.B. Temperatur des Waschwassers
  - Prozeßparameter, Sollwerte
    z.B. die voreingestellte Waschtemperatur für Kochwäsche
- Wirkungen sind meist externe Größen:
  - Einwirkungen: Zufluß von Verarbeitungsgut, Stellgrößen vom Rechner, Störgrößen aus der Umgebung (z.B. Gewalteinwirkung) oder prozeßinterne Defekte (z.B. ein Leck)
  - Auswirkungen: Abfluß von Verarbeitungsgut, Meßgrößen zum Rechner, Alarme zum Rechner oder zum Bediener
  - Rückwirkungen entstehen, wenn durch Abfluß von Verarbeitungsgut interne Zustände geändert werden, so als ob eine äußere Einwirkung stattgefunden hätte. z.B. beim Ablauf von Waschwasser wird sich die die Temperatur des Restwassers bei gleichbleibender Heizleistung stärker erhöhen, als ob sich die Heizleistung erhöht hätte.



b) Prozeßdaten des Prozeßrechners:
- Eingabedaten für den Prozeßrechner stammen entweder vom Bediener oder vom Prozeß, sie können aus analogen oder aus digitalen Daten bestehen; eine Sonderform sind Ereignisse, bei denen ihr Ursprung und der Zeitpunkt ihres Eintretens die wichtigsten Informationen sind.
- Ausgabedaten gelangen entweder zum Bediener oder zum Prozeß; auch sie können analoge oder digitale Daten enthalten.
- Digitale Daten enthalten diskrete Werte, vergleichbar mit den ganzen Zahlen (INTEGER)
- Analoge Daten enthalten kontinuierliche Werte, wie etwa die reellen Zahlen (REAL)



c) Zusammenwirken zwischen Rechner und Prozeß:

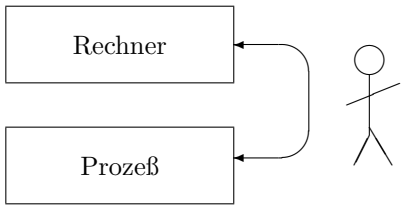


Der Prozeßzustand (die Menge aller Zustandsgrößen) wird durch die Eingangsgrößen bestimmt: durch die Stellgrößen, Störgrößen und durch den Zufluß von Verarbeitungsgut; der Abfluß von Verarbeitungsgut wirkt auf den Prozeßzustand zurück.
Der Prozeßzustand bestimmt die Ausgangsgrößen: den Abfluß von Verarbeitungsgut und die Meßgrößen; Alarme werden durch besondere änderungen des Prozeßzustands ausgelöst und führen am Rechner zu Ereignissen.
Die Ausgabedaten des Prozeßrechners werden zu Eingangsgrößen (Stellgrößen) des Prozesses; Eingabedaten werden aus den Ausgangsdaten (Meßgrößen) des Prozesses gewonnen.
Der Rechner erhält weitere Eingabedaten als Sollwerte vom Bediener und liefert weitere Ausgabewerte als Anzeigen für den Bediener.

4.1.11 Prozeßkopplung

a) Indirekt gekoppelter Betrieb (off-line)

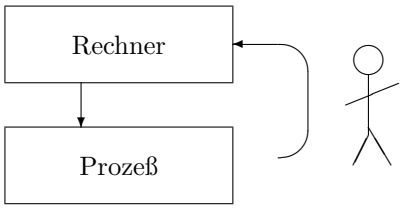
Die übertragung der Ausgabedaten vom Rechner zum Prozeß und der Eingaben vom Prozeß zum Rechner erfolgt manuell.



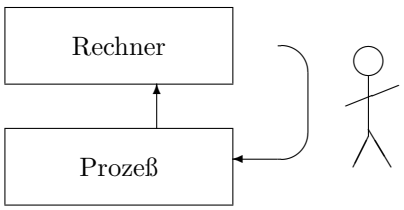
b) Direkt gekoppelter Betrieb (on-line)

- Offener Prozeßbetrieb (open loop control)

Steuerung eines Prozesses druch einen Rechner, die Meßdatenerfassung erfolgt manuell.

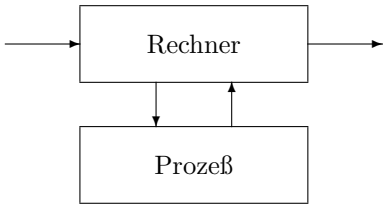


Erfassung von Meßdaten durch den Rechner, die Steuerung erfolgt manuell.



- Geschlossener Prozeßbetrieb (closed loop control)

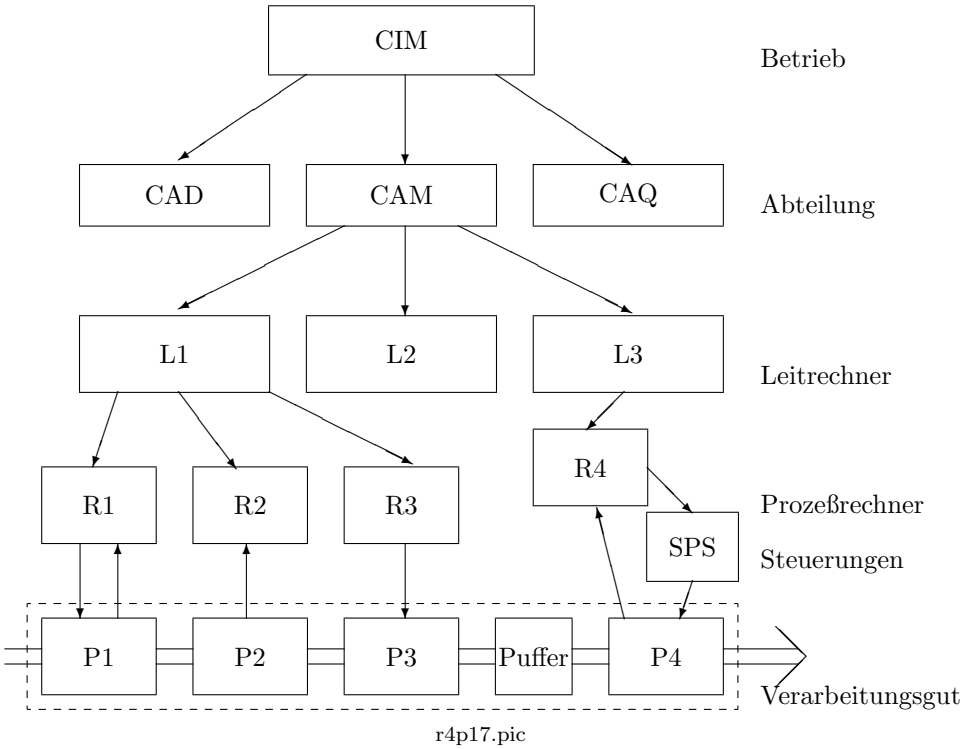
Regelung:
manuelle Eingriffe werden nicht mehr benötigt, die Steuerdaten werden aus den Meßdaten bestimmt



Anmerkung: Das englische Wort controlïst nicht gleichbedeutend mit dem deutschen Wort "Kontrolle", es entspricht vielmehr dem Begriff "Steuerung".

4.1.12 Verteilte Verarbeitung

Eine verteilte Verarbeitung ist in der Regel hierarchisch strukturiert.



r4p17.pic

Lokale Autonomie: Teilstrecken der Verarbeitung können weiterarbeiten, auch wenn in der Umgebung Staus auftreten. Dazu sind Puffer (Lager) notwendig, deren Kapazität ausreichend groß sein muß.

4.2 Modellierung von Prozessen

4.2.1 Prozeßmodelle

Modell = abstrakte Darstellung der Realität.

a) Das gegenständliche Modell:
vereinfachte physische Darstellung zur Darstellung des Prozeßverhaltens, Beobachtung der Auswirkungen unter dem Einfluß von Einwirkungen.
z.B. Auto im Windkanal

b) Die Simulation:
Ersatz des gegenständlichen Modells durch ein Computerprogramm, das bei gleichen Eingaben dieselben Ausgaben erzeugt, wie das Original.
Hierzu notwendig ist:

c) Das abstrakte, mathematische Modell:
Beschreibung eines Automaten (s.w.o.) durch die Funktionen

$$A = \lambda(E, Z) \text{ und } Z' = \delta(E, Z)$$

Beschreibung des Verhaltens eines Prozesses durch die mathematische Funktionen (λ, δ). Für δ sind es in der Regel Differentialgleichungssysteme. Die Lösung solcher Gleichungssysteme bei der Simulation im Computer ist nur durch Differenzengleichungen möglich. Typischer Anwendungsfall ist die FEM (Finite Elemente Methode), bei der statisches Verhalten von Konstruktionen und deren zeitliche Veränderungen simuliert werden.

4.2.1.1 Prozeßbeschreibung

Ein Prozeß kann in einem Stufenplan sowohl nach unterschiedlichen Gesichtspunkten (Betrachtungseinheiten) als auch nach unterschiedlichem Detaillierungsgrad (Betrachtungsebenen) gegliedert und beschrieben werden.

Je mehr unterschiedliche Beschreibungsmethoden und Verfeinerungsstufen verwendet werden, umso größer ist die Wahrscheinlichkeit, das System vollständig zu beschreiben. Eine iterative Verbesserung ist möglich.

a) Statische Beschreibung von Bau- und Funktionseinheiten in Bau- bzw. Funktionsplänen. Beschreibung der (statischen) Prozeßkennwerte.

b) Dynamische Beschreibung (Wirkungen)
- **Stationäre Beschreibung** von Vorgängen und Abläufen, kontinuierlichen Veränderungen:
  Beschreibung der Prozeßzustandsvariablen und -parameter
- **Transiente Beschreibung** von Ereignissen, diskontinuierlichen abrupten Veränderungen.

c) Einbeziehung von Störgrößen
erfordert eine Verfeinerung des Modells und eine Erweiterung um Teilsysteme aus der Umgebung, mit denen Wirkungsbeziehungen bestehen.
Z.B. bei Wärmeverlusten aus einem Heizkessel muß die Wärmeleitfähigkeit des Inhalts, des Behälters und der Umgebung berücksichtigt werden sowie die Wärmekapazität der Umgebung.

4.2.1.2 Zwei Beispiele

Beispiel 1: Kochtopf (Fließprozeß)
Statische Beschreibungen
1. Tabelle: 2 Einzelprozesse (Elementarprozesse)
- Heizplatte: Umwandlung von elektrischer in Wärmeenergie
- Wasserkessel: Speicherung von Material

2. Tabelle: Prozeßkenngrößen
- Fassungsvermögen V des Kessels (in l)
- Widerstand R des Heizdrahts (in Ω)
  bei einer Betriebsspannung U ergibt sich eine Heizleistung $P = U^2/R$ (in W)

Bauplan:

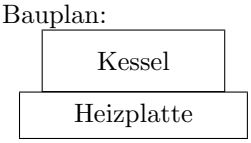
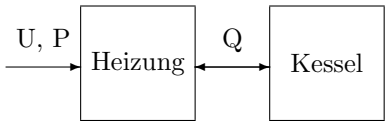


Bild r4p26

. .Funktionsplan:



r4p27.pic

Stationäre Beschreibung

Prozeßparameter:
c = spezifische Wärmekapazität der Flüssigkeit,
m = Masse der Flüssigkeit

(die Prozeßkennwerte des Kessels werden zunächst nicht berücksichtigt.)

Prozeßzustandsgrößen:

Q = Wärmeinhalt,
T = Temperatur
P = (Heiz)Leistung,
t = Zeit

. .Funktionsplan

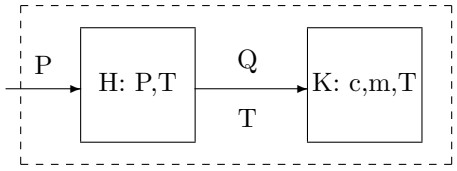


Bild r4p28

Formeln

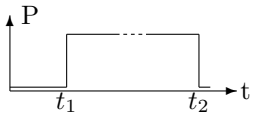
$$\Delta Q = c * m * \Delta T$$

ΔQ = Wärmezufuhr,
ΔT = Temperaturerhöhung

$$\Delta Q = P * \Delta t$$

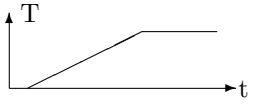
Δt = Heizzeit

Transiente Beschreibung (Ereignisse):
Einschalten (t_1), Ausschalten (t_2).



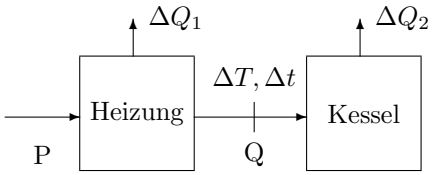
r4p29.pic

Sieden



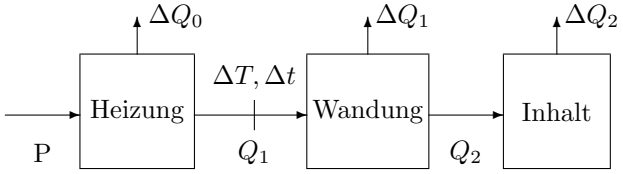
r4p30.pic

Störeinwirkungen
– Wärmeverluste der Heizplatte und des Kessels ΔQ_i
– Wärmewiderstand zwischen Heizplatte und Kessel erfordert einen Wärmetransport, der zu einer Temperaturdifferenz ΔT und einer zeitlichen Verzögerung Δt führt.



r4p31.pic

Verfeinerung
Unterscheidung zwischen Kesselwand und Kesselinhalt mit unterschiedlichen spezifischen Wärmekapazitäten (c_w, c_i), Massen (m_w, m_i), unterschiedlichen Wärmeverlusten, etc.



r4p32.pic

Beispiel 2:
Das Parkhaus (Stückprozeß)
Statische Beschreibung s. Bauplan.

Dieser Prozeß ist nicht vollständig automatisiert: die Einwirkung der Anzeige auf den Prozeß (das Beachten der Anzeige durch die Autofahrer) erfordert noch menschlichen Eingriff.

Anlagedaten: Kapazität M

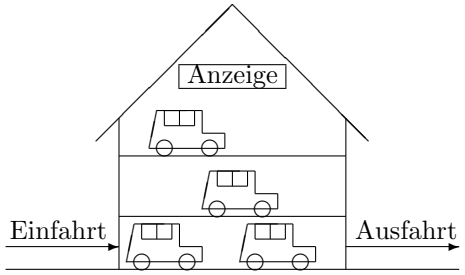


Bild r4p33: Bauplan

Transiente Beschreibung:
Automat = $\{\{A\},\{E\},\{Z\},Z_0,\delta,\lambda\}$ (s.w.o.)
$\{E\}$ = { Auto-Einfahrt, -ausfahrt }
$\{A\}$ = { Anzeige der freien Plätze }
$\{Z\}$ = Belegung {leer, gefüllt, voll}
= $N \in \{0,..,M\}$

Z_0 = Anfangsszustand = "leer" (N = 0)
λ : Anzeige = Kapazität – Belegung = M – N
(Moore-Automat)

δ : N = N+1 falls E = Einfahrt
N = N–1 falls E = Ausfahrt

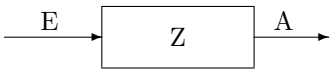


Bild r4p34

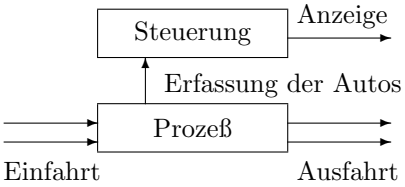


Bild r4p35

Zustandsübergangsdiagramm:

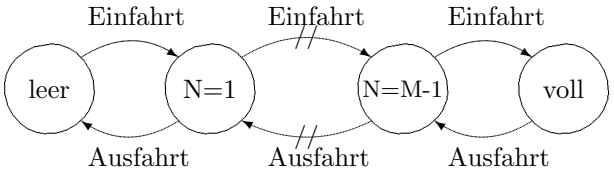


Bild r4p36

Stationäre Beschreibung
Struktogramm:
Der obere Teil der Endlosschleife zeigt die Zustandsübergangsfunktion δ, der untere Teil die Ausgabefunktion λ.

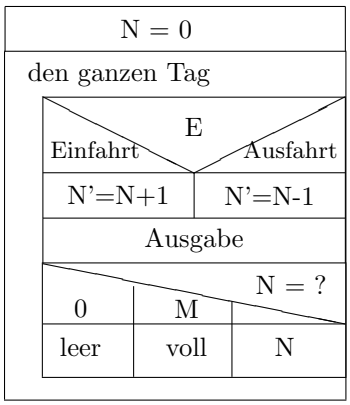


Bild r4p37

4.2.2 Realzeitbetrieb

Der Realzeitbetrieb ist dadurch gekennzeichnet, daß
- Eingabewerte vom Prozeß erfaßt werden müssen
- Ausgabewerte berechnet werden müssen
- auf Ereignisse reagiert werden muß
- Realzeitbedingungen einzuhalten sind.

Im folgenden wird nicht weiter unterschieden, ob Eingaben (E) als Meßwerte oder Ereignisse zu betrachten sind, in jedem Fall ist mit ihnen ein Zeitpunkt verknüpft, an dem sie auftreten und gegebenfalls eine Wert besitzen.

Ausgaben (A) erfolgen als Antworten auf Eingaben und können irgendwelche Aktionen im Prozeßgeschen bewirken. Dazuwischen muß die CPU des Rechners aktiv werden und Berechnungen durchführen (EVA: Eingabe-Verarbeitung-Ausgabe).

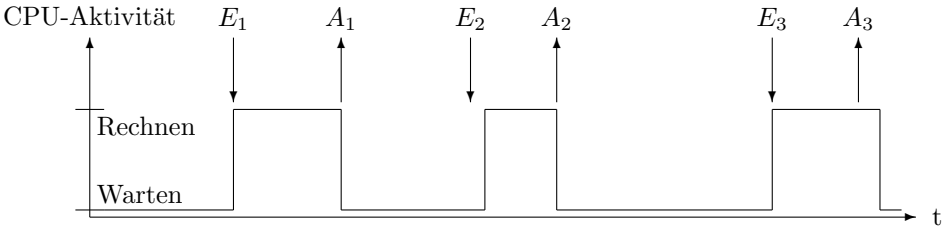


Bild r4p41

4.2.2.1 Betriebsarten

a) Abrufbetrieb (Polling-Betrieb):
- Abfrage der Meßwerte (in einer Schleife)
- Berechnung der Ausgabewerte (in einem Unterprogramm)
- Ausgabe der Stellwerte
- Fortsetzung der Abfrage

b) Anforderungsbetrieb (Interrupt-Betrieb):
- Ereignis löst eine Unterbrechungsanforderung aus
- Unterbrechung mit Sprung in eine Bearbeitungsroutine (ISR, Interrupt Service Routine)
- Berechnung der Ausgabewerte in der ISR
- Ausgabe der Stellwerte
- Rücksprung zum unterbrochenen Programm

c) Vergleich der Betriebsarten:

	Polling	Interrupt
Vorteile	einfache Hard- und Software	CPU-Belastung $< 100\%$ hohe Effektivität
Nachteile	CPU-Belastung $= 100\%$ geringe Effektivität	komplexe Hard- und Software

4.2.3 Prozeß- und Rechenzeiten

Vgl DIN 19226

a) Rechenzeiten

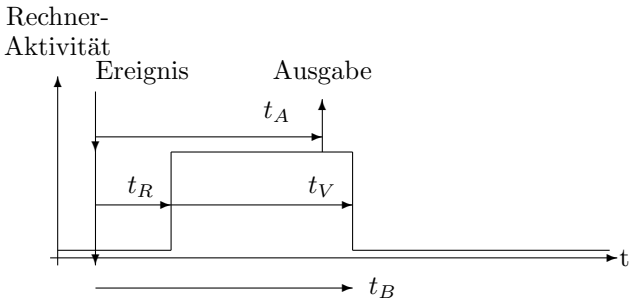


Bild r4p43

t_R: Reaktionszeit (stochastisch)
t_V: Verarbeitungszeit – durch die CPU – (deterministisch)
t_A: Antwortzeit (stochastisch)
t_B: Bearbeitungszeit (stochastisch)
t_S: Stellzeit – im Prozeß – (deterministisch)
t_P: Prozeß/Zykluszeit (deterministisch oder stochastisch)
t_Z: maximal zulässige Antwortzeit (fest), hier muß die (bekannte) Stellzeit t_S berücksichtigt sein

b) Prozeßzeiten

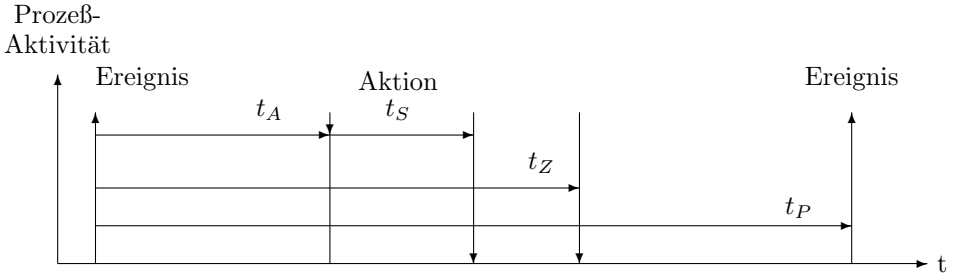


Bild r4p42

c) Zeitverhältnisse:

$t_R \geq 0$ ($t_R = 0$ ist nur der Idealfall)
$t_B = t_R + t_V$ (da meist $t_R \ll t_V$, kann $t_B = t_V$ gesetzt werden
$t_A \leq t_B$ (in der Regel kann sogar $t_A = t_B$ angenommen werden)
$t_A < t_Z$ (das ist die Hauptforderung der PDV)

4.2.4 Realzeitbedingungen

a) schritthaltende Verarbeitung: $t_A < t_Z$
Diese Bedingung muß in jedem Fall eingehalten werden.

b) Auslastung
Der Auslastungsgrad $\eta = t_{Pi}/t_{Bi} \leq 1$
Als Mittelwert: $\bar{\eta} = \Sigma t_{Pi}/\Sigma t_{Bi}$

c) zeitkritische Prozesse $\eta_i \leq 1$ für alle i;
Hier muß jedes Ereignis abgearbeitet sein, bevor ein neues eintritt.
Bei zeitunkritischen Prozessen genügt $\eta \leq 1$, d.h. alle Ereinisse werden früher oder später abgearbeitet.

d) Konflikte

- Konkurrenz: "gleichzeitige" Ereignisse
- Warteschlagenüberlauf

- Verlust
- Zurückstellung
- Unterbrechung

4.2.5 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)

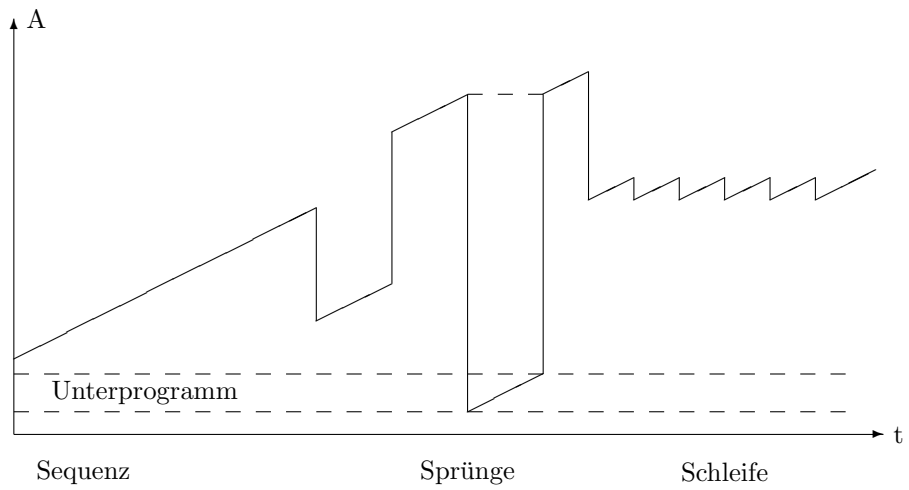
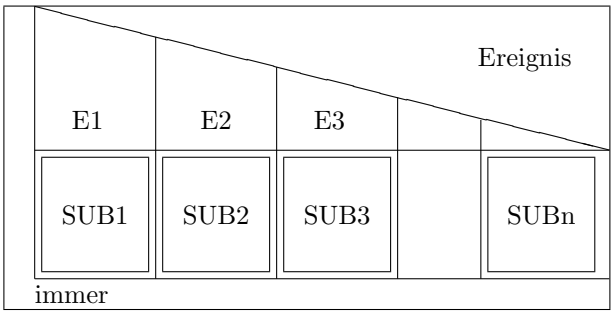


Bild r4p44

- Sequenz (Folge von Instruktionen) ergibt eine Gerade, deren Steigung die Verarbeitungsgeschwindigkeit des Prozessors wiedergibt.
- Sprünge zeigen sich als senkrechte Übergänge
- bei Unterprogrammsprüngen wird die vorhergehende Sequenz zurückgekehrt,
- Schleifen ergeben einen Sägezahnverlauf.

4.2.6 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:

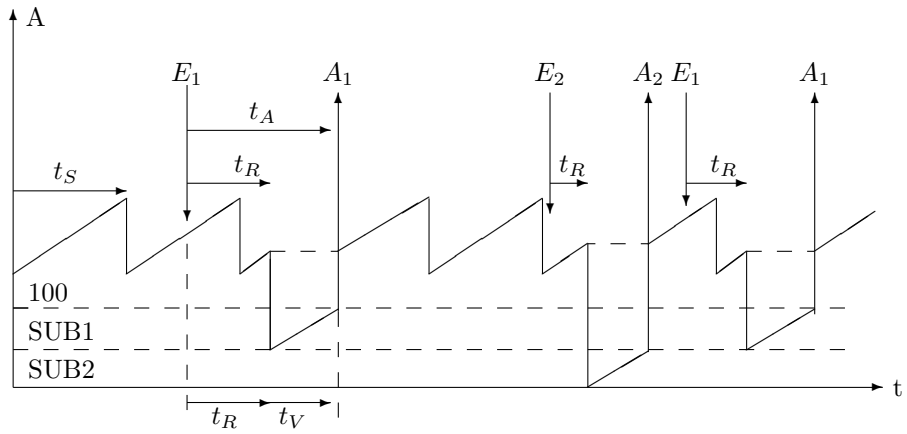


Bild r4p46

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

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

und im Mittel wird

$$\bar{t}_A \leq t_s + n \cdot \bar{t}_V$$

(wobei $\bar{t}_V$ die mittlere Verarbeitungszeit ist); da auch t_s linear mit der Anzahl n ansteigt, wird t_A stets linear mit der Anzahl n der zuberücksichtigenden unterschiedlichen Ereignisse 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.

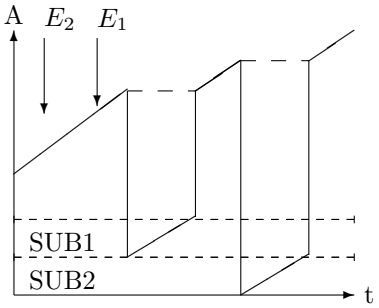


Bild r4p47

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.

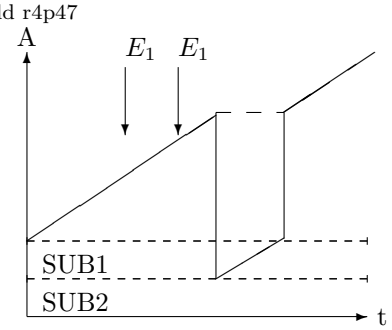


Bild r4p48

4.2.7 Zeitverhalten im Interrupt-Betrieb

Interrupt- bzw. Anforderungsbetrieb

4.2.7.1 Interrupt-Erzeugung

Quelle für (fast) jede Interrupt-Anforderung ist ein Geräte-Anschluß.
Seine Interrupt-Logik hat folgende Aufgaben:
- äußere Ereignisse und Meldungen erfassen und eventuell speichern,
- Interrupt-Anforderung (Interrupt Request, IRQ) an die CPU senden und auf deren Bestätigung (Interrupt Grant, IRG) warten,
- weitere Daten über den geforderten Interrupt senden, (Priorität des Interrupts HWP, Zeiger (IV) auf einen Eintrag in der IVT (vektorisierte Interrupts).)

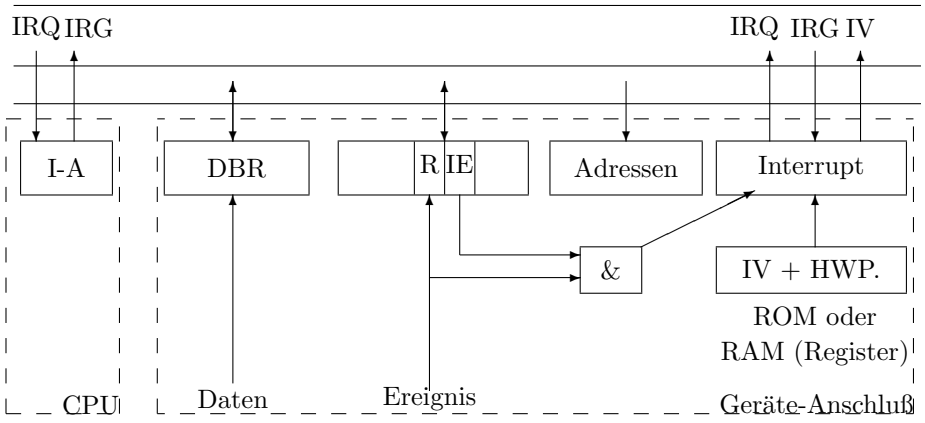


Bild r4p55

4.2.7.2 Filterung der Interrupt-Anforderungen

In der CPU im Interrupt-Verarbeitungswerk (Interrupt Arbitrator, I-A)

a) Maskierung, Verriegelung
Maskenregister wird unter Programmkontrolle geladen, dessen Bits einzelnen Geräte-Anschlüssen zugeordnet sind und deren Interruptanforderungen weitergeben oder blockieren.

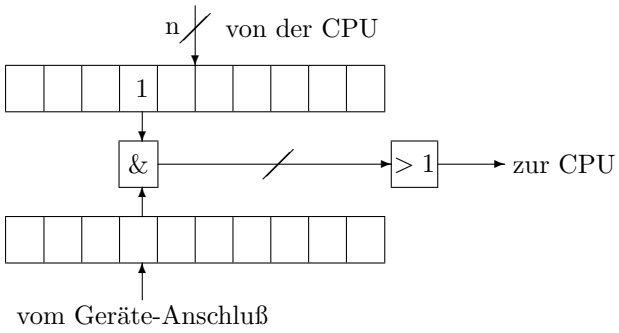


Bild r4p56

b) Interrupt-Gewichtung
Die von den Geräte-Anschlüssen ankommenden IRQs werden 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.

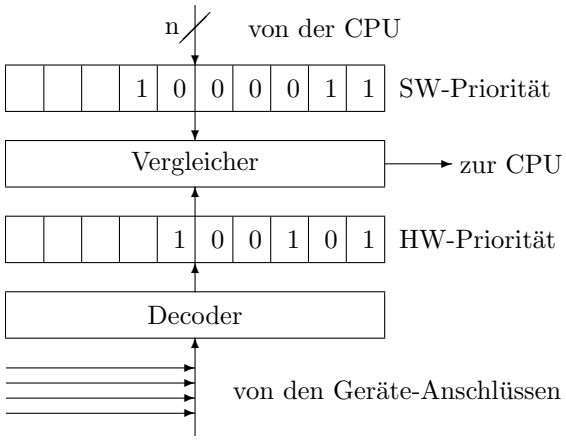


Bild r4p57

c) Darstellung der Prioritätensteuerung durch 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.

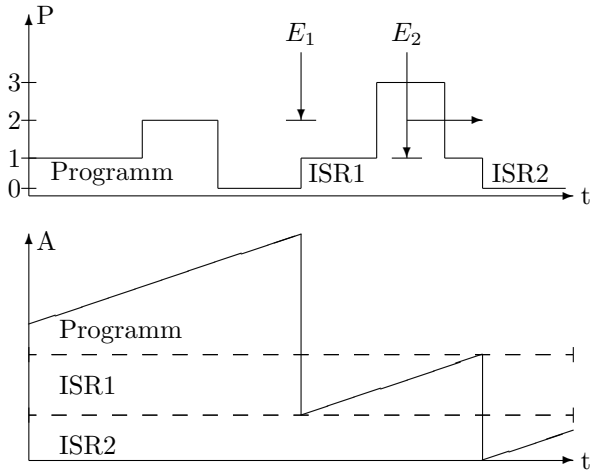
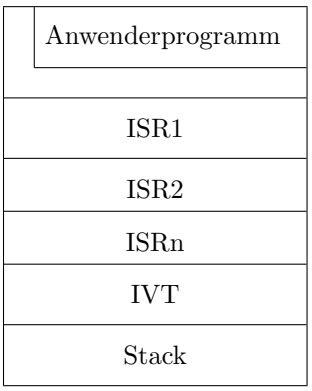


Bild r4p58

4.2.7.3 Bearbeitung des Interrupts

durch das Mikroprogramm der CPU



- 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.

Bild r4p49

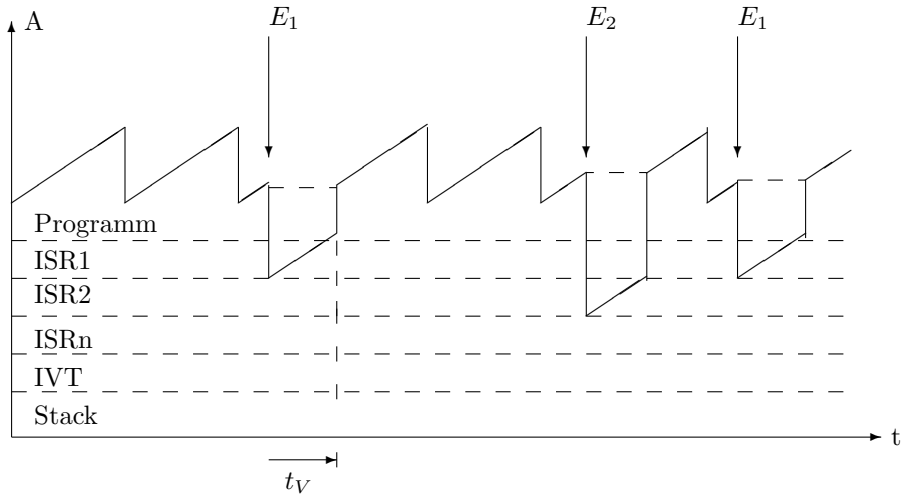


Bild r4p50

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.

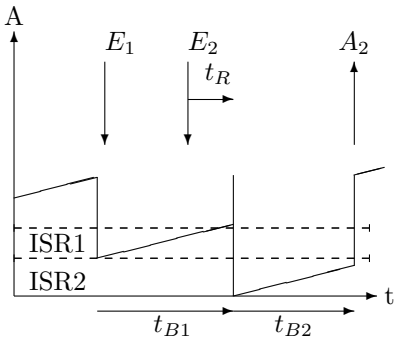


Bild r4p51

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.

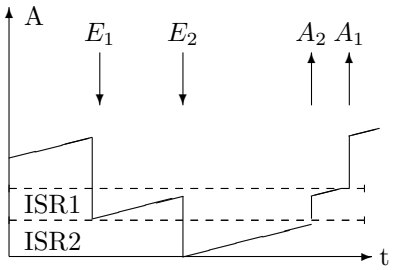


Bild r4p52

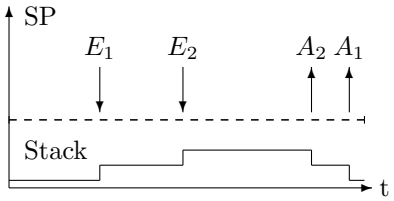


Bild r4p53

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.

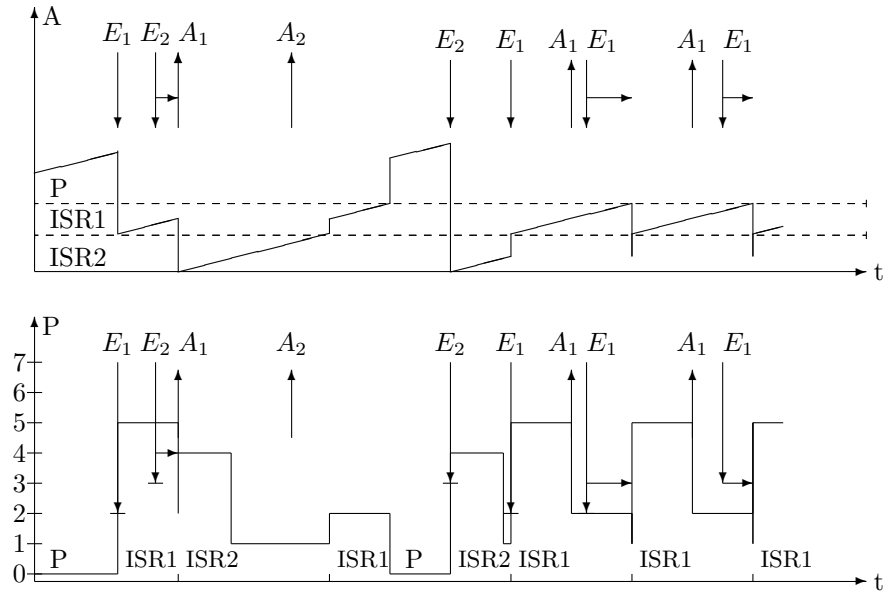


Bild r4p54

.

Kapitel 5

Regelungstechnik

Eine Einführung

Grundbegriffe (DIN 19226 Regelungs- und Steuerungstechnik)

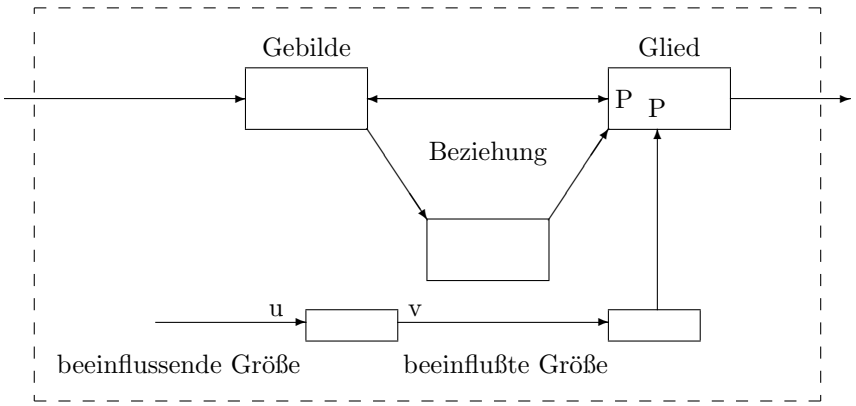
System: eine abgegrenzte Anordnung von Gebilden, die miteinander in Beziehung stehen. (N.B. Gebilde können auch System sein, d.h. die Definition ist rekursiv)

Struktur: die wirkungsmäßige Art und Zuordnung der Gebilde in einem System.

Parameter: das Ausmaß der Wirkung der Teilsysteme aufeinander.

Wirkung: die Veränderunge einer Größe (beeinflußte Größe) durch eine andere (verursachende Größe)

Prozeß: (s.w.o.) ein Vorgang zur Umformung, zum Transport oder zur Speicherung von Materie, Energie oder Information. (DIN 19226 bzw DIN 66201)



r5p0a.pic

5.1 Wirkungen

Wirkungsplan: symbolische Darstellung aller Wirkungen in einem System. Netz aus Übertragungsgliedern (Blöcken) und Wirkungslinien (vgl. NIK).

Elemente eines Wirkungsplans:

Block: ein Glied, System (ein Prozeß) oder Gebilde, das eine beeinflußte Größe darstellt, und auf das eine oder mehrere beeinflussende Größen wirken (veraltet: Strecke).

Wirkungslinie: der Weg einer Größe im Wirkungsplan, der durch eine Richtung gekennzeichnet ist.

Addition, Konnektor: ein Glied, das die Summe von mehrerern Größen bilden kann; die Subtraktion wird durch ein negatives Polaritätszeichen am Eingang des Konnektors dargestellt.

Verzweigung, Verteiler: ein Glied, das ein und dieselbe Größe auf mehrere Wirkungswege verteilt.

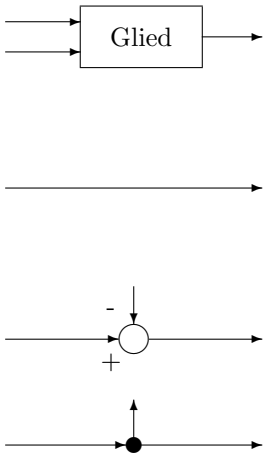


Bild r5p0b

Der Wirkungsweg: Verlauf einer Wirkung durch ein System. Angenommen wird:

- gerichtete Wirkung ($\vec{u} \rightarrow \vec{v}$), bzw.
  das EVA-Prinzip (Eingabe → Verarbeitung → Ausgabe).
- Rückwirkungsfreiheit:
  keine Veränderung eines Systems durch die von ihm beeinflußten Systeme

Grundstrukturen:

Reihenstruktur: Wirkungsweg, der sequentiell mehrere Glieder durchläuft

Parallelstruktur: Wirkungsweg, der sich auf mehrere parallele Wirkungswege aufteilt

Kreisstruktur: Wirkungsweg, der eine Ausgangsgröße wieder als Eingangsgröße enthält. Eine Kreisstruktur stellt einen geschlossenen Wirkungsweg dar.

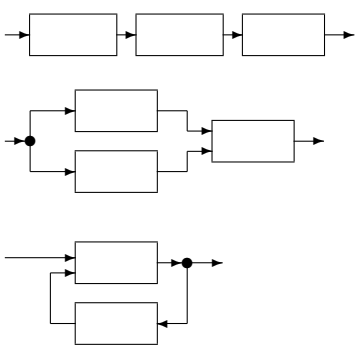


Bild r5p00

Das Verhalten eines Systems wird bestimmt durch
– die Eigenschaften seiner Übertragungsglieder,
– die Verknüpfung der Übertragungsglieder (die Struktur).

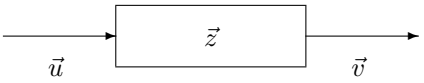
5.2 Übertragungsglieder

a) Beschreibung als Automat (nach DIN 19 226 Teil 1: 1994):

Eingangsgrößen: auf ein System einwirkende (verursachende) Größen; die Gesamtheit aller auf ein System einwirkenden Größen wird als Eingangsvektor $\vec{u} = \{u_1, u_2, ..., u_n\}$ bezeichnet.

Ausgangsgrößen: erfaßbare Größen eines Systems, die nur von seinen Zustands- und den Eingangsgrößen abhängen; die Gesamtheit der Ausgangsgrößen bildet den Ausgangsvektor $\vec{v} = \{v_1, v_2, ..., v_m\} \quad (m \neq n)$.

Zustandsgrößen: interne zeitlich veränderliche Größen eines Systems, die das Verhalten des Systems bestimmen und deren Veränderungen nur von den Eingangsgrößen abhängen; die Gesamtheit der Zustandsgrößen bildet den Zustandsvektor $\vec{z} = \{z_1, z_2, ..., z_x\}$.



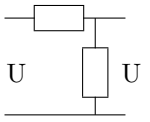
r5p01.pic

Übertragungsglieder werden als rückkwirkungsfrei angenommen, d.h. $u(t)$ hängt nicht von $v(t)$ ab.

b) Elementarglieder (ideal)
Elementarglieder besitzen nur eine Eingangsgröße und eine Ausgangsgröße.

P-Glied (proportional):

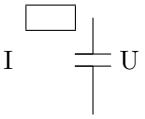
$$v(t) = k_p * u(t)$$



r5p02.pic

I-Glied (integrierend):

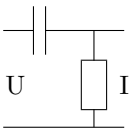
$$v(t) = k_I * \int_0^t u(t')\, dt'$$



r5p03.pic

D-Glied (differenzierend)

$$v(t) = k_D * \frac{du(t)}{dt}$$

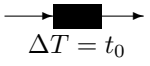


r5p04.pic

T-Glied (verzögernd um t_0)

$$v(t) = u(t - t_0)$$

Nur mit einem Speicher realisierbar.



r5p05.pic

c) Antwortverhalten auf spezielle Eingangsfunktionen $u(t)$

Glied:	P	I	D	T
Eingangsfunktion $u(t)$	Ausgangsfunktion $v(t)$			
	$v(t)/k_P$	$v(t)/k_I$	$v(t)/k_D$	$v(t)/1$
Sprungfunktion $\sigma(t)$ $\sigma(t) = \begin{cases} 0 \ für\ t < 0 \\ 1 \ für\ t \geq 0 \end{cases}$	σ	g	δ	σ
Impulsfunktion $\delta(t)$ $\delta(t) = \begin{cases} \infty \ für\ t = 0 \\ 0 \ sonst \end{cases}$	δ	σ	δ'	δ
Rampenfunktion $g(t)$ $g(t) = \begin{cases} 0 \ für\ t < 0 \\ g \cdot t \ für\ t \geq 0 \end{cases}$	g	t^2	σ	g
Sinusfunktion $sin(\omega t)$				t_0

Bild r5p06

$sin(\omega t)$	$-\frac{1}{\omega} cos(\omega t)$	$\omega \cdot cos(\omega t)$	$sin(\omega(t - t_0))$
	$\frac{1}{\omega} sin(\omega - \pi/2)$	$\omega \cdot sin(\omega t + \pi/2)$	$sin(\omega t - \varphi_0)$
			$\varphi_0 = \omega t_0$

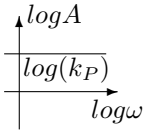
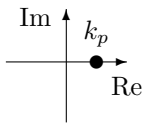
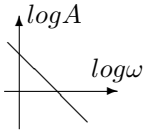
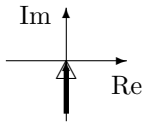
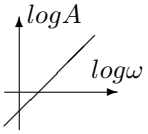
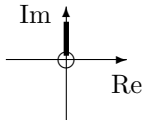
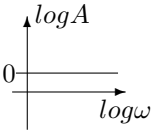
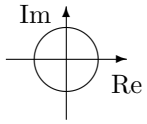
r5p06.pic

5.2.1 Der Frequenzgang von Übertragungsgliedern

Darstellung der Sinusantwort eines Systems:

$u(t) = A_0 \cdot sin(\omega t)$ bzw. $u(t) = A_0 \cdot e^{i\omega t}$
$v(t) = A_S \cdot sin(\omega t + \varphi)$ $v(t) = A_S \cdot e^{i(\omega t+\varphi)}$
mit $A_S = A_0 * f(\omega)$ und $\varphi = \varphi(\omega)$

Frequenzgang: $F(\omega) = v(\omega)/u = \frac{A_S(\omega)}{A_0} \cdot e^{i\varphi} = f(\omega) \cdot e^{i\cdot\varphi(\omega)}$
Amplitudengang: $f(\omega) = |F(\omega)| = \sqrt{Im^2(F) + Re^2(F)}$
Phasengang: $\varphi(\omega) = \angle F(\omega) = tan^{-1}(Im(F)/Re(F))$
Bode-Diagramme
In der Regel wird $f(\omega)$ doppelt logarithmisch dargestellt $\{log(f(log(\omega))\}$, $\varphi(\omega)$ halblogarithmisch $\{\varphi(log(\omega))\}$
Ortskurven sind die graphischen Darstellungen in Polarkoordinaten: Der komplexe Wert von $F(\omega)$ wird in der Gauß'schen Zahlenebene $(Im(F), Re(F))$ aufgetragen. D.h. $\varphi = \varphi(\omega)$ ergibt die Richtung, $f(\omega)$ die Entfernung vom Ursprung an.

Elementarglieder	**Bode-Diagramm**	**Ortskurve**
P-Glied: $F(\omega) = k_P = const$ $f(\omega) = k_p = const, \quad \varphi = 0$	 Bild r5p07	 Bild r5p11
I-Glied: $F(\omega) = 1/i\omega$ $f(\omega) = 1/\omega, \quad \varphi = -\pi/2$	 Bild r5p08	 Bild r5p12
D-Glied: $F(\omega) = i\omega$ $f(\omega) = \omega, \quad \varphi = +\pi/2$	 Bild r5p09	 Bild r5p13
T-Glied: $F(\omega) = e^{i\omega t_0}$ $f(\omega) = 1 = const, \quad \varphi = t_0 \cdot \omega$	 Bild r5p10	 Bild r5p14

5.2.2 Die Übertragungsfunktion G(s)

Bei komplizierteren Übertragungssystemen ist die Sinusantwort bzw, der Frequenzgang nicht immer algebraisch bestimmbar, dann werden die Eingangsfunktion $u(t)$ und die Ausgangsfunktion $v(t)$ zunächst einer Laplace-Transformation unterworfen, welche die Funktionen $U(p)$ und $V(p)$ liefert, deren Quotient $G(p) = V(p)/U(p)$ in jedem Fall gebildet werden kann.

Die Laplace-Transformation $\mathcal{L}$ $\quad u(t) \overset{\mathcal{L}}{\longleftrightarrow} U(s)$

Definition:

$$\mathcal{L}(u(t)) = \int_0^\infty u(t) \cdot e^{-st} dt = U(s)$$

mit der komplexen Variablen $s = i\omega + \alpha \equiv p$
(N.B. für $\alpha = 0$ erhält man die Fourier-Transformation; die Laplace-Transformation kann man also als eine Faltung mit einer gedämpften Schwingung betrachten, im Gegensatz zu einer ungedämpften im Falle der Fourier-Transformation.)

Umkehrung:

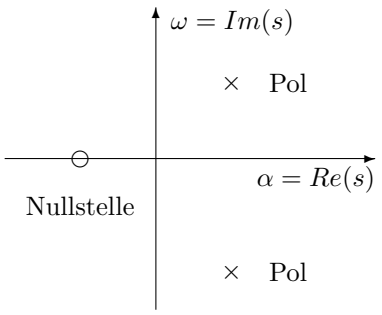
$$\mathcal{L}^{-1}(U(s)) = \frac{1}{2\pi i} \oint U(s) \cdot e^{st} ds = u(t)$$

Zur Lösung dieses (Laplace-)Integrals benötigt man oft den Residuensatz.

Die komplexe Übertragungsfunktion G(s) wird definiert durch

$$G(s) := \frac{V(s)}{U(s)} = \frac{\mathcal{L}(v(t))}{\mathcal{L}(u(t))}$$

Die Darstellung dieser Übertragungsfunktionen erfolgt anhand ihrer Nullstellen ($G(s) = 0$) und Pole ($G(s) = \infty$) in der komplexen Ebene für s. (Man beachte, daß hier die komplexe Variable s dargestellt wird, und nicht die komplexe Funktion G(s) !)



r5p15.pic

Falls für ein System der Frequenzgang $F(\omega)$ bekannt ist, und dieser nicht explizit von der Zeit t abhängt (s. LTI-Systeme), kann die Übertragungsfunktion $G(s)$ einfach durch das Ersetzen von $i\omega$ durch s erfolgen.

Die Übertragungsfunktionen der Elementarglieder:

P-Glied:	$G(s) = k_P = const$	keine Nullstelle, kein Pol
I-Glied:	$G(s) = k_I/s$	Pol bei $s = 0$
D-Glied:	$G(s) = k_D \cdot s$	Nullstelle bei $s = 0$
T-Glied:	$G(s) = e^{st_0}$	keine Nullstelle, kein Pol
T_1-Glied:	$G(s) = 1/(1 + a \cdot s)$	RC – Tiefpaß, Pol bei $s = -1/a = -1/RC$

Anwendung: Wenn die Übertragungsfunktion G(s) eines Systems bekannt ist, kann man mit ihr die Ausgangssignale für beliebige Eingangssignale berechnen.

Folgende Schritte sind durchzuführen:

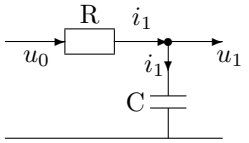
1. Laplace-Transformation des Eingangangssignals: $U(s) = \mathcal{L}(u(t))$
2. Berechnung der Ausgangsfunktion: $V(s) = G(s) \cdot U(s)$
3. Bestimmung des Ausgangssignals durch das Laplace-Integral:
   $v(t) = \mathcal{L}^{-1}(V(s)) = \frac{1}{2\pi i} \oint V(s) \cdot e^{st} ds$

5.2.3 Reale Übertragungsglieder

Reale Übertragungsglieder haben stets kompliziertere Übertragungsfunktionen als die (idealen) Elementarglieder. Hier ist in der Regel sowohl der Amplitudengang ($f(\omega)$) als auch der Phasengang ($\varphi(\omega)$) von der Frequenz abhängig.

Ein Beispiel: der RC-Tiefpaß.
Berechnung des Frequenzgang $F(\omega)$:
Für eine Sinusantwort ($u(t) = A \cdot sin(\omega t)$)
liefert die Spannungsteilerschaltung $u_1 = \frac{R_C}{R+R_C} \cdot u_0$
(mit $R_C = 1/i\omega C$).

$$F(\omega) = v(t)/u(t) = u_1/u_0 = 1/(1 + i\omega RC)$$



r5p16.pic

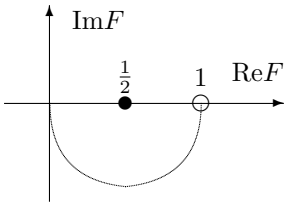
Für den Amplitudengang erhält man

$$f(\omega) = |F| = 1/\sqrt{(1 + \omega^2 R^2 C^2)}$$

mit Werten zwischen 0 und 1.
Für den Phasengang erhält man

$$\varphi(\omega) = tan^{-1}(-\omega RC)$$

mit Werten zwischen 0 und -90 Grad.

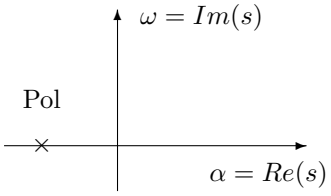


r5p17.pic

Die Ortskurve für $F(\omega)$ ist ein Halbkreis um $(\frac{1}{2}, 0)$, der für $\omega = 0$ bei $(1, 0)$ startet.

Die Übertragungsfunktion G(s) ergibt sich wie der Frequenzgang zu

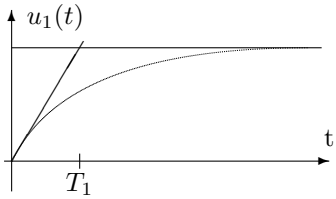
$$G(s) = 1/(1 + sRC)$$



r5p18.pic

Diese Funktion hat einen Pol für $s = -1/RC$

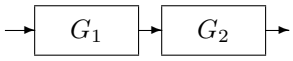
Die Antwort auf eine Sprungfunktion $u(t) = \sigma(t)$ ist ein gedämpfter Sprung $v(t) = u_1 = (1 - e^{-t/T_1})$ mit $T_1 = RC$, dem Schnittpunkt der Tangente vom Kurvenanfang mit dem Grenzwert (1).



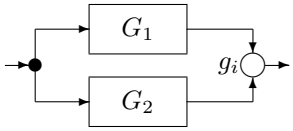
r5p18a.pic

5.2.4 Verknüpfung von Übertragungsgliedern

Reihenstruktur: Hier werden die Übertragungsfunktionen einfach multipliziert:
$G(s) = G_1(s) \cdot G_2(s)$



Parallelstruktur: Hier werden die Übertragungsfunktionen (unter Umständen gewichtet oder invertiert) addiert (subtrahiert):
$G(s) = \sum g_i G_i$



Kreisstruktur: Hier können Rückwirkungen dargestellt werden, die zu komplizierteren Verknüpfungen der Übertragungsfunktionen führen. (Ausführlicheres bei den Regelkreisen weiter unten.)

$V = G_1 \cdot (U \pm V \cdot G_2)$
$G = G_1/(1 \mp G_2)$

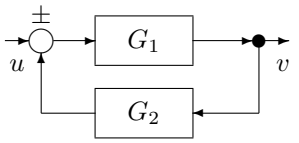


Bild r5p19

Beispiele:
1. Der RC-Tiefpaß (s.w.o. r5p16.pic) besteht aus der Reihenschaltung eines P-Glieds (R) und eines I-Glieds (C):
- der Widerstand R erzeugt aus der Eingangssgröße, der Spannung $u_0(t)$, eine proportionale Ausgangsgröße, den Strom $i_1(t) = u_0(t)/R$; die (frequenzunabhängige) Übertragungsfunktion lautet $G_R = 1/R$.
- der Kondensator C ist ein ideales I-Glied, dessen Eingangsgröße, der Strom $i_1(t)$, aus dem P-Glied geliefert wird, und dessen Ausgangsgröße, die an ihm liegende Spannung $u_1 = \frac{1}{C}\int i_1(t')dt'$ ist. Die Übertragungsfunktion lautet hier $G_C = 1/Cs$.
- Tatsächlich liegt aber am Widerstand R gar nicht die von außen angebotene Spannung u_0 an, dieser wirkt vielmehr die sich am Kondensator C aufbauende Spannung u_1 entgegen; also wirkt nur die Differenz $u_0 - u_1$ die man sich durch ein Verknüpfungsglied erzeugt denken kann.

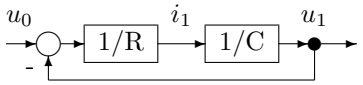


Bild r5p20

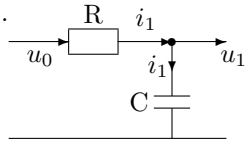


Bild r5p16

Das Resultat erhält man (iterativ) aus:

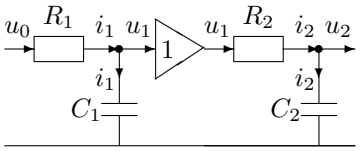
$$V(s) = G_R(s) \cdot G_C(s) \cdot (U(s) - V(s)) = \frac{1}{R} \cdot \frac{1}{Cs} \cdot (U(s) - V(s))$$

Die Auflösung nach $V(s)$ ergibt:

$$V(s) = U(s) \cdot \frac{1}{sRC} / (1 + \frac{1}{sRC}) = U(s) \cdot 1/(sRC + 1)$$

Die Übertragungsfunktion wird also wieder $G(s) = 1/(1 + sRC)$ mit $s = (i\omega + \alpha)$

2. Der ideale zweistufige RC-Tiefpaß besteht aus 2 in Serie geschalteten RC-Tiefpässen, die über ein geeignetes Entkopplungsglied (ein P-Glied mit Verstärkung 1) verbunden sind, damit an der Knotenstelle R_1C_1 kein Strom abgezweigt wird, die Rückwirkungsfreiheit also gegeben ist.



r5p21.pic

Aus der bekannten Übertragungsfunktion

$$F_i(\omega) = 1/(1 + i\omega T_i)$$

für einen einfachen RC-Tiefpaß (mit $T_i = R_iC_i$) erhält man für die Reihenschaltung zweier solcher Glieder:

$$\boxed{F = F_1 \cdot F_2 = 1/(1 + i\omega T_1) \cdot (1 + i\omega T_2)} \qquad \text{bzw} \qquad \boxed{G = \frac{1}{(1+sT_1)} \cdot \frac{1}{(1+sT_2)}}$$

Für den Amplitudengang erhält man das Produkt

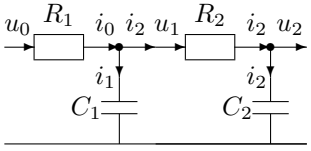
$$f(\omega) = |F| = 1/\sqrt{(1 + \omega^2T_1^2)(1 + \omega^2T_2^2)} = |F_1| \cdot |F_2|$$

mit Werten zwischen 0 und 1.

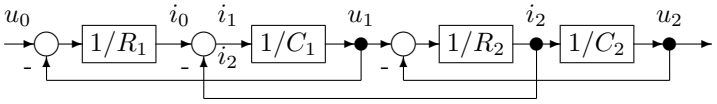
Für den Phasengang erhält man $\boxed{\varphi(\omega) = tan^{-1}(\frac{-\omega(T_1+T_2)^2}{1-\omega^2(T_1T_2)})}$

mit Werten zwischen 0 und 180 Grad.

3. Der reale zweistufige RC-Tiefpaß ohne Entkopplungsglied enthält mehrere Rückwirkungen: 2 mal die Rückwirkung der Ladespannungen U_1 und U_2 aber auch die Stromverzweigung im Inneren des 1. RC-Glieds, die als negative Rückwirkung verstanden werden muß. Damit ergibt sich eine mehrfach verschlungene Rückwirkung, wie sie im Bild gezeigt ist.



r5p23.pic



r5p22.pic

Die Lösung ergibt sich nach kurzer Rechnung zu:

$$\boxed{V(s) = U(s)/(1 + k_1 \cdot i\omega + k_2 \cdot (i\omega)^2)} \qquad \text{bzw} \qquad \boxed{G_s = \frac{1}{(1+k_1s+k_2s^2)}}$$

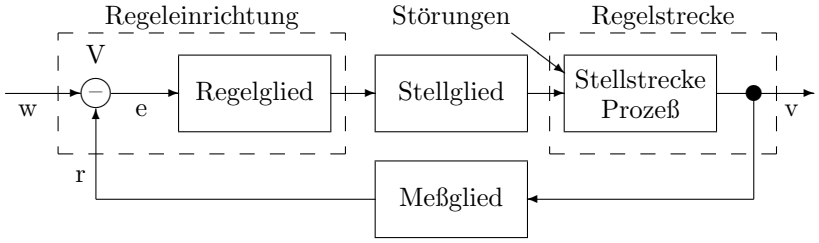
mit $k_1 = T_1 + T_{12} + T_2 = R_1 \cdot C_1 + R_1 \cdot C_2 + R_2 \cdot C_2$
und $k_2 = T_1 \cdot T_2 = R_1 \cdot C_1 \ \cdot \ R_2 \cdot C_2$
wie man es auch für den Frequenzgang $F(\omega)$ aus den Kirchhoffschen Sätzen leicht hätte ausrechnen können.

Typische reale Übertragungsglieder

- PT1-Glied $G = k_p/(1 + sT_1s)$
- PT2-Glied $G = k_p/(1 + 2dsT_2 + s^2T_2^2)$
- PD-Glied $G = k_p \cdot (1 + sT_1)$
- PID-Glied $G = k_p \cdot (1 + \frac{1}{sT_I} + sT_D)$

5.3 Regelung

a) Der Wirkungsplan: Ein System (von Übertragungsgliedern) mit Rückwirkung

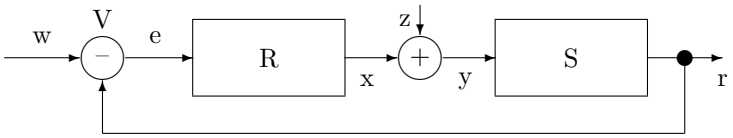


r5p30.pic

w = Führungsgröße (Sollwert)
v = Ausgangsgröße (Istwert)
r = Regel-, Rückführungsgröße $r = G_m * v$
e = Regelabweichung, -differenz $e = w - r$
z = Störgrößen
x = Steuergröße
y = Stellgrößen
S = Regelstrecke = zu regelndes System, Prozeß
V = Vergleichsglied (Differenzbildung von Ist- und Sollwert)
M = Meßeinrichtung, oBdA ist $G_m = 1$
R = Regelglied, -einrichtung

Die Konstruktion des Regelglieds aus einfachen oder elementaren Übertragungsgliedern (P, I, D, PID) zur Erreichung des Regelziels ist Aufgabe der Regelungstechnik.

b) Das Regelziel:



r5p31.pic

Das Regelziel ist, daß der Istwert am Ausgang (r), dem Sollwert w am Eingang möglichst genau folgt, und von Störeinwirkungen z nicht beeinflußt wird, d.h. $r = w$ ist.

Kreisverstärkung: $G_0 = G_R \cdot G_S$ = Verhalten des offenen Regelkreises

Grundgleichungen:

$$R = (X + Z) \cdot G_S$$

$$X = E \cdot G_R$$

$$E = W - R$$

Damit läßt sich das notwendige Verhalten des Regelglieds berechnen:

$$R = (E \cdot G_R + Z) \cdot G_S = E \cdot G_R \cdot G_S + Z \cdot G_S$$

$$R = (W - R) \cdot G_0 + Z \cdot G_S$$

$$R \cdot (1 + G_0) = W \cdot G_0 + Z \cdot G_S$$

$$R = W \cdot \frac{G_0}{1 + G_0} + Z \cdot \frac{G_S}{1 + G_0}$$

$$R = W \cdot 1 + Z \cdot 0 \text{ für } G_0 \to \infty$$

Das Regelziel wird also genau dann erreicht, wenn die Kreisverstärkung $= \infty$ wird.
Da das Übertragungsverhalten G_S der Regelstrecke durch den Prozeß vorgegeben ist, gilt es für den Regler eine Übertragungsfunktion G_R zu finden, durch welche $G_0 = G_R \cdot G_S = \infty$ wird, bzw. $G_R = G_0 \cdot G_S^{-1}$
Diese Bedingung ist natürlich nie exakt zu erfüllen,
- es wird immer $G_0 < \infty$ sein,
- die Inverse G_S^{-1} wird nicht immer exakt zu bestimmen sein (s.w.u.).

c) Das Regelproblem: Die Selbsterregung.
Falls $G_0(i\omega_1) = -1$ wird, wird der Ausgabewert R beliebig groß ($R \to \infty$) Dann beginnt das System mit der entsprechenden Kreisfrequenz (ω_1) und ansteigender Amplitude zu schwingen, was in vielen Fällen zur Selbstzerstörung führt.

Auch in den Fällen, in denen die Ortskurve von $G(s)$ nicht direkt durch den Punkt -1 der reellen Achse führt, kann dies jedoch während des Einschaltens des Systems passieren. Hierbei wird sich die wirkliche Kreisverstärkung $G_0(t)$, von der Verstärkung 0 ausgehend, erst aufbauen und in einer Zwischenstufe auch direkt durch den kritischen Punkt verlaufen. Ob dies vorkommen kann, hängt von der Gestalt der Ortskurve von G_0 ab. Das Nyquistkriterium gibt hierfür eine Regel an:

d) Das Nyquistkriterium: Ein Regelkreis ist nur dann stabil, wenn der kritische Punkt beim Durchlaufen der Ortskurve $G_0(i\omega)$ von $\omega = 0$ nach $\omega = \infty$ im Gebiet links der Ortskurve liegt. (Nyquist 1923)

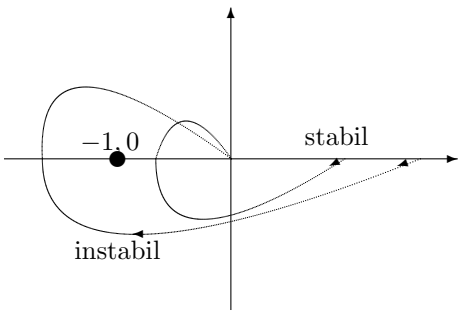


Bild r5p32

5.3.1 Klassen von Übertragungsgliedern

a) Übertragungseigenschaften und -merkmale

Die Ausgangsgröße wird formal als $V = \phi(U)$ dargestellt.

- Linearität:
  — Additiv: $\phi(U_1 + U_2) = \phi(U_1) + \phi(U_2)$
  — Homogenität: $\phi(c \cdot U) = c \cdot \phi(U)$
- Stabilität: $\phi(U + \Delta U) = \phi(U) + \Delta\phi(U)$
  d.h. wenn U von beschränkter Schwankung ist, dann ist auch V von beschränkter Schwankung.
  Ein Gegenbeispiel ist das D-Glied, das z.B.. eine (beschränkte) Sprungfunktion σ in eine unbeschränkte Impulsfunktion δ umwandelt.

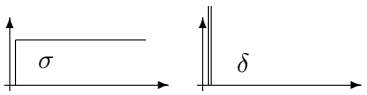


Bild r5p33

- Zeitinvarianz: $\phi(U_1(t - t_0)) = \phi(U_2(t))$ falls $U_1(t - t_0)) = U_2(t)$ ist.
  D.h. wenn die Übertragungsfunktion nicht selber zeitabhängig ist.
  Ein Gegenbeispiel wäre $v(t) = t \cdot u(t)$

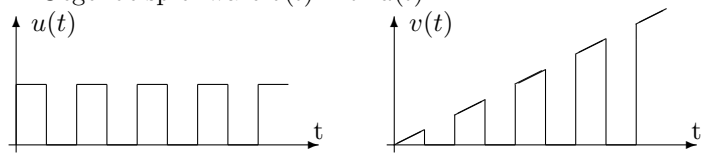


Bild r5p34

- Speicherfreiheit: die Ausgabe v(t) hängt nicht von der Vorgeschichte ab.
  Beispiele: P-Glied, D-Glied.
  Gegenbeispiele: I-Glied, T-Glied.
- Kausalität: die Ausgabe hängt nicht von der Zukunft ab. D.h. zum Beispiel, daß ein T-Glied nur verzögern kann ($t_0 \geq 0$).
- Umkehrbarkeit: Die Umkehrfunktion $U = \phi^{-1}(V)$ ist eindeutig bestimmbar.
  Beispiele: I-Glied und D-Glied
  Gegenbeispiele: beim Quadierer $v(t) = u^2(t)$ kann das Vorzeichen nicht mehr rekonstruiert werden, das T-Glied hat eine nicht-kausale Umkehrfunktion.

b) Klassen von Systemen:

- Kontinuierliche Systeme verarbeiten analoge Signale, $u(t)$ und $v(t)$ sind (aus einem meist eingeschränkten) Bereich beliebig wählbar ($u, v, t \in \mathcal{R}$)
- Zeitdiskrete Systeme verarbeiten zeitdiskrete Signale, also solche, die nur zu bestimmten Zeitpunkten $t_i (= n \cdot T)$ definiert sind; in der Regel sind sie durch eine Zeitbasis (Takt T) vorgegeben und äquidistant.
- Digitale Systeme verarbeiten nur diskrete Werte $u_i(t)$ und $v_i(t)$
- LTI-Systeme (Linear Time-Invariant) sind die am meisten betrachteten Systeme, da sie leicht zu berechnen sind. Andere Systeme lassen sich oft durch eine Kombination solcher Systeme für begrenzte Wertebereiche nachbilden.

5.3.2 Nichtlineare zeitinvariante Übertragungsglieder

Die Übertragungskennlinien für einige wichtige NTI-Glieder:

a) Der Begrenzer
Alle realen Übertragungssysteme besitzen obere und untere Wertegrenzen, bei deren Erreichen man auch von Übersteuerung spricht. Manchmal werden solche Systeme bewußt als Begrenzungsglieder eingesetzt, z.B. zum Schutz von Systemeingängen.

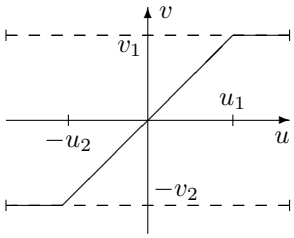


Bild r5p35

$$v = \begin{cases} v_1 & für \quad u > u_1 \\ c \cdot u & sonst \\ -v_2 & für \quad u < -u_2 \end{cases}$$

b) Das Zweipunktglied
Ähnlich wie beim Begrenzer gibt es beim Zweipunktglied 2 Grenzwerte des Ausgangswert v, zwischen denen bei einem bestimmten Wert u_s der (kontinuierlich) veränderlichen Eingansspannung u abrupt umgeschaltet wird. Realisiert durch eine hohe Steilheit des Begrenzers. Hier ist nur ein asymmetrisches Glied gezeigt, wie es in Rechner typisch ist; aber auch nagtive Werte können zusätzlich implementiert sein.

Dieses Glied kann als einfachster Analog-Digital-Wandler (hier mit 1 bit Breite) betrachtet werden. (Beispiel: Bimetallschalter)

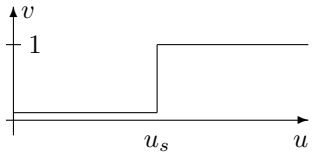


Bild r5p36

$$v = \begin{cases} 1 & für \quad u > u_s \\ 0 & für \quad u < u_s \end{cases}$$

c) Der Schmitt-Trigger (Das Zweipunktglied mit Schaltdifferenz)
Ein Zweipunktglied mit Schaltdifferenz schaltet bei unterschiedlichen Eingangswerten u_1 und u_2, jenachdem ob u ansteigt oder fällt. Man spricht hier auch von Hysterese, wenn die Kennlinie symmetrisch zum Nullpunkt ist. Dieses System besitzt ein Gedächtnis, ist also nicht speicherfrei. Solche Übertragungsglieder sind typisch für alle Speicherbausteine der EDV.

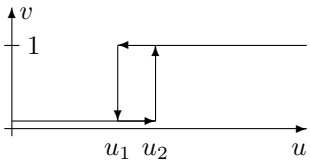


Bild r5p37

$$v = \begin{cases} 0 \; für \; u < u_1 \\ 0 \; für \; u_1 < u(t) < u_2 \quad und \quad u(t-\delta) < u_2 \\ 1 \; für \; u_1 < u(t) < u_2 \quad und \quad u(t-\delta) > u_1 \\ 1 \; für \; u > u_2 \end{cases}$$

5.3.3 Abtastende Übertragungsglieder

Abtastende Übertragungsglieder machen eine Umsetzung von zeitkontinuierlichen (meist auch analogen, d.h. wertkontinuierlichen) Signalen in zeitdiskrete Signale. Die weitere Umsetzung in wertdiskrete Signale wird dann von Analog-Digital-Wandlern durchgeführt.

a) Das Abtast-und-Halte-Glied (Sample & Hold)
Das Abtastglied liefert von einem zeitkontinuierlichen Signal $u(t)$ die Werte zu vorgegebenen Zeitpunkten t_n. In der Regel werden diese anhand einer Zeitbasis T äquidistant (periodisch) gesetzt $t_n = n \cdot T$. Das Halteglied speichert diese Werte, so daß sie (zeitkontinuierlich) einer weiteren Verarbeitung zur Verfügung stehen.

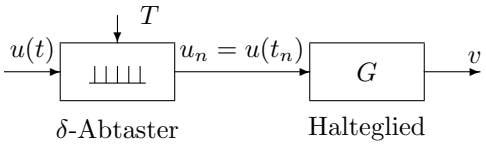


Bild r5p38

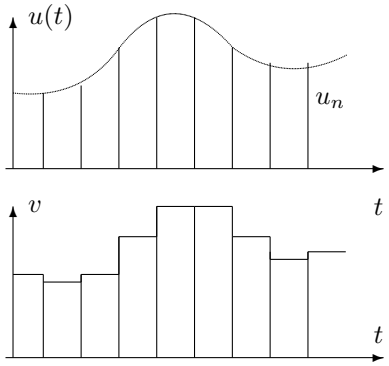


Bild r5p39

b) Das Abtasttheorem von Shannon
- Das Signal $x(t)$ sei bandbegrenzt, d.h. $X(i\omega) = 0$ für $\omega > \omega_0$
- dann ist $x(t)$ eindeutig bestimmt durch Abtastwerte $x_n = x(n \cdot T)$ mit $n \in \mathcal{N}$
- falls die Abtastrate $\omega_A = \frac{2\pi}{T} > 2 \cdot \omega_0$ ist.
- Die Rekonstruktion erfolgt durch ein ideales Tiefpaßfilter mit der Verstärkung $V = T \cdot \omega_0$ und der Grenzfrequenz ω_g mit $\omega_0 < \omega_g < \omega_A - \omega_0$
- Das rekonstruierte Signal hat gegenüber dem Original eine Phasenverschiebung von $T/2$.

c) Die Z-Transformation
Zur quantitativen Berechnung von diskreten Systemen muß an Stelle der Laplace-Transformation die Z-Transformation zur Darstellung von Signalen $X(z)$ und von Übertragungsfunktionen $G(z)$ verwendet werden.
Die Z-Transformation transformiert eine diskrete Funktion $u(n)$, z.B. eine Impulsfunktion $u(t_n)$ oder eine Wertefolge $\{u_n\}$ in eine (stetige) Funktion $U(z)$ der komplexen Variablen $z = r \cdot e^{i\varphi}$, die den Platz von $e^{iw+\alpha}$ einnimmt. Die Abtastung wird als periodisch (mit der Periode T) angesetzt.

$$\mathcal{Z}(u(n)) = U(z) = \sum_{n=-\infty}^{\infty} u(n) z^{-n}$$

Beispiel:
Das Eingangssignal sei die Sprungfunktion $\sigma(t)$, die nach der Abtastung den Wert $u_n = 1$ für alle $n \geq 0$ hat. Wendet man hierauf die z-Transformation an, so erhält man:

$$U(z) = \sum_{n=0}^{\infty} 1 \cdot z^{-n} = \frac{z}{z-1}$$

5.3.4 Digitale Regelung

In einer digitalen Regelung wird grundsätzlich ein PID-Glied verwendet, dessen Parameter dem zu regelnden Prozeß angepaßt sein müssen. Im Prinzip muß hier die Inverse der z-Transformierten des Prozesses nachgebildet werden.

Tatsächlich wird das PID-Glied rein formal gebildet und die Koeffizienten empirisch oder adaptiv angepaßt.

Es seien (vgl.)
r_i die Eingabewerte vom Prozeß zum Rechner
y_i die Ausgabewerte vom Rechner an den Prozeß
w_i die Sollwerte im Rechner
e_i die Regelabweichungen, die im Rechner leicht gebildet werden können: $e_i = w_i - r_i$

Dann erhält man als allgemeinen Algorithmus:

$$\begin{aligned} y_i &= k_p \cdot e_i \qquad \text{(P-Glied)} \\ &+ k_d \cdot (e_i - e_{i-1}) \qquad \text{(D-Glied)} \\ &+ k_i \cdot (e_i + e_{i-1} + e_{i-2} + e_{i-3} + e_{i-4} + e_{i-5}..) \qquad \text{(I-Glied)} \end{aligned}$$

Der vorhergehende Wert war aber:

$$\begin{aligned} y_{i-1} &= k_p \cdot e_{i-1} \\ &+ k_d \cdot (e_{i-1} - e_{i-2}) \\ &+ k_i \cdot (e_{i-1} + e_{i-2} + e_{i-3} + e_{i-4} + e_{i-5}...) \end{aligned}$$

Unter Verwendung dieses Werts wird:

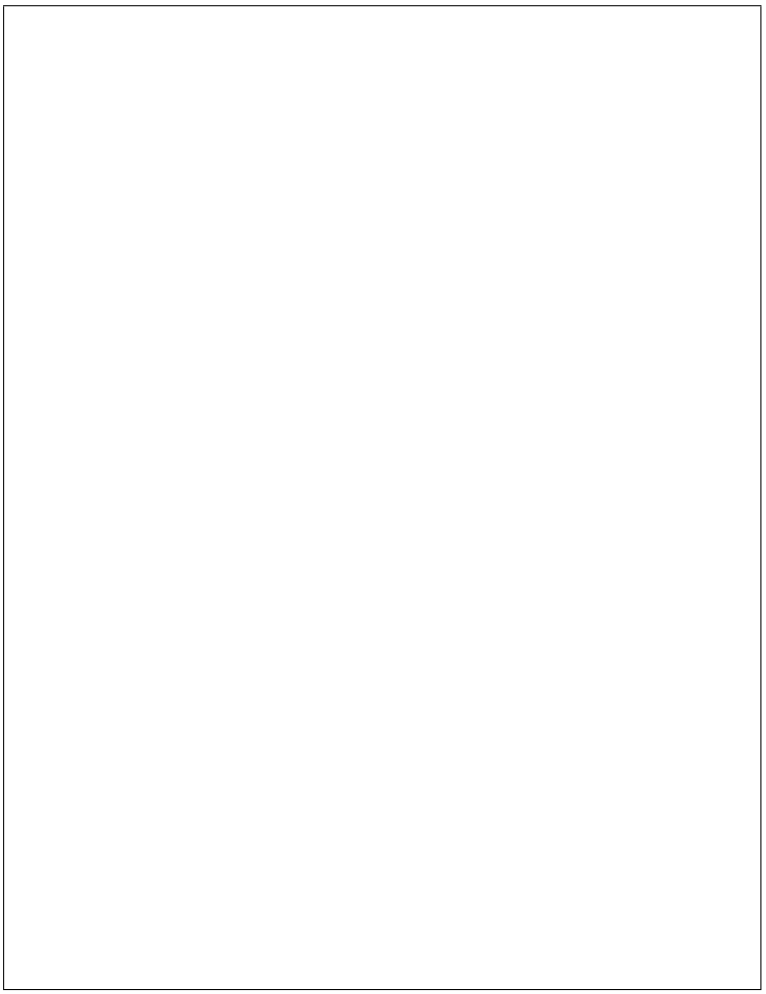
$$y_i = d_0 \cdot y_{i-1} + d_1 \cdot e_i - d_2 \cdot e_{i-1}$$

Hiernach müssen nur noch 3 Werte gespeichert werden und die Summation aller Werte e_i für den Integralteil (der leicht zum Überlauf oder zu Rundungsfehlern führen kann) entfällt.
Der Zusammenhang zwischen den Parametern ist:

$$\begin{aligned} k_p &= d_1 - (2d_0 - 1)(d_0 d_1 - d_2) \qquad \text{und} \quad d_0 = k_P + k_I \cdot T_A + k_D/T_A \\ k_d &= (d_0 - 1)(d_0 d_1 - d_2) \qquad\qquad\quad d_1 = -k_P - 2k_D/T_A \\ k_I &= d_0(d_0 d_1 - d_2) \qquad\qquad\quad d_2 = k_p + k_D/T_A \end{aligned}$$

5.4 Übertragungsfunktionen

Die Grundtypen von Regelkreisgliedern und ihre Darstellung als Übergangsfunktion und als Frequenzgang.



5.5 Laplace-Integrale

Zeitfunktion $f(t)$	Laplace-Transformierte $F(s) = \mathcal{L}(f(t))$
$\delta(t)$ (Impuls)	1
$\sigma(t)$ (Sprung)	1/s (für s > 0)
c = const	c/s
t	$1/s^2$
t^n	$n!/(s^{n+1})$
$e^{\pm at}$	$1/(s \mp a)$ (für $(Re(s) > Re(a))$
te^{-at}	$1/(s+a)^2$
t^2e^{-at}	$2/(s+a)^3$
t^ne^{-at}	$n!/(s+a)^{n+1}$
$1 - e^{-at}$	$a/(s(s+a))$
a^t	$1/(s - ln\|a\|)$
$sin(at)$	$a/(s^2+a^2)$
$cos(at)$	$s/(s^2+a^2)$
$sin(\omega t + \varphi)$	$cos\varphi \cdot \frac{\omega}{s^2+\omega^2} + sin\varphi \cdot \frac{s}{s^2+\omega^2}$
$sinh(at)$	$a/(s^2-a^2)$
$cosh(at)$	$s/(s^2-a^2)$
$e^{at} - e^{bt}$	$(a-b)/(s-a)(s-b)$
$e^{-at}sin(bt)$	$b/((s+a)^2+b^2)$
$e^{-at}cos(bt)$	$(s+a)/((s+a)^2+b^2)$

.

Kapitel 6

Prozeßrechner-Hardware

6.1 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

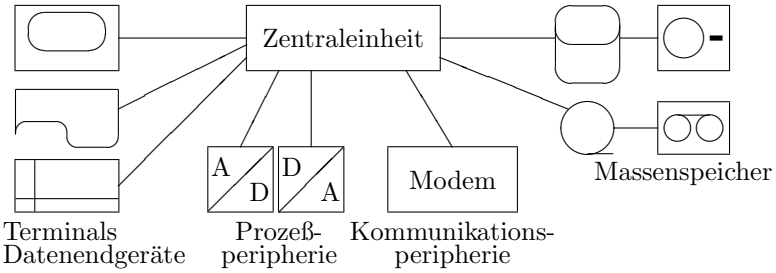


Bild r6p01

6.2 Die Zentraleinheit

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

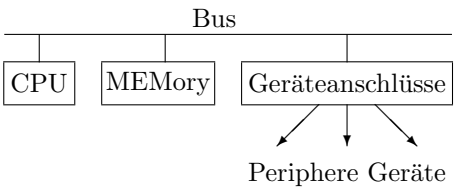


Bild r6p02

- Der Bus verbindet die Funktionseinheiten (auch andere Topologien z.B. Stern)
- Der Arbeitsspeicher (MEMory) enthält Daten und Instruktionen
- Die Geräteanschlüsse sorgen für die Kommunikation mit den peripheren Geräten (oft enthalten sie eigene Prozessoren)

6.2.1 Die CPU

- Leitwerk: interpretiert Instruktionen anhand eines Mikroprogramms.
- Rechenwerk: führt Operationen aus, meist nur auf ganze Zahlen (Integer)
- Das Unterbrechungswerk (Interrupt Arbitrator, IA) bearbeitet Interrupt-Anforderungen.
- Coprozessoren für Gleitpunktoperationen (Floating Point Processor), Textoperationen (Byte Processor)
- Register enthalten die wichtigsten Daten

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

Die Flags im Prozessorstatuswort

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

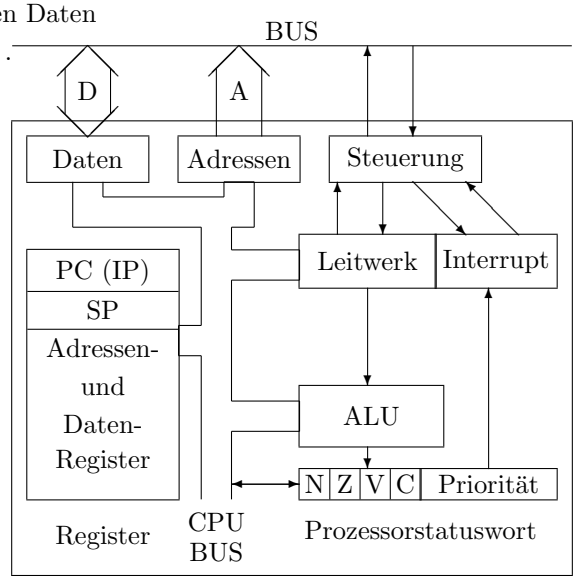


Bild r6p03

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

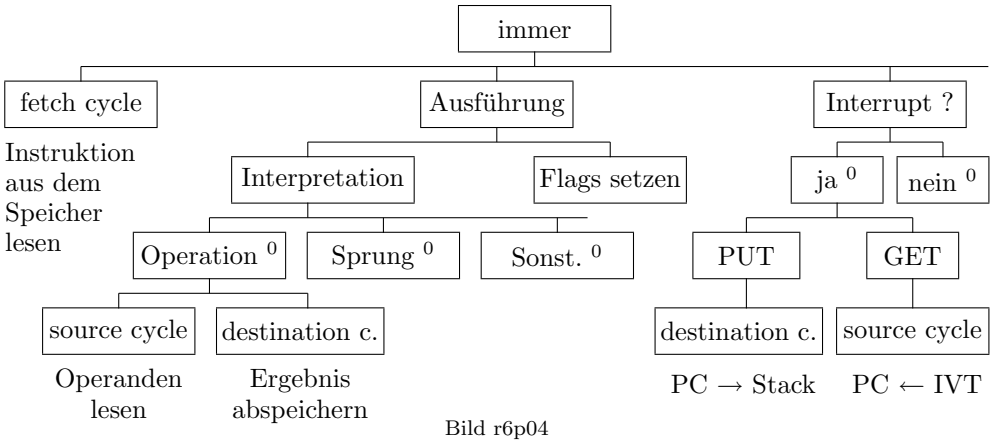


Bild r6p04

6.2.2 Der Bus

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

Das Busprotokoll für einen asynchronen parallelen Bus (im Raummultiplex)
Adressen und Daten werden als Worte parallel übertragen.
Die Steuersignale sorgen im Handshaking-Verfahren für eine gesicherte Datenübertragung.

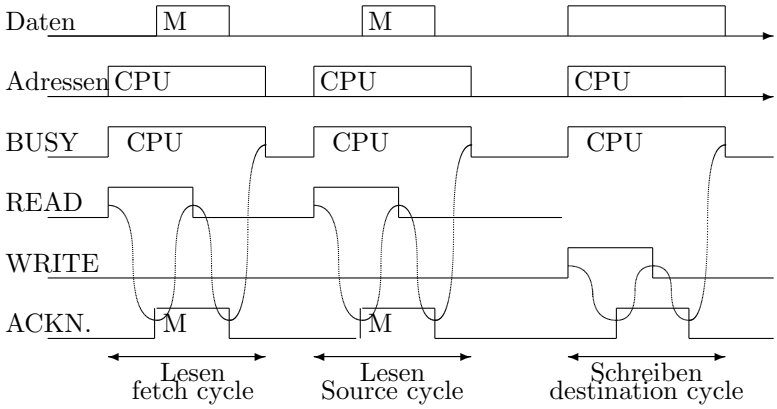


Bild r6p05

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

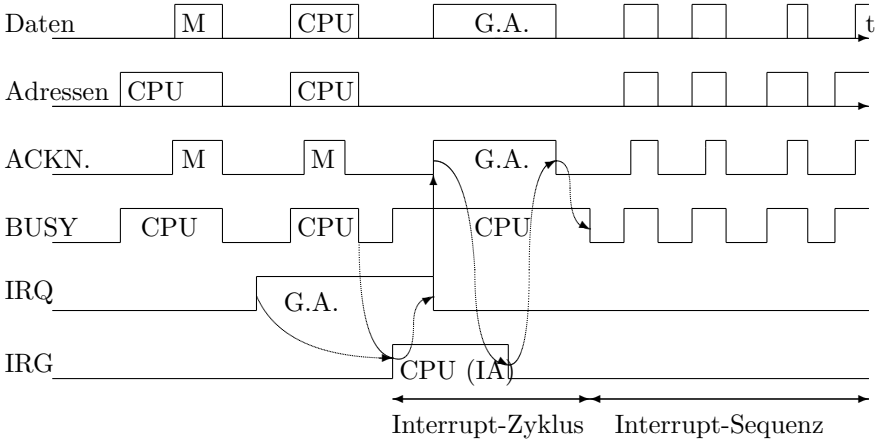


Bild r6p06

6.2.3 Der Arbeitsspeicher (Memory)

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 über die höchstwertigen (MSB) Adreßbits oder die niederwertigsten (LSB), "interleaving". Über die Steuersignale (READ/WRITE) wird die Richtung der Datenübertragung bestimmt (R/W).

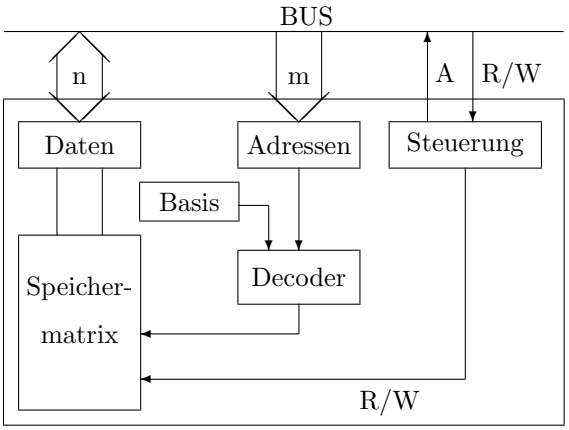


Bild r6p07

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

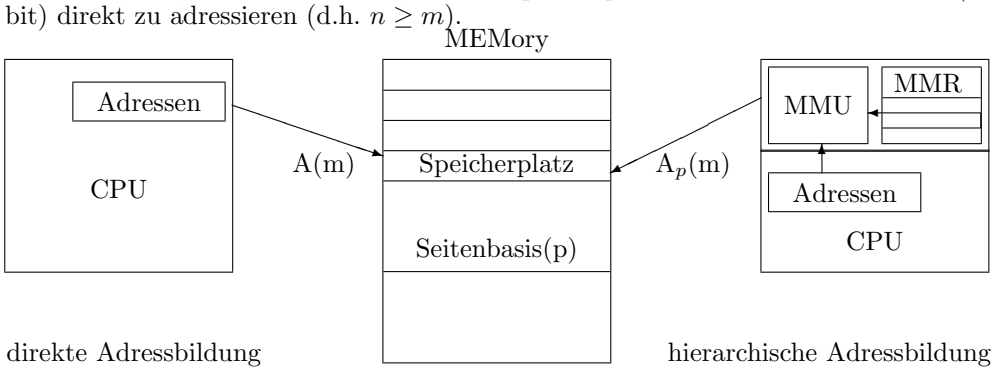


Bild r6p09

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. Die MMU kann Bestandteil der CPU sein, sie kann aber auch in Form eines Geräteanschlusses ausgebildet sein.

6.2.4 Geräteanschlüsse

- Interfaces: byteweise Übertragung zu und und von Datenendgeräten
- Controller: blockweise Übertragung von und zu Massenspeichern

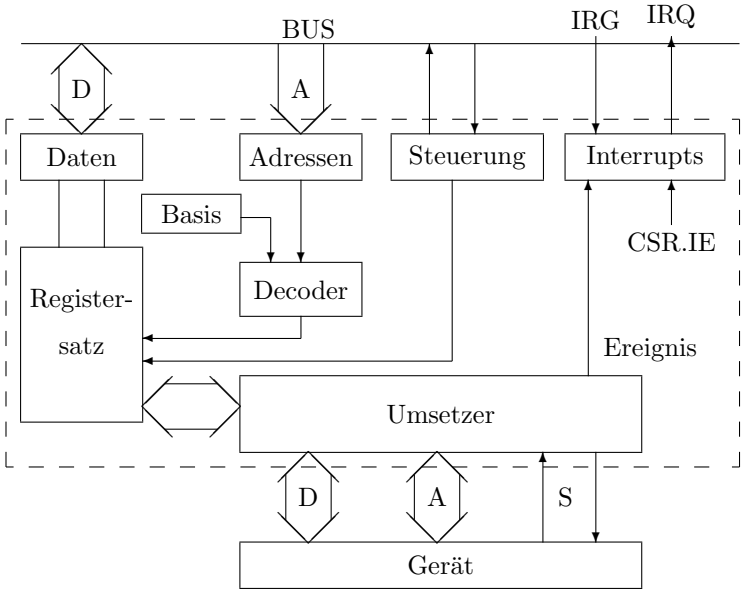


Bild r6p11

a) Busschnittstelle für den Zugriff der CPU auf
- Datenregister (Data Buffer Register, DBR)
- Steuerregister (Control and Status Register, CSR)
- Word Count Register (WCR) bei DMA-Controllern
- Memory Address Register (MAR) bei DMA-Controllern
- Device Address Register (DAR) bei DMA-Controllern

Für alle Geräteanschlüsse einheitlich.

b) Geräteschnittstelle zur Ankopplung von peripheren Geräten, für jeden Gerätetyp unterschiedlich:
- Terminalschnittstelle, seriell, asynchron (V.24, RS 232-C)
- Druckerschnittstelle, parallel (Centronics)
- SCSI (Small Computer System Interface), DMA-Controller
- IEC-Bus (IEEE 488) für diverse Geräte, auch Prozeßperipherie

c) Umsetzungsprotokolle entsprechend der Geräteschnittstelle
- Parallel-Anschlüsse werden direkt zum DBR verbunden
- Serielle Anschlüsse erfordern einen Parallel-Seriell-Umsetzer (z.B. UART, Universal Asynchronous receiver and Transmitter oder PUSART, Programmable Universal Synchroneous and Asynchroneous Receiver and Transmitter)
- DMA-Controller für schnelle Blockübertragung

6.3 Standardperipherie

- Ein/Ausgabegeräte für den Benutzer:
  - Terminal (Bildschirm und Tastatur)
  - Drucker
  - Plotter
  - Maus
- Massenspeicher
  - Magnetplatten, Floppydisk
  - Magnetband, Cassetten

6.3.1 Terminal

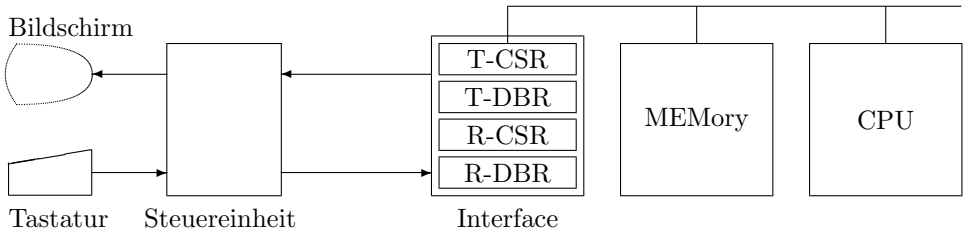


Bild r6p15

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.

Der Bildschirm (Monitor)

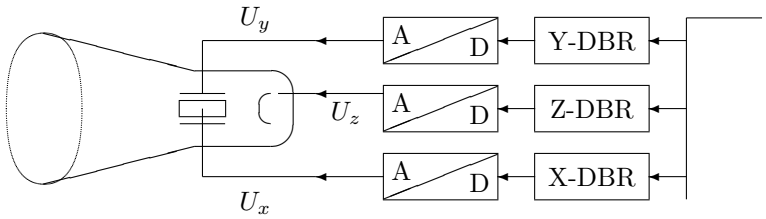


Bild r6p16

Als Bildschirm wird üblicherweise eine Kathodenstrahlröhre (CRT, Cathode Ray Tube) verwendet, wie beim Fernseher. Die Ablenkung in X- und in Y-Richtung erfolgt durch Analogspannungen (U_x und U_y), die aus Digitalwerten durch A/D-Wandler (s.w.u.) erzeugt werden. Die Helligkeit wird durch eine weitere Spannung (U_z) gesteuert. Bei Farbbildröhren werden 3 solcher Spannungen benötigt für die Grundfarben Rot, Grüen, 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.

Neuere Geräte verwenden LCDs (Flüssigkristall-Anzeigen, Liquid Crystal Displays) bei denen sich im Kreuzungspunkt von Ansteuerleitungen (vgl. Tastatur) Flßsigkristalle befinden, die bei angelegter Spannung Licht durchlassen, sonst aber nicht.

Die Tastatur
In modernere Tastaturen werden im Kreuzungspunkt von Ansteuerleitungen Verbindunge durch die Tasten hergestellt, die von einem Micro-Prozessor abgefragt und erkannt werden. An die CPU wird nur der "Scan-Code" geliefert, ein Zahlenwert, der durch Software (Treiber) entsprechend einer Tabelle (z.B.) in entsprechende Zeichen umgewandelt wird.

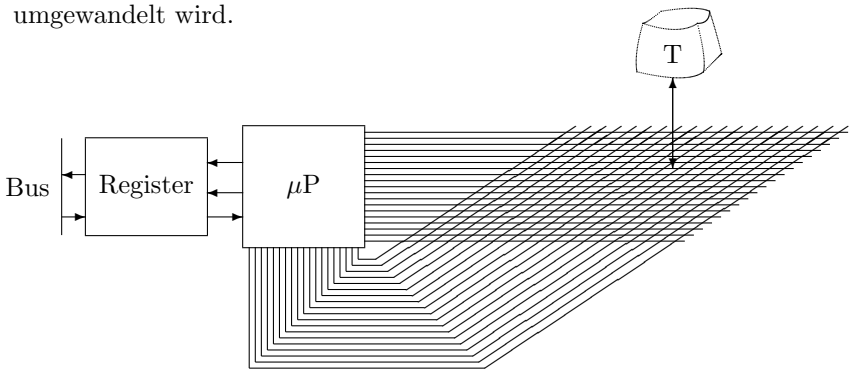


Bild r6p17

6.3.2 Magnetplatten

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

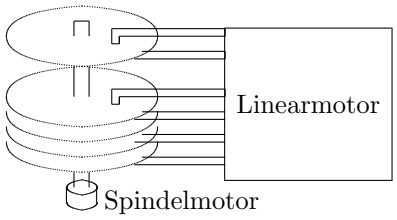


Bild r6p18

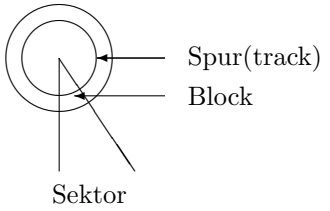


Bild r6p19

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.

Ein Datenblock wird aus eine (sequentiellen) Folge von Bits gebildet. Zusätzlich zu den Nutzdaten enthält ein Block weitere Felder: eine Präambel zur Synchroni-

sation, 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.

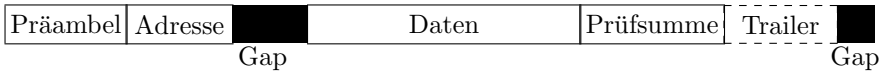


Bild r6p20

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 ($U_{ind} = d\Phi/dt$).

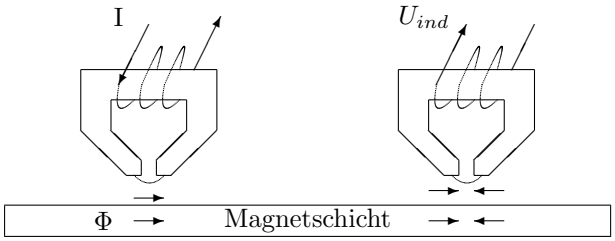


Bild r6p21

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".

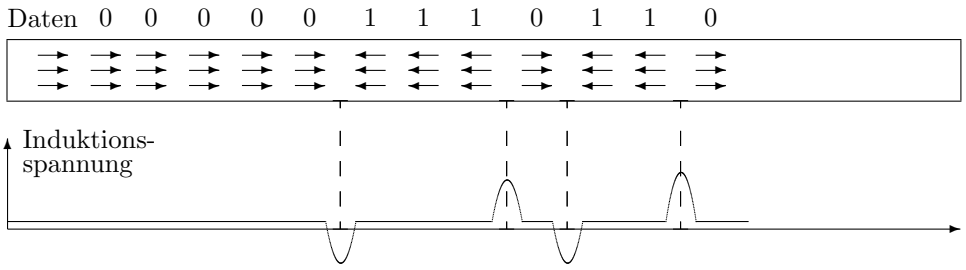


Bild r6p22

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 Blockkapazität: In einem Block werden typischerweise 4096 bit Nutzdaten (seriell) aufgezeichnet, die im Rechner in 512 Byte abgespeichert werden.

Die Spurkapazität ist die Datenmenge in einer Spur, die durch die Anzahl S der Sektoren bestimmt wird; typischerweise ist S = 8 bis 100.

Die Plattenkapazität berechnet sich aus der Blockkapazität und der Anzahl der Blöcke: Seiten · Spuren · Sektoren. Typisch sind heute: 512 Byte · 20 Seiten · 800 Spuren · 50 Sektoren · = 400 Mbyte

Die Übertragungsrate von der Platte in den Arbeittspeicher 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.

6.3.3 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).

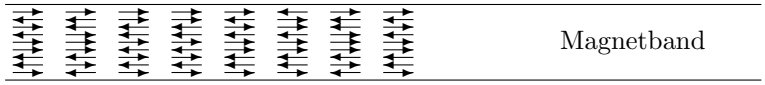


Bild r6p23

6.3.4 CD-ROM

Merkmale:
- Spiralförmige Anordnung (wie bei Schallplatte)
- Datenblöcke konstanter (linearer) Länge (variable Rotations- oder Übertragungsgeschwindigkeit)
- Sequentieller Zugriff (“intelligente“ Controller können Spiralarme überspringen)
- bitserielle (binäre) Aufzeichnung (Reflexion hell/dunkel)

6.3.5 Graphische Ein- und Ausgabe

Die Maus

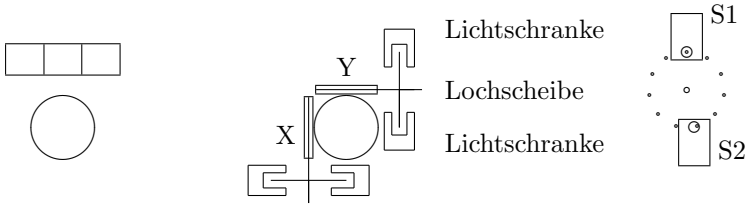


Bild r6p24

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.

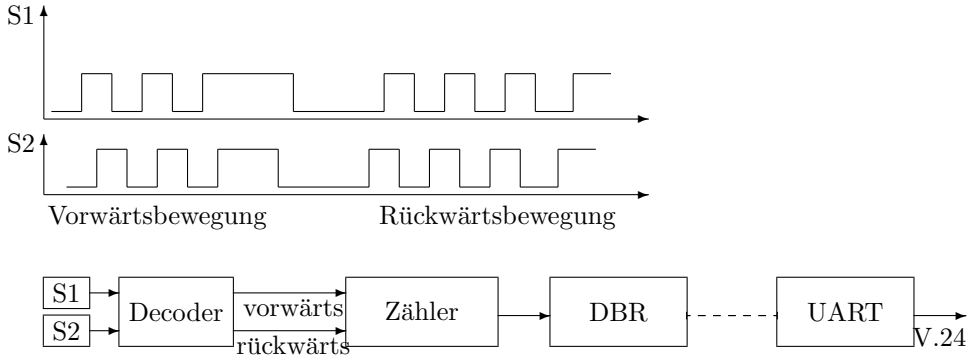


Bild r6p25

Der Plotter

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:

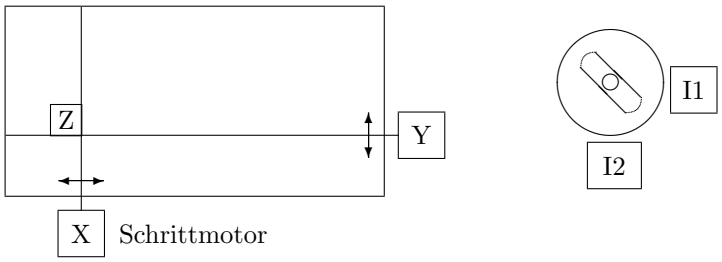


Bild r6p27

Die Impuls erzeugen über Elektromagnete. Kraftwirkungen auf den (magnetischen) Rotor. In der Praxis werden hier meist 7 Paare verwendet, und die Ansteuerung wird von spezielle 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.

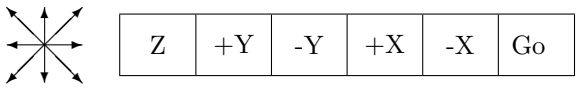


Bild r6p28

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.

6.4 Prozeßperipherie

Digitale Ein- und Ausgabe
Analogein- und -ausgabe, ADC, DAC

6.4.1 Digitale Ausgabe

Digitale Ausgabe (digital output, DO) erfolgt in der Regel bitparallel direkt aus einem Datenregister (DBR).
Die Nachbearbeitung erfolgt in einem Übertragungsglied (D/D)

- zur Verstärkung der elektrischen Spannung oder Leistung
- zur Differenzierung (Impulsbildung)
- zum Integrieren (Zähler).

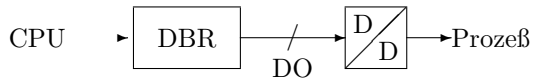


Bild r6p30

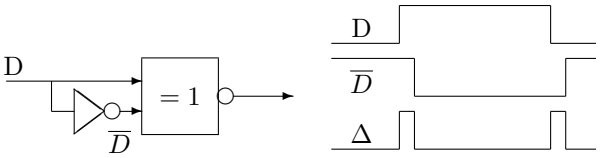


Bild r6p31

Differenzieren mit Hilfe eines Inverters (mit geeigneter Durchlaufverzögerung) und einem XOR-Glied.

6.4.2 Digitale Eingabe

Digitale Eingabe (digital input, DI) erfolgt ebenfalls meist bitparallel in ein Datenregister (DBR) und benötigt vorangehende Signalverarbeitung (D/D).

- zur Impulsformung (Begrenzer, Hoch- oder Tiefpaß)
- zur Differenzierung (Impulsbildung)
- zum Integrieren (Zähler).

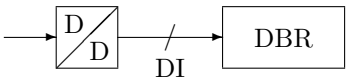
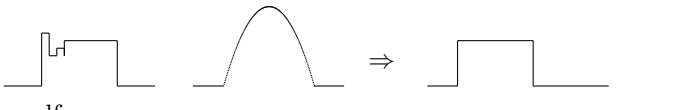


Bild r6p32



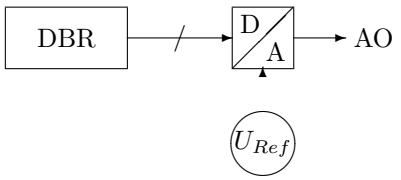
Impuslformer

Differenzierer

Bild r6p33

6.4.3 Analogausgabe

Bei der Analogausgabe (analog output, AO) werden die Daten (D) aus einem Datenregister (DBR) mittels eines Digital-Analog-Wandlers (Digital to Analog Converter, DAC) in elektrische Spannungswerte (U) umgesetzt.



$U_x = U_{ref} \cdot D/2^n$

Bild r6p34

a) Linear-Umsetzer (Potentiometric Ladder)

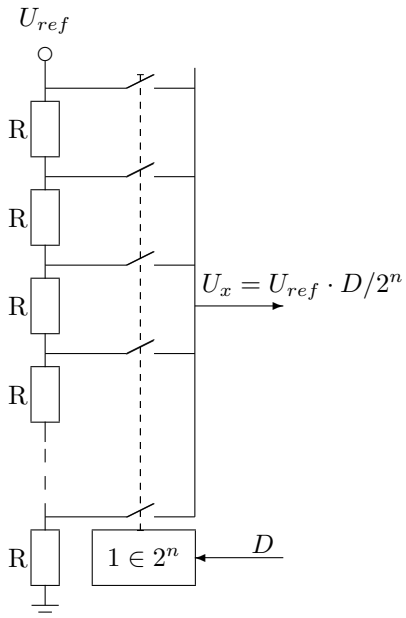


Bild r6p35

. .Beim Linearumsetzer wird von einer Spannungsteilerkette ein Spannungswert durch eine geeignete Anordnung von Schaltern abgegriffen. Es bildet also ein diskretisiertes Potentiometer.
Hier ist eine vereinfachte Anordnung gezeigt, für die eine geeignete Decodierung der n Datenbits für einen von 2^n Schaltern erforderlich ist ($1 \in 2^n$).
In der Praxis wird meist ein binärer Baum von Schaltern verwendet.

Eigenschaften
- schnell ($t_W < 1\mu s$)
- grob ($n = 10$ Auflösung ca. 1:1000)
- aufwendig (je 2^n Widerstände und Schalter)

b) binär gewichteter Stromaddierer (Stromsummenwandler)
Beim Stromaddierer werden mit den Bits b_i des Dualwerts Stromflüsse I_i geschaltet und addiert, so daß daraus eine dem Digitalwert entsprechende Analogspannung U_x entsteht.

$$U_x = R \cdot \sum_{i=1}^{n} I_i \cdot b_n \qquad \text{mit} \qquad I_i = U_{Ref}/R \cdot 2^{n-i-1}$$

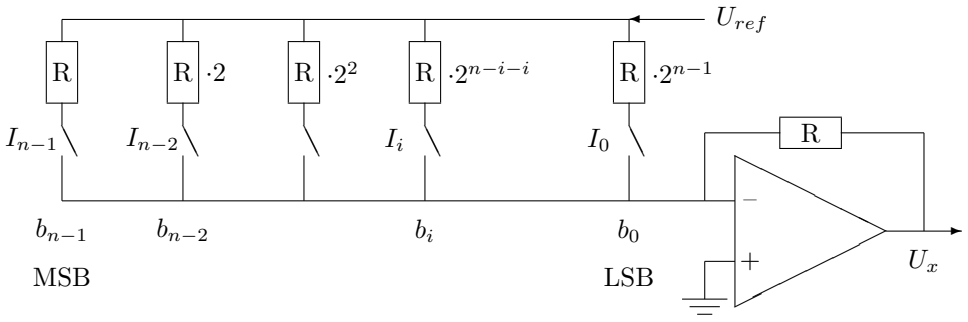
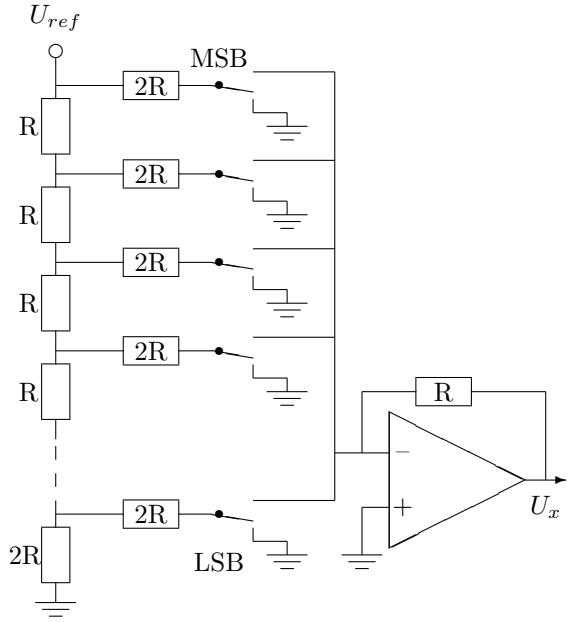


Bild r6p36

Eigenschaften
- geringer Aufwand (n Widerstände, n Schalter, 1 Verstärker)
- mittelschnell ($t_W > 1\mu s$)
- fein ($n \geq 16$ Auflösung ca. 1:1000)
- Monotoniefehler (Sprung beim Umschalten zum höchstwertigen Widerstand)

c) R-2R-Netzwerk



. .Mittels eines geeigneten Schalt- und Widerstandsnetzwerks werden Ströme erzeugt und geschaltet, die dem Dualwert D entsprechen. Der Vorteil dieser Schaltung liegt darin, daß alle Widerstände in derselben Größenordnung (R, 2R) liegen und fertigungstechnisch leichter auf den Sollwert abgeglichen werden können; auch die Langzeitstabilität verbessert sich hierdurch deutlich.

Bild r6p37

Eigenschaften
- geringer Aufwand (2n Widerstände, 2n Schalter, 1 Verstärker)
- mittelschnell ($t_W > 1\mu s$)
- fein ($n \geq 16$ Auflösung ca. 1:100 000)
- keine Monotoniefehler, geringe Linearitätsfehler

d) Bipolare Ausgangsspannungen

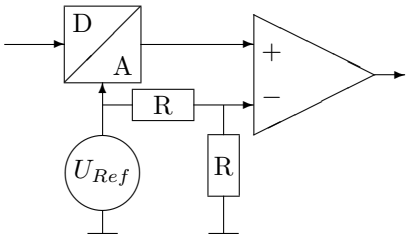


Bild r6p38

. .Um Ausgangsspannungen beliebiger Polarität (±) zu erzeugen, wird eine Vorspannung (Bias) mit dem halben Wert der Referenzspannung (U_{Ref}) auf den verwendeten Operationsverstärker gegeben. Dann müssen die Digitalwerte im 2-er Offset dargestellt werden (nicht im 2-er Komplement):

D	U_x
00000	$-U_{Ref}/2$
10000	0
11111	$+U_{Ref}/2$

e) Qualitätskriterien

- Auflösung: Die Anzahl der Datenbits n ergibt die Zahl der Stufen und die kleinste Schrittweite ($U_{Ref}/2^n$).
- Linearität: Die Stufen können durch unterschiedliche Widerstandswerte innerhalb des Umsetzers unterschiedliche Höhen besitzen.
- Monotonie: Bei groben Abweichungen der Widerstandswerte vom Sollwert kann die Monotonie gestört werden, d.h. Ausgangsspannungen, die zu einem höheren Digitalwert gehören, sind kleiner als der Vorgängerwert.
- Geschwindigkeit: wird meist durch den verwendeten Operationsverstärker bestimmt. Typisch sind hier $5V/\mu s$

Die hier vorgestellten Verfahren:

- Linearumsetzer: sehr schnell ($3ns$), sehr linear, monoton, geringe Auflösung ($n \leq 8$), teuer
- Stromsummenwandler: wenig linear, wenig monoton, langsam ($1\mu s$), hohe Auflösung ($n \leq 18$), billig
- R-2R-Wandler: linear, mäßig monoton, hohe Auflösung ($n \leq 18$), preiswert

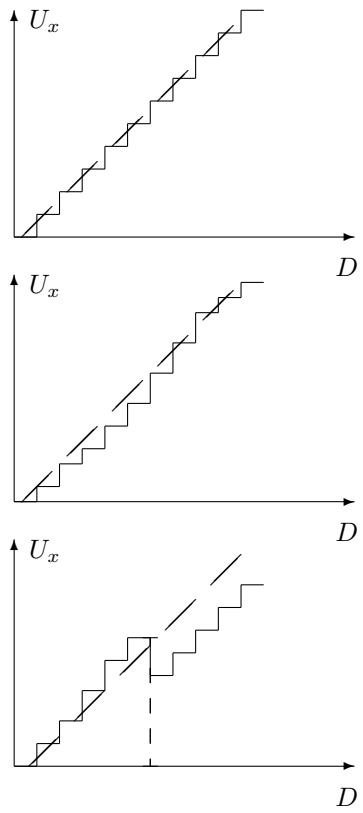


Bild r6p39

6.4.4 Analogeingabe

Bei der Analogeingabe (analog input, AI) werden die Daten durch Vergleich mit einer Referenzspannung U_{Ref} in einem Analog-Digital-Wandler (Anlog to Digital Converter, ADC) gewonnen und im Datenregister DBR abgelegt. Im Gegensatz zur Anlogausgabe müssen in den meisten Fällen Zeitbedingungen beachtet werden, denn die Erfassung und Umwandlung eines Meßwerts erfordert hier eine beträchliche Zeit (die Konvertierungszeit t_c).

Zur Steuerung werden zusätzliche Informatio-. . nen benutzt z.B. ein BUSY-Bit im CSR, welches anzeigt, daß eine Wandlung gerade erfolgt, bzw. ein READY-Bit, welches eine erfolgreiche Wandlung anzeigt. Manche Wandler benötigen ein Startsignal, das vom Programm durch das Setzen eines Bits (GO) im CSR erzeugt werden muß.
Hierin unterscheiden sich die Verfahren stark.

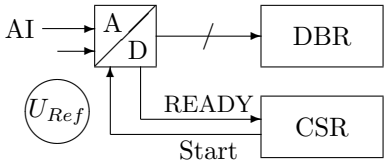


Bild r6p40

$$D = \frac{U_x}{U_{Ref}}(2^n - 1)$$

Die Qualitätskriterien sind vergleichbar mit denen der Digital-Analog-Wandler.

a) Parallelwandler (Flash-Converter)

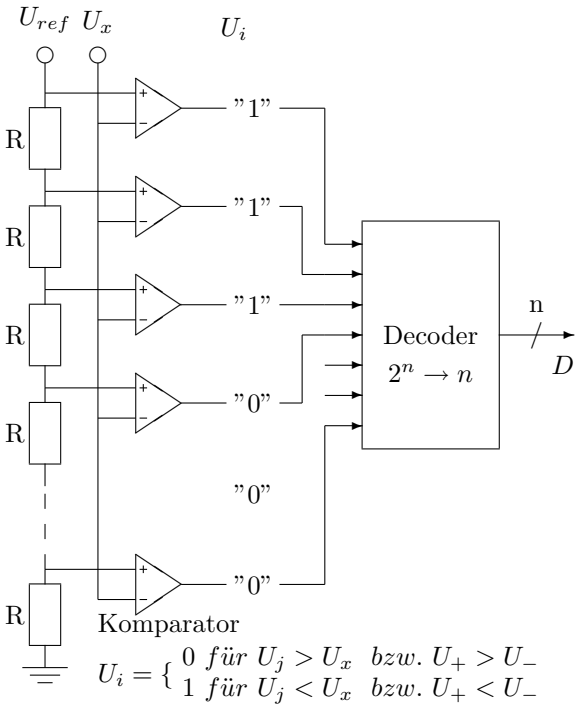


Bild r6p41

. .Im Parallelwandler wird die Eingangsspannung U_x mit einer Reihe von Spannungen verglichen, die durch potentiometrische Teilung aus der Referenzspannung U_{Ref} gewonnen werden. Die hierbei entstehende Reihe von Binärwerten stellt aber noch keine Dualzahl dar, sondern muß durch einen geeigneten Decoder erst umgewandelt werden. Der Vorteil dieses Wandlers ist seine hohe Geschwindigkeit ($t_c \leq 10ns$), der Nachteil seine geringe Auflösung ($n \approx 8$).

Eigenschaften
- sehr schnell ($t_W \leq 10ns$)
- sehr grob
  ($n = 8$ Auflösung ca. 1:250)
- sehr aufwendig
  (je 2^n Widerstände und Komparatoren)

b) Integrationswandler

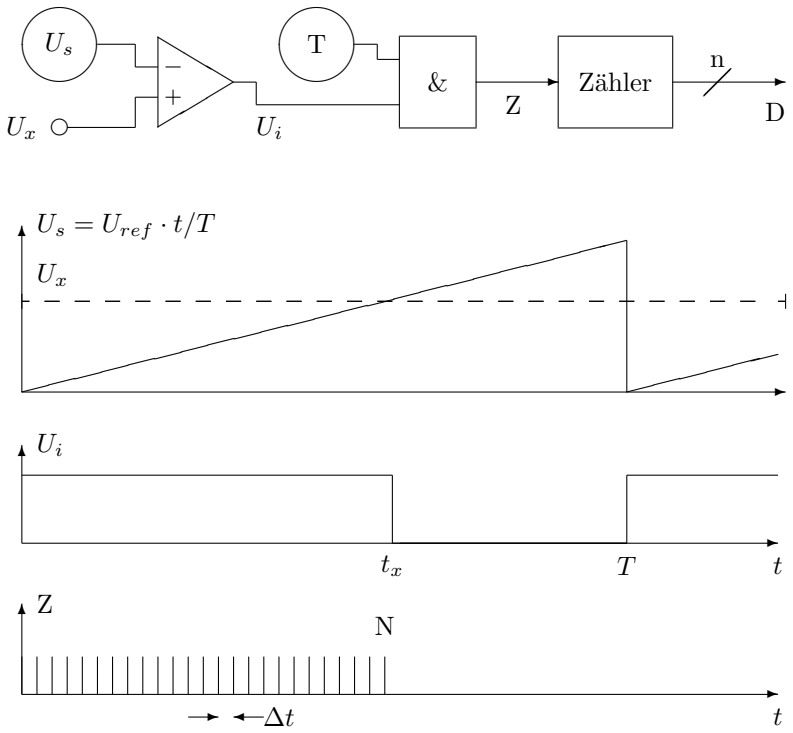


Bild r6p42

Beim Integrationswandler wird die Eingangsspannung U_x mittels eines einzigen Komparators mit einer zeitlich linear ansteigenden (Sägezahn)Spannung U_s verglichen. Bis zum Erreichen des Istwerts U_x wird eine Impulsfolge Z gezählt, deren Wert N den Digitalwert ergibt:

$$\boxed{N = t_x/\Delta t = \frac{T}{\Delta t} \cdot U_x/U_{Ref}}$$

Zur Erzeugung der Sägezahnspannung kann vorteilhaft ein DAC eingesetzt werden, der über einen Zähler eine monoton ansteigende Folge von Werten erhält. Die Auflösung diese Wandlers kann praktisch beliebig groß gemacht werden, dazu muß nur die Impulsfolgefrequenz $f_p = 1/\Delta t$ oder die Meßzeit T großgemacht werden. Letzteres verringert aber die Geschwindigkeit ($t_c = T$). Typische Wandlungszeiten liegen bei 10 ms bei einer Auflösung von 20 bit.
Eigenschaften
- sehr langsam ($t_W \geq 1s$)
- sehr fein ($n \geq 16$ Auflösung ca. 1:100 000)
- aufwendig (verschiedenste Komponenten)
- Linearität abhängig von Sägezahn)

c) Stufenumsetzer

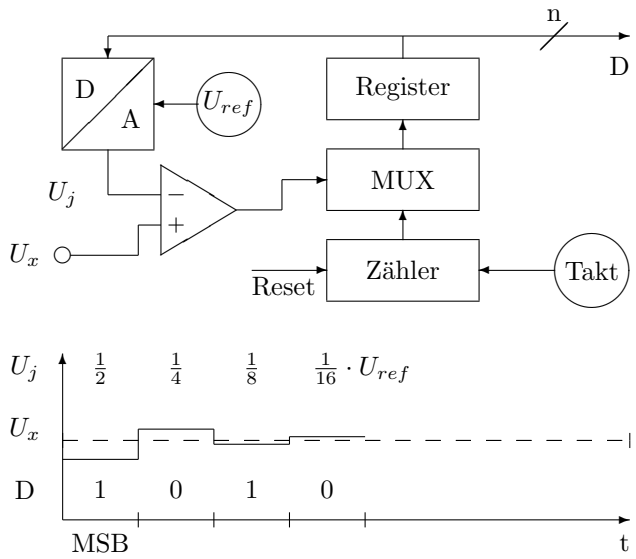


Bild r6p43

Der Stufenumsetzer (Wägecodierer) arbeitet nach dem Prinzip der sukzessiven Approximation. Ähnlich wie beim Integrationswandler wird hier die Eingangsspannung U_x mit einer Spannung U_j verglichen, die jedoch nicht linear durchgefahren, sondern in einem Rückkopplungsverfahren in Binärstufen variiert wird. Der Endwert wird so schon nach n Schritten erreicht. Dieses Verfahren ist recht schnell, pro Schritt werden typ. 1 μs benötigt, so daß eine Wandlung mit einer Auflösung von typ. 16 bit insgesamt $t_c = 16\ \mu s$ benötigt, was einer Konvertierungsfrequenz von $62.5kHz$ entspricht.
Eigenschaften
- schnell ($t_W \approx n \cdot 1\mu s$)
- sehr fein ($n \geq 16$ Auflösung ca. 1:100 000)
- aufwendig (verschiedenste Komponenten)
- Linearität und Monotonie abhängig vom DAC)

6.4.5 Zeitgeber

a) absolute Zeit: Die Systemuhr

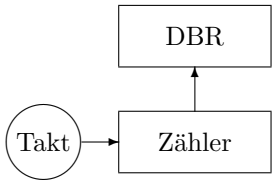


Bild r6p45

. .Die absolute Zeit erhält man von einer Digitaluhr, die im wesentlichen nur aus einem Taktgeber und einem Zähler besteht.
Funkuhr: Von der PTB (Physikalisch Technische Bundesanstalt in Braunschweig) wird ein Signal hoher Frequenzstabilität über den Sender DCF 77 ausgestrahlt, dem die aktuelle Zeit digital aufmoduliert ist.

b) relative Zeit: Timer

Zeiterfassung (Stoppuhr):

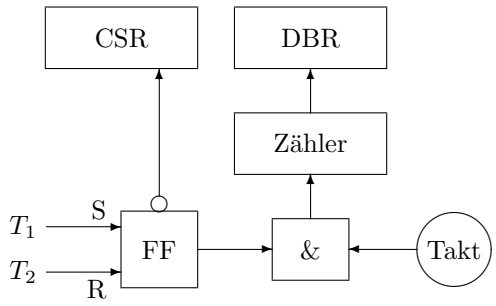


Bild r6p46

..Bildung der Zeitdifferenz ΔT aus einem Start-Ereignis T_1 und einem Stopp-Ereignis T_2.

$$\Delta T = T_2 - T_1$$

Zeitgeber (Wecker):

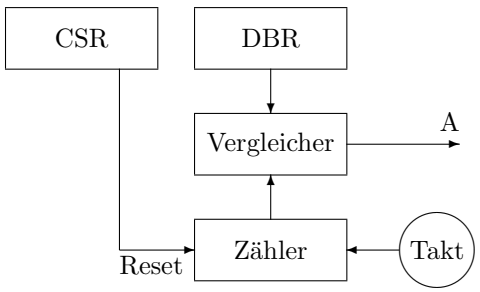


Bild r6p47

..Zählung einer Impulsfolge bekannter Frequenz f bzw. einer Zeitbasis $T = 1/f$

$$\Delta T = N \cdot T$$

während der das Ausgangssignal $A = 1$ ist.

6.5 Prozeßglieder

Prozeßketten
- Meßketten zur Erfassung von Meßwerten
- Steuerketten zur Erzeugung von Stellwerten

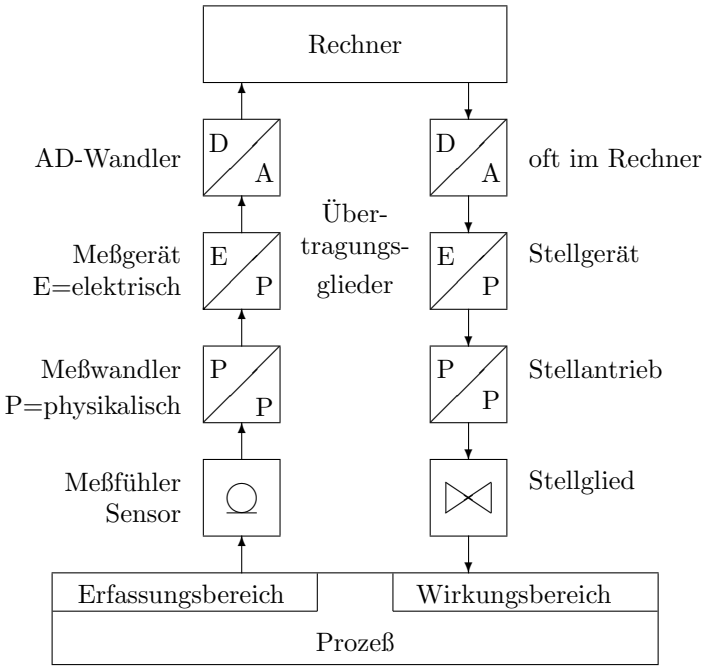


Bild r6p50

Die Prozeßperipherie bildet zwischen Rechner und Prozeß jeweils eine Kette zur Erfassung von Meßgrößen als Eingabedaten und zur Erzeugung von Stellgrößen aus den Ausgabedaten des Rechners.
Die wichtigsten Glieder in diesen Ketten bewirken Umsetzungen zwischen digitalen und analogen Werten (D/A) sowie zwischen elektrischen und nichtelektrischen physikalischen Größen (E/P). Auch Umwandlungen zwischen verschiedenen physikalischen Größen kommen vor (P/P).

6.5.1 Meßglieder, -wandler, und -geräte

6.5.1.1 Übersicht

Meßgröße	Meßglied/-prinzip	Meßbereich (-genauigkeit)	Meßergebnis
Ort (x)	Schleifdraht	1cm bis 10m (±0.1 mm)	el. Widerstand
	Codelineal	dito (± 1 Bit)	Digitalwert (opt., el.)
	Lichtschranken	0.1 mm (±0,1 mm)	Binärwert
	Strichgitter	1cm bis 1m (±0,01 mm)	binäre Impulsfolge
	RADAR	1m bis >100 km (± 1 cm)	Laufzeit
Winkel(φ)	Potentiometer	0 bis 360° $<$ (±1°)	el. Widerstand
	Codescheibe	dito (± 1 Bit)	Digitalwert (opt., el.)
Weg (s = Δx)	Dehnungsmeßstreifen	0.1 bis 100 nm	el. Widerstand
	kapazitive und	100 nm bis 1 mm	Kapazität, Frequenz
	induktive Weggeber	100 nm bis 1 m	Induktivität, Frequenz
	Interferometer	100 nm bis 10 m	Helligkeit, Impulsfolge
Dicke (d = s)	Interferometrie:		
(z.B. Draht)	Optische Beugung	500 nm bis 0.5 mm	Helligkeitsmuster
Zeitintervall	Uhr, Zähler	10^{-9} bis 10^{+9}s ($\pm 10^{-12}$)	Impulsfolge
Frequenz (f)	Zähler	1 Hz bis 1 GHz	binäre Impulsfolge
	Teiler	$>$ 1 GHz	Frequenz
	Überlagerung	bis 10^{15}Hz	Frequenz
Geschwindigkeit	$\bar{v} = \Delta s/\Delta t$		
	Dopplereffekt	$<$ c	Frequenzverschiebung
	Tachometer		Drehzahl
Drehzahl	Tachogenerator	0.1 bis 100000 /min	el. Spannung
	Rasterscheiben		Impulsfolge
Beschleunigung	$\bar{a} = \Delta v/\Delta t$		
	Trägheitssensor	0,1 bis 1000 m/s^2	Trägheitskraft $F = m * a$
Masse (m)	Waage	10^{-6}g bis 1000 to	Kraft
Kraft (F)	elast. Verformung	10^{-6}N bis 10^{6}N	Weg, Winkel
	Hydraulik		Druck
Druck	Manometer	10^{-5} Pa bis 10^{10} Pa	Weg, Winkel
(p= F/A)	Piezoeffekt		el. Spannung
Drehmoment M	Hebel (r)		Kraft $F = M/r$
Durchfluß Q	Meßuhr	10^{-6} bis 10^{6} m^3/s	Drehzahl
	Schwebekörper		Weg
	Staudruckmesser		Druckdifferenz
Dichte	ρ = m / V		
	Auftriebskörper		Kraft, Weg
Feuchte	Hygroskop.Material	0 - 100 %	el. Widerstand
	Dehnung	dito	Weg
	Verdunstung	dito	Temperaturdifferenz
Temperatur	Ausdehung	-270 bis 1000 °C	Weg
	Thermoelement	-273 bis 2000 °C	el. Spannung
	Widerstand (z.B. NTC)	-273 bis 2000 °C	el. Widerstand
Helligkeit	Photoelement	10^{-8} bis 10^{6} W/m^2	el. Spannung
(Intensität)	Photowiderstand, -zelle	10^{-6} bis 10^{6} W/m^2	el. Widerstand
	Photodiode,-transistor	dito	el. Widerstand
	Photonenzähler	$< 10^{-7} W/m^2$	Impulsfolge
Farbe	Spektrometer		Ort, Helligkeit
Stoffzusammen-	Chromatograph		Ort, Farbe,
setzung	Spektrograph		Geschwindigkeit

6.5.1.2 Ausgewählte Verfahren

a) Ortsmessungen

– **Schleifdraht** (Potentiometer) . .
Die abgegriffene Spannung U_x ist genau proportional zum Abstand x:

$$U_x = U_{Ref} \cdot x/l$$

Diese Spannung kann an einen Analogeingang (AI) gelegt werden.

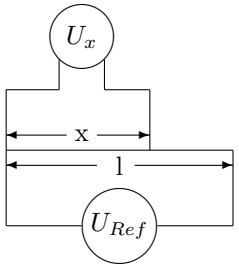


Bild r6p51

– **Rasterlineal** (Strichgitter) ..
Umwandlung (Diskretisierung) der analogen Größe in eine digitale schon auf der physikalischen Ebene.
Bei der Abtastung wird eine (serielle) Bitfolge erzeugt, die von einem Zähler gezählt und auf einen Digitaleingang (DI) gegeben werden kann.

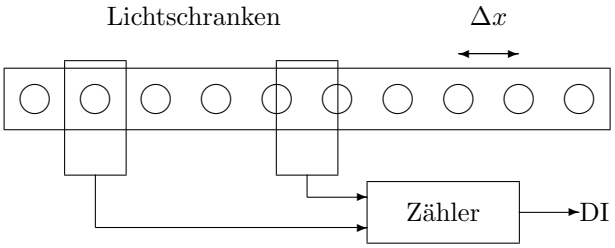


Bild r6p52

Zur Erkennung von Vorwärts- und Rückwärtsbewegungen müssen 2 Sensoren im Abstand $d = (2n + 1) \cdot \Delta x/2$ („auf Lücke“) verwendet werden.

– **Codelineal**

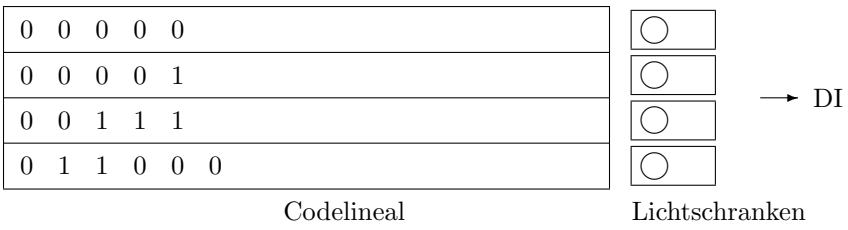


Bild r6p53

Diskretisierung und Digitalisierung des Orts.
Zur Darstellung von ”0” und ”1” können schwarze und weiße Felder bei Reflektionsabtastung verwendet werden, und Löcher für die ”1”-en bei Transmissionsabtastung.
Bei der Abtastung werden n bit breite Digitalwerte erzeugt.
Zur Verringerung der Störanfälligkeit wird oft der Gray-Code verwendet, bei dem sich von Feld zu Feld immer nur 1 Bit ändert. Zur Umwandlung in eine Dualzahl ist ein Decoder notwendig.

b) Geschwindigkeitsmessungen
- Durchschnittsgeschwindigkeit $\bar{v} = \Delta x/\Delta t$
- Momentangeschwindigkeit:
  Tachometer, Tachogenerator: $U_{ind} = C \cdot v$ (Induktionsgesetz)
- Doppler-Radar

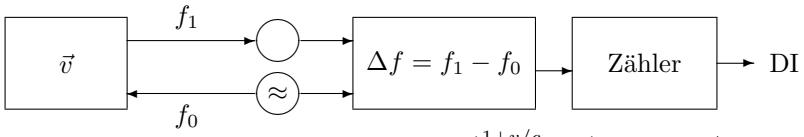


Bild r6p54 Frequenzverschiebung: $\Delta f = f_0(\frac{1+v/c}{1-v/c} - 1) \approx f_0 \cdot 2v/c$

($c = 3 \cdot 10^8 m/s$ ist die Lichtgeschwindigkeit, die stets sehr viel größer ist als v)

c) Beschleunigungsmessungen
- Durchschnittsbeschleunigung: $\bar{a} = \Delta v/\Delta t$
- Trägheitssensor: $\vec{F} = m \cdot \vec{a}$

d) Kraftmessungen
- Federwaage: $\vec{F} = D \cdot \vec{y}$
- Piezo-Effekt: $U = k \cdot A$ $p = k \cdot F$

e) Temperaturmessungen
- Thermische Ausdehnung: $\Delta l = \alpha \cdot \Delta T$ (bis ca. $300°C$)
- Thermospannung: $\Delta U = k \cdot \Delta T$ (bis ca. $1000°C$)
- Strahlungsintensität: $I \sim T^4$ (bis ca. $8000°C$)

6.5.1.3 Fehler

Die einzelnen Verfahren sind jeweils mit Fehlern behaftet, die sich in der Mekette fortpflanzen. Dies muß bei der Auswertung berücksichtigt werden.

6.5.2 Stellglieder, -wandler, und -geräte

6.5.2.1 Übersicht

Stellglied	Stellgröße	Eigenschaft	Steuergröße
Elektromotor	Ort Geschwindigkeit Beschleunigung Drehmoment Kraft Winkel Drehzahl	stetig	el. Leistung
Schrittmotor	dito	diskret	Impulszahl
Elektromagnet	Ort, Weg	diskret	el. Leistung
Stellventil	Durchfluß	stetig	Elektromotor
Drossel	dito		
Sperrventil	Durchfluß	diskret	Elektromagnet
Schieber	Transportfluß	stetig	Elektromotor
Weiche	Transportrichtung	diskret	Elektromotor
Pumpe	Durchfluß	stetig	Elektromotor
Verdichter	dito		
Heizung	Temperatur	stetig	el. Leistung
Elektroden	Stoffeigenschaften	stetig	el. Strom / Spannung
Relais,Schütz	el. Strom	diskret	el. Leistung
Transistor,	el. Strom	stetig	el. Spannung
Thyristor	el. Strom	diskret	el. Spannung
el. Widerstand	el. Strom	stetig	Widerstandswert
Potentiometer	el. Widerstand	stetig	Winkel, Weg
Stelltransformator	el. Spannung	stetig	Winkel, Weg
Lampe	Helligkeit	stetig	el. Leistung

6.5.2.2 Ausgewählte Verfahren

a) Motoren (für x, v, a)
- analog: $v = r \cdot n$ mit r = Radius, n = Drehzahl $\sim P$ = el. Leistung
- digital: (Schrittmotor): $v \sim f_p$ mit f_p= Impulsfolgefrequenz. Ähnlich wie beim Strichgitter müssen stets 2 (orthogonale) Impulsfolgen erzeugt werden, um eine Drehrichtung (links/rechts) festzulegen.

b) Ventile (für Durchfluß und Richtung)
- Sperrventile (diskret, auf/zu)
- Drosselventile (kontinuierlich)
- Weiche (Richtung)

c) Heizung
- elektrische Widerstandsheizung:
  Temperaturerhöhung ΔT durch elektrische Leistung $P = U \cdot I$ während einer Zeitspanne Δt

  $P \cdot \Delta t = \Delta Q = c \cdot \Delta T$

6.5.3 Übertragungsglieder

a) Leitungen
- Freie Leitung, R, C, L
- Verdrillte Leitungen, R, C, L, $R = \sqrt{L/C}$
- Coaxialleitung, Wellenwiderstand R, Dämpfung $\delta(\omega)$
- Lichtwellenleiter, Dämpfung $\delta(\lambda)$, Dispersion $n(\lambda)$
- Funk und Ultraschall, Reflexionen

b) Koppelglieder
- Galvanische Trennung durch
  - Optokoppler
  - Transformatoren
- Frequenztrennung durch
  - Filter
  - Hoch-, Tief-, Bandpässe
  - Weichen

c) Umformer
- Verstärker, Abschwächer
- Diskriminatoren, Begrenzer
- Schmitt-Trigger (Hysterese)

Inhaltsverzeichnis