Skriptum

PRRU
Prozeßdatenverarbeitung

Inhaltsverzeichnis

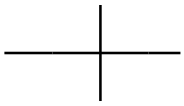
I Grundlagen für PRRU

1 PHYSIK

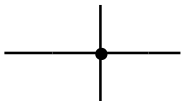
1.1 Einheiten und Symbole

Stromstärke (Strom):	I	Einheit:	1 Ampere	1 A
Spannung:	U	Einheit:	1 Volt	1 V
Widerstand:	R	Einheit:	1 Ohm	1 Ω
Frequenz:	f	Einheit:	1 Hertz	1 Hz = 1 s^{-1}
Kapazität:	C	Einheit:	1 Farad	1 F = 1 As V^{-1}
Leistung:	P	Einheit:	1 Watt	1 W=1VA

Leitung

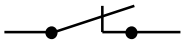


2 einander kreuzende Leitungen ohne elektrischen Kontakt

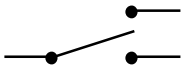


2 einander kreuzende Leitungen mit elektrischen Kontakt

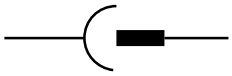
Schalter, Einschalter (Schließer, Arbeitskontakt)



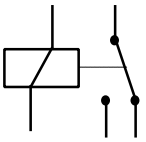
Ausschalter (Öffner, Ruhekontakt)



Umschalter

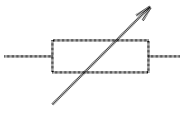


Buchse und Stecker

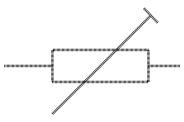


Relais

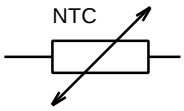
Widerstand



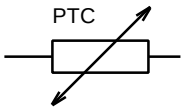
regelbarer Widerstand (Potentiometer)



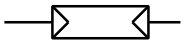
Trimmpotentiometer



temperaturabhängiger Widerstand (negativer Temperaturkoeffizient)



temperaturabhängiger Widerstand (positiver Temperaturkoeffizient)

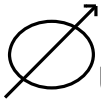


lichtabhängiger Widerstand (LDR)

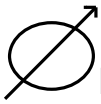
Glühbirne

Gleichstromquelle

Wechselstromquelle



Spannungsmeßgerät



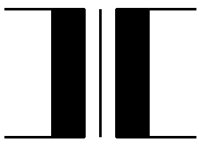
Strommeßgerät

Kondensator (Kapazität)

Drehkondensator (regelbare Kapazität)

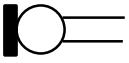
Trimmkondensator (regelbare Kapazität)

Elektrolytkondensator (Kapazität)

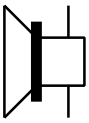


Transformator

Motor



Mikrofon

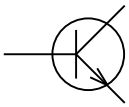


Lautsprecher

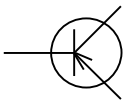
Diode

Leuchtdiode

Photodiode



npn-Transistor



pnp-Transistor

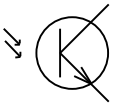


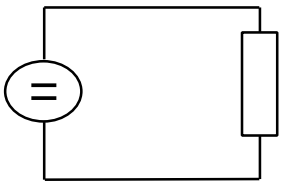
Photo-Transistor

1.2 Ohm´sches Gesetz

Die Stromstärke I in einem Leiter ist proportional zur angelegten Spannung U:

$$I = \frac{U}{R}$$

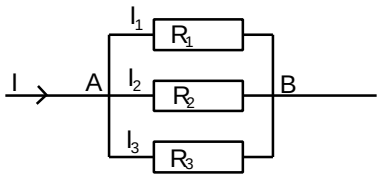
R ist material- und temperaturabhängig.
Bsp.:



U=5 V
R=150 Ω ⇒ R≈ 33 mA

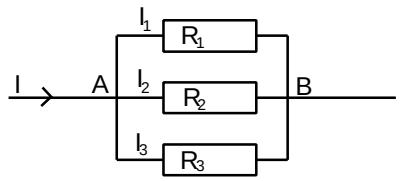
1.3 Kirchhoff´sche Regel

1. Kirchhoff´sche Regel: Die Gesamtstromstärke ist bei einer Verzweigung gleich der Summe der Einzelstromstärken.
⇒ "Im geschlossenen Stromkreis geht kein Strom verloren und es wird kein Strom hinzugefügt".



$I=I_1+I_2+I_3$ sowohl am Punkt A als auch am Punkt B

2. Kirchhoff´sche Regel: Zwischen 2 Verzweigungspunkten liegt an allen Zweigen die gleiche Spannung.



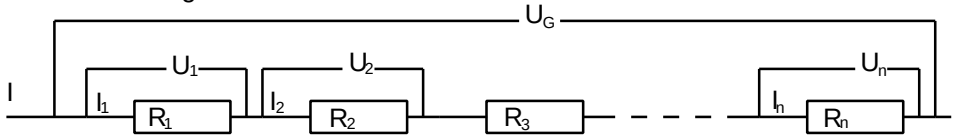
$U_1=I_1*R_1$

$U_2=I_2*R_2$ $\quad U_1=U_2=U_3=U_{AB}$

$U_3=I_3*R_3$

1.4 Verhältnisse bei mehreren Widerständen

Serienschaltung von Widerständen



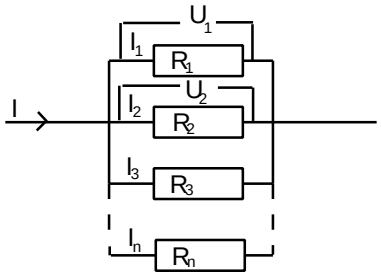
$R_G=R_1+R_2+R_3+ \ldots +R_n$

$I_1=I_2= \ldots =I_n =I$

$U_1=I*R_1 \qquad U_2=I*R_2 \qquad \ldots \qquad U_n=I*R_n$

$\Rightarrow \quad U_G=U_1+U_2+ \ldots +U_n$

Parallelschaltung von Widerständen



$$\frac{1}{R_G}=\frac{1}{R_1}+\frac{1}{R_2}+\ldots+\frac{1}{R_n}$$

$U_1=U_2= \ldots =U_n=U$

$$I_1=\frac{U}{R_1}, I_2=\frac{U}{R_2}, \ldots, I_n=\frac{U}{R_n}$$

Sonderfall $R_1=R_2= \ldots =R_n=R \quad \Rightarrow \quad R_G=R/n$

$$\Rightarrow I=\frac{U}{R+R_i}$$

1.5 Quellen- und Klemmenspannung

Bei Kurzschluß einer Spannungsquelle müßte nach dem Ohm´schen Gesetz der Strom beliebig groß werden, daß dem nicht so ist, kann man mittels eines Innenwiderstands der Spannungsquelle erklären.

R_i ... Innenwiderstand

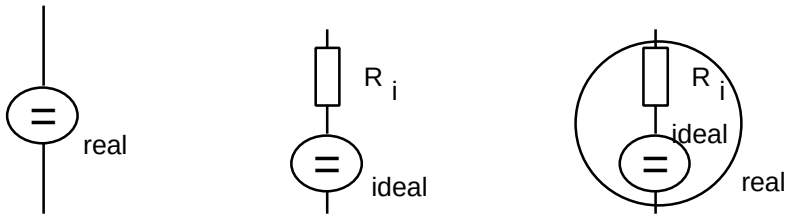
Der Kurzschlußstrom ist daher

$$I_K=\frac{U}{R_i}$$

Dieser Innenwiderstand verringert ständig die Quellenspannung U, sodaß an den Klemmen der Spannungsquelle nur mehr die Klemmenspannung U_{KL} anliegt.

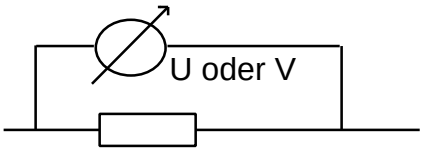
$$U_{KL}=U-R_i \cdot I$$

Reale Spannungsquelle =Ideale Spannungsquelle + Innenwiderstand



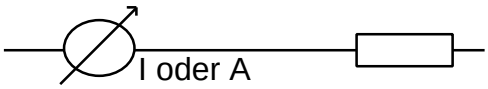
1.6 Spannungs- und Strommessung inkl. Meßfehler

Spannungsmeßgeräte dürfen die zu messende Spannung nicht wesentlich beeinflussen, daher müssen diese Geräte einen hohen Innenwiderstand (>20MΩ) besitzen und parallel zur zu messenden Spannung geschaltet werden.



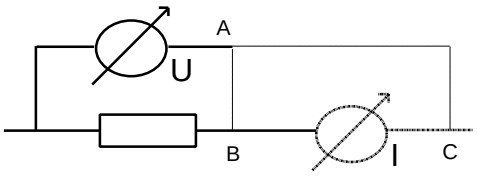
Probleme mit der Genauigkeit entstehen daher, wenn der Widerstand, an dem die Spannung gemessen werden soll, von der gleichen Größenordnung wie der Innenwiderstand des Meßgerätes ist.

Strommeßgeräte müssen analog zu oben einen möglichst kleinen Innenwiderstand (<1Ω) besitzen und in Serie geschaltet werden.



Analog zu oben wird die Messung bei kleinen Widerständen ungenau.

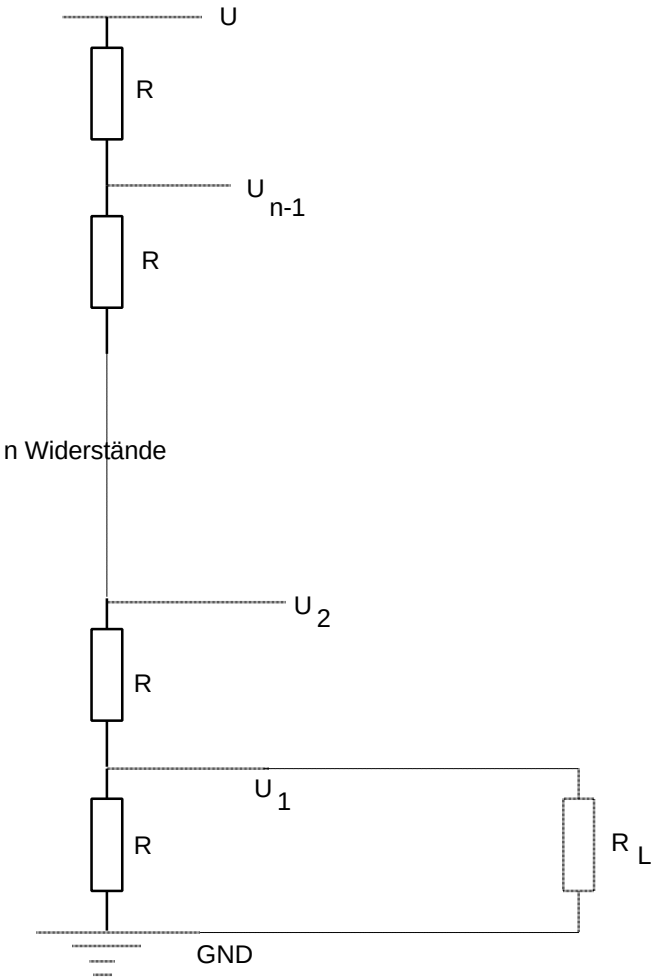
Problem:



Wird A mit B verbunden, mißt das Strommeßgerät die Summe aus dem Strom durch den Widerstand und dem Strom durch das Spannungsmeßgerät (interessant ist aber nur der Strom durch den Widerstand; das Spannungsmeßgerät zeigt die Spannung am Widerstand korrekt an)). Wird A mit C verbunden, mißt das Spannungsmeßgerät die Summe aus der Spannung am Widerstand und am Strommeßgerät (interessant ist aber nur die Spannung am Widerstand; das Strommeßgerät zeigt den Strom durch den Widerstand korrekt an).
Ferner ist noch zu berücksichtigen das die Leitungen ebenfalls einen Widerstand darstellen und das Widerstände temperaturabhängig sind.
Je nach notwendiger Genauigkeit müssen oben genannte Einflüsse auf die Messung berücksichtigt werden. Im Allgemeinen sind diese Einflüsse auf Messungen im Unterricht so gering, daß sie vernachlässigt werden können.

1.7 Spannungsteiler

Widerstandsnetzwerke können als Spannungsteiler eingesetzt werden



$$U_i = \frac{i \cdot U}{n}$$

R_L ... Lastwiderstand

Solche Spannungen können allerdings nicht stark belastet werden, da sonst U_i kleiner wird.

$$\Rightarrow \qquad R_L >> i \cdot R$$

1.8 Elektrische Leistung

Die elektrische Leistung ist im Rahmen dieses Gegenstandes besonders zum Abschätzen der Belastbarkeit von Bauteilen und Geräten von Bedeutung; für Gleichstromquellen beträgt die elektrische Leistung

$$P = U * I$$

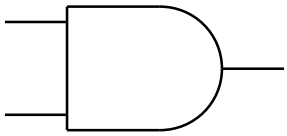
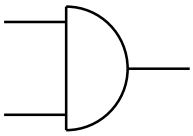
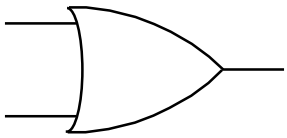
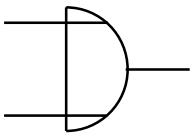
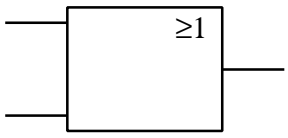
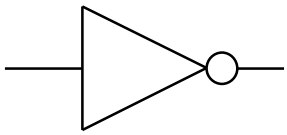
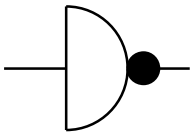
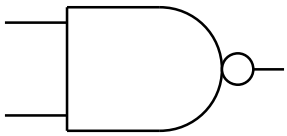
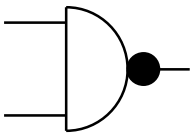
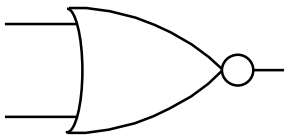
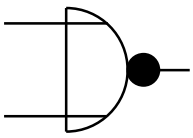
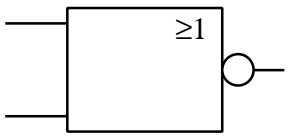
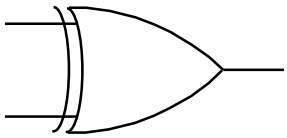
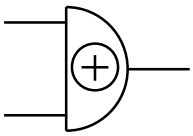
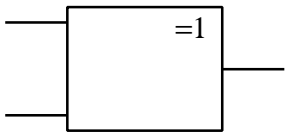
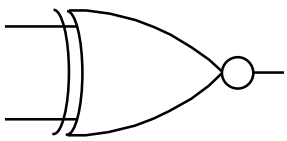
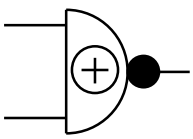
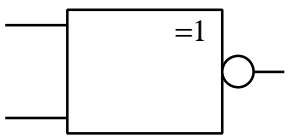
und durche Einsetzen des ohm´schen Gesetzes erhält man:

$$P = U * I = I^2 * R = \frac{U^2}{R}$$

2 GEDV

2.1 Logische Verknüpfungen und ihre Darstellung

Die Symbole für logischen Verknüpfungen müssen leider in 3 Normen bekannt sein, da sonst bestehende Literatur (Datenblätter) nicht verstanden wird:

	ANSI	DIN (alt)	DIN (neu)
AND			
OR			
NOT			
NAND			
NOR			
XOR			
XNOR			

Die Wahrheitstafeln für diese logischen Verknüpfungen lauten:

AND

A	B	E
0	0	0
0	1	0

1	0	0
1	1	1

OR

A	B	E
0	0	0
0	1	1
1	0	1
1	1	1

NOT

A	E
0	1
1	0

NAND

A	B	E
0	0	1
0	1	1
1	0	1
1	1	0

NOR

A	B	E
0	0	1
0	1	0
1	0	0
1	1	0

XOR

A	B	E
0	0	0
0	1	1
1	0	1
1	1	0

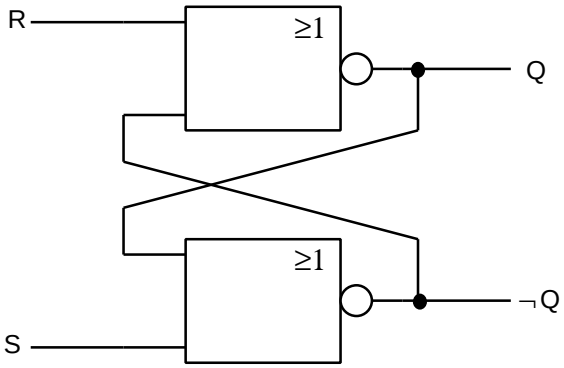
XNOR

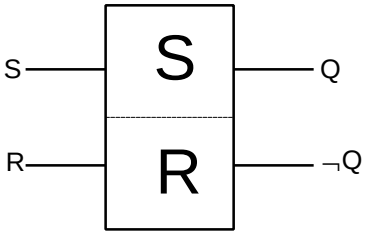
A	B	E
0	0	1
0	1	0
1	0	0
1	1	1

2.2 Schaltalgebra

Eine für PRRU sehr wichtige Anwendung sind alle Arten von Flip-Flops, das sie in der Lage sind Information zu speichern:

2.2.1 RS-Flip-Flop



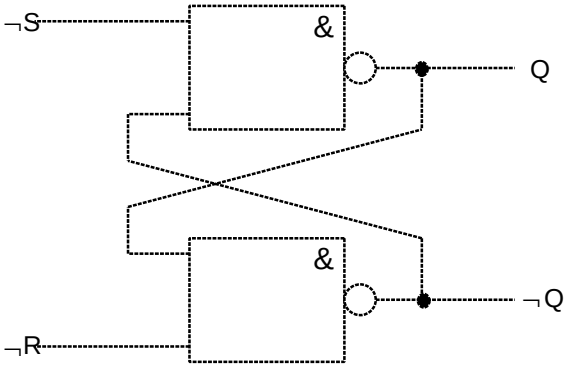


Wahrheitstabelle des RS-Flip Flops:

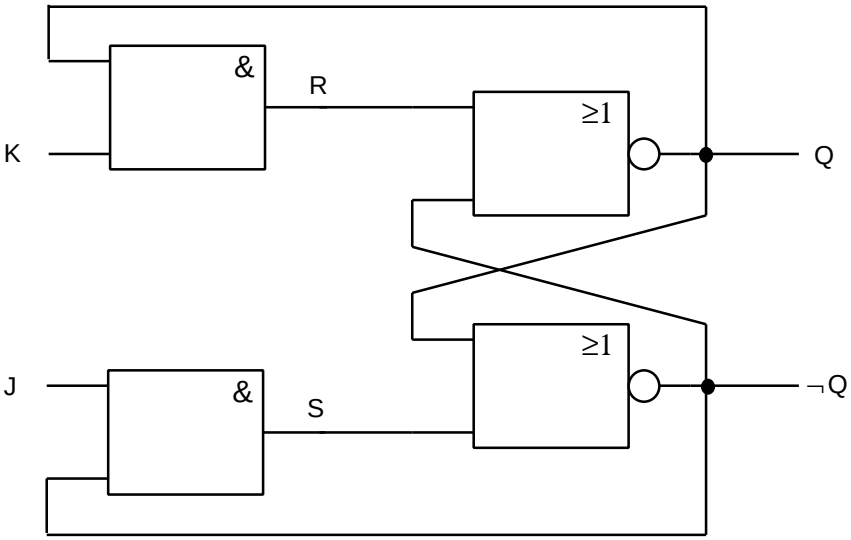
R_n	0	0	0	0	1	1	1	1
S_n	0	0	1	1	0	0	1	1
Q_n	0	1	0	1	0	1	0	1
Q_{n+1}	0	1	1	1	0	0	?	?

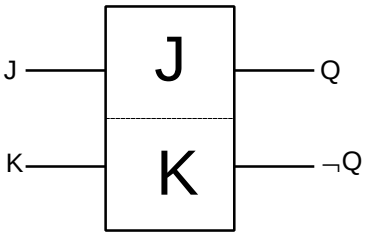
? ... Dieser Zustand ist nicht definiert

2.2.2 RS-Flip-Flop mit NAND-Gatter



2.2.3 JK-Flip-Flop

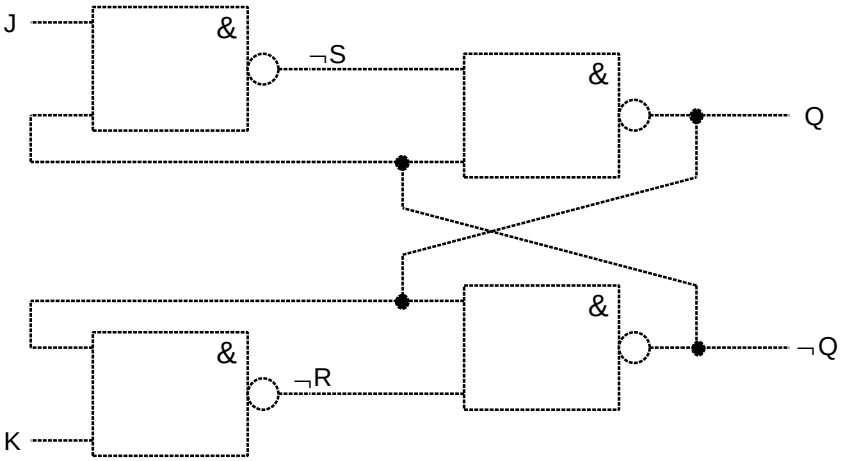




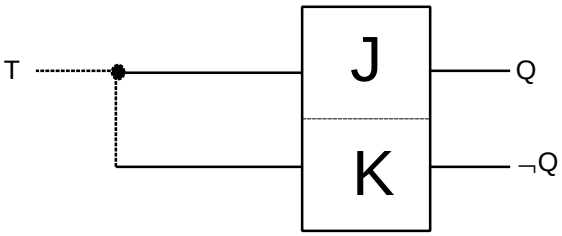
Wahrheitstabelle des JK-Flip Flops

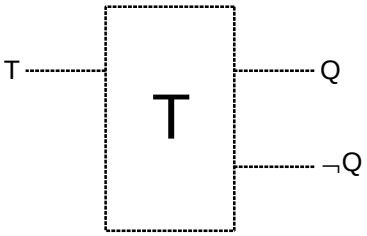
K_n	0	0	0	0	1	1	1	1
J_n	0	0	1	1	0	0	1	1
Q_n	0	1	0	1	0	1	0	1
Q_{n+1}	0	1	1	1	0	0	1	0

2.2.4 JK-Flip-Flop mit NAND-Gatter



2.2.5 T-Flip-Flop (Toggle)

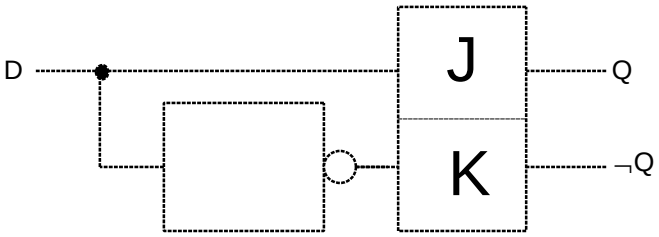


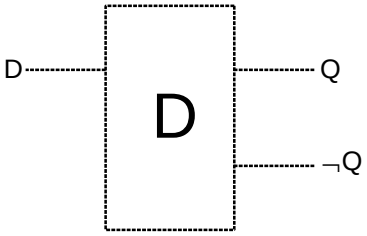


Wahrheitstabelle des T-Flip Flops

T_n	0	0	1	1
Q_n	0	1	0	1
Q_{n+1}	0	1	1	0

2.2.6 D-Flip-Flop (Delay, Data)





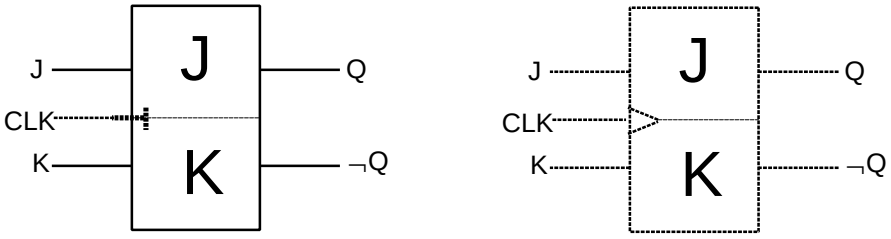
Wahrheitstabelle des D-Flip Flops

D_n	1	1	0	0
Q_n	0	1	0	1
Q_{n+1}	1	1	0	0

2.2.7 Getaktete Flip-Flops

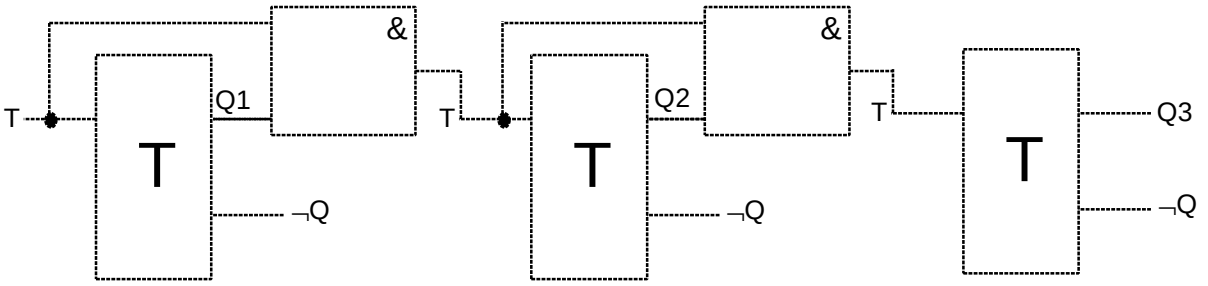
In fast allen Fällen sollen Flip Flops nur zu definierten Zeiten schalten, daher werden ihre Eingänge noch mit einem Taktsignal (oder Clocksignal) verknüpft, sodaß sich Änderungen am Eingang nur zu genau definierten Zeitpunkten auswirken können. Je nach Verknüpfungsart schaltet der High- oder Lowzustand des Taktsignals das Flip Flop. Um den Zeitpunkt noch genauer bestimmen zu können, wird im allgemeinen nur der Umschaltpunkt des Taktsignals (meist die steigende Flanke) als Enable für das Flip Flop verwendet.

In den Symbolen wird der Takteingang durch ein „liegendes T“ oder ein Dreieck angedeutet:



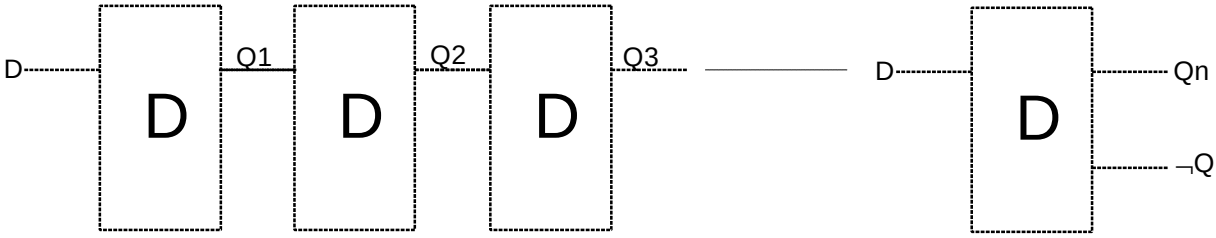
2.2.8 Zähler

Eine wichtige Anwendung von T-Flipflops sind Zähler:
z.B: Ein 3-Bit-Zähler



2.2.9 Schieberegister

Eine mögliche Anwendung von D-Flipflops sind Schieberegister:



2.2.10 Realisierung

Realisiert werden die logischen Funktionen mittels digitaler IC´s (IC=Integrated Curcuit)
z.B: der TTL-Serie 74xxx/54xxx (siehe Anhang)

Je nach Integrationsart von IC´s werden verschiedene Typisierungen vorgenommen:

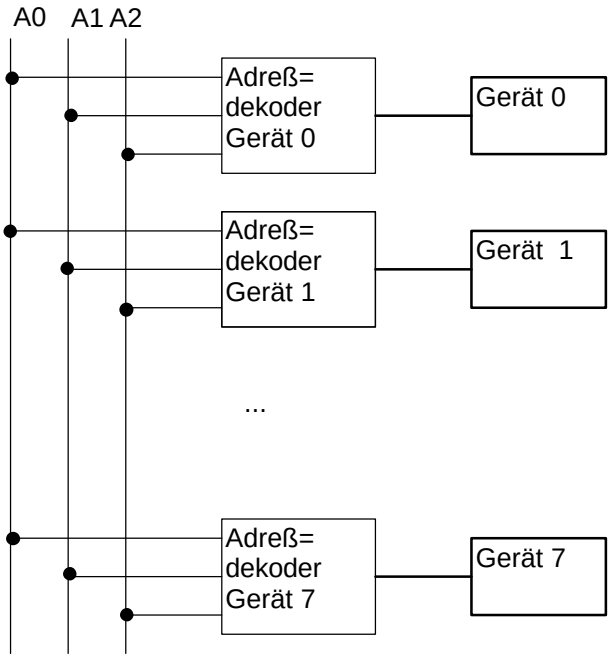
Abkürzung	Bedeutung	Anzahl der Gatter	Anzahl der Transistoren
SSI	Small Scale Integration	<30 Gatter	~100 Transistoren
MSI	Medium Scale Integration	30..100 Gatter	~1.000 Transistoren
LSI	Large Scale Integration	100..500 Gatter	~10.000 Transistoren
VLSI	Very Large Scale Integration	<100.000 Gatter	~100.000 Transistoren
V^2LSI	Very VLSI	<1.000.000 Gatter	~1.000.000 Transistoren
V^3LSI	Very V^2LSI	<10.000.000 Gatter	~10.000.000 Transistoren

2.3 Adreßdekoder

2.3.1 Einführung und Begriffe

Wenn innerhalb einer Schaltung (Computer, ...) viele Komponenten angesprochen werden müssen, wird dies nicht mehr mit der Ansteuerung jeder einzelnen Komponente durchgeführt, da sonst die Zahl der notwendigen Leitungen zu groß werden würde. In diesem Fall werden die einzelnen Komponenten mit einer Adresse versehen und alle Komponenten an gemeinsame Adreßleitungen (oft Adreßbus) angeschlossen. Jede Komponente soll selbstverständlich auch jetzt nur reagieren, wenn sie benötigt wird. Dazu wird die Adresse der benötigten Komponenten an die Adreßleitungen gelegt und über andere Leitungen (Datenleitungen, Datenbus, ...) die notwendigen Informationen ausgetauscht. Damit jede Komponente nur bei ihrer Adresse aktiv wird, ist eine passende Auswahlschaltung notwendig, diese wird **Adreßdekoder** genannt.

z.B.: Acht Einheiten, von denen eine ausgewählt werden soll. Dazu sind mindestens drei Adreßleitungen notwendig (2^3=8). Die möglichen Adressen lauten im Binärsystem 000, 001, 010, 011, 100, 101, 110, wobei bei diesem Beispiel alle vergeben sind.



Wie jetzt ein Adreßdekoder realiert wird, hängt von vielen Faktoren ab, z.B.: ob am Ausgang des Dekoders eine 0 oder eine 1 benötigt wird.
Sollte am Ausgang des Dekoders (das ist der Steuereingang des Gerätes) eine 0 benötigt werden, ist der Aufbau der Dekoder in unserem Beispiel für das erste bzw. letzte Gerät eine einfache logische Verknüpfung: OR bzw. NAND. Für die anderen Geräte ist der Aufwand etwas größer, da vor der Verknüpfung noch eine oder zwei Leitungen invertiert werden müssen. Generell werden für einfache Adreßdekoder nur Verknüpfungen aus folgender Tabelle verwendet:

	Ausgang=0	Ausgang=1
Eingänge überwiegend 0	OR	NOR
Eingänge überwiegend 1	NAND	AND

Bei einem Adreßdekoder mit OR oder NOR müssen alle Eingänge auf Null gebracht werden, d.h. die Eingänge die bei der gültigen Adresse 1 sind, werden invertiert zum Dekoder geführt; Eingänge, die bei der gültigen Adresse 0 sind, werden direkt angeschlossen.

Adreßdekoder werden oft nicht nur für eine Adresse, sondern für einen ganzen Block von Adressen benötigt. Ein Block besteht im allgemeinen aus direkt aufeinander folgenden Adressen; die Anzahl ist meist eine Potenz von 2 (1, 2, 4, 8, 16, ...) und der Beginn liegt an einer ganzen Blockgrenze (Die erste Adresse ist ganzahlig durch die Blockgröße teilbar). Jeder Block ist dann eindeutig durch eine Basisadresse und seine Größe definiert und wird auch nur so angegeben. Adreßdekoder für Blöcke

unterscheiden sich in ihrem Aufbau nicht von Adreßdekodern für Einzeladressen, allerdings dekodieren sie nur die höchstwertigsten Adreßleitungen, die niederwertigen (2^{Anzahl}=Blockgröße) werden an das Gerät weitergeleitet .

Folgende Begriffe werden im Zusammenhang mit Adreßdekodern zur Beschreibung verwendet :

Adreßraum	Größe (=Anzahl der zur Verfügung stehenden Adressen) z.B.: 64KB
Adreßbereich	Ein genaue Angabe von ... bis ... z.B.: 0_{Hex}..$FFFF_{Hex}$
Adreßleitungen	Anzahl der Leitungen (2^{Anzahl}=Adreßraum)

Generell unterscheidet man zwischen internem und externem Adreßraum (-bereich, ...), wobei intern auf das Gerät bezogen ist und extern das Gesamtsystem bedeutet (z.B.: hat eine Netzwerkkarte einen internen Adreßraum von 32 Byte, der in einem externen Adreßraum von 1024 Byte des PC´s eingeblendet wird)

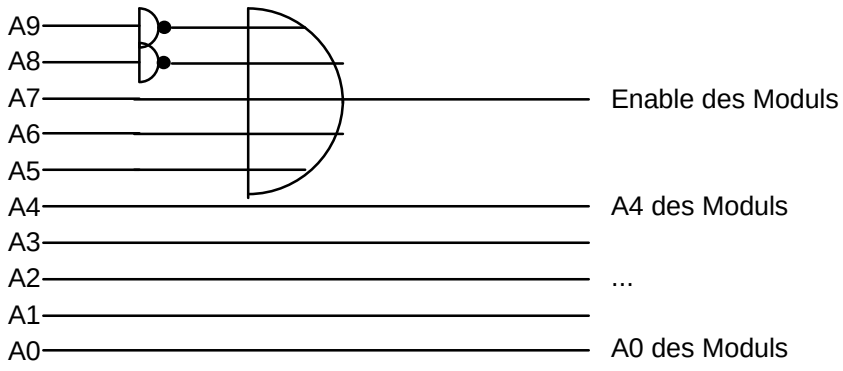
Bei vielen Mikroprozessoren werden Arbeits- und Ein-/Ausgabespeicherbereiche unterschieden, wofür der Mikroprozessor Steuerleitungen zur Verfügung stellt, die bei bei Bedarf in die Dekodierung einbezogen werden können oder müssen.

2.3.2 Mögliche Arten (nach der Funktion unterschieden)

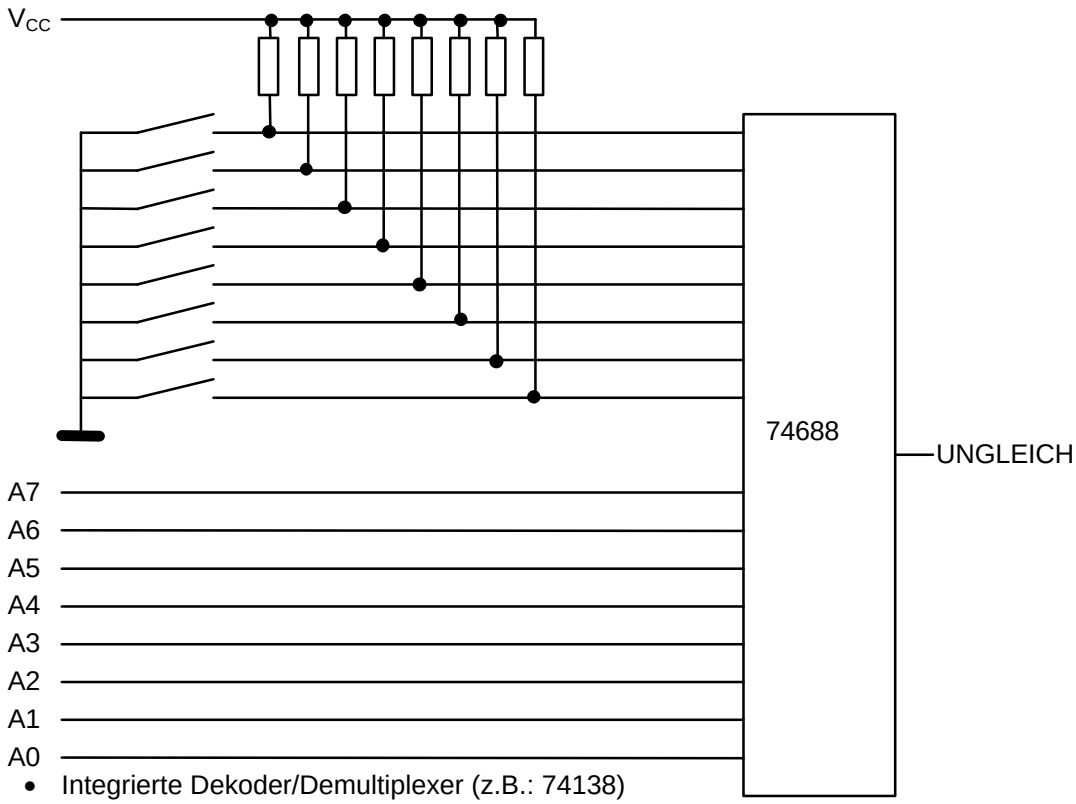
- Eine Einzeladresse aus einem Adreßraum; der Dekoder besitzt nur einen Steuerausgang
- Der gesamte Adreßraum wird von einem Dekoder in mehrere (2,4,8,...) gleich große Bereiche geteilt, ein solcher Dekoder besitzt dann für jeden Bereich einen Steuerausgang.
- Ein Block (definiert durch Basisadresse und Größe); der Dekoder besitzt einen Steuerausgang, zusätzlich werden aber die niederwertigsten Adreßleitungen dem Gerät zugeführt.
- Ein Dekoder für einen gesplitteten Block (Adressen nicht aufeinanderfolgend), dieser muß speziell für eine Anwendung betrachtet werden.

2.3.3 Mögliche Arten (nach ihrer Realisierung unterschieden)

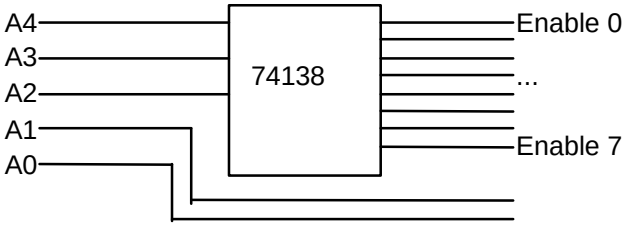
- Diskret mit Gattern aufgebaute Dekoder
  z.B.: Ein Modul mit 32 Adressen und Low-Enable liegt ab Adresse 300_{Hex} in einem Bereich, der mit 10 Adreßleitungen adressiert werden kann.



- Integrierte Vergleicher (z.B.: 74688)
  z.B.: Ein voreinstellbare Adresse soll mit der anliegenden Adresse verglichen werden. (Die voreinstellbare Adresse wird hier mit Schaltern - z.B.: DIP-Switches - realisiert, dabei bedeutet ein offener Schalter 1(High) und ein geschlossener Schalter 0(Low). Die eingezeichneten Widerstände haben üblicherweise einen Wert zwischen 4,7 kΩ und 10 kΩ)



- Integrierte Dekoder/Demultiplexer (z.B.: 74138)
  z.B.: Ein Adreßraum von 32 Byte soll in acht Bereiche á 4 Byte geteilt werden



- Kaskadierte Adreßdekoder (Mehrstufige Dekoder, d.h. in einer ersten Stufe wird ein größerer Adreßbereich dekodiert, der in einer weiteren Stufe (oder mehreren weiteren Stufen) in kleinere Bereich geteilt wird.

- EPROM-Dekoder (Dabei werden an bestimmten Adressen Datenbits gesetzt oder nur diese nicht gesetzt und die Datenleitungen des EPROMs werden mit den entsprechenden Geräten verbunden. Bei Anliegen von bestimmten Adressen oder Adreßbereichen an das EPROM werden diese Geräte aktiviert.

II PRRU-Theorie

1 SCHNITTSTELLEN

1.1 Parallele Schnittstelle (Centronics)

1.1.1 Allgemeines

Bei einer parallelen Schnittstelle werden mehrere (meist 8) direkt einstellbare Bits vom Rechner zur Verfügung gestellt. Jedes Bit kann einzeln den logischen Wert 0 oder 1 annehmen. Üblicherweise werden diese Bits mittels TTL-Pegel (0=0V; 1=5V) oder negativer TTL-Pegel (0=5V; 1=0V) übertragen. Da hier quasi binäre Zustände elektronisch zur Verfügung stehen, werden parallele Schnittstellen in der Prozeßsteuerung auch zur Übermittlung digitaler Informationen verwendet. So können vom Rechner durch Schreiben auf parallele Schnittstellen einfach Schrittmotore, Relais und ähnliches gesteuert werden bzw. durch Lesen von parallelen Schnittstellen der Zustand von Schaltern, Lichtschranken und dgl. abgefragt werden. Die bekannteste Parallelschnittstelle ist die Centronicsschnittstelle, die vor allem zum Anschluß von Druckern Verwendung findet.

1.1.2 Zur Verwendung der parallen Schnittstelle eines PC´s

a.) reservierte Bereiche im I/O-Adreßraum:

Hexadezimal:	3BC-3BF	378-37F	278-27F
Dezimal:	956-959	888-895	632-639
Binär:	1110111100-1110111111	1101111000-1101111111	1001111000-1001111111

b.) Basisadressen der PORT´S in der BIOS-Tabelle

0040h:0008h für LPT1: 0040h:000Ah für LPT2: 0040h:000Ch für LPT3:

c.) Register einer parallen Schnittstelle:

Datenregister (Basisadresse): WRITE ONLY!	D7	D6	D5	D4	D3	D2	D1	D0
Pin an DB25 (PC)	9	8	7	6	5	4	3	2
Pin an Centronics	9	8	7	6	5	4	3	2

Statusregister (Basisadresse+1): READ ONLY!	S7	S6	S5	S4	S3	S2	S1	S0
Pin an DB25 (PC)	11	10	12	13	15			
Pin an Centronics	11	10	12	13	32			

Busy	S7	-BUSY	(0=Drucker beschäftigt) invertiert?
Acknowledge	S6	-ACK	(0=Drucker ist bereit)
Paper error	S5	PE	(1=Papier fehlt)
Selected	S4	SLCT	(1=Drucker ist online)
Error	S3	-ERROR	(0=Fehler)
Nicht belegt	S2,S1,S0		

Controllregister (Basisadresse+2): READ/WRITE!	C7	C6	C5	C4	C3	C2	C1	C0
Pin an DB25(PC)					17	16	14	1
Pin an Centronics					36	31	14	1

Nicht belegt	C7,C6,C5		
IRQ Enable	C4	IE	(1=Interrupt, wenn -ACK auf 0)

Select Input	C3	SLCT IN	(1=Drucker online schalten)
Initialisieren	C2	-INIT	(0=Druckerreset)
Automatic Linefeed	C1	AUTOFEED	(1=LF nach CR)
Strobe	C0	-STROBE	(0=Daten liegen an)

d.) DB25-Stecker:

1	-STROBE	14	AUTOFEED
2	D0	15	ERROR
3	D1	16	INIT
4	D2	17	SLCT IN
5	D3	18	GND
6	D4	19	GND
7	D5	20	GND
8	D6	21	GND
9	D7	22	GND
10	-ACK	23	GND
11	-BUSY	24	GND
12	PE	25	GND
13	SLCT		

e.) Centronics-Stecker:

1	-STROBE	19	GND
2	D0	20	GND
3	D1	21	GND
4	D2	22	GND
5	D3	23	GND
6	D4	24	GND
7	D5	25	GND
8	D6	26	GND
9	D7	27	GND
10	-ACK	28	GND
11	-BUSY	29	GND
12	PE	30	GND
13	SLCT	31	-INIT
14	AUTOFEED	32	-ERROR
15	NC	33	GND
16	NC	34	NC
17	Shield	35	NC
18	NC	36	SLCT IN

f.) Kommunikationsablauf:

PC: Daten anlegen (D0-D7)
PC: -STROBE setzen und wieder wegnehmen (soll ca. 1Microsekunde auf 0 sein)
PRN: -BUSY setzen (auf 1 da es vom Rechner beim Empfang invertiert wird)
PRN: Zeichen verarbeiten
PRN: -ACK setzen (auf 0)
PRN: -BUSY zurücksetzen
PRN -ACK zurücksetzen

g.) Signalrichtung:

PC >-> PRN: -STROBE, D0..D7, AUTOFEED, -INIT, SLCT IN
PC <-< PRN: -ACK, -BUSY, PE, SLCT, -ERROR
PC <-> PRN: GND

1.2 Serielle Schnittstelle (RS-232C, V.24)

1.2.1 Allgemeines

Die serielle Schnittstelle zeichnet sich durch nur eine Leitung für die Informationsübertragung pro Richtung aus. Bei ihr werden die Informationen in Blöcken Bit für Bit hintereinander (seriell) übertragen. Anfang und Ende dieser Blöcke ist üblicherweise speziell gekennzeichnet. Eine relativ einfache serielle Schnittstelle ist die RS232C-Schnittstelle (nach EIA-Norm) bzw. die sehr ähnliche sogenannte V.24-Schnittstelle (nach CCITT-Norm) - sogenannt weil die V.24-Norm nur die Leitungen normiert, die elektrischen Pegel auf diesen Leitungen sind in V.28 festgelegt - auf die wir uns im folgenden beschränken wollen.
Bei der Übertragung wird zuerst ein Startbit (logisch 0), danach der Datenteil (üblicherweise im ASCII-Code) mit dem LSB beginnend, ein eventuelles Paritätsbit und abschließend noch 1 bis 2 Stopbits (logisch 1) gesendet. Ferner gibt es noch mehrere Betriebsarten, von denen heute eindeutig die Betriebsart **fullduplex** (Übertragung gleichzeitig in beide Richtungen möglich) dominiert; daneben existieren aber noch die Betriebsarten **simplex** (Übertragung nur in eine Richtung möglich) und **halfduplex** (Übertragung zwar in beide Richtungen, aber nicht gleichzeitig $\rightarrow$ erhöhter Protokollaufwand).
Heute werden auf der seriellen Schnittstelle nur mehr 2 verschiedene Protokolle verwendet: das hardware-orientierte DTR(Data Terminal Ready)-Protokoll und das software-orientierte XON/XOFF-Protokoll (wegen der verwendeten ASCII-Codes auch Ctrl-S/Ctrl-Q-Protokoll genannt).
Ursprünglich ist diese Schnittstelle zwischen einem Modem (oder allgemeiner einer Datenübertragungseinrichtung (DÜE) bzw, engl. Data Circuit termination equipment (DCE)) und einem Terminal (oder allgemeiner einer Datenendeinrichtung (DEE) bzw, engl. Data Terminal Equipment (DTE)) definiert worden, daher war auch eindeutig wer auf welcher Leitung senden bzw. empfangen konnte; heute gilt diese Einschränkung nicht mehr, daher ist auch nicht immer eindeutig wer der beiden Kommunikationspartner die Rolle des DCE bzw, des DTE übernimmt wodurch Übertragungsprobleme entstehen, die u.a. auch durch ein sogenanntes **Nullmodemkabel** gelöst werden können (Nullmodem bedeutet, daß zwei DEE´s ohne dazwischenliegende Modems miteinander kommunizieren können. Auf Grund der technischen Gegebenheiten können auch zwei DÜE´s mittels Nullmodemkabel kommunizieren).
Die Übertragung auf einer seriellen Schnittstelle kann asynchron (üblicher) oder synchron (schneller) erfolgen, in beiden Fällen spielt dabei der Takt (die Bitzeit) eine sehr wichtige Rolle, da bei ungleicher Zeitbasis die Übertragung unmöglich ist. Die Taktrate wird in Bit pro Sekunde bzw. in Baud angegeben (1 Bd =1 Baud =1 Informationswechsel/s=N Bit/s).
Damit die Übertragung funktioniert müssen folgende Parameter auf beiden Seiten zusammenpassen:

- DCE/DTE-Rollenverteilung
- Baudrate (z.B.: 50, 75, 150, 300, 600, 1200, 2400, 4800, 9600, ...)
- Anzahl der Datenbits (5, 6, 7, 8; im PC-Bereich heute fast immer 8)
- Anzahl der Stopbits (1; 1,5; 2; im PC-Bereich heute fast immer 1)
- Betriebsart (im PC-Bereich heute fast immer fullduplex)
- Parität (Even, Odd, None; im PC-Bereich heute fast immer keine)
- Protokoll

1.2.2 Zur Verwendung der seriellen Schnittstelle eines PC´s

a.) reservierte Bereiche im I/O-Adreßraum:

Hexadezimal:	3F8-3FF	2F8-2FF	3E8-3EF	2E8-2EF
Dezimal:	1016-1023	760-767	1000-1007	744-751
Binär:	1111111000-1111111111	1011111000-1011111111	1111101000-1111101111	1011101000-1011101111

b.) Basisadressen der PORT´S in der BIOS-Tabelle

0040h:0000h für COM1:	0040h:0002h für COM2:
0040h:0004h für COM3:	0040h:0006h für COM4:

c.) Konfiguration und Status der seriellen Schnittstelle (INT14):

Konfiguration: C7 C6 C5 C4 C3 C2 C1 C0
(INT 14,
AH=0, AL=Konfiguration, DX=Nummer der Schnittstelle (COM1=0)
AH=Status, AL=Modemstatus)

Baudrate	C7,C6,C5	000	110
		001	150
		010	300
		011	600
		100	1200
		101	2400
		110	4800
		111	9600
Parität	C4,C3	00	Keine
		01	Ungerade
		11	Gerade
Anzahl der Stopbits	C2	0	1 Stopbit
		1	2 Stopbit (1,5 Stopbit)
Anzahl der Datenbits	C1,C0	10	7 Datenbits
		11	8 Datenbits

Status: S7 S6 S5 S4 S3 S2 S1 S0

Time Out	S7	1	Time-Out-Fehler
Shift Register	S6	1	Shift Register leer
Data Register	S5	1	Datenregister leer
Unterbrechung	S4	1	Leitung unterbrochen
Protokoll	S3	1	Protokollfehler
Parität	S2	1	Paritätsfehler
Überlauf	S1	1	Zeichen wurde überschrieben
Daten bereit	S0	1	Zeichen empfangen

Modemstatus: M7 M6 M5 M4 M3 M2 M1 M0

Verbindung	M7	1	Verbindung aufgebaut
RI	M6	1	Telefon läutet
Modem	M5	1	Modem eingeschaltet
Senden	M4	1	Zum Senden bereit
Verbindung	M3	1	M7 hat sich geändert
RI	M2	1	M6 hat sich geändert
Modem	M1	1	M5 hat sich geändert
Senden	M0	1	M4 hat sich geändert

d.) Abbildung eines DB-9- auf einen DB-25-Stecker (PC)

1 ... 8	4 ... 20	7 ... 4
2 ... 3	5 ... 7	8 ... 5
3 ... 2	6 ... 6	9 ... 22

1.2.3 DB25-Stecker

Pin	Kürzel			Beschreibung	DTE DCE	G
	CCITT	EIA	Abk.			
1	101	AA	GND	Protective Ground (Masse)	↔	E
2	103	BA	TD	Transmitted Data (Sendedaten)	→	D
3	104	BB	RD	Receives Data (Empfangsdaten)	←	D
4	105	CA	RTS	Request To Send (Sendeteil einschalten)	→	S
5	105	CB	CTS	Clear To Send (Sendebereitschaft)	←	S

6	107	CC	DSR	Data Set Ready (Betriebsbereitschaft)	←	S
7	102	AB	SG	Signal Ground (Betriebserde)	↔	E
8	109	CF	CD	Carrier Detect (Empfangssignalpegel)	←	S
9				Reserved		F
10				Reserved		F
11				Unassigned		F
12	122	SCF	2CD	Secondary Carrier Detect	←	Z
13	121	SCB	2CTS	Secondary Clear To Send	←	Z
14	118	SBA	2TD	Secondary Transmitted Data	→	Z
15	114	DB	TC	Transmit Clock (Sendeschrittakt)	←	T
16	119	SBB	2RD	Secondary Received Data	←	Z
17	115	DD	RC	Receive Clock (Empfanfsschrittakt)	←	T
18				Unassigned		F
19	120	SCA	2RTS	Secondary Request To Send	→	Z
20	108	CD	DTR	Data Terminal Ready (Terminal betriebsbereit)	→	S
21	110	CG		Signal quality detect (Empfangsgüte)	←	S
22	125	CE	RI	Ring Indicator (Ankommender Ruf)	←	S
23	11	CH		Data rate selector	→	S
24	113	DA		Transmitter signal element timing (Transmit clock)	→	T
25				Unassigned		F

Legende:

Pin	Pin eines DB-25-Steckers	
CCITT	Kurzbezeichnung nach CCITT V.24	
EIA	Kurzbezeichnung nach EIA RS-232C	
Abk.	gebräuchliche Abkürzung	
DTE/DCE	Richtung	
	↔	bidirektional
	←	DTE ← DCE
	→	DTE → DCE
G	Gruppe	
	E	Erde
	D	Daten
	S	Steuerdaten
	T	Takte
	Z	Zusatzkanal
	F	Frei

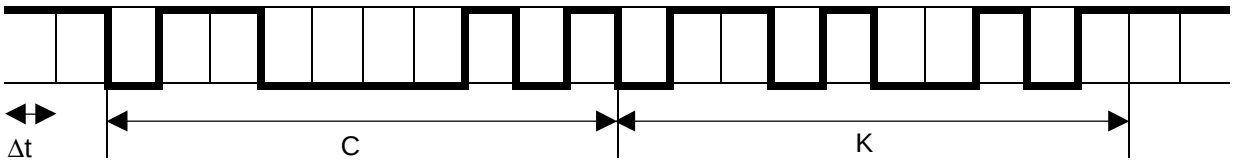
1.2.4 Verwendete Spannungen

logische Zustände	V.24/V.28	RS-232 C
„1“ auf Datenleitungen	-3 .. -12 V	-5 .. -25 V
„0“ auf Datenleitungen	+3 .. +12 V	+5 .. +25 V
„1“ auf Steuerleitungen	+3 .. +12 V	+5 .. +25 V
„0“ auf Steuerleitungen	-3 .. -12 V	-5 .. -25 V

Trotz der unterschiedlichen Spannungen in dieser Tabelle sind V24 und RS-232 C miteinander verträglich, da in der RS 232 C-Norm festgelegt ist, daß bei entsprechender Belastung (z.B.: durch die V.24-Schnittstelle eines anderen Rechners) die Spannung kleiner sein muß.

1.2.5 Übertragungsbeispiel

„CK“ mit 8 Datenbits, ohne Parität, 1 Stopbit und 9600Bits/s („C“..67..01000011, „K“..75..01001011)

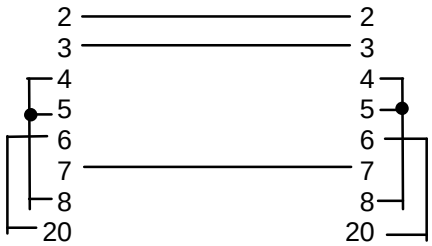


$\Delta t=1/9600s=0{,}000104$ s daher ist die Übertragungsdauer für die zwei Zeichen ca. 0,00208 s.

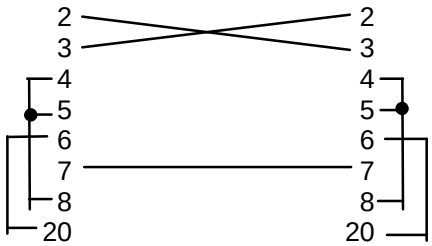
1.2.6 Kabel

Hier werden nur die Belegungen für die DB25-Verbindungen angegeben, die DB9 Verbindungen sind mit Hilfe der oben angeführten Abbildungen herzustellen.

- Modemkabel
  Alle 25 Pole sind 1:1 verbunden (d.h. Pin1 mit Pin1 usw.)
- Modemkabel - reduzierte Variante
  Nur die Pins 2,3,4,5,6,7,8 und 20 sind 1:1 verbunden
- Modemkabel - Sparvariante
  Vorteil: Nur drei Verbindungen zwischen den Geräten
  Nachteil: Softwarehandshake ist notwenig, Hardwarehandshake ist unmöglich



- Nullmodemkabel
  Alle 25 Pins sind verbunden, aber „Kommunkationspaare“ (2-3, 4-5, 6-20, 13-19 und 14-16) sind ausgekreuzt (d.h. Pin2 mit Pin3 der anderen Seite, Pin2 mit Pin3 der anderen Seite usw.) und die restlichen Pins sind 1:1 verbunden
- Nullmodemkabel - reduzierte Variante
  Nur die Pins 2,3,4,5,6,7,8 und 20 sind wie beim Nullmodemkabel verbunden
- Nullmodemkabel - Sparvariante
  Vorteil: Nur drei Verbindungen zwischen den Geräten
  Nachteil: Softwarehandshake ist notwenig, Hardwarehandshake ist unmöglich



1.2.7 Normen für andere serielle Schnittstellen

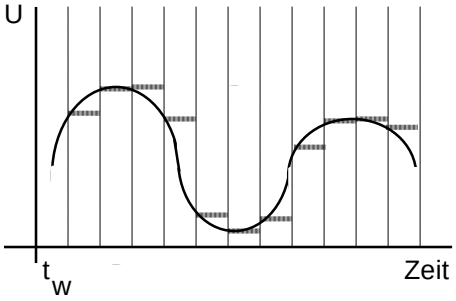
X.21 und V.11

1.3 ADC

1.3.1 Allgemeines

In der realen Welt (nicht im Computer) wird meist mit analogen Signalen gearbeitet, die für die Verarbeitung mit dem Computer erst aufbereitet werden müssen. Dabei entstehen eine Reihe von Problemen, derer sich der Anwender bewußt sein sollte, da es sonst zu unerwünschten Ergebnissen kommen kann.
Analoge Signale sind im allgemeinen dadurch gekennzeichnet, daß sie einen „beliebig“ großen Wertebereich zur Verfügung haben und auch - zumindest für die relevanten Größenordnungen - beliebig unterteilt werden können (zwischen zwei Werten gibt es immer noch Zwischenwerte). Daher wird auch oft von zeitkontinuierlichen Signalen gesprochen (es existieren keine Sprungstellen, wenn das Signal in Abhängigkeit von der Zeit betrachtet wird).
Digitale Signale haben einen festgesetzten Wertebereich, d.h. es gibt einen kleinsten und einen größten Wert, die nicht unter- bzw. überschritten werden können. Außerdem existiert ein kleinster

Abstand (1) zwischen zwei benachbarten Werten. Hier wird von zeitdiskontinuierlichen Signalen gesprochen, da keine beschreibende Kurve ohne Sprungstellen existiert, wenn man das Signal in Abhängigkeit von der Zeit betrachtet.
Wenn nun ein Computer (Mikroprozessor, ...) mit analogen Signalen arbeiten soll, müssen diese digitalisiert (Analog-Digital-Wandler, ADC) bzw. wieder in analoge Form gebracht werden (Digital-Analog-Wandler, DAC). Diesen Vorgang der Digitalisierung läßt sich am einfachsten mit dem Ablesen eines Analogmeßgerätes (z.B.: Quecksilberthermometer) vergleichen, da sich dabei die ablesende Person ebenfalls auf einen fixen Wertebereich beschränkt (z.B.: wegen der endlichen Genauigkeit der Skala, ...).



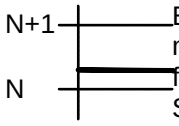
Je kleiner dabei das t_w ist, desto genauer kann das analoge Signal wiedergegeben werden.

In der Praxis verwendet man praktisch ausschließlich Wandler die als Analogsignal eine elektrische Spannung benötigen; sollen andere physikalische Größen gemessen werden, ist ein geeigneter Sensor notwendig.

1.3.2 Die wesentlichen Fehler bei der Digitalisierung sind:

a.) Der Digitalisierungsfehler (Quantisierungsfehler)

Da zum Runden eine weitere binäre Stelle benötigt würde, ist die letzte Stelle eines Wandlers immer ungenau (es kann nicht ausgesagt werden, ob ein Analogwert näher dem kleineren oder näher dem größeren Wert eines Paares ist)
z.B.:

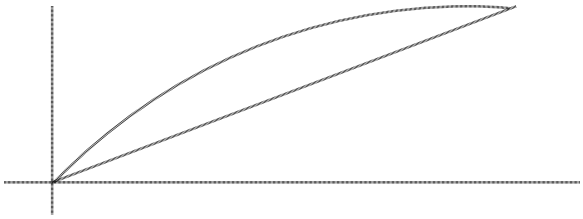


Es kann nicht festgestellt werden, ob U_x näher dem Wert N oder näher dem Wert N+1 ist, da dazu die Mitte zwischen N und N+1 festgestellt werden müßte und dies im Binärsystem einer weiteren Stelle entspricht.

Ein Wert N ist daher immer nur bis auf ±1 genau.

b.) Integrale Nichtlinearität

Weicht die Konversionskurve zwischen Analog- und Digitalwert von der idealen Geraden ab, so spricht man von integraler Nichtlinarität.
z.B.:



c.) Differentielle Nichtlinearität

Weichen die einzelnen Punkte der Konversionskurve nach verschiedenen Richtungen von der idealen Geraden ab, spricht man von differentieller Nichtlinearität, die insbesondere bei Verteilungsmessungen problematisch ist, da gleich große Intervalle im Analogbereich nicht mehr durch gleich große Intervalle im Digitalbereich repräsentiert werden und es im Extremfall sogar zu einer nicht monotonen Zuordnung (Umkehr der Ziffernfolge) kommen kann.

z.B.:

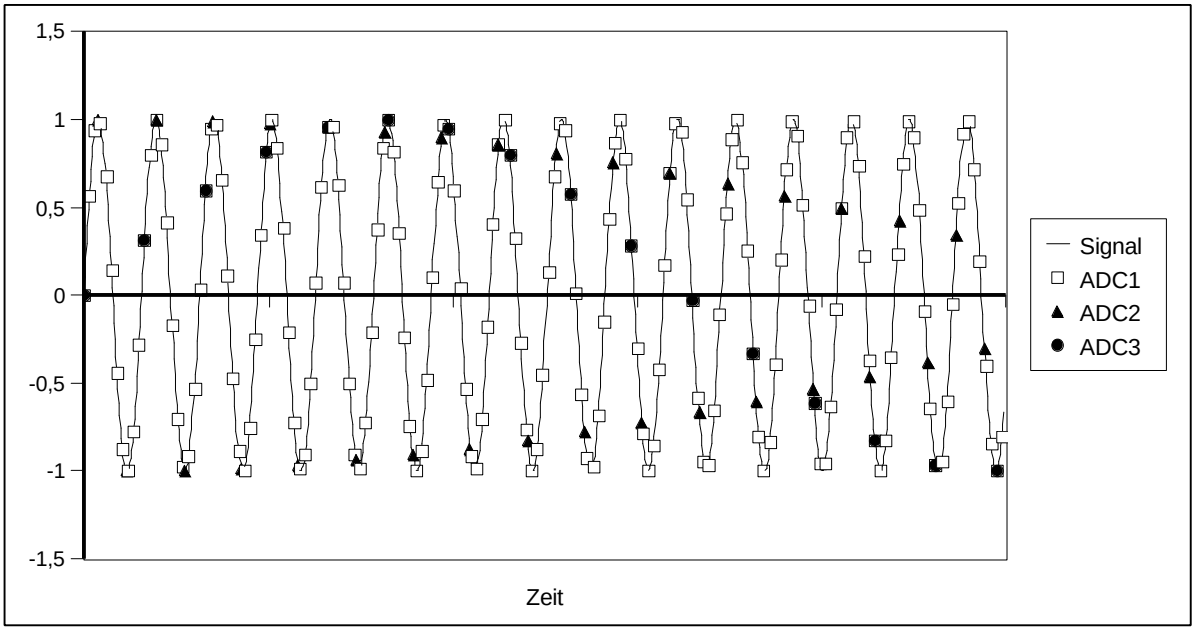
Das Intervall D1-D2 ist deutlich kleiner als das Intervall D2-D3, obwohl die zugehörigen Intervall A1-A2 und A2-A3 gleich groß sind. D4 ist vor D3 obwohl A3 vor A4 ist.

d.) Rauschen

Durch das Rauschen ist die Auflösung in etwa auf 10^{-5} bis 10^{-6} begrenzt.

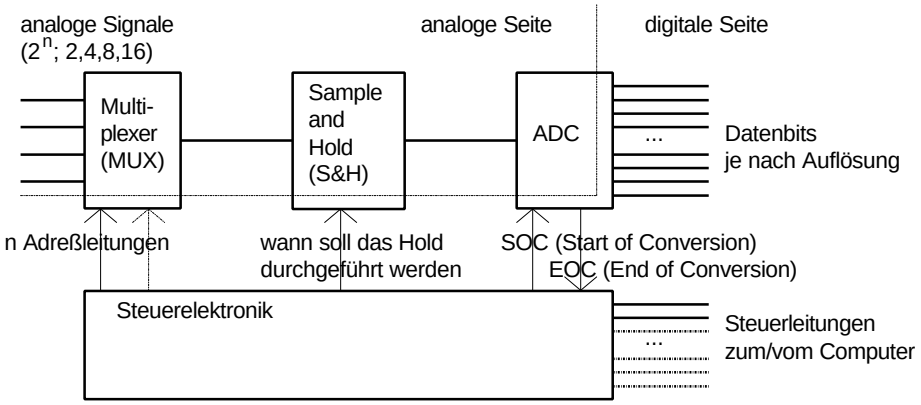
e.) Abtasttheorem

Das ist kein Fehler der Digitalisierung, führt aber bei Nichtbeachtung zu falschen Ergebnissen. Das Abtasttheorem besagt, daß bei der Messung von Signalen pro Periode mindestens 2 Meßwerte erfaßt werden müssen, um ein aussagefähiges Ergebnis zu bekommen



In obigem Bild wird ein Signal mit 3 ADCs mit verschiedenen Wandlungsraten gemessen. ADC1 mißt mit einer Samplingrate die ca. 9 mal so groß ist wie die Frequenz des Signals, daher werden pro Periode 9 Meßpunkte erfaßt und das Signal mit i.a. ausreichender Genauigkeit beschrieben. ADC2 mißt mit ca. der doppelten Frequenz des Signal und zeigt die oben beschriebene Grenze (Frequenz und Amplitude des Signals wird noch erfaßt, Details nicht mehr. Hinweis: Sollte genau mit der doppelten Frequenz gemessen werden und im Nulldurchgang des Signal begonnen werden, kann auch diese Aussage nicht mehr getroffen werden, da der ADC Punkte auf einer Gerade - der Nullinie des Signals - mißt.). ADC3 mißt mit einer Rate, die geringfügig höher ist, als die Frequenz des Signals, sodaß weniger als ein Meßpunkt pro Periode erfaßt wird. Das Ergebnis ist in diesem Fall wieder ein Sinussignal mit einer anderen Frequenz; das Meßergebnis entspricht nicht der Realität, obwohl alle Punkte korrekt gemessen wurden.

1.3.3 Blockschaltbild



1.3.4 Kenndaten

a.) Anzahl und Art der Eingänge

Aus Kostengründen und aus dem Zeitverhalten (Schnelle CPUs, langsame Prozesse) hat man ADC mit einem Eingangsmultiplexer versehen, um einen ADC für verschiedene analoge Signale verwenden zu können. Üblich sind ADCs mit 1(ohne Multiplexer), 2, 4, 8, 16 oder 32 Eingängen; diese können „**differential**" (jeder Analog-Eingang hat eigene Masse) oder „**single-ended**" (alle Analog-Eingänge haben eine gemeinsame Masse) sein

b.) Spannungsbereich

Der Spannungsbereich wird im allgemeinen mit einer Referenzspannung U_{Ref} und der Polaritätsangabe (unipolar oder bipolar) bezeichnet, da die Wandler entweder einen Bereich von 0 V bis U_{Ref} (=unipolar) oder von $-U_{Ref}$ über 0 V bis + U_{Ref} (=bipolar) haben. Seltener gibt es um 0 V unsymmetrische Wandler, die dann durch ihre kleinste und ihre größte Spannung, die sie wandeln können, beschrieben werden.
Die kleinste analoge Spannung wird durch den kleinsten digitalen Wert repräsentiert, die größte analoge Spannung durch den größten digitalen Wert (kein Zweierkomplement, sondern eine Offsetdarstellung).
Übliche Spannungsbereiche sind U_{Ref} =5V oder 10V sowohl uni- als auch bipolar.

c.) Auflösung

Die kleinste Differenz im Analogwert, die durch einen Unterschied im digitalen Wert dargestellbar ist, wird Auflösung genannt und durch die Anzahl der Bits des ADCs beschrieben. Ein N-Bit Wandler kann 2^N-Werte und 2^N-1 Intervalle repräsentieren, daher errechnet sich die kleinste Differenz aus dem Spannungsbereich und der Anzahl der Intervalle.
Übliche Auflösungen sind 8, 10, 12 oder 16 Bit.

d.) Wandlungsrate

Die Wandlungsrate wird entweder in als „Sampling rate" in Hertz(Hz; Samples per second) angegeben oder als Wandlungszeit in Sekunden (s). Einfache Wandler schaffen eine Wandlungsrate von wenigen 10 kHz (=100 µs); teure Wandler aus der Videotechnik erreichen hier ca. 500 MHz (=2 ns).

e.) Fehler

Alle oben angeführten Fehler sind meist getrennt im Datenblatt eines Wandlers angeführt, Gesamtfehler werden i.a. als eine Summe von einem Absolutfehler und einem prozentuellen

Fehler angegeben (z.B.: ±(1 + 0,5% vom Meßwert) bedeutet, daß das gemessene Ergebnis um 1+ ½ % des Meßwertes vom wahren Wert abweichen kann.

f.) Sonstiges (Umgebungsbedingungen, Stromverbrauch, ...

Jeder Wandler arbeitet nur in einem spezifierten Bereich (Datenblatt), außerhalb dessen größere Fehler auftreten können oder im schlimmsten Fall auch Defekte in der Hardware auftreten können. Bei „normalen“ Verhältnissen (Temperatur=Raumtemperatur, Luftfeuchtigkeit ist nicht zu hoch, Druck ist Luftdruck, ...) funktionieren üblich Wandler ohne Probleme. Der Stromverbrauch und die Belastbarkeit müssen schaltungstechnisch berücksichtigt werden.

g.) Beispiel zu Spannungsbereich, Auflösung und Fehler

Ein unipolare Wandler mit U_{Ref} =5V, 8 Bit Auflösung und einem Gesamtfehler ±(1 + 0,5% vom Meßwert) liefert 192 als Digitalwert, Wie groß ist die angelegte Spannung?

Analogwert	Digitalwert	Beschreibung
+5 V	255	U_{Max}
...	...	
U_x	192	Gesuchte Spannung
...	...	
0 V	0	U_{Min}

U_x=3,76V±0,04V d.h. die angelegte Spannung liegt im Bereich zwischen 3,72 und 3,80 V.

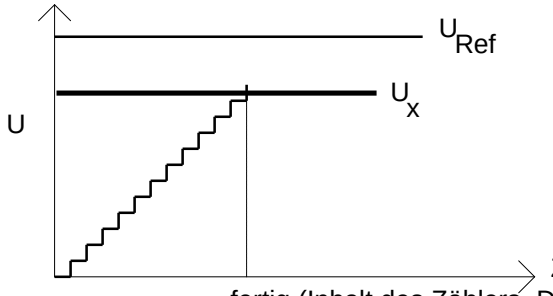
1.3.5 Formeln für die Umrechnung

	unipolar	bipolar
digital-analog	$\text{Analogwert} = \frac{\text{Digitalwert} * U_{Ref}}{2^N - 1}$	$\text{Analogwert} = \frac{\text{Digitalwert} * 2 * U_{Ref}}{2^N - 1} - U_{Ref}$
analog-digital	$\text{Digitalwert} = \frac{\text{Analogwert} * 2^N - 1}{U_{Ref}}$	$\text{Digitalwert} = \frac{(\text{Analogwert} + U_{Ref}) * 2^N - 1}{2 * U_{Ref}}$

1.3.6 Verfahren

a.) Zählverfahren

„Zutaten": 1 Zähler, 1 DAC, 1 Komparator



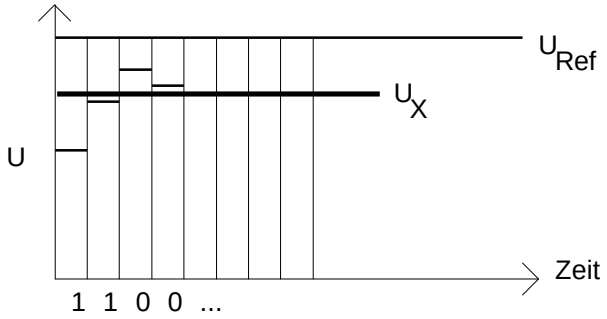
fertig (Inhalt des Zählers=Digitalwert)

Der Zähler wird mit 0 initialisiert, der Inhalt des Zähler wird mittels des DACs in einen analogen Wert umgewandelt und mit der angelegten Spannung verglichen. Wenn der Wert des DACs kleiner als die angelegt Spannung ist, wird um eins weiter gezählt und wieder gewandelt, bis die angelegte Spannung kleiner als die Spannung des DACs ist. Der Wert des Zählers zum Zeitpunkt des Abbruchs entspricht dem Digitalwert des analogen Signals.
Vorteile: einfach, billig

Nachteile: langsam, Wandlungszeit nicht konstant
verschieden lange Zeit, um den Wert zu messen ⇒ Zeitkorrelation geht verloren (im Schnitt werden 2^{N-1} Taktzyklen benötigt).

b.) Sukzessive Approximation

„Zutaten": 1 Steuerlogik, 1 DAC, 1 Komparator



Die sukzessive Approximation arbeitet nach dem Verfahren „Binäres Suchen“ (High-Low). Ist das angelegte Signal größer als der Vergleichswert, wird '1' geliefert, ist es kleiner, wird '0' geliefert.

Vorteil: Konstante und relativ kurze Wandlungszeit (n Bits ⇒ n Taktzyklen), akzeptable Kosten

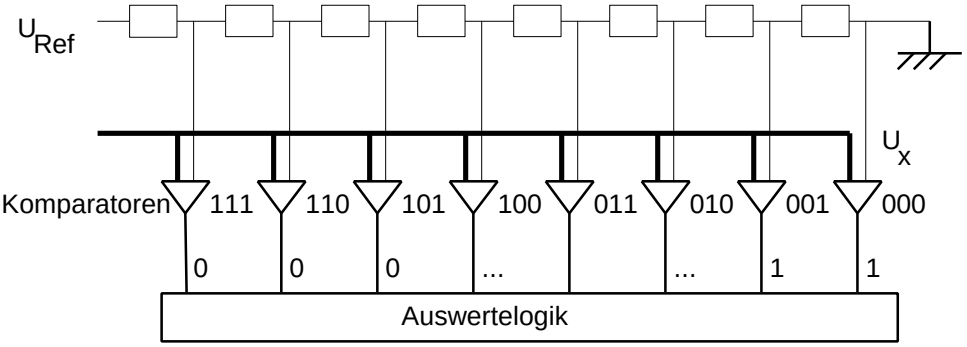
Nachteil: Steuerlogik

Da die Vorteile die Nachteile überwiegen ist die sukzessive Approximation das am häufigsten eingesetzte Wandlungsverfahren.

c.) Flashwandler (Parallelwandler)

„Zutaten": 2^N Widerstände (⇒ Widerstandsnetzwerk), 2^N Komparatoren, Auswertelogik

z.B.: 3-Bit-Wandler



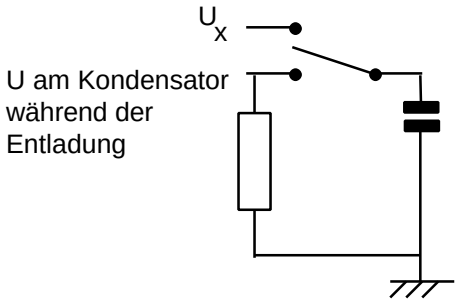
Die angelegte Spannung wird gleichzeitig mit allen möglichen Werten des ADCs verglichen und die Auswertelogik wandelt das wieder in den entsprechenden Wert zurück. Bei der Auswertungen muß der Übergang von 0 auf 1 gesucht werden und die Adresse des Komparators stellt den zugehörigen Digitalwert dar.

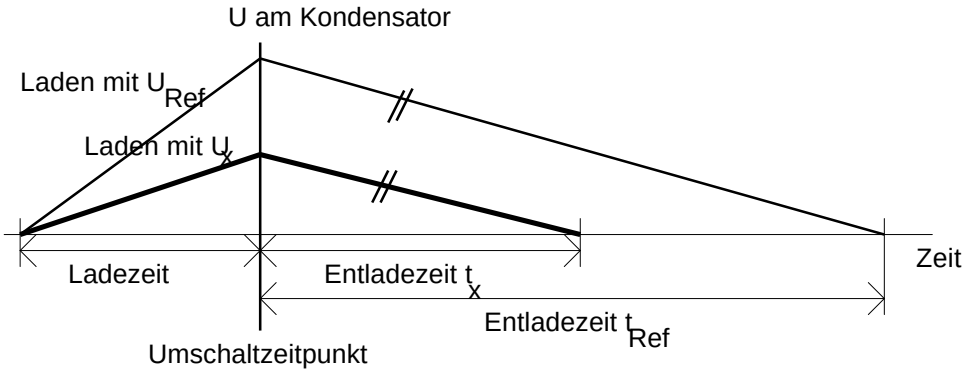
Vorteil: schnellstes Wandlersystem

Nachteil: sehr viele Einzelteile bei steigender Bitzahl notwendig (z.B. 16-Bit-Wandler: 2^{16} Widerstände und Komparatoren), die sehr genau aufeinander abgestimmt sein müssen ⇒ Kosten

d.) Dual-Slope-Wandler

„Zutaten": Stopuhr, elektronischer Schalter, Kondensator, Komparator





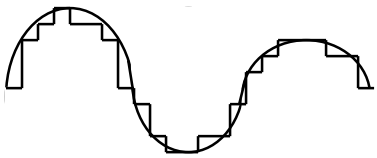
Der Kondensator wird eine bestimmte Zeit (Ladezeit) mit der angelegten Spannung aufgeladen. Danach wird auf Entladung umgeschalten und die Entladezeit gemessen (Zählen von Zeitimpulsen, daher nur eine Sonderform des Zählverfahrens). U_x kann dann wie folgt errechnet werden:

$$U_x = \frac{U_{Ref} * t_x}{t_{Ref}}$$

Eigenschaften: siehe Zählverfahren.

e.) Deltawandler

„Zutaten: Komparator



1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0

Es wird gemessen, ob das Signal größer oder kleiner als das vorangegangene Signal ist; die Größe der Spannungsunterschiede sind dabei ohne Bedeutung.

Vorteil: billig und schnell

Nachteil: konstante Eingangsspannungen können nicht gemessen werden, Absolutwerte gehen verloren

Wandler diesen Typs werden hauptsächlich in der Audiotechnik eingesetzt.

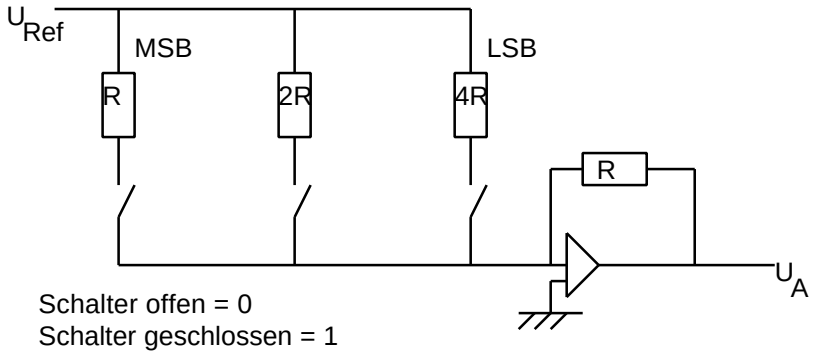
1.4 DAC

1.4.1 Allgemeines

Grundsätzliches zur Problematik der Konversion zwischen analogen und digitalen Signalen siehe ADC. DACs sind im Gegensatz zu ADCs immer diskret aufgebaute Parallelwandler und haben daher i.a. eine höhere Wandlungsrate als vergleichbare ADCs. Auch hier werden in der Praxis ausschließlich Wandler verwendet, die als Analogsignal eine elektrische Spannung liefern; sollen andere physikalische Größen ausgegeben werden, ist ein geeigneter Aktuator notwendig.

1.4.2 Blockschaltbild

am Beispiel 3-Bit-Wandler



1.4.3 Kenndaten

Anzahl und Art der Ausgänge

Im Gegensatz zu den ADCs werden DAC meist nur mit einem Ausgang betrieben, da die auszugebende Spannung immer anliegen sollte und daher nicht wie beim ADC zwischen mehreren Signalen geschalten werden kann.

Spannungsbereich

siehe ADC

Auflösung

siehe ADC

Wandlungsrate

siehe ADC

Fehler

siehe ADC

Einstellzeit

Die Einstellzeit (settling time) ist die Zeit, die der DAC benötigt, um ein ankommendes digitales Signal in ein stabiles analoges Signal umzuwandlen. Sie ist umso größer je größer der Sprung (die Änderung) zwischen zwei aufeinander folgenden Werten ist. Sie begrenzt auch die Wandlungsrate nach oben, da sie nicht beliebig kurz werden kann.

Sonstiges (Umgebungsbedingungen, Stromverbrauch, ...

siehe ADC

1.4.4 Formeln für die Umrechnung

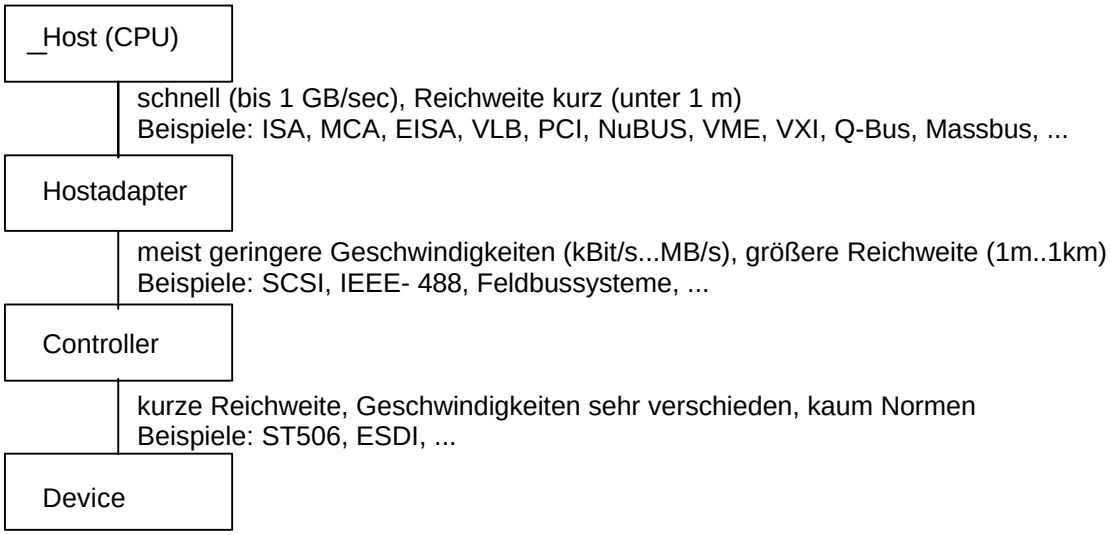
siehe ADC

2 BUSSYSTEME

2.1 Allgemeine Grundlagen

Bussysteme dienen der Kommunikation zwischen dem eigentlichen Rechner und verschiedenen Peripheriegeräten. Im Unterschied zu Schnittstellen können an Bussysteme mehr als zwei Komponenten angeschlossen werden, daher sind hier immer bidirektionale Leitungen, ein Adressierungsschema und Buszugriffsmechanismen vorhanden. Bussysteme bestehen aus 4 Leitungsgruppen: den Adreßleitungen (A0..An), den Datenleitungen (D0..Dn), den Steuerleitungen (z.B.: IOR, IOW, MEMR, MEMW, DRQ, DACK, IRQ, ...) und den Versorgungsleitungen (eine oder mehrere Versorgungsspannungen und Masseleitungen). Bei manchen Bussystemen sind nicht alle Gruppe mit eigenen Leitungen ausgestattet, um die Anzahl der Leitungen geringer zu halten; hier

werden über Mulitplexverfahren Leitungen mehrfach genutzt (z.B.: zuerst die Adreßinformation und im Anschluß die Daten).



Controller und Device sind meist direkt aneinander gekoppelt.

2.2 IEEE 488-Bus

2.2.1 Allgemeines

Anhand dieses typischen Meßgerätebussystems sollen auch einige grundlegende Eigenschaften von Bussystemen zwischen Hostadapter und Controller dargestellt werden.
Dieses Bussystem wurde ursprünglich von HP (HPIB) entwickelt und dann von IEEE und anderen Organisationen genormt. Die wichtigsten Normen sind:

IEEE 488-1978
IEC 625.1
ANSI MC1.1
DIN IEC 66.20

Die Unterschiede beschränken sich auf zwei verschiedene Ausführungen des Steckers:
der Centronics-ähnliche 24polige IEEE-488-Stecker und
der 25polige Sub-Min-D Stecker nach IEC-Norm.
Weiters ist dieses Bussystem unter einigen Markennamen bekannt:

HPIB
GPIB
PLUSBUS
ASCIIbus

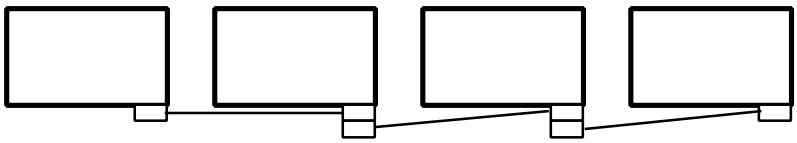
2.2.2 Eigenschaften und Kennwerte

An einem Bus dürfen maximal 15 Geräte angeschlossen werden (allerdings sind 31 Adressen vorhanden).
Die maximale Gesamtlänge des Kabels beträgt 20m, wobei zwischen zwei Geräten maximal 4m und im Durchschnitt pro Geräte nicht mehr als 2m verwendet werden dürfen.
z.B.:

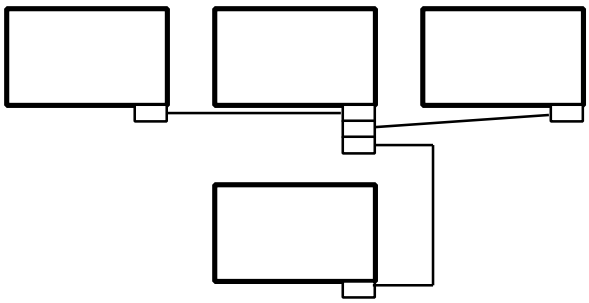
GerätA..2m..GerätB..4m..GerätC..2m..GerätD
GerätA..1m..GerätD..4m..GerätE..3m..GerätF

Jeder Stecker an einem Kabel ist gleichzeitig Buchse, damit sofort weiterverkabelt werden kann. Die mechanische Belastung durch die Verbindungen ist nicht vernachlässigbar, daher sind auch die 24poligen IEEE-488-Stecker mit massiven Schrauben (leider sowohl metrische als auch nach Zoll-Standard) versehen.

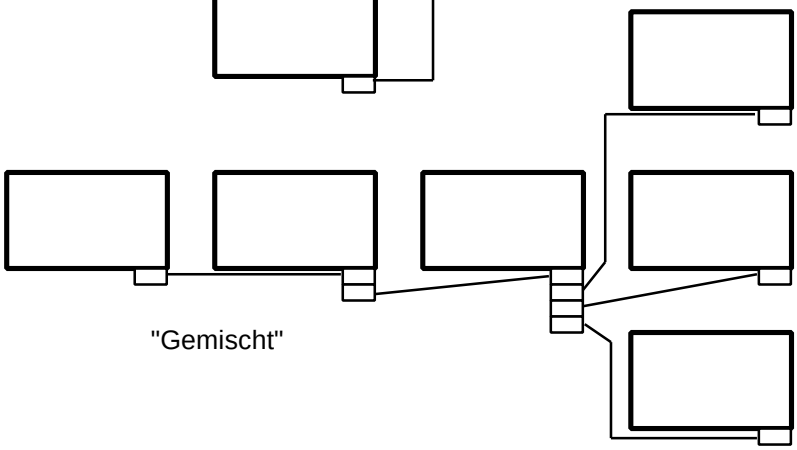
Die Anordnung der Geräte kann linienartig, sternförmig oder gemischt sein:



Linienartig



Sternförmig



"Gemischt"

Die Übertragungsleistung beträgt 250..500 KB/s (heute auch schon bis 1 MB/s), doch ist diese vom langsamsten System, das an der Kommunikation beteiligt ist, abhängig.

Auf den Leitungen wird neg. TTL-Logik verwendet, d.h.:

0(false)	>	2 V
1(true)	<	0,8 V

2.2.3 Gerätearten

Die Geräteeigenschaften werden mit einer Vielzahl von Buchstaben (meist neben dem Stecker) beschrieben, die drei grundlegenden Eigenschaften sind:

T	Talker	Gerät kann Daten senden
L	Listener	Gerät kann Daten empfangen
C	Controller	Gerät kann den Bus kontrollieren, hier werden 2 Arten unterschieden: System-Controller (Hardwareeigenschaft) Controller-In-Charge (momentaner Controller).

2.2.4 Leitungen

Die 24 Leitungen werden wie folgt unterschieden:

8 Steuerleitungen
- 5 Management - Leitungen (ATN, IFC, REN, SRQ, EOI)
- 3 Handshake - Leitungen (NRFD, DAV, NDAC)

8 Datenleitungen (DIO1..DIO8)

8(9) Masseleitungen (für jede Steuerleitung eine und eine für die Datenleitungen)

a.) Managementleitungen:

ATN(ATtentioN)

Entscheidet ob Daten (ATN=False) von 1 Talker zu ein od. mehreren Listenern oder ein Busbefehl (ATN=True; meist Adressierungsinformationen) von einem Controller übertragen werden. Diese Leitung wird vom Controller-In-Charge nach Bedarf gesetzt.

IFC (InterFace Clear)

Darf nur vom System-Controller gesetzt werden. Dabei gehen alle Geräte in den „Power-Up"-Zustand und der System-Controller wird Controller-In-Charge.
In der Praxis gibt es hier manchmal Schwierigkeiten, weil alle Geräte - nach der Norm - innerhalb von 100 µs reagieren müssen (entweder sind sie mit dem Reset fertig oder melden dem Controller-In-Charge ihre Nichtbereitschaft) und das nicht immer eingehalten wird.

REN (Remote ENable)

Darf nur vom System-Controller gesetzt werden. Alle Geräte, die diese Leitung unterstützen, schalten in den Remote-Zustand (d.h. sie lassen sich nur mehr über den Bus und nicht mehr über Tasten bedienen). Dadurch können Fehlbedienungen leichter vermieden werden.

SRQ (Service ReQuest)

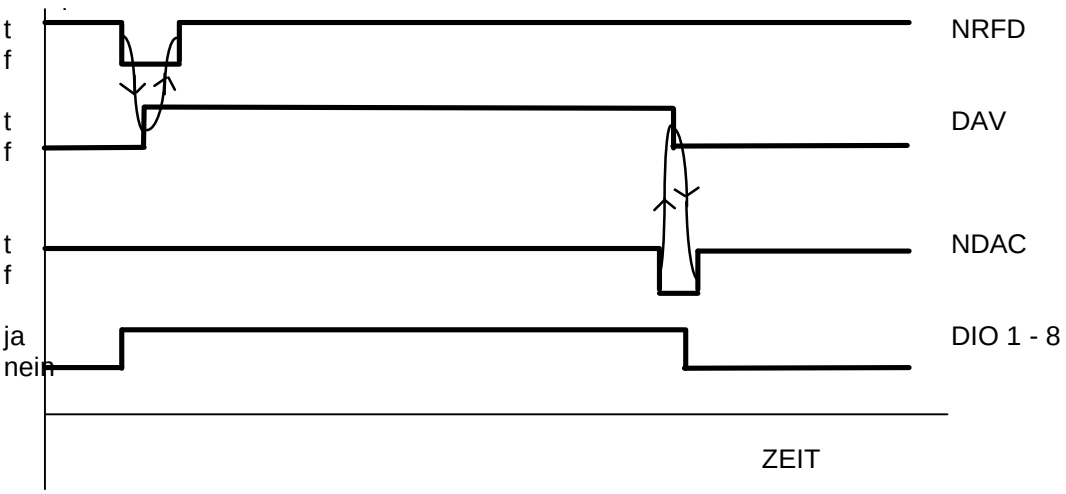
Diese Leitung kann von allen Stationen gesetzt werden und wird vom Controller-In-Charge überwacht. Damit kann jedes Gerät dem Controller melden, daß es Service benötigt (z.B.: Meßwert, übergehende Buffer, Alarmmeldung). Dabei entsteht das Problem, daß es maximal 14 (der Controller selbst wird kaum von sich selber Service erbitten) angeschlossene Systeme gibt und der Controller nun eruieren muß, welche System Service benötigt.
Dafür werden zwei Polling-Verfahren eingesetzt:

Seriell Poll	Jede Station wird einzeln gefragt (hintereinander)
Parallel Pol	Jedem Gerät wird eine Datenleitung zugeteilt (in einem einmaligen Initialisierungsprozeß) und dann können bis zu acht Geräte auf einmal gefragt werden (mit zwei Durchgängen sind alle Geräte abgefragt).

EOI (End Of Identify)

Die zwei Funktionen werden mittels der ATN-Leitung unterschieden:
Während einer Datenübertragung (ATN=false) wird dies vom aktiven Talker gesetzt, um mitzuteilen, daß die Übertragung abgeschlossen ist.
Wenn die ATN true ist, wird das vom Controller zum Auslösen eines Parallel Polls benutzt.

b.) Handshakeleitungen



NRFD	Not Ready For Data	t	true
DAV	DAta Valid	f	false
NDAC	Not Data Accepted	ja	Daten vorhanden
DIO 1 - 8	Datenleitung von 1 - 8	nein	Daten nicht vorhanden

NRFD, NDAC sind für den/die Listener, DAV und die Datenleitungen für den Talker.

Bei mehreren Listenern wird NRFD und NDAC erst bei letzten auf False gesetzt, damit alle die Daten korrekt lesen können. Dieses Dreileitungshandshake hat den Vorteil, daß jeder an der Kommunikation Beteiligte zu jedem Zeitpunkt die Situation der anderen Teilnehmer kennt. Der Talker(Controller) wartet bis alle seine Zuhörer(Geräte) bereit sind Daten zu übertragen (NRFD wird false), legt seine Daten an und setzt sie gültig (DAV=true). Darauf antworten die Zuhörer sofort mit der Zurücknahme der Bereitschaft (NRFD wird true) und beginnen mit dem Einlesen der Daten; wenn die Zuhörer die Daten gelesen haben, wird das Akzeptieren der Daten gemeldet (NDAC wird false). Der Sender reagiert darauf mit der Rücknahme der Gültigkeitsanzeige (DAV=false) und der Freigabe der Datenleitungen; die Zuhörer bestätigen dies mit der Zurücknahme der Übernahmebestätigung (NDAC wird true).

2.2.5 Bussteuercommandos

In der Norm sind 126 definiert, die in mehrere Gruppen eingeteilt sind. Die wichtigsten sind:

a.) Universal Command Group (alle Geräte sind betroffen)

LLO	Local Lock Out
DC	Device Clear
PPU	Parallel Poll Unconfigure
SPE	Serial Poll Enable
SPD	Serial Poll Disable

b.) Addressed Command Group (gilt nur für definierte Zuhörer)

SDC	Selected Device Clear
GTL	Go To Local
PPC	Parallel Poll Configure
TC	Take Control (um andere Controller zum Contoller in Charge zu machen)

c.) Listener Address Group (Gerät wird als Empfänger/Zuhörer definiert)

(M)LA00	(My)Listener Address
.	
.	
(M)LA30	
UNL	UN Listen

d.) Talker Address Group (Gerät wird als Sender/Talker definiert)

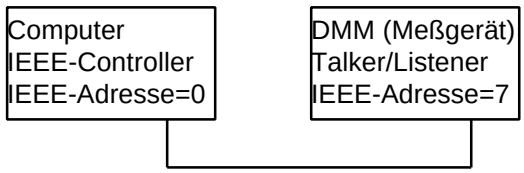
(M)TA00	(My)Talker Address
.	
.	
(M)TA30	
UNT	UN Talker

e.) Secondary Address Group

(M)SA00	(My) Secondary Address	PPE01 Parallel Poll Enable
		...
		PPE08
...		PPE11
		...
		PPE18
		PPD
(M)SA30		

2.2.6 Kommunikationsablauf

Im folgenden Beispiel soll ein Meßwert vom Digitalmultimeter eingelesen werden



ATN	Daten	Bemerkung
T	MLA07	DMM ist Empfänger/Listener
T	MTA00	Computer ist Sender/Talker
F	„F0R0X“	Geräteabhängige Anweisungen (z.B.: Funktion automatisch, Bereich automatisch und Messung ausführen)
T	UNT	Sendeerlaubnis entziehen
T	UNL	Empfänger deaktivieren
T	MLA00	Computer ist Empfänger
T	MTA07	DMM ist Sender
F	„NDCV0.1234E+01“	Meßwert in ASCII-Form, dieser ist ebenfalls geräteabhängig (z.B.: N=Normaler Modus, DCV=Gleichspannung, 1,234 V)
T	UNT	Sendeerlaubnis entziehen
T	UNL	Empfänger deaktivieren

Für einen Meßwert wurden in diesem Beispiel 27 Byte übertragen.

2.3 PC-Bus

2.4 ISA-Bus

Entstand 1985 bei der Markteinführung des IBM-AT, war aber bereits kurz nach seiner Einführung technisch überholt. Das Design der Adreßleitungen (A16...A23) und der umständlichen 8/16-Bit-Erkennung, welche die Kompatibilität zum alten 8-Bit-PC-Bus sicherstellen sollte, erwies sich bald als Flaschenhals und bremste bereits die ersten 286er-PCs aus. Der eher gemächliche Bustakt von 6, später 8MHz erlaubte bei 16-Bit-Busbreite eine theoretische Übertragungsrate von 8MByte/s, in der Praxis wurden aber nur Spitzenwerte von 5-6MByte/s erreicht.
Mit dem Aufkommen der 386er Prozessoren (32 Bit Daten und 32 Bit Adressen) wurde der Datenbus (16 Bit) zu eng, auch der Adreßraum zu klein (2^{24}=16 MB, 2^{32}=4GB) . ISA-Erweiterungskarten können nur auf die unteren 16MByte zugreifen. Diese Einschränkung war vor allem bei Plattencontrollern und Grafikkarten störend. Bei Plattenkontrollern ist diese Einschränkung auch heute noch von Bedeutung, da noch viele ISA-Kontroller (z.B.: Multi-I/O-IDE Kontroller, AHA1540/1542) nur 16 MB mittels DMA ansprechen können, in vielen Rechnern aber schon mehr Speicher installiert ist.

2.5 EISA-BUS

Wurde von „IBM-Clone-Herstellern“ als Konkurrenzprodukt zum MCA-System entwickelt, daher so wie dieser Multimasterfähig (d.h. mehrere Geräte können die Kontrolle über das Bussystem übernehmen, dadurch ist eine Entlastung der CPU möglich und in Summe eine höhere Leistung) ISA-kompatibler, 32Bit-Bus; weiterhin 8MHz Bustakt (Kompatibilität), aber Transferrate von 16 MByte/s im Standardmode durch einen breiteren Bus (32 Bit Datenbus). Der sogenannte Burst-Mode (Startadresse im ersten Takt, in den folgenden Takten jeweils ein Datenwort) steigerte die Übertragungsrate bis 33MByte/s. Diese Leistung war aber nur auf Grund eines nicht unerheblichen Hardware-Mehraufwandes möglich, weshalb die ersten EISA-Rechner sehr teuer waren und hauptsächlich als Server eingesetzt wurden. Später wurde die Übertragungsrate mit Hilfe des Enhanced Master Burst (EMB) auf bis zu 133MByte/s gesteigert, diese Entwicklung kam allerdings zu spät. Heute eignet sich das EISA-System besonders als zusätzlicher I/O-Bus in PCI/EISA Kombiboards.

2.6 MCA

Wurde 1987 von IBM eingeführt und sollte die Limitationen des ISA-Bussystems aufheben. Verfügte über einen 32-Bit Daten- und Adreßbus, Multimaster-Fähigkeit und schnelle Datenübertragung bis zu 16MByte/s.
Die damalige restriktive Lizenzpolitik IBM (zwecks Ausschaltung der Clone-Hersteller), das Fehlen von 1 oder zwei ISA Slots zwecks Abwärtskompatibilität und die horrenden Preise für MCA-Systeme (IBM PS/2) verhinderte die Etablierung dieses Bussystems.
Mit MCA 2 öffnete IBM die Lizenzpolitik, diese Technologie konnte sich aber trotzdem nicht mehr durchsetzen.

2.7 Local Bus (VLB)

Herstellern von Grafikkarten war sowohl MCA als auch ISA ein Dorn im Auge, da sie für ihre Produkte einen schnellen und vor allem billigen Bus benötigten. Zunächst erschienen Lösungen in Form von "Local Bus-Systemen", bei denen die Graphikchips direkt auf dem Motherboard untegebracht waren. Der Markt reagierte zunächst verhalten auf dieses neue System, wehalb sich die in der VESA (Video Electronics Standards Association) organisierten Anbieter den VESA-Local-Bus präsentierten (Transferraten bis 80MByte/s bei On-Board Systemen, 75MByte/s bei Boards mit Slots). Der VESA-Local-Bus (VLB) ist ein leicht modifizierter 486er Prozessor-Bus, wodurch Boardhersteller mit sehr geringem Aufwand aus Basis existierender 486-ISA-Designs VLB-Boards entwickeln. Der VLB sollte möglichst schnell und billig sein, deshalb wurde auf aufwendige Dinge wie Ressource-Sharing oder Autokonfiguration verzichtet, was aber später (in der Version 2.0) zu Problemen führte. In der ersten Version erlaubte die Spezifikation nur dann Slots, wenn die Taktfrequenz nicht höher als 40MHz war, was auch zu Problemen führte. Die Zukunft des VLB hängt in erste Linie von der Entwicklung und Preisgestaltung des technisch wesentlich fortschrittlicheren PCI-Bus ab.

2.8 PCI

Wurde von Intel für den 486er-Nachfolger Pentium entwickelt, wobei neben der Performance Anwenderfreundlichkeit und Zukunftssicherheit aber auch Abwärtskompatibilität im Vordergrund standen. CPU- und Erweiterungsbus sind streng voneinander getrennt und durch sogenannte Host-Bridges (setzt Schreib- und Leseanforderungen in PCI-Bus-Zyklen um) miteinander verbunden. Diese Technologie soll die Kompatibilität zu den nächsten Prozessorgenerationen gewähleisten. Möglichst viele Eigenschaften von bereits etablierten Bussystemen sollten übernommen werden (z.B. IRQ-Sharing, Multimasterfähigkeit, ...). Datentransferraten: 32-Bit 133 MByte/s; 64Bit (Revision 2.0) bis 266 MByte/s. Der PCI-Bus ist viel detaillierter und genauer spezifiziert als dies bei EISA und VL der Fall ist (exakte Leitungslängen auf Erweiterungskarten, Design der Anschlüsse etc.). Um eine Etablierung auf dem Markt zu forcieren, gestaltet Intel die Lizenzpolitik sehr liberal (US$ 2.500 pro Jahr). Jüngstes aller bisher vorgestellten Bussysteme. Ob sich der PCI-Bus durchsetzen wird, bleibt abzuwarten, es hängt in erster Linie von der Preisgestaltung ab. Technisch bietet er jedenfalls den am weitesten entwickelten Standard.

2.9 VME/VXI-Bus

Versa Module Europe. Versa ist ein Bussystem von Motorola, dieses wurde 1981 von Motorola, Mostek, Signetics und Philips zu einem rechnerunabhängigen asynchronen Parallelbussystem unter Verwendung des Europaformat für Printplatten weiterentwickelt.

2.9.1 Eigenschaften

Kartenformat 10 x 16 cm (Europaformat)
- einfach 96 pol. Stecker, 16 Datenl., 24 Adreßl.
- doppelt 2 x 96 pol. Stecker, 32 Datenl., 32 Adreßl.

multiprozessorfähig
6 Address-Modifier-Lines

- 16/24/32-Bit Adressen
- Anwendung/OS
- Code/Daten
- Speicher/IO
- geschützter/ungeschützter Speicherbereich

Arbitrierungsverfahren (Anforderung bis Zuteilung des Bus) über vier Prioritätsebenen, wobei die Teilnehmer ihre Priorität dynamisch verändern können.

enthält IIC - BUS (InterIntelligence Bus) einen synchronen seriellen Bus zum Austausch von Nachrichten zur Systemsteuerung, Synchronisierung und Programmunterbrechung. Auf diesem Bus werden 38 Bit lange Nachrichten nach einem genau definierten Protokoll ausgetauscht.

Maximale Transferrate: 24 MByte/s
7 Interruptebenen
3 Fehlerbehandlungssignale
4 verschiedene Versorgungsspannungen (+5V, +12V, -12V und +5V Standby)

2.9.2 VXI (3 x 96 pol. Stecker)

Nachfolger von VME, hier wird eine dritte Größe eines Boards mit 3x96 pol. Stecker definiert und die gegenseitige Beeinflußung von Einsteckkarten bzw. deren stärkere Kommunikation untereinander stärker berücksichtigt.

2.10 SCSI-Bus

Aus dem 1979 von Shugart entwickelten SASI-Bus (Shugart Associates System Interface) entwickelt und 1982 standardisiert (ANSI X3T9.2, ECMA ..., ISO....)

2.10.1 Merkmale

50 poliges, maximal 6 m langes Flachbandkabel mit 8 Datenleitungen
Multimasterfähig
Max. Transferrate: 1,5 MByte/s asynchron, 4 MByte/s synchron
Eine Unterbrechungsebene
Maximal 8 Geräte

2.10.2 Befehlssatz

Common Command Set mit 8 Gruppen zu je 32 Befehlen

Gruppe 0	Grundbefehle
Gruppe 1	erweiterte Befehle
Gruppe 2	reserviert für Erweiterungen
Gruppe 3	reserviert für Erweiterungen
Gruppe 4	reserviert für Erweiterungen
Gruppe 5	erweiterte Befehle
Gruppe 6	geräteabhängige Befehle
Gruppe 7	geräteabhängige Befehle

3 Mikrokontroller

Mikrokontroller-Mikroprozessor-Mikrocomputer

4 Prozessorarchitektur

Überblick

5 Speicher

Überblick

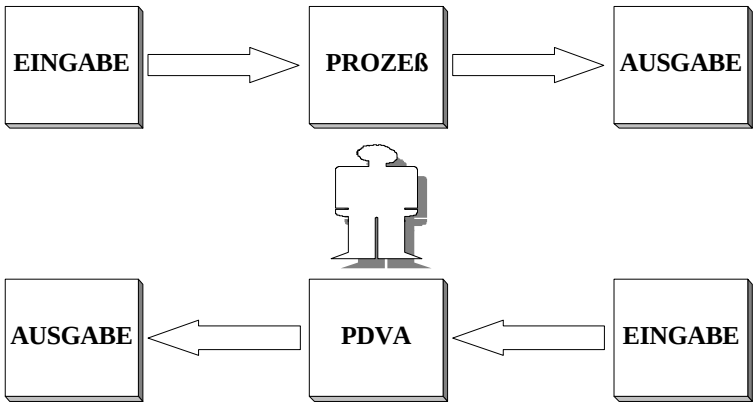
6 Prozeßregelung

6.1 Begriffsbestimmung

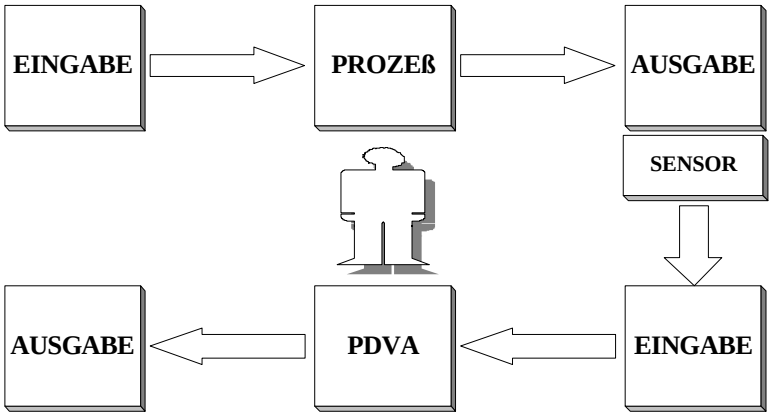
Daten	Informationen - all jene Dinge die einen Informationsgehalt haben digitalisierbar
BIT	Binary Digit - Binärzustand - kleinste Informationseinheit
Byte	8 Bit
Wort	verschiedene Größen möglich
	8 Bit PIC, XT
	12 Bit Instruktionswort eines PIC
	16 Bit 286, 8052
	32 Bit 386, 486, HOST
	60 Bit CDC
	64 Bit ALPHA, R4000
	256 Bit Very Long Instruction Word [VLIW]
Datenverarbeitung	Manipulation von Daten - verknüpfen von Daten zu neuen Daten
Programm	Folge von Verarbeitungsanweisungen
File	Aneinanderreihung von Daten, die in einem logischen Bezug stehen, unter einem gemeinsamen Namen
Kanal	Ein Kanal ist eine Zuordnung zwischen einer internen (logischen) und einer externen (physischen) Einheit - kann aber auch eine Zuordnung zwischen Hard- und Software sein. Diese Zuordnung kann vom Benutzer, Operator, Arbeitsvorbereiter und auch von einem Programm kommen.
Prozeß	Ein Prozeß ist die Umformung und/oder der Transport von Materie, Energie und/oder Information
Techn. Prozeß	Ein Prozeß der mit technischen Mitteln gemessen, geregelt und/oder gesteuert werden kann
Hardware	alles was man angreifen kann - materiell
Software	nicht materiell - betriebsnotwendig (um die HW benutzen zu können)
PDVA	Prozeßdatenverarbeitungsanlage - Rechner, die mit Prozeßdaten umgehen können und echtzeitfähig sind eignen sich als PDVA.

6.2 Verbindung zwischen Prozeß und Prozeßdatenverarbeitungsanlage

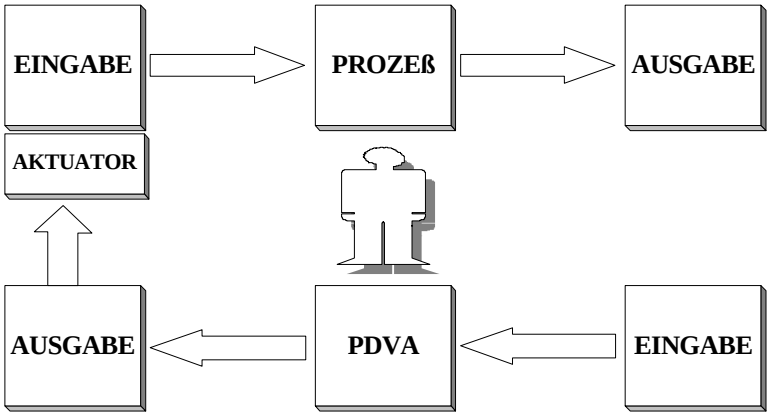
a.) Offline



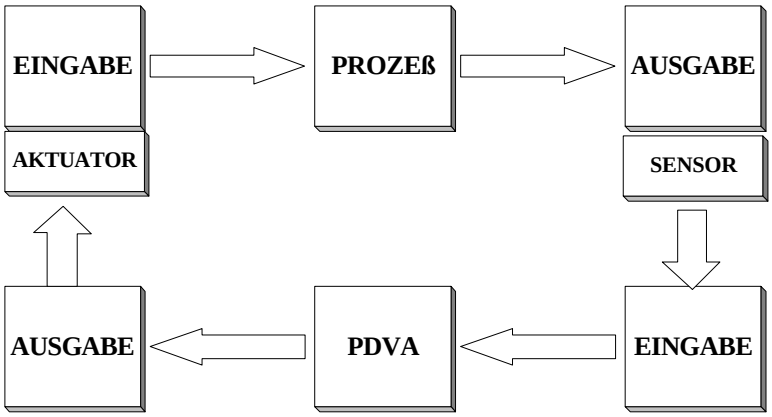
b.) Online - Datenerfassung



c.) Online -Steuerung



d.) Online - Closed Loop



Wieso finden sich bei den vorangegangenen Grafiken die Koppelelektroniken (Aktuator, Sensor) immer auf der oberen, also auf der Prozeßseite ?

1) alles spezifische zum Prozeß verlagern
2) digitale Daten können besser (weniger fehleranfällig) übertragen und rekonstruiert werden
   D [dB} = 20 log (Ue/Ua)
3) auf der anderen Seite kann oft technische Unmöglichkeit herrschen
   z.B. Temperatur in einem Hochofen messen: stellt man den Computer neben den Hochofen wird er kaputt, stellt man ihn weiter weg entspricht die Temperatur nicht mehr der im Hochofen

6.3 Forderungen an PDVA

Schnelligkeit — Echtzeitverarbeitung - je schneller desto schneller kann man auf Abweichungen reagieren - der Begriff „schnell“ ist relativ (abhängig von Prozeß).Schnell bedeutet so schnell, daß die Reaktion rechtzeitig erfolgt

Synchrone Daten — regelmäßige Daten - Bildschirm

Asynchrone Daten — unregelmäßige Daten - Serielle und Parallele Schnittstelle - langsamer da Steuerinfo

Gleichzeitigkeit — Prozesse bestehen häufig aus Teilprozessen die oft gleichzeitig ablaufen müssen - Problem : Synchronisation

Sicherheit — a) Ausfallsicherheit (z.B. Prozeßrechner im Airbus)
höhere Ausfallsicherheit durch Redundanz (ein anderes System schaltet sich dazu) oder Clustersystem (andere Systeme laufen parallel und verkraften den Ausfall)
b) Prozeßsicherheit (z.B.Ampel darf nie in 2 Richtungen grün zeigen)
muß geeignete Mechanismen haben um Fehler abzuwenden

Geeignete Überträger
Aktuatoren, Sensoren

Ein Sensor/Aktuator wandelt eine physikalische Größe in eine andere physikalische Größe um. Es ist also praktisch ein A/A-Wandler.
Ein Bauelement kann sowohl Aktuator als auch Sensor sein.

Der Sensor wird so eingesetzt, daß er auf der Ausgangsseite ein Signal hat, daß der Computer verarbeiten kann (z.B. Spannung).
Der Aktuator wird so eingesetzt, daß er auf der Eingangseite ein Signal hat, daß der Computer verarbeiten kann.

Aktuatoren	Relais (Schalter)
	Schrittmotor
	Hydraulik
	Lautsprecher
Sensoren	Thermometer (Quecksilber, Bimetall)
	Thermoelement
	Lichtschranke
	Feuchtefühler
	Mikrophon
	PTC, NTC

6.4 Einsatzgebiete

Industrielle Produktion	Überwachung von Meßwerten, vorgeben von Sollwerten
Energietechnik	Lastverteiler automatisch schalten (wann wird wo wieviel Energie benötigt)
Energienutzung	Waschmaschine, Kopierer
Verkehrstechnik	Ampelsteuerung, Flugplatzsicherung, Flugzeugsteuerung, Eisenbahnsignalsteuerung, beim Auto ABS und Transistorzündungen, Verkehrsleitsysteme
Laborautomatisierung	Meß- und Regeltechnik
Medizin	Bilderfassung, Echtzeitröntgenbilder, Expertensysteme

6.5 Wesentliche Aspekte des Echtzeitbetriebes

Daten sind einmalig	
Daten sind fehlerhaft	Meßfehler, Übertragungsfehler, Umwandlungsfehler
Daten sind zeitabhängig	Zeitpunkt der Messung nicht egal
Menge unbegrenzt	Filter ansetzen um sinnfvolle Datenmenge zu erhalten

6.6 Interrupts

a.) Definition:

Ein Interrupt ist ein Hardware-gesteuerter Vorgang durch den die Ausführung eines Programmes unterbrochen wird, der momentane Programmzustand in einen Zwischenspeicher kopiert wird und eine sogenannte Interrupt Service Routine (ISR) aufgerufen wird.

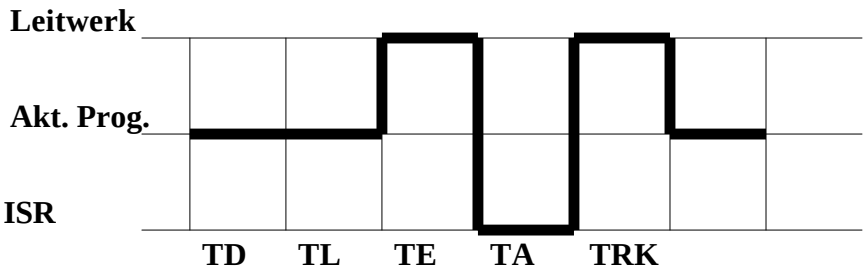
Nach Beendigung der ISR wird mit dem ursprünglichen Programm weitergemacht.
Das Umschalten von einem Prozeß zu einem anderen wird Context switching genannt.
Information die abgespeichert wird Vektor genannt:

HOST:	PC (Program Counter)	PSW
PC:	IP (Instruction Pointer)	Flags

b.) Teilaufgaben bei einem Interrupt

Unterbrechungsmeldung speichern
Meldung an das Leitwerk
Zuordnungder ISR (über Vektor)
Auswertung der Priorität
Context switching
Zwischenspeicher löschen

c.) Zeitlicher Ablauf



TD = Durchlaßzeit (Unterbrechungsmeldung speichern)
TL = Latenzzeit (Jene Zeit die verstreicht bevor das Leitwerk aktiv wird)
TE = Erkennungszeit (Laden des Vektors, Priorität auswerten)
TA = Ausführungszeit (die Zeit die die ISR braucht)
TRK = Rückkehrzeit (Context Switching)

TD+TL	= tw (Wartezeit)	
TE+TA+TRK	= tv (Verarbeitungszeit)	
TD+TL+TE	= tr (Reaktionszeit)	
TD+TL+TE+TA	= ta (Antwortzeit)	(Jetzt ist Interrupt fertig bearbeitet)
TD+TL+TE+TA+TRK	= tg (Gesamtzeit)	

d.) Schritthaltende Verarbeitung

tp.....Prozeßzeit
Zeit zwischen 2 gleichartigen Interrupts, sollte diese schwanken (asynchrones Ereignis) dann minimale Zeit betrachten.

tz.....Maximal zulässige Antwortzeit

tz <= tp
Für ein Einzelereignis darf diese Zeit auch einmal überschritten werden, aber in Summe sollte es nicht länger sein.

Echtzeitbedingung

Die Gesamtzeit muß kleiner als die Antwortzeit sein. tg < tv

$$tg_i < tz_i$$

Belastung

Hängt mit Verarbeitungszeit zusammen, je länger desto größer Auslastung

$$tv_i/tp_i$$

Gesamtbelastung

$$\text{Summe } (tv_i/tp_i) <= 1$$

Faustregel: Je kürzer die Verarbeitungszeit desto höher ist die Priorität zu wählen
Je kürzer die Prozeßzeit desto höher ist die Priorität zu wählen

Beispiel

tp_1=10ms tp_2=3ms
tv_1=4ms tv_2=1ms
$tp_i=tz_i$

Belastung = 4/10 + 1/3 =0,73

e.) Folgen für die Programmierung

asynchrone Programmierung (Zeitstruktur wird nicht vom Programmierer, wie bei synchroner Programmierung vorgegeben sondern vom Prozeß)

Merkmale asynchroner Programmierung:

statt eines Event Managers wird für jedes Event eine Interrupt Routine festgelegt.

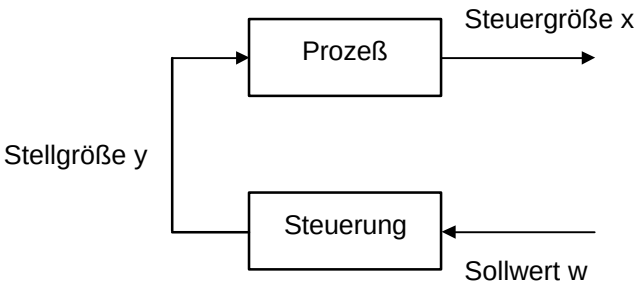
Vorteile
Programmteile laufen Interrupt gesteuert
auf wichtige Ereignisse kann sofort reagiert werden
Auch „unverhergesehene“ Ereignisse können behandelt werden

Nachteile
höherer Programmieraufwand
Synchronisation muß händisch erfolgen
die Aufeinanderfolge der Programmteil ist dynamisch und nicht statisch
(das kann Probleme bei der Optimierung geben)

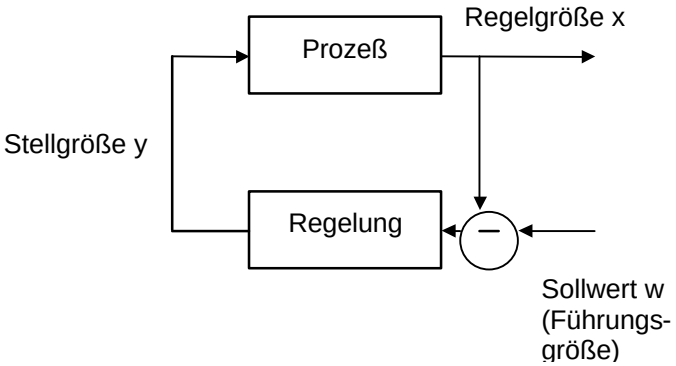
6.7 Regelungstechnik

6.7.1 Abgrenzung Regelung - Steurung

a.) Steuerung



b.) Regelung



z.B. Temperaturmessung

über Steuerung: man muß genau wissen bei welcher Außentemperatur wieviel geheizt werden soll, da man keine Rückmeldung bekommt ob die gewünschte Temperatur erreicht wurde.

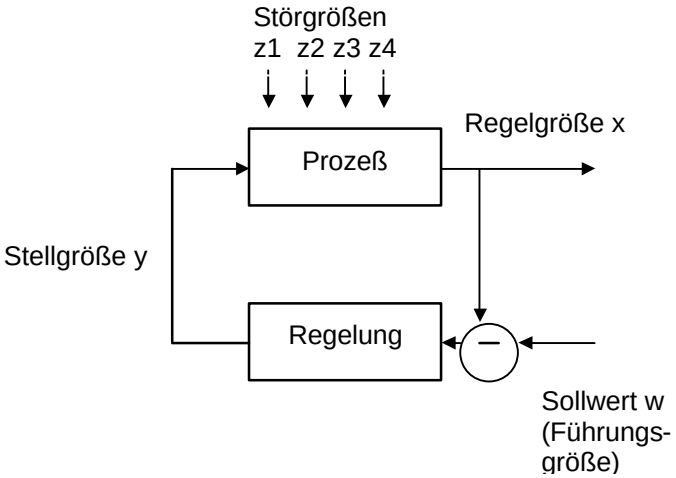
über Regelung: man kann aufgrund der Rückmeldungen entscheiden wie man zum Ziel kommt, also ob z.B. noch mehr geheizt werden muß oder ob die Maßnahmen die man getroffen hat schon das gewünschte Ergebnis erzielt haben.

6.7.2 Allgemeines zur Regelungstechnik

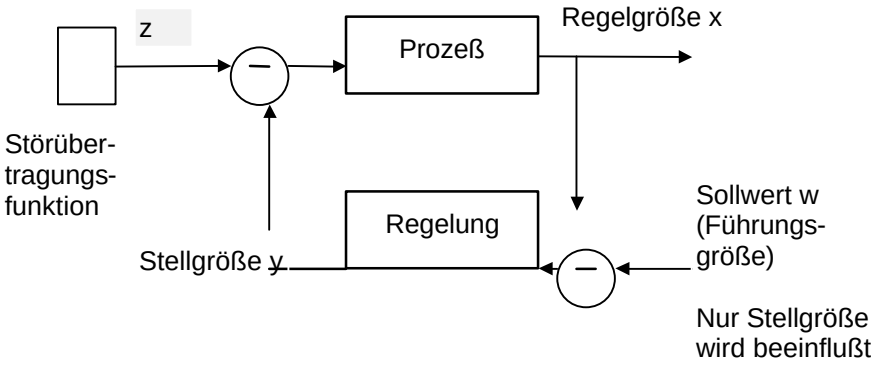
Festwertregelung	der Sollwert bleibt über eine bestimmte Zeit konstant (z.B. immer 200C)

Folgewertregelung Sollwert ändert sich abhängig vom Prozeß
(z.B. konstante Drehgeschwindigkeit bei einer Drehbank, je mehr man zur Mitte kommt desto schneller muß die Drehgeschwindigkeit werden)

Störung

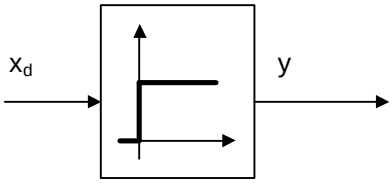


zi sind die Störgrößen. Aus der Störgröße versucht man eine Hauptstörgröße zu extrahieren. Es kann sich dabei um eine einzelne oder um eine aus mehreren Störgrößen zusammengesetzte (imaginären) Störgröße handeln.



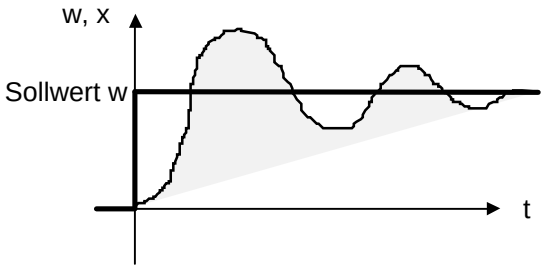
6.7.3 Proportionalregler (P-Regler)

Blockdiagramm



$y = K_p * x_d$

Zeitverhalten



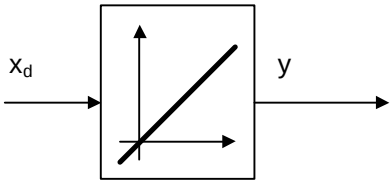
x_d ... Differenz zwischen Soll und Ist, Regeldifferenz
y ... Stellgröße
K_p ... Konstante (Anpassung d. Reglers an den Prozeß)

X_p ... Reglerverstärkung x_d/y_n in %
kann statt K_p in der Formel eingesetzt werden

Kann nie den Sollwert w erreichen, da wenn Sollwert erreicht ist ist das Ergebnis der Formel 0 und die Heizung dreht sich ab.
Je größer K_p desto schneller die Annäherung. Stört überschwanken dann K_p kleiner machen, wenn man Überschwanken erhöhen möchte dann auch K_p erhöhen.
Einsatz: Bei Anwendungen wo das Regelergebnis nicht ganz so genau sein muß (z.B. Temp.)

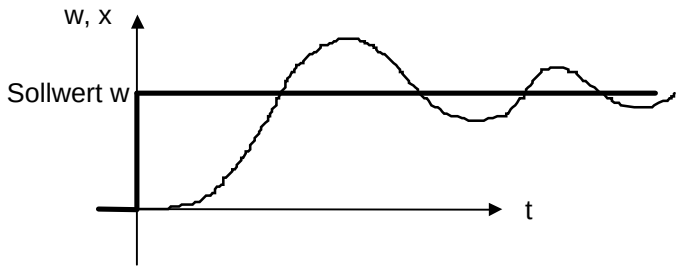
6.7.4 Integralregler (I-Regler, Nachstellregler)

Blockdiagramm



$$y = K_I \int_0^{t1} x_d(t)\, d_t$$

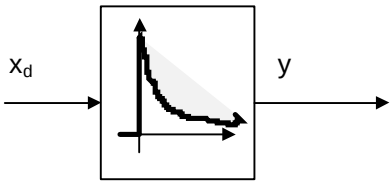
Zeitverhalten



Lange Anlaufzeit, Überschwingen unvermeidbar, der Sollwert wird erreicht.

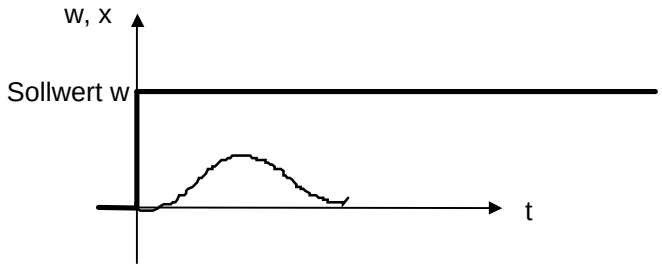
Der I-Regler ist eher träge. Wird eingesetzt wenn der Sollwert über längeren Zeitraum konstant bleibt. Das Blockdiagramm wird folgendermaßen interpretiert: zu Zeitpunkt t_0 tritt x_d auf und bleibt konstant. Ki sollte eher kleiner gewählt werden.

6.7.5 Differentialregler (D-Regler, Vorhalteregler)



$$y(t) = K_D * \dot{x}_d(t)$$

Zeitverhalten



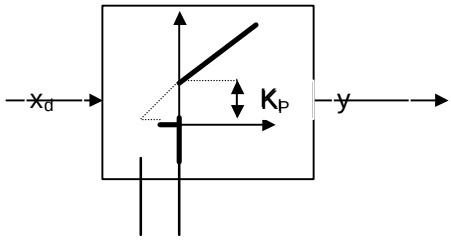
$$\dot{a} = \frac{d_a}{d_t}$$ zeitliche Ableitung $$a' = \frac{d_a}{d_x}$$ räumliche Ableit.

Ist dann sinnvoll, wenn ständige Änderungen des Sollwertes auftreten, dann liefert der D-Regler viel, ist der Sollwert einmal erreicht ändert sich nichts mehr und er liefert nichts. Reagiert nur auf Änderungen von x_d, nicht auf x_d selbst. Brauchbar, da am Anfang hoher Sollwert, aber nur in Verbindung mit anderen Reglern. Vorallem für sprungartige Störgrößen oder wenn sich die Sollwerte häufig sprungartig ändern.

Dieser Regler ist alleine sinnlos, aber in Kombination mit einem anderen Regler sehr sinnvoll, da dieser Regler sehr gut zu Beginn ist, wenn sich noch viel ändert.

6.7.6 Kombinationen

a.) PI-Regler



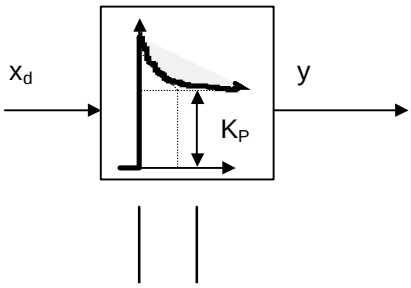
T_n ... Nachstellzeit

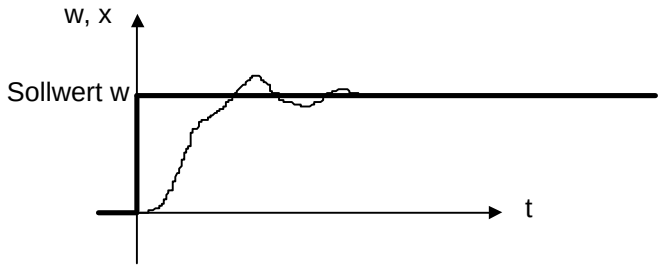
Je größer Tn desto kleiner KI (flacher)
Tn = KP/KI

PI-Regler sind nicht so träge wie I-Regler und schlagen nicht so weit aus wie P-Regler, Sollwert wird erreicht, Anfang P-Regler, Ende(Sollwert erreicht) I-Regler
$y(t) = KP * xd(t) + \int xd(t)\,dt$

z.B. $y(t1)=K_P * x_d(t1) + \int_0^{t_1} x_d(t1)\,d_t$

b.) PD-Regler



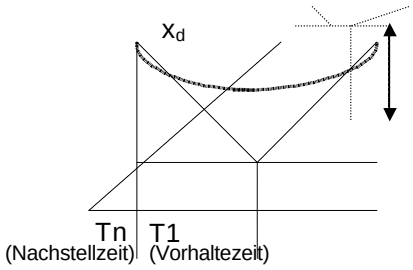


Tv ... Vorhaltezeit

$T_V = K_D/K_P$

Sollwert wird nicht erreicht, da: Anfang: D-Regler, Ende: P-Regler
$y(t) = K_P * x_d(t) + K_d * x(t)$

c.) PID-Regler



y

Sollwert w

Anfang D-Regler, Ende I-Regler Sehr gut Anpassung Stellgrößen (KP, KI, KD)

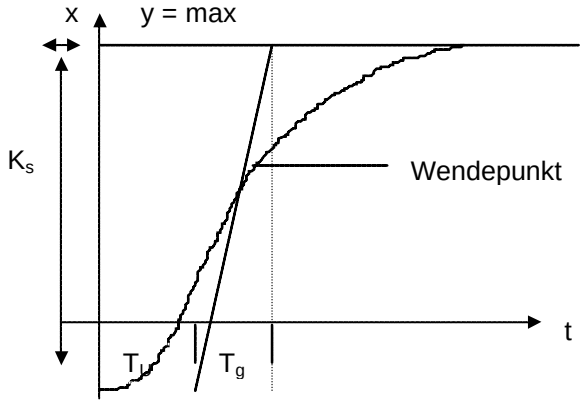
$$y(t) = K_P * x_d(t) + K_I * \int_0^{t_1} x_d(t)\, d_t + K_d * x(t_1)$$

Mitte p-Regler, durch 3

KP = 0 ⇒ ID-Regler, KI = 0 ⇒ PD-Regler, KD= 0 ⇒ PI-Regler
geht Tn nach unendlich ⇒ PD-Regler, geht T1 nach 0 ⇒ PI-Regler
Jegliche Sollgröße kann erreicht werden. Fast kein Überschwingen. Durch Anpassung möglich, kein Überschwingen zu haben. Integrationsband: 4. Stellgröße, nur innerhalb wird dieser wird I-Regler aktiviert.
Wenn das I-Band zu groß ist ⇒ zu viel überschwingen. I-Band eigentlich sinnlos.
Wenn das I-Band zu klein ist ⇒ durch P-Regler wieder „oben hinaus“, alle Infos von I-Regler gehen verloren.

6.7.7 Finden der Regelparameter

Es gibt immer für jeden der Parameter (K_D, K_P, K_I, I-Band) immer <u>einen</u> optimalen Wert. Diesen gilt es zu finden.



Einstellregeln Chien - Hrones - Reswick

	Führungsregler	Störungsregler
P	$X_P = 3{,}3 * \frac{K_S * T_U}{T_g}$	$X_P = 3{,}3 * \frac{K_S * T_U}{T_g}$
PI	$X_P = 2{,}9 * \frac{K_S * T_U}{T_g}$ $T_n = 1{,}2\ T_g$	$X_P = 1{,}6 * \frac{K_S * T_U}{T_g}$ $T_n = 4\ T_U$
PD	$X_P = 0{,}56 * \frac{K_S * T_U}{T_g}$ $T_1 = 0{,}5\ T_U$	$X_P = 0{,}56 * \frac{K_S * T_U}{T_g}$ $T_1 = 0{,}5\ T_U$
PID	$X_P = 1{,}6 * \frac{K_S * T_U}{T_g}$ $T_n = T_g$ $T_1 = 0{,}5 * T_U$	$X_P = 1{,}05 * \frac{K_S * T_U}{T_g}$ $T_n = 2{,}4\ T_g$ $T_1 = 0{,}42 * T_U$

„Einstellregeln aus der Praxis“

-) reiner P-Regler (gerade so, daß er nicht überschwingt, dann haben wir KP)
-) I-Regler dazuschalten (nun haben wir PI-Regler und kriegen KI)
-) D-Regler dazuschalten (PID-Regler)

Sollwertsprünge, jetzt KD so stellen, daß die Sprünge so schnell wie möglich ausgeregelt sind. Kp ist so einzustellen, daß kein Überschwingen entsteht. Danach ist der I-Anteil hinzuzuschalten, K_I wird so eingestellt, daß eine Ausregelung zwischen Ist- und Sollwert geschieht. Anschließend wird der D-Anteil hinzugeschalten und es werden einige Sollgrößensprünge gemacht. K_D so groß wählen, daß ohne Überschwingen der nächste Sollwert erreicht wird.

6.8 Modelle zur Prozeßregelung

6.8.1 Prozeßmodell

Ziel ist es, daß die Regelung des Prozesses nach bestimmten Richtlinien optimiert wird.

Kriterien:

Maximierung des Gewinns
Minimierung der Kosten
Optimale Ausnützung von Material / Energie
Minimierung des Fertigungszeit
Maximierung der Produktqualität
Minimierung der Umweltschäden

Für die Modellbildung werden folgende Informationen benötigt:

Spezifikation des Produktes

Kosten des Produktes
Einzelkomponenten des Produktes
Wert des Einzelkomponenten des Produktes
Fertigungskosten des Produktes
Qualtitätsanforderungen an das Produkt
Zusammensetzung des Rohmaterials
Kosten des Rohmaterials
Funktion, Grenzen und Energieverbrauch der Fertigungseinheit
Fertigungseinrichtunge (Zeit, Kosten, verwendete Werkzeuge)
Kosten der Energie *
Kosten der Arbeitskraft

Um mit diesen Daten den Prozeß zu modellieren, ist viel Zeit und eine Person, die sich mit dem Prozeß gut auskennt, nötig.

* Die Firma Zumtobel hat ein sehr interressantes Modell entwickelt.
Es handelt sich dabei um eine Jalousienregelung. Im Winter werden die Jalousien geöffnet (die wenige Sonne nutzen) im Sommer werden die Jalousien geschloßen. Allerdings ist diese Regelung etwas komplizierter, da sowohl die Außen als auch die Innentemperatur berücksichtigt wird. Das System besteht aus 2 Fuzzyreglern. Sagt der Benutzer „zu“ gehen die Jalousien zu, dh zu Beginn ist der Benutzerwunsch Gesetz. Aber nach einigen Minuten läßt die Absolutheit dieser Anweisung nach und der Regler übernimmt mehr und mehr die Kontrolle des Geschehens. Dieser Vorgang wird vom 2. Fuzzyregler geregelt. Der 1. ist für die Temperaturregelung zuständig.

6.8.2 Analytische Modell

Man versucht das Prozeßmodell indem man den Prozeß physisch betrachtet und dann das Modell bildet.
Der physische Prozeß wird möglichst vollständig mit mathematischen Formeln beschreiben.
Eignet sich nur für sehr einfache physische Prozesse.

Bei dynamischen Prozessen können folgende Probleme auftreten:

Das Modell wird sehr groß (kann nicht berechnet werden)
Eine oder mehrere wichtige Parameter fehlen
Die Wechselwirkung zwischen einzelnen Parametern wird oft nicht genug berücksichtigt.
Bsp: Isolation einer Hauswand. Die Isolierungswerte der Wand und des Fensters sind leicht zu bestimmen, aber im Übergangsbereich ist es schwierig.

6.8.3 Experimentelles Modell

Sammeln von Meßwerten und daraus dann ein Modell formen.
Die Funktion muß linearisiert werden (Man stellt, den für das Modell interressanten Teil einer Funktion/Kurve als Gerade dar)

Das Modell gilt nur für den erstellten Prozeß
Die Experimentdaten sind vom Experimenttator abhängig
Bei einer Streung der Meßwerte ist eine hohe Anzahl an Experimenten nötig
Die Wechselwirkung wird fast vollständig vernachlässigt

6.8.4 Adaptives Modell

VERFEINERUNG DES MODELLS

GROBMODELL	
while (MODELL NICHT FEIN GENUG)	
	AUSWAHL WICHTIGER PARAMETER
	KONTROLLE DURCH MESSUNGEN

Vorteile gegenüber den anderen Modellen:

stark minimierte Rechenzeit
Anzahl der Versuche hält sich in Grenzen
Es vereinigt die Vorteile vom Analytischen und Experiementellem Modell

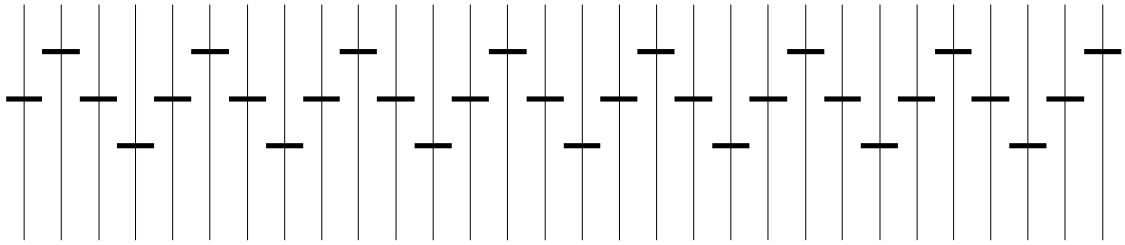
6.8.5 Prozedurales Modell

Ähnliches Vorgehen wie bei der Programmierung eines Taschenrechners. (Folge eintippen
Man muß versuchen herauszufinden wie der Ablauf der Tätigkeiten ist. (Oft Probleme, da die Mitarbeiter einer Firma diese oft nicht sagen möchten, außerdem denkt man oft nicht an einige Details, oder weiß bei eher seltenen Abläufen z.B. Notfälle nicht den genauen Ablauf)

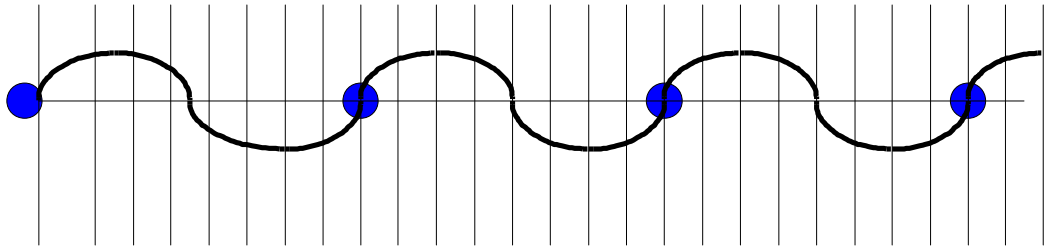
6.8.6 Kombination von digitalen und analogen Modellen

Probleme bei der Zuordnung

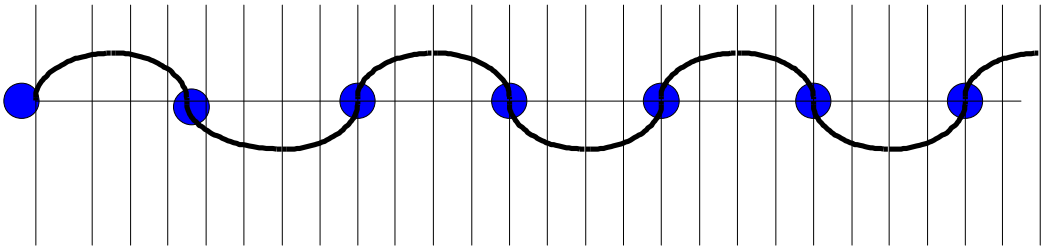
analog : Zeit kontinuierlich, mit beliebigen Zwischenwerten
digital : Stufen, sprunghaft. Zeit diskontinuierlich mit Werten die bestimmt sind



1mal pro Periode abtasten (ergibt Gerade)

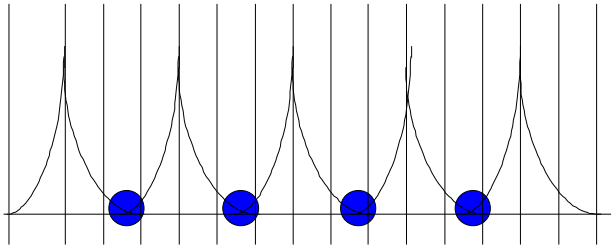


2 mal pro Periode abtasten (ergibt bei Pech auch Gerade)



Dieses Problem kann man lösen indem man sich die Form des Signales anschaut.

Bei diesem Signal darf man nicht nur in den „Tiefs“ messen sonst wird die Kurve völlig verfälscht.



6.9 Temperaturregler

$T_n=T_{n-1} + A * (Uh_{n-1})^2 - B$

Konstante A....
B....Abflußwärme

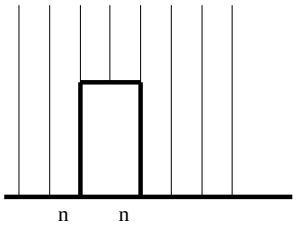
Diese 2 Konstanten müssen an die Realität angepaßt werden.

$T_n=T_{n-1} + A * (Uh_{n-1})^2 - C(T_{n-1} - T_U)$

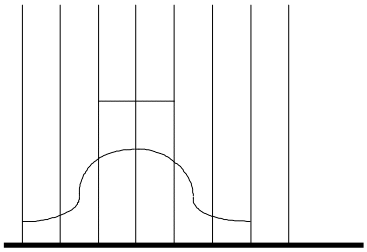
T_U....Umgebungstemparatur
$T_{n-1} - T_U$....Wärmeabflüsse

Da aber die vom Regler getroffene Aktion nicht sofort wirkt muß man noch eine Verzögerung einbauen

$T_n=T_{n-k} + A * (Uh_{n-k})^2 - C(T_{n-k} - T_{U\ n-k})$
Verbesserungen:



Heizung aufdrehen und sofort wieder abdrehen



Die obere (erste) Grafik stimmt nicht, denn es wird
nicht heiß und dann kalt sondern es wird nie
richtig warm!!!!

Die Flächen unter der Kurve müssen aber in beiden obigen Zeichnungen gleich sein. (Integral über alle K's)

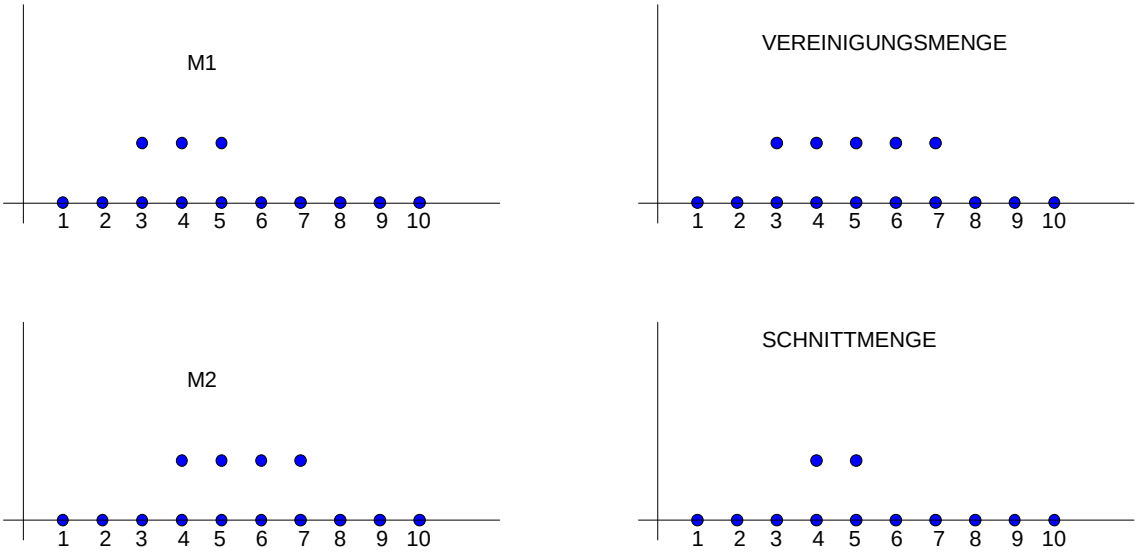
7 Fuzzy Logic

Klassische Logik	Ja, Nein
Fuzzy	Ja, Nein, Vielleicht (viele Vielleichts)

Problem leichter darlegbar

- linguistische Variablen (an die Sprache angepaßt)
- Lösungen die auf den ersten Blick nicht reichen können, können möglicherweise besser sein als andere

Grundmenge
M = {1,2,3,4,5,6,7,8,9,10}
M = (x|(xE |N) ^ (x <= 10)}
M_1={3,4,5}
M_1 = (x|(xE |N) ^ (x >2)^(x<6)}
M_1 C M M_1 ist Teilmenge von M
M_2={4,5,6,7} C M



Zugehörigkeitsfunktion von M_1
Fm_1={0,0,1,1,1,0,0,0,0,0}
Fm_2={0,0,1,1,1,1,1,0,0,0}
Fm_1 ^ Fm_2={0,0,0,1,1,0,0,0,0,0}
Fm_1 v Fm_2={0,0,1,1,1,1,1,0,0,0}

M_1 v \M_1 = M
M_1 ^ \M_1 = {} bzw. 0

In der Fuzzylogic
M_{F1} v \M_{F1} <= M
M_{F1} ^ \M_{F1} > 0

7.1 Scharfe Mengen (Crisp Sets)

NULLELEMENT	m_1 ^ 0 = 0
	m_1 v 0 = m1
EINSELEMENT	m_1 ^ 1 = m_1
	m_1 v 1 = 1
IDEMPOTENZ	m_1 ^ m_1 = m_1
	m_1 v m_1 = m_1
KOMMUTATIVGESETZ	m_1 ^ m_2 = m_2 ^ m_1
ASSOZIATIVGESETZ	m_1 ^ (m_2 ^ m_3) = (m_1 ^ m_2) ^ m_3
	m_1 v (m_2 v m_3) = (m_1 v m_2) v m_3
DISTRIBUTIVGESETZ	m_1 ^ (m_2 v m_3) = (m_1 ^ m_2) v (m_1 ^m_3)
DE MORGANSCHE REGEL	!(m_1 ^ m_2) = !m_1 v !m_2
	!(m_1 v m_2) = !m_1 ^ !m_2

! = nicht

Alle diese Gesetze gelten nur dann wenn M_1 und M_2 E M (dh. die beiden Mengen müssen die gleiche Grundmenge haben)

Wenn es sich nicht um die gleiche Grundmenge handelt, dann Zahlentriple (z.B. 1 Punkt im Raum)
=> M x N dann Produktmenge verwenden {(x,y) | (x E M) ^ (y E N) }

Beispiel Datenbankabfrage

FIRMA	UMSATZ	PROFIT

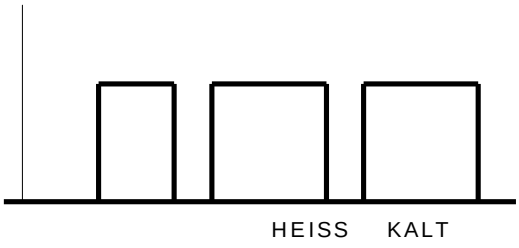
A	500	7
B	600	14
C	800	17
D	850	12
E	900	18
F	1000	15
G	1100	14
H	1200	13
I	1400	6
J	1500	12

Jetzt Abfrage in welche Firma sollte man investieren. (Üblicherweise in SQL realisieren)
Wenn Hoher Umsatz: Umsatz => 1000
guter Profit: Profit => 14%

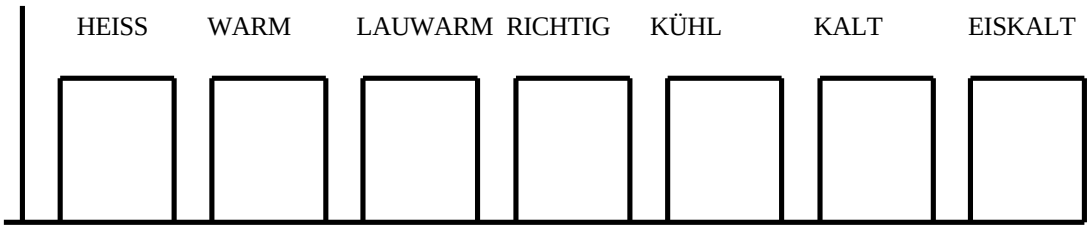
FIRMA	U	P	U ^ P
A	0	0	0
B	0	1	0
C	0	1	0
D	0	0	0
E	0	1	0
F	1	1	1
G	1	1	1
H	1	0	0
I	1	0	0
J	1	0	0

Trotzdem ergibt sich hier eine eher schlechte Lösung denn am besten wäre Firma E.

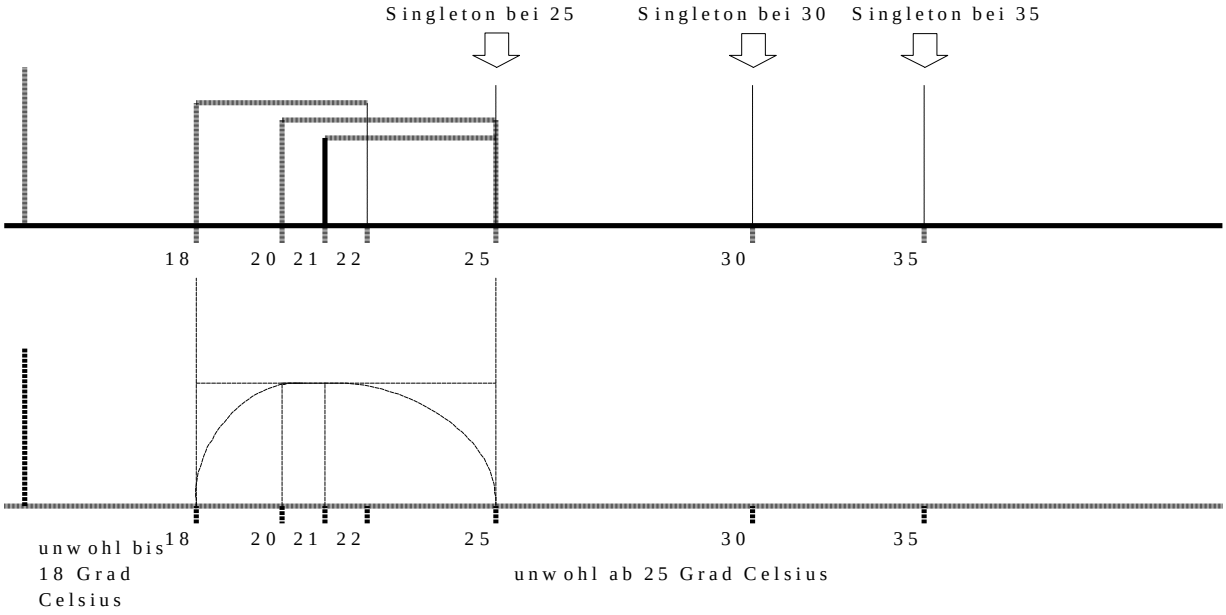
Intervallvariable



Bei dieser Lösung ist kein wirklicher Übergang zwischen Heiß und Kalt.
Daher:

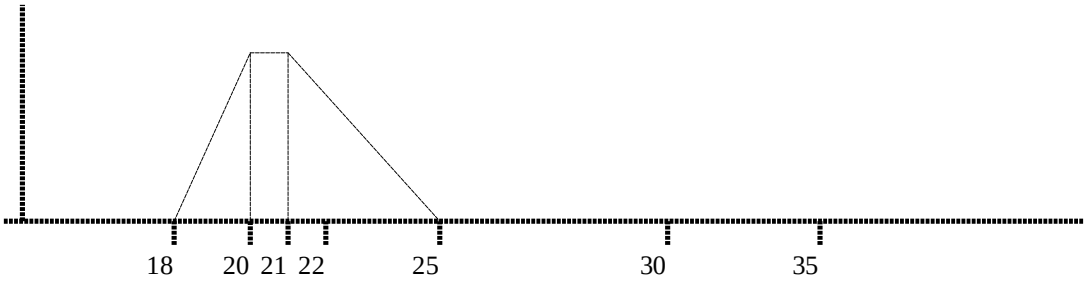


7.2 Unscharfe Mengen (Fuzzy Sets)



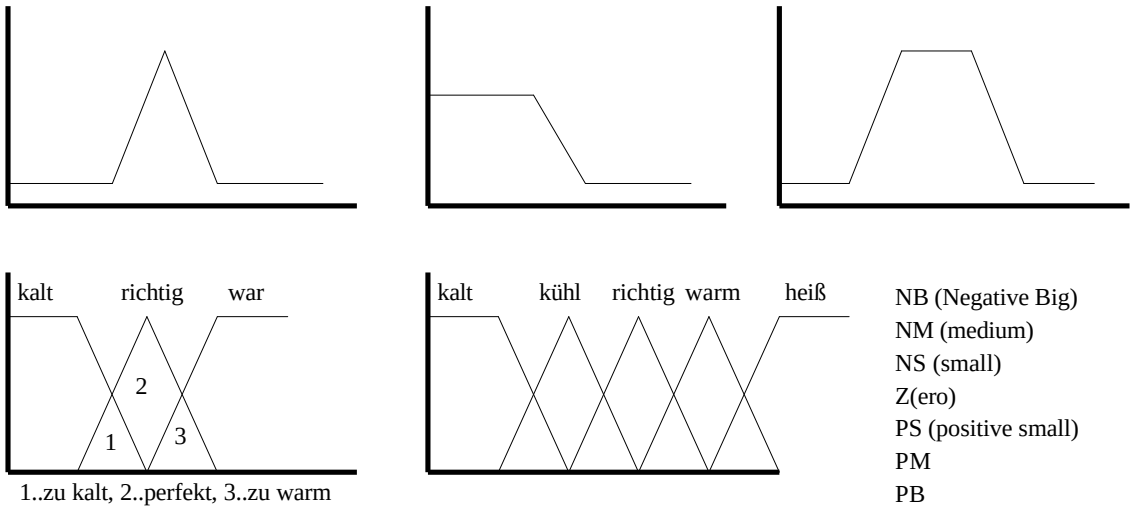
Diese Funktion besitzt einen Übergang von wahr auf falsch und besteht nicht nur aus 0 und 1.

Derartige Funktionen sind allerdings schwer zu berechnen daher werden diese Funktionen folgendermaßen dargestellt:



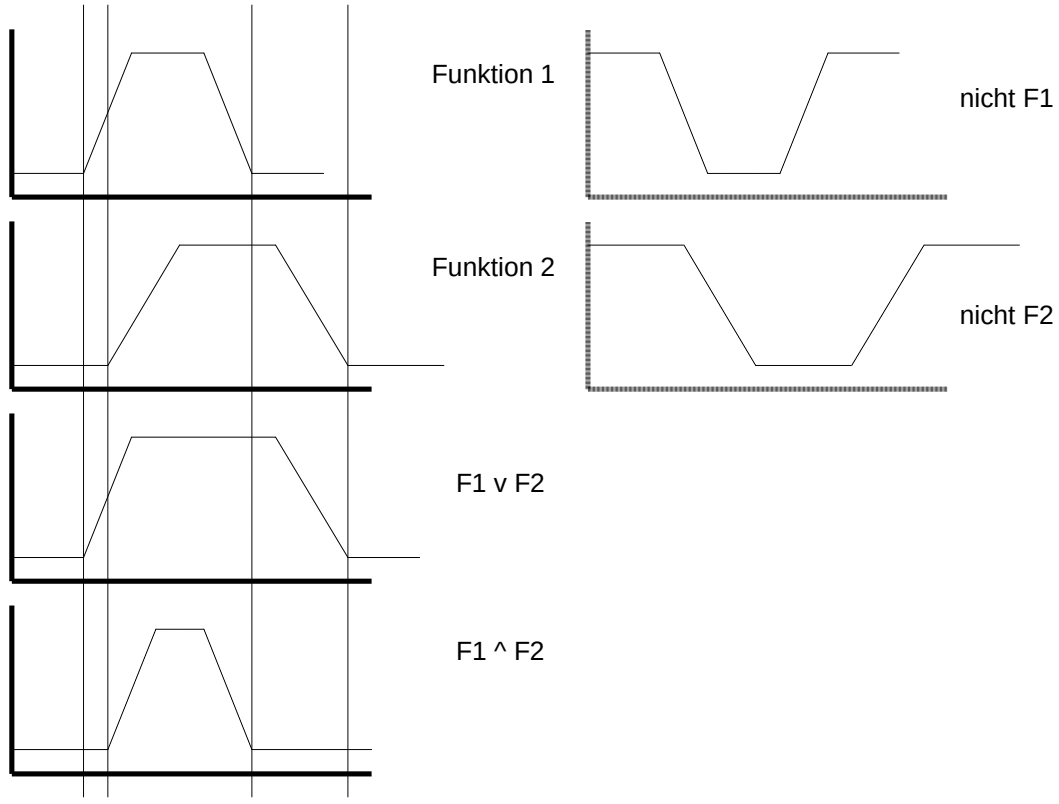
Diese Funktion ist stetig aber nicht stetig differenzierbar (dh. in den sind keine Unstetigkeits-Stellen also keine Sprünge z.B. Steigungsfolge 0,2,0,-2,0,2..... die erste Grafik ist stetig differenzierbar)

Zugehörigkeitsfunktionen



1..zu kalt, 2..perfekt, 3..zu warm

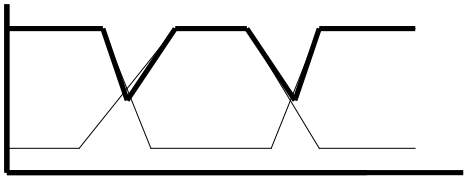
Wie aber stellt man eine v bzw. ^ Verknüpfung dar?



$f_{F1\ v\ F2} = MAX(f_1,f_2)$
$f_{F1\ ^\ F2} = MIN(f_1,f_2)$

Ein Singleton geschnitten mit einer Menge ist das Singleton wenn es innerhalb der Menge liegt und nicht das Singleton wenn es außerhalb der Menge liegt.

Vereinigungsmenge:



f_1 ^ nicht f_1 => 0
f_1 v nicht f_1 < 1 im Gegensatz zur scharfen Menge

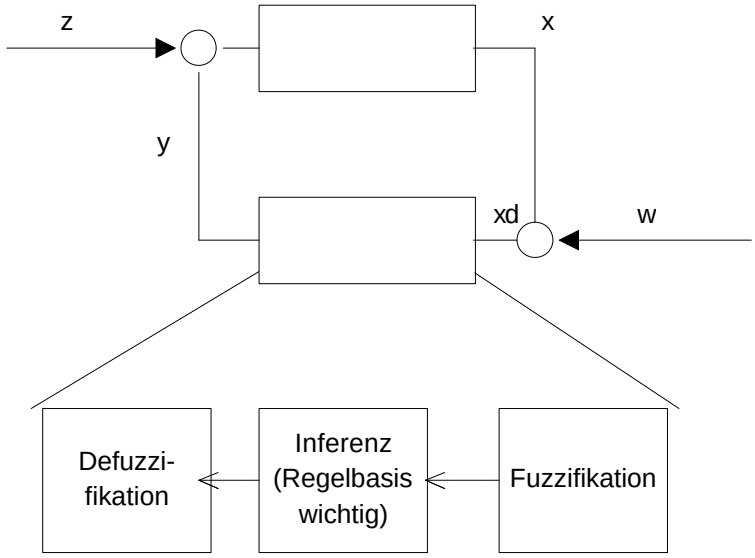
Noch einmal zurück zur kleinen Datenbank. Welches Ergebnis würde die gleiche Abfrage nun liefern?

FIRMA	U	P	U ^ P
A	0	0	0
B	0	0,33	0
C	0,36	0,83	0,36
D	0,45	0	0
E	0,55	1	0,55
F	0,73	0,5	0,5
G	0,91	0,33	0,33
H	1	0,17	0,17
I	1	0	0
J	1	0	0

Hier würden wir folgende Unternehmen bekommen (absteigend gereiht):
E, F, C, G, H

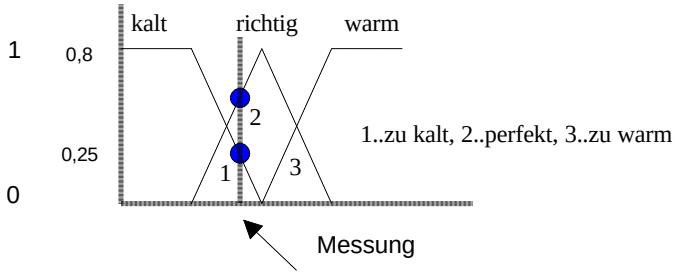
- Nicht soviele Fuzzy Sets/linguistische Variablen
- Überschneidungen großzügig
- Großer Bereich abgedeckt

<u>Wie funktioniert ein Fuzzy Regler?</u>



Ein scharfer Wert kommt als Eingabe in den Regler. Dieser wird in eine linguistische Form umgeformt werden (Fuzzifikation), eine linguistische Variable besteht aus mehreren Fuzzysets. Die Inferenz könnte z.B. sein: „Wenn es kalt muß geheizt werden“.
Am Ende muß wieder ein scharfer Wert erreicht werden (Defuzzifikation).

FUZZIFIZIERUNG



Diese Messung liefert einen Crisp-Wert. Wie aber drückt man nun aus, daß ein Crisp Wert zu zwei Mengen gehört?
Dazu wird die Zugehörigkeit des Wertes zu den einzelnen Mengen untersucht:
Zur Menge KALT hat dieser Wert eine Zugehörigkeit von z.B. 0,25
Zur Menge RICHTIG hat dieser Wert eine Zugehörigkeit von z.B. 0,8
Zur Menge WARM hat dieser Wert eine Zugehörigkeit von z.B. 0

Die Fuzzifizierung ist damit durchgeführt.

INFERENZ

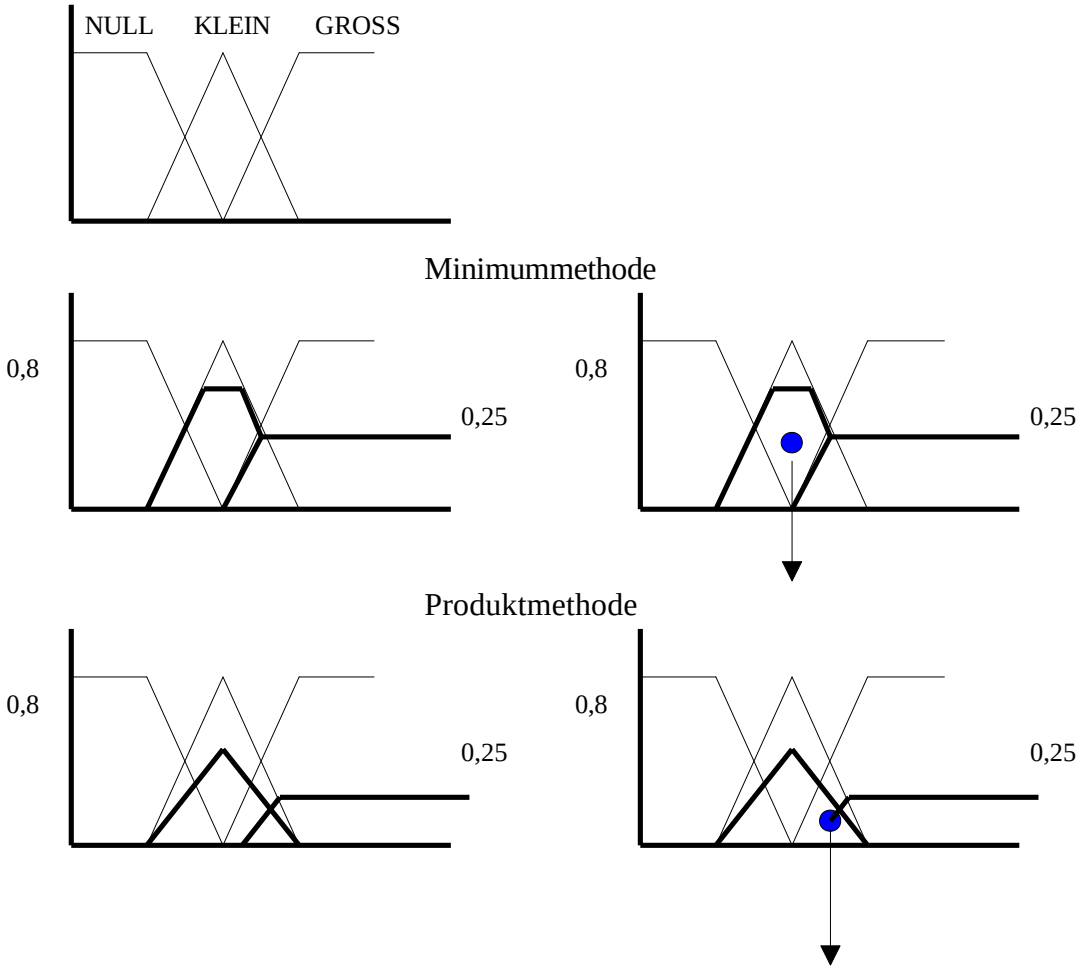
TEMPERATUR	***HEIZLEISTUNG***	***WAHRHEITSWERT DER REGEL***	***BEIM OBIGEN BEISPIEL***
KALT	GROSS	1	0,25
RICHTIG	KLEIN	1	0,8

WARM	NULL	1	0

Bei der Inferenz können auch Regeln aus der Praxis einfließen.

DEFUZZIFIKATION

Zur Defuzzifikation können verschiedene Methoden eingesetzt werden.
Zum Beispiel die Minimummethode oder die Produktmethode.



Der Unterschied zwischen Minimum und Produktmethode ist, daß bei der Minimummethode bei (in unserem Beispiel) 0,8 abgeschnitten wird, bei der Produktmethode nur die Spitze des Dreiecks auf 0,8 verlegt und somit nur verkleinert wird. Bei beiden Methoden werden dann die entstehenden Flächen addiert. Beide Methoden arbeiten mit der COG (Center of Gravity) der Schwerpunkts-Methode (dabei wird der Schwerpunkt der Gesamtfläche ermittelt).

ANDERE FUZZY OPERATOREN

	t-Norm	**s-Norm**	
Minimum	$f_3 = \min(f_1,f_2)$	$f_3 = \max(f_1,f_2)$	**Maximum**
Algebraisches Produkt	$f_3 = f_1 * f_2$	$f_3 = f_1 + f_2 - f_1 * f_2$	**Algebraische Summe (<1)**
Gewichtete Differenz	$f_3 = \max(0,f_1+f_2-$	$f_3 = \min(1,f_1+f_2)$	**Gewichtete Summe**

	1)		
	$f_3 = f_1 \wedge f_2 = t(f_1,f_2)$	$f_3 = f_1 \vee f_2 = s(f_1,f_2)$	

FUZZY UND

$u_c = y*\min(u_A,u_B) + (1-y)* \frac{1}{2} * (u_A+u_B)$

FUZZY ODER

$u_c = y*\max(u_A,u_B) + (1-y)* \frac{1}{2} * (u_A+u_B)$

y...Parametrisierung im Bereich 0..1

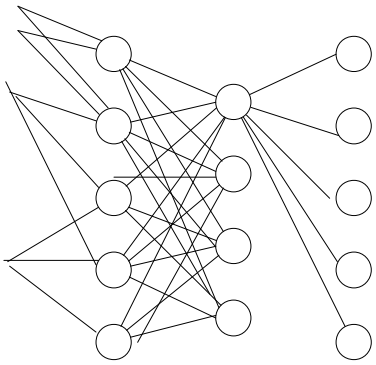
PARAMETRIERTE KOMBINATION

$u_c = y*\min(u_A,u_B) + (1-y)* \max(u_A,u_B)$

wenn y..1 dann FUZZY UND
wenn y..0 dann FUZZY ODER

8 Neuronale Netze

Ein neuronales Netz besteht aus Neuronen die in mehreren Schichten verteilt sind.
Aus mehreren Eingangswerten macht ein Neuron einen Ausgangswert.



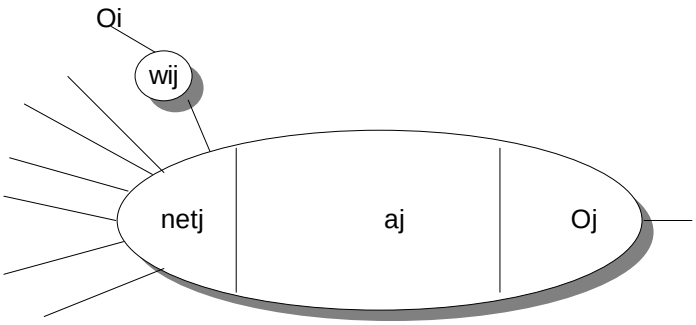
Die, in der Grafik, mittlere Schicht wird als „hidden layer“ oder auch „versteckte Schicht“ bezeichnet. Eine Layer ist also die Anzahl der „Spalten“ und nicht die der Zeilen. Je mehr Schichten ein neuronales Netz hat, desto besser wird sein Lernvermögen.

Wie schon aus der Grafik ersichtlich wird aus einer Menge von Eingabewerten 1 Ausgabewert erzeugt, der dann (gewichtet) weiteren Schichten (und dort Neuronen) zur Verfügung gestellt wird.

Die Eingangsfunktion für neuronale Netze ist:

$net_j = SUMME(w_{ji}O_i)$

net_j...Nettoinputfunktion
w_{ji}...Gewichtungen
O_i...Outputfunktion des vorigen



a_j..Aktivierungsfunktion
oft ist die Aktivierungsfunktion der Einfachheit halber mit der Nettoinputfunktion gleichgesetzt

$$a_j = net_j$$

Im allgemeinen kann aber gesagt werden $\mathbf{a_j = f(net_j)}$
$O_j = f_1(a_j)$

Zu aller erst werden alle (gewichteten) Eingangswerte aufsummiert (oder das Produkt der Eingangswerte gebildet, oder das Minimum,..).

LERNEN IN NEURONALEN NETZEN

Lernen bedeutet in diesem Fall : Gewichte anpassen.

Es gibt 2 Arten von Lernen:

1. Überwachtes Lernen
2. Nicht überwachtes Lernen

ad 1)
Das Ergebnis muß wieder mit einem vorgegebenen Muster verglichen werden.
ad 2)
Hier werden Dinge „unbewußt“ gelernt.

III Datenblätter, Bedienungsanleitungen, ...

1 Rule

1.1 Allgemeines

RULE (RechnerUnterstützter LeiterplattenEntwurf) ist ein kleines CAD-System zum Entwickeln von Platinen.

1.2 Tasten

↓↑→←	Cursortasten	Cursor (Fadenkreuz) bewegen
␣	Leertaste	Schnellmodus ein/aus
↵	RETURN	Menüpunkt wählen
Esc	ESC	Abbruch, Modus beenden, Menü
Delete	DEL	Markierte Elemente löschen
Insert	INS	Bauteil einfügen
←(BS)	BACKSPACE	letzen Baulteil löschen
Page Up	PAGE UP	Maßstab verkleinern
Page Down	PAGE DOWN	Maßstab vergrößern
Home	HOME	Bezugspunkt setzen
1	1	Lötpunkt setzen
2	2	Leiterbahn zeichnen
3	3	Dreieck zeichnen
4	4	Recheck zeichnen
9	9	Kreis zeichnen
0	0	Zentrierungsbohrung
␣	ß	Schriftmodus
F1	F1	Speichern
F2	F2	Laden
F3	F3	Drucken
F4	F4	Zoll ja/nein
F5	F5	Raster ja/nein
F6	F6	Fangen aus/ein (Cursor am Raster ausrichten)
F7	F7	Schneller Bildaufbau aus/ein
F8	F8	Winkelfangen aus/ein (Leiterbahnen nur in 45°-Raster)
F9	F9	Automatisches Panorama aus/ein (Randpunkt des Bildschirms wird neuer Mittelpunkt)
F10	F10	Ende
A	Ä	Element ändern
D	D	Element drehen

F	F	Fenster markieren
K	K	Element kopieren
N	N	Neuzeichnen
P	P	Panoramafunktion
S	S	Element spiegeln
V	V	Element verschieben
W	W	Element wählen
Z	Z	Knickpunkte ziehen

1.3 Menü

INTERFACE
- Benennen — Zeichnung mit Namen versehen
- Speichern — Zeichnung speichern
- Laden — Zeichnung laden (bisherige wird aus dem Speicher gelöscht)
- Drucken — Zeichnung ausdrucken
- Bildschirm — Bildschirmdaten eingeben (um am Schirm 1:1 Bild zu erhalten)
- Pfade — Zugriffspfade für das Programm festlegen
- Ende — Rule beenden

BAUTEILE
- Bibliothek — auswählen der Bibliothek
- Import — Aus der Bibliothek auswählen
- Export — Bauteil in die Bibliothek übertragen
- Neu — Leere Bibliothek anlegen
- Reorganisation — Bibliothek neu anordnen

OPTIONEN
- Leiterbahn — Einstellungen der Leiterbahnen
- Schrift — Schriftart, - höhe und -breite
- Recheck — Ebene für Rechecke
- Dreieck — Ebene für Dreiecke
- Kreis — Ebene für Kreise
- Punkt — Einstellungen für Lötaugen
- Zentrierungen — Einstellungen für Zentrierungen

SYSTEM
- Ebenen — Parameter für die einzelnen Ebenen
  - 0 — CU-unten
  - 1..10 — CU
  - 11 — Schrift
  - 12 — Bauteile
  - 13 — Lötstopmaske unten
  - 14 — Lötstopmaske oben
  - 15 — Bohrschablone
- Maßstab — Einstellen des Maßstabes
- Platine — Eingabe der Platinengröße
- Spezial — Sonstige Optionen
- Fenster — Fenstereinstellungen
- Info — Resourcenauslastung

1.4 Richtlinien zur Platinenentwicklung

Lötaugen so groß als möglich
Leiterbahnen so breit als möglich
Nicht mehr als eine Leiterbahn zwischen zwei Lötaugen
Bohrlöcher sehr klein halten (nur zum Ansetzen des Bohrers)

2 Logikbausteine

2.1 Allgemeines zur 74/54-Serie

Namensgebung: X4[x[x[x[x]]]]nn[n[n]]

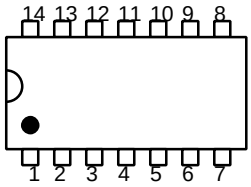
mit
- X 5 oder 7 für die Serie (7=Normal, 5=Military)
- xxxx Bauart (z.B.: S, L, LS, ALS, ...)
- nnn Typ (z.B.: 00 ... 4 NANDs)

Die Bauart hat Auswirkungen auf den Stromverbrauch und die Belastbarkeit der Bauteile aber nicht auf die Pinbelegung.

Temperaturbereich:
- 54xxx -55°C bis +125°C
- 74xxx 0°C bis +70°C

Versorgungsspannung:
- 54xxx 4,5 V bis 5,5 V
- 74xxx 4,75 V bis 5,25 V

Packung: meist DIP ... Dual Inline Package mit 14, 16, ... Pins (PIN...Anschluß)



Reihenabstand 3/10" oder 6/10"

Pinabstand 1/10"

1/10" (Zoll) = 2,54 mm

Pin 1 wird an der Markierung oder der Kerbe erkannt

„Lesehilfe“ zu den IC-Beschreibungen/Datenbüchern:
Bei IC´s mit einfachen Funktionen (z.B.: NAND) ist keine Wahrheitstafel angegeben, da diese als bekannt vorausgesetzt werden können. In solchen Fällen ist eine charakteristische Gleichung angegeben, um die Belegung der PINs leichter lesen zu können. Bei mehreren gleichen Funktionsgruppen (Gattern) in einem IC sind diese durch eine Ziffer an der ersten Stelle der PIN-Beschreibung unterschieden. In den Wahrheitstafeln bei den IC´s werden die üblichen Symbole aus den Datenbüchern verwendet, die aber von der Funktion abhängig sind und daher hier nicht vollständig aufgeführt sein können. Allgemein gültige Symbole sind:

- H Highzustand (meist identisch mit logisch 1 oder wahr)
- L Lowzustand (meist identisch mit logisch 0 oder falsch)
- X Zustand beliebig (Don´t care)
- ↑ Steigende Flanke (i.A. bei Taktsignalen).

Synonyme für die „Versorgungsspannungen“:

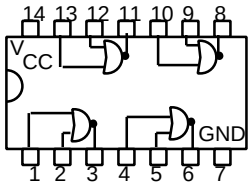
Name	Symbol	Spannung	Kurzsymbol	Pegel	Logisch	Weitere Synonyme
Versorgungsspannung	V_{CC}	5 V	+	High H	1	
Masse	GND	0 V	-	Low L	0	Ground, Erde

Die Zuordnung der logischen Zustände ist willkürlich; in der Praxis ist diese Zuordnung üblich es existiert aber auch die umgekehrte (negative TTL, H...0 und L...1).
Ausgenommen die konkreten Spannungswerte werden diese Synonyme auch außerhalb der TTL verwendet.

2.2 Wichtige Typen

2.2.1 7400 (4 NAND-Gatter mit je 2 Eingängen)

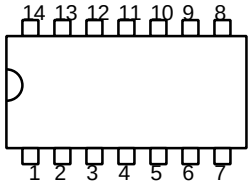
Charakteristische Gleichung: $Y = \neg(A \uparrow B)$



1 ... 1A	8 ... 3Y	7 ... GND
2 ... 1B	9 ... 3B	14 ... V_{CC}
3 ... 1Y	10 ... 3A	
4 ... 2A	11 ... 4Y	
5 ... 2B	12 ... 4B	
6 ... 2Y	13 ... 4A	

2.2.2 7402 (4 NOR-Gatter mit je 2 Eingängen)

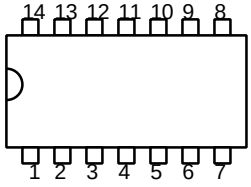
Charakteristische Gleichung: $Y = \neg(A \vee B)$



1 ... 1Y	8 ... 3A	7 ... GND
2 ... 1A	9 ... 3B	14 ... V_{CC}
3 ... 1B	10 ... 3Y	
4 ... 2Y	11 ... 4A	
5 ... 2A	12 ... 4B	
6 ... 2B	13 ... 4Y	

2.2.3 7404 (6 NOT-Gatter)

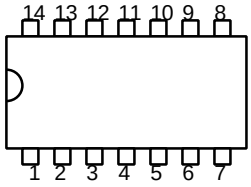
Charakteristische Gleichung: $Y = \neg A$



1 ... 1A	8 ... 4Y	7 ... GND
2 ... 1Y	9 ... 4A	14 ... V_{CC}
3 ... 2A	10 ... 5Y	
4 ... 2Y	11 ... 5A	
5 ... 3A	12 ... 6Y	
6 ... 3Y	13 ... 6A	

2.2.4 7408 (4 AND-Gatter mit je 2 Eingängen)

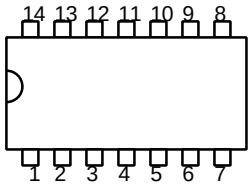
Charakteristische Gleichung: $Y = A \wedge B$



1 ... 1A	8 ... 3Y	7 ... GND
2 ... 1B	9 ... 3B	14 ... V_{CC}
3 ... 1Y	10 ... 3A	
4 ... 2A	11 ... 4Y	
5 ... 2B	12 ... 4B	
6 ... 2Y	13 ... 4A	

2.2.5 7411 (3 AND-Gatter mit je 3 Eingängen)

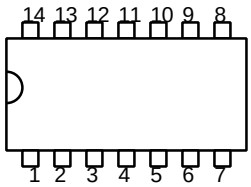
Charakteristische Gleichung: $Y = A \wedge B \wedge C$



1 ... 1A	8 ... 3Y	7 ... GND
2 ... 1B	9 ... 3C	14 ... V_{CC}
3 ... 2A	10 ... 3B	
4 ... 2B	11 ... 3A	
5 ... 2C	12 ... 1Y	
6 ... 2Y	13 ... 1C	

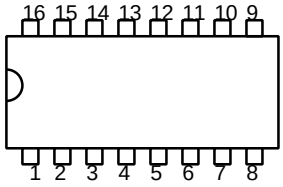
2.2.6 7432 (4 OR-Gatter mit je 2 Eingängen)

Charakteristische Gleichung: $Y = A \vee B$



1 ... 1A	8 ... 3Y	7 ... GND
2 ... 1B	9 ... 3B	14 ... V_{CC}
3 ... 1Y	10 ... 3A	
4 ... 2A	11 ... 4Y	
5 ... 2B	12 ... 4B	
6 ... 2Y	13 ... 4A	

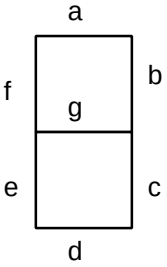
2.2.7 7448 (BCD zu 7-Segment Dekoder/Anzeigentrieber)



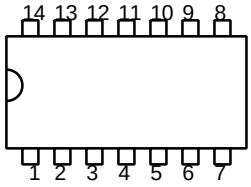
1 ... B	9 ... e	8 ... GND
2 ... C	10 ... d	16 ... V_{CC}
3 ... Lamptest	11 ... c	
4 ... BI/RBO	12 ... b	
5 ... RBI	13 ... a	
6 ... D	14 ... g	
7 ... A	15 ... f	

Wert	Eingänge						RBO	Ausgänge						
/Fkt.	LT	RBI	D	C	B	A	/BI	a	b	c	d	e	f	g
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L
BI	X	X	X	X	X	X	L	L	L	L	L	L	L	L
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L
LT	L	X	X	X	X	X	H	H	H	H	H	H	H	H

Segmentnamen einer 7-Segment-Anzeige:



2.2.8 7474 (2 Flipflops mit Preset und Clear)

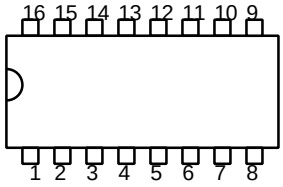


1 ... 1¬CLR	8 ... 2¬Q	7 ... GND
2 ... 1D	9 ... 2Q	14 ... V_{CC}
3 ... 1CLK	10 ... 2¬PRE	
4 ... 1¬PRE	11 ... 2CLK	
5 ... 1Q	12 ... 2D	
6 ... 1¬Q	13 ... 2¬CLR	

Eingänge				Ausgänge	
¬PRE	¬CLR	CLK	D	Q	¬Q
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	instabil	instabil
H	H	↑	H	H	L
H	H	↑	L	L	H

H	H	L	X	Q_0	$\neg Q_0$

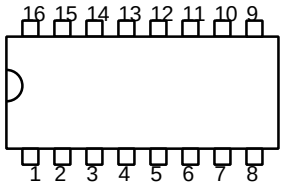
2.2.9 74109 (2 J-¬K-Flipflops mit Preset und Clear)



1 ... 1¬CLR	9 ... 2¬Q	8 ... GND
2 ... 1J	10 ... 2Q	16 ... V_{CC}
3 ... 1¬K	11 ... 2¬PRE	
4 ... 1CLK	12 ... 2CLK	
5 ... 1¬PRE	13 ... 2¬K	
6 ... 1Q	14 ... 2J	
7 ... 1¬Q	15 ... 2¬CLR	

Eingänge					Ausgänge	
¬PRE	¬CLR	CLK	J	¬K	Q	¬Q
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	instabil	instabil
H	H	↑	L	L	L	H
H	H	↑	H	L	Toggle	Toggle
H	H	↑	L	H	Q_0	$\neg Q_0$
H	H	↑	H	H	H	L
H	H	L	X	X	Q_0	$\neg Q_0$

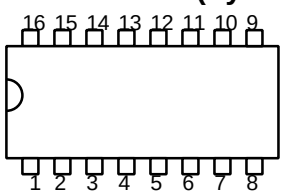
2.2.10 74138 (3-Bit-Binärdekoder/Demultiplexer, 3 zu 8)



1 ... A	9 ... Y6	8 ... GND
2 ... B	10 ... Y5	16 ... V_{CC}
3 ... C	11 ... Y4	
4 ... ¬G2A	12 ... Y3	
5 ... ¬G2B	13 ... Y2	
6 ... G1	14 ... Y1	
7 ... Y7	15 ... Y0	

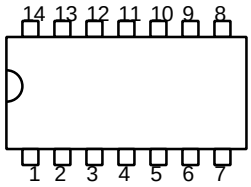
Eingänge					Ausgänge							
Enable		Select										
G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

2.2.11 74160 (Synchroner programmierbarer Dezimalzähler mit Clear)



1 ... ¬CLR	9 ... ¬LOAD	8 ... GND
2 ... CLK	10 ... Enable T	16 ... V_{CC}
3 ... A	11 ... Q_D	
4 ... B	12 ... Q_C	
5 ... C	13 ... Q_B	
6 ... D	14 ... Q_A	
7 ... Enable P	15 ... RCO	

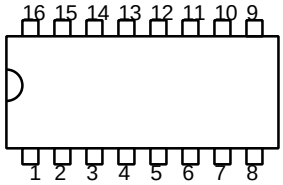
2.2.12 74164 (8-Bit Schieberegister mit Parallelausgabe und Clear)



1 ... A
2 ... B
3 ... Q_A
4 ... Q_B
5 ... Q_C
6 ... Q_D
8 ... CLK
9 ... ¬CLR
10 ... Q_E
11 ... Q_F
12 ... Q_G
13 ... Q_H
7 ... GND
14 ... V_{CC}

Eingänge				Ausgänge			
¬CLR	CLK	A	B	Q_A	Q_B	...	Q_H
L	X	X	X	L	L	...	L
H	L	X	X	Q_{A0}	Q_{B0}	...	Q_{H0}
H	↑	H	H	H	Q_{An}	...	Q_{Gn}
H	↑	L	X	L	Q_{An}	...	Q_{Gn}
H	↑	X	L	L	Q_{An}	...	Q_{Gn}

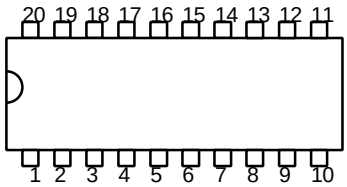
2.2.13 74165 (8-Bit Schieberegister mit Paralleleingang)



1 ... Shift/¬Load
2 ... CLK
3 ... E
4 ... F
5 ... G
6 ... H
7 ... ¬Q_H
9 ... Q_H
10 ... Serial IN
11 ... A
12 ... B
13 ... C
14 ... D
15 ... CLK Inhibit
8 ... GND
16 ... V_{CC}

Eingänge					Interne Ausgänge		Ausgang
Shift/ ¬Load	Clock Inhibit	Clock	Serial	Parallel A..H	Q_A	Q_B	Q_H
L	X	X	X	a..h	a	b	h
H	L	L	X	X	Q_{A0}	Q_{B0}	Q_{H0}
H	L	↑	H	X	H	Q_{An}	Q_{Gn}
H	L	↑	L	X	L	Q_{An}	Q_{Gn}
H	H	X	X	X	Q_{A0}	Q_{B0}	Q_{H0}

2.2.14 74373 (8-Bit D-Latch)

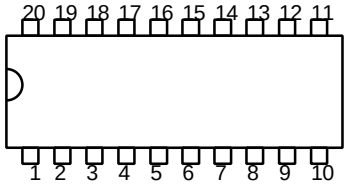


1 ... ¬OC
2 ... 1Q
3 ... 1D
4 ... 2D
5 ... 2Q
6 ... 3Q
7 ... 3D
8 ... 4D
9 ... 4Q
10 ... GND
11 ... Enable G
12 ... 5Q
13 ... 5D
14 ... 6D
15 ... 6Q
16 ... 7Q
17 ... 7D
18 ... 8D
19 ... 8Q
20 ... V_{CC}

¬OC	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

OC...Output Control
Z...Hochohmiger Zustand (High Impedance State)

2.2.15 74573 (8-Bit D-Latch)

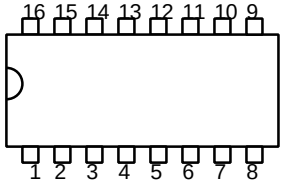


1 ... ¬OC	6 ... 5D	11 ... Enable G	16 ... 4Q
2 ... 1D	7 ... 6D	12 ... 8Q	17 ... 3Q
3 ... 2D	8 ... 7D	13 ... 7Q	18 ... 2Q
4 ... 3D	9 ... 8D	14 ... 6Q	19 ... 1Q
5 ... 4D	10 ... GND	15 ... 5Q	20 ... V_{CC}

¬OC	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

OC...Output Control
Z...Hochohmiger Zustand (High Impedance State)

2.2.16 74595 (8-Bit Schieberegister mit „Output-Latches“)



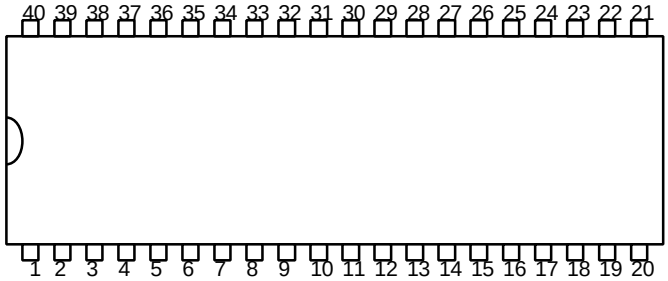
1 ... Q_B	9 ... Q_H´	8 ... GND
2 ... Q_C	10 ... ¬SCLR	16 ... V_{CC}
3 ... Q_D	11 ... SCK	
4 ... Q_E	12 ... RCK	
5 ... Q_F	13 ... ¬G	
6 ... Q_G	14 ... SER	
7 ... Q_H	15 ... Q_A	

RCK	SCK	¬SCLR	¬G	Funktion
X	X	X	H	Q_A bis Q_H sind Hochohmig
X	X	L	L	Schieberegister wird gelöscht, Q_H´=0
X	↑	H	L	Schieberegister aktiv (Q_N=Q_{N-1}, Q_0=SER)
↑	X	H	L	Schieberegister wird in das Output Latch kopiert

SCLR...Shift Register Clear
SCK...Shift Register Clock
RCK...Latch Register Clock
SER...Serial In
↑...Steigende Flanke

Q_A...Q_H Ausgänge des Output Latch
Q_H´ Ausgang des Schieberegister (zum Kaskadieren)

2.2.17 8255 (24-Bit PIO(Parallel-Input-Output)-Schnittstelle)

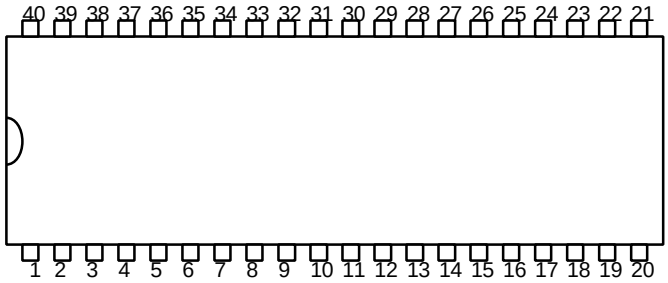


1 ... PA0	11 ... PC6	21 ... PB3	31 ... D3
2 ... PA1	12 ... PC5	22 ... PB4	32 ... D2
3 ... PA2	13 ... PC4	23 ... PB5	33 ... D1
4 ... PA3	14 ... PC0	24 ... PB6	34 ... D0
5 ... ¬RD	15 ... PC1	25 ... PB7	35 ... RESET
6 ... ¬CS	16 ... PC2	26 ... V_{CC}	36 ... ¬WR
7 ... GND	17 ... PC3	27 ... D7	37 ... PA7
8 ... A1	18 ... PB0	28 ... D6	38 ... PA6
9 ... A0	19 ... PB1	29 ... D5	39 ... PA5
10 ... PC7	20 ... PB2	30 ... D4	40 ... PA4

Control Word

D7	Mode Set Flag (1=Active)	
D6	Group A	Mode Selection High
D5		Mode Selection Low (00=Mode 0; 01=Mode 1; 1X=Mode 2)
D4		Port A (0=Output; 1=Input)
D3		Port C Upper (0=Output; 1=Input)
D2	Group B	Mode Selection (0=Mode 0; 1=Mode 1)
D1		Port B (0=Output; 1=Input)
D0		Port C Lower (0=Output; 1=Input)

3 8052AH



1 ... T2/P1.0	11 ... Serial Out	21 ... A8	31 ... +/¬EA
2 ... T2EX/P1.1	12 ... ¬DMA Req	22 ... A9	32 ... AD7
3 ... PWM/P1.2	13 ... ¬INT1	23 ... A10	33 ... AD6
4 ... P1.3	14 ... T0	24 ... A11	34 ... AD5
5 ... P1.4	15 ... T1	25 ... A12	35 ... AD4
6 ... ¬PE/P1.5	16 ... ¬WR	26 ... A13	36 ... AD3
7 ... P1.6	17 ... ¬RD	27 ... A14	37 ... AD2
8 ... P1.7	18 ... XTAL2	28 ... A15	38 ... AD1
9 ... RESET	19 ... XTAL1	29 ... ¬PSEN	39 ... AD0
10 ... Serial In	20 ... V_{SS}	30 ... ALE	40 ... V_{CC}

4 MCS-BASIC (für 8052AH)

4.1 Editor

Das MCS-Basic enthält nur einen minimalen Zeileneditor; d.h. eine eingegebene Zeile kann nur durch Neueingabe geändert werden. Während der Eingabe einer Zeile kann mittels <DEL> das zuletzt eingegebene Zeichen und mittels <Ctrl>-D die gesamte Zeile gelöscht werden.

4.2 Variablenkonzept

Variablennamen können bis zu 8 Zeichen (Buchstaben, Ziffern (nicht an erster Stelle) und Unterstreichung) lang sein, dabei darf allerdings innerhalb eines Variablennamens kein Schlüsselwort enthalten sein.
Felder können nur eindimensional und bis maximal 254 Elemente angelegt werden; für den Index werden dabei runde Klammern verwendet. Der benötigte Speicherplatz für Felder beträgt 6*(#Elemente+1).

Integer (2 Byte)
Ganze Zahlen können Hexadezimal oder Dezimal im Bereich von 0 bis 65535 angegeben werden, sie müssen allerdings mit einer Ziffer beginnen (65535 oder 0FFFFH, 40H oder 64,...)

Real (8 Byte)
Der Gleitkommabereich beträgt: ±1E-127 bis ±.99999999E+127

Strings
MCS-Basic kennt nur ein eindimesionales Stringfeld mit maximal 255 verschiedenen Zeichenketten gleicher Länge, die mit $(0) bis $(254) angesprochen werden. Der Speicherplatz muß dafür vorher mit dem STRING-Befehl reserviert werden.

4.3 Operatoren und Funktionen

+	Addition	=	gleich
-	Subtraktion	<>	ungleich
*	Multiplikation	>	größer
/	Division	>=	größer gleich
**	Potenzierung (max. hoch 255)	<	kleiner
		<=	kleiner gleich

.AND.	Logisches Und
.OR.	Logisches Oder
.XOR.	Logisches Exclusiv-Oder

ABS(Ausdruck)	Absolutbetrag
ASC(Zeichen)	ASCII-Wert von Zeichen
ASC(str [,stelle])	ASCII-Wert des Zeichens an der Stelle stelle im String str
ASC(str [,stelle])=Wert	Ersetzen des Zeichens an der Stelle stelle im String str durch CHR(Wert) Zeichen mit ASCII-Code Wert
ATN(Ausdruck)	Arcus-Tangens
CHR(Ausdruck)	Zeichen mit ASCII-Code Ausdruck
COS(Ausdruck)	Cosinus
EXP(Ausdruck)	e hoch Ausdruck (mit e=2.7182818)
INT(Ausdruck)	ganzahliger Anteil
LOG(Ausdruck)	Natürlicher Logarithmus
NOT(Ausdruck)	Einer-Komplement (bitweises invertieren)
PI	Konstante (3.1415926)
RND(Ausdruck)	Pseudozufallszahl zwischen 0 und 1 (nach 65535 Aufrufen Wiederholung)
SGN(Ausdruck)	+1 wenn Ausdruck >=0, sonst -1
SIN(Ausdruck)	Sinus
SQR(Ausdruck)	Quadratwurzel von positiven Zahlen
TAN(Ausdruck)	Tangens

4.4 Spezielle Operatoren und Funktionen

CBY(Ausdruck)	Inhalt der Speicherstelle Ausdruck im Programmspeicher
DBY(Ausdruck)	Inhalt der Speicherstelle Ausdruck im internen Datenspeicher
DBY(Addr)=wert	Ändert Speicherstelle Addr im internen Datenspeicher auf wert
FREE	Freier Speicherplatz für Programme (FREE=MTOP-LEN-511)
GET	Wert des Zeichens vom Eingabegerät (0=Kein Zeichen)
IE	IE-Register der 8052-CPU
IP	IP-Register der 8052-CPU
LEN	Wieviel Speicherplatz verbraucht das selektierte Program
MTOP	Speicherplatz für MCS-Basic (oberhalb von MTOP wird kein Speicher verwendet)
PORT1	I/O-Port 1 der 8052-CPU
PCON	PCON-Register der 8052-CPU
RCAP2	16Bit-Wert der Register RCAP2H:RCAP2L
T2CON	T2CON-Register der 8052-CPU
TCON	TCON-Register der 8052-CPU
TIME	Vorkommaanteil des Werts der Realtimeclock in Sekunden (Nachkommaanteil mit DBY(47H); Genauigkeit 5ms)
TIMER0	16Bit-Wert des internen Timers 0 TH0:TL0
TIMER1	16Bit-Wert des internen Timers 1 TH1:TL1
TIMER2	16Bit-Wert des internen Timers 2 TH2:TL2
TMOD	TMOD-Register der 8052-CPU
XBY(Ausdruck)	Inhalt der Speicherstelle Ausdruck im externen Datenspeicher
XBY(Addr)=wert	Ändert Speicherstelle Addr im externen Datenspeicher auf wert
XTAL	Wert der Quarzfrequenz (Default: 11059200)

4.5 Kommandos

CONT	Fortsetzung des Programmes nach STOP oder <Ctrl>-C
FPROG1..FPROG6	Siehe PROG1,,PROG6 allerdings wird der Intel Intelligent Eprom Programmier-Algorithmus verwendet
LIST	Ausgabe des Programmtextes auf der Konsole
LIST#	Ausgabe des Programmtextes auf dem Drucker
LIST@	Ausgabe des Programmtextes über eigenen Treiber, dabei muß der Treiber ab Adresse 403CH stehen und das Bit 7 an 27H gesetzt sein (DBY(27H)=DBY(27H).OR.80H)
NEW	Löscht das Programm im RAM, setzt alle Variablen, Strings und Interrupts zurück aber nicht die Realtimeclock, die Stringreserviserung und den internen Stackpointer
NULL [integer]	Anzahl der Nullbytes nach Ausgabe eines <CR>, 0<integer<255
PROG	Selektiertes Programm wird in das EPROM kopiert ("gebrannt")
PROG1	Programmiert die Baudrate der Konsole in das EPROM
PROG2	PROG1und automatischer Start des 1.Programmes beim Einschalten
PROG3	PROG1 und MTOP wird ebenfalls in das EPROM prgrammiert
PROG4	PROG2+PROG3
PROG5	PROG3 und Einlesen der externen Speicherstelle 5FH und 5EH (5FH enthält 0A5H ... externer Speicher wird nicht gelöscht 5EH enthält 034H ... 1.Programm wird gestartet)
PROG6	PROG5 und Starten eines Assemblerprogrammes an Adresse 4039H nach Rückkehr aus dem Assemblerprogramm 3 Möglichkeiten: Carrybit gelöscht ... Autobaudroutine Carrybit gesetzt; Akkumulator=0 ... gespeicherte Baudrate Carrybit gesetzt; Akkumulator<>0 ... 1.Programm starten
RAM	Wahl des Programms im RAM-Speicher
ROM [integer]	Wahl des integer. Programmes im ROM-Speicher
RUN	Zurücksetzen aller Variablen und Interrupts und Start des Programmes mit der ersten Zeile
XFER	Selektiertes Programm wird vom ROM in das RAM kopiert

4.6 Befehle (Statements)

BAUD expr — Setzen der Baudrate für den Druckeranschluß (Default: 1Bd; Datenformat: 1 Startbit, 8 Datenbits, 2 Stopbits)

CALL address — Aufruf eines Assembler-Programmes. (Für address=0.127 wird an Adresse 4100H+2*address, sonst an address verzweigt).

CLEAR — Löschen aller BASIC-Variablen und Zurücksetzen der Stacks und Interrupts.

CLEARI — Setzt alle vom BASIC verwendeten Interrupts zurück.

CLEARS — Setzt alle vom BASIC verwendeten Stacks zurück.

CLOCK0 — Schaltet die Uhr ab

CLOCK1 — Schaltet die Uhr ein (5ms-Ticks, TIME zwischen 0 und 65535.995)

DATA expr [,expr] — Folge von Werten, die durch READ gelesen werden können.

DIM var(dim)[,var(dim)] — Reserviert Speicherplatz für Felder.

DO - UNTIL bedingung — Diese Schleife wird sooft durchgeführt, bis die Bedingung WAHR ist

DO - WHILE bedingung — Diese Schleife wird sooft durchgeführt, bis die Bedingung FALSCH ist

END — Programmende (muß nicht letzte Zeile sein).

FOR var=begin TO end [STEP step]
  befehle

NEXT var — Die Befehle werden sooft durchgeführt, solange var kleiner gleich end ist. Default-Schrittweite=1.

GOSUB zeile — Sprung zur angegebene Zeilennummer
(RETURN = Rücksprung zur Zeile hinter dem GOSUB-Befehl).

GOTO zeile — Sprung zur angegebenen Zeile.

IDLE — Warten auf Interrupt (ONEX1 oder ONTIME), dabei wird der Pin P1.6 des Prozessors auf Low gesetzt.

IF bed THEN befehl
  [ELSE befehl] — Verzweigung

INPUT ["Fragetext"] var {,var,...} — Eingabe(n) während dem Programmablaufs.

LD@ expr — Lädt Realzahl von Adresse expr auf den Argument Stack.

LET var = ausdruck — Die Variable var erhält den Wert des Ausdruckes ausdruck

ON expr GOSUB znr,znr,...

ON expr GOTO znr,znr,... — Sprung zur Zeilennummer znr, die an der expr. Stelle steht.

ONERR znr — Bei nachfolgendem arith. Fehler, erfolgt Sprung in die Zeile znr. Der Fehlercode steht an externer Speicherstelle 257:

Fehlercodes:
- 10.... dividiert durch 0
- 20.... Arith. Overflow
- 30.... Arith. Underflow
- 40.... Bad Argument

ONEX1 znr — Sprung an Zeilennummer znr, wenn nachfolgend der INT1-Pin am 8052H auf 0 gesetzt wird.

ONTIME expr, znr — Sprung an Zeilennummer znr, wenn nachfolgend TIME gleich oder größer als der Ausdruck expr ist.

P. oder ? — siehe PRINT

P.# oder ?# — siehe PRINT#

PGM — Programmiert Eprom von einem Basicprogramm aus, dafür müssen die entsprechenden Parameter im Speicher abgelegt sein:

- 1BH:19H — Startadresse der zu programmierenden Daten
- 1AH:18H — Zieladresse-1 im EPROM
- 1FH:1EH — Anzahl der Bytes, die programmiert werden sollen
- 40H:41H — Programmierimpulsbreite
- 26H,Bit3 — 0=Normale/1="Intelligente" Programmierung

PH0 — Hexadezimale Ausgabe von Zahlen im Bereich von 0..65535 ohne führende Nullen, sonst funktioniert PH0 wie normales Print.

PH1 — wie PH0, allerdings werden für die hexadezimale Ausgabe immer 4 Stellen verwendet.

PHO# — Wie PH0, aber Ausgabe auf den Drucker.

PH1# — Wie PH1, aber Ausgabe auf den Drucker.

PH0@ — Wie PH0, aber Ausgabe über eigenes Treiberprogramm (siehe LIST@).

PH1@ — Wie PH1, aber Ausgabe über eigenes Treiberprogramm (siehe LIST@).

PRINT liste — Ausgabefunktion für Text auf der Konsole

Spezielle Formatierungsbefehle für PRINT:

-TAB(expr)	Zwischenraum bis Spalte expr
- SPC(expr)	expr Leerzeichen werden ausgegeben
- CR	Es wird kein Line Feed durchgeführt
- U.	siehe USING
- USING(Fx)	x: Anzahl der Stellen für Fließkommaformat x=0: Ausgabe nachfolgender Nullen wird unterdrückt
- USING(#.#)	zum Formattieren einer Zahl (#..Ziffer)

PRINT# Wie PRINT, aber Ausgabe auf den Drucker.

PRINT@ Wie PRINT, aber Ausgabe über eigenes Treiberprogramm (siehe LIST@).

PUSH expr Die Ausdrücke nach dem PUSH-Statement werden ausgewertet und werden auf den Argument-Stack geschrieben

POP var Es wird vom Argument-Stack gelesen und in die nachfolgende Variable geschrieben

PWM high, low, periods PULSE WIDTH MODULATION

Dieser Befehl generiert eine vom Benutzer definierten Sequenz am Pin P1.2 (am 2.Bit vom I/O - Anschluß1) des Prozessors

high Anzahl der High-Zyklen (ein Zyklus=1,085µs

low Anzahl der Low-Zyklen

periods Anzahl der Perioden.

READ var {,var} Ordnet den(r) Variable(n) den(die) nächste(n) Wert(e) in der DATA-Liste zu

REM Kommentar Kommentar

RESTORE Setzt den internen Lesezeiger (DATA-Zeiger) an den Anfang der Liste

RETURN Springt zum letzten verwendeten GOSUB-Statement zurück

RETI Es wird benutzt, um aus allen verwendeten Interrupts auszusteigen

RROM [integer] Startet Program mit der Nummer integer aus dem EPROM, dabei Variablen und Basicinterrupts gelöscht.

ST@ expr Speichert eine Realzahl vom Argument Stack an der Adresse expr.

STOP Nach dem STOP-Statement stopt das Programm und meldet die nächste Zeilennummer. Nach der Eingabe von CONT wird das Programm weiterverarbeitet

STRING [expr1][,expr2] Reservierung von Speicherplatz für Strings und Löschen aller bisher definierten Zeichenketten; expr1 gibt den gesamten für Strings zu verwendenden Speicherplatz an; expr2 die Länge einer Zeichenkette; die Anzahl der Strings ergibt sich aus (expr1-1)/(expr2+1), da für jede Zeichenkette ein zusätzliches Byte und eines insgesamt benötigt wird.

z.B.: STRING 100,10 führt zu 9 Zeichenketten, die mit $(0)..$(8) verwendet werden können. STRING 0,0 hebt die Reservierung wieder auf.

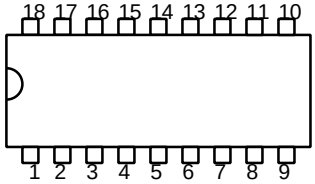
UI0 Umschalten von benutzerspezifischer Eingaberoutine auf Konsole.

UI1 Umschalten auf benutzerspezifische Eingaberoutine an Adresse 4033H.

UO0 Umschalten von benutzerspezifischer Ausgaberoutine auf Konsole.

UO1 Umschalten auf benutzerspezifische Ausgaberoutine an Adresse 4039H.

5 PIC 15C56



1 ... RA2	2 ... RA3	3 ... RTCC	4 ... ¬MCLR	
5 ... V_{SS}	6 ... RB0	7 ... RB1	8 ... RB2	9 ... RB3
10 ... RB4	11 ... RB5	12 ... RB6	13 ... RB7	
14 ... V_{dd}	15 ... OSC2	16 ... OSC1	17 ... RA0	18 ... RA1

6 STAMP-BASIC

6.1 Editor

Der Stamp-eigene Editor läuft am PC und nicht am PIC selbst, prinzipiell kann am PC selbstverständlich jeder ASCII-Editor zum Erstellen der Programme verwendet werden, allerdings wird der Stamp-Editor zum Laden der Programme in den Stamp benötigt. Er wird ähnlich den anderen PC-Fullscreeneditoren bedient und enthält zusätzlich einige Sonderfunktionen, die speziell für die Verwendung mit dem PIC gestaltet sind.
Editiertasten:

Enter	Eingabe und nächste Zeile	Tab	Eingabe und nächste Zeile
↑	Eine Zeile hinauf	↓	Eine Zeile hinunter
→	Ein Zeichen rechts	←	Ein Zeichen links
Ctrl →	Ein Wort rechts	Ctrl ←	Ein Wort links
Home	Zeilenanfang	End	Zeilenende
Page Up	Eine Bildschirmseite hinauf	Page Down	Eine Bildschirmseite hinunter
Ctrl Page Up	Dateianfang	Ctrl Page Down	Dateiende
Shift ↑	Markiere eine Zeile hinauf	Shift ↓	Markiere eine Zeile hinunter
Shift →	Markiere ein Zeichen rechts	Shift ←	Markiere ein Zeichen links
Shift Ctrl →	Markiere ein Wort rechts	Shift Ctrl ←	Markiere ein Wort links
Shift Home	Markiere bis Zeilenanfang	Shift End	Markiere bis Zeilenende
Shift Page Up	Markiere eine Bildschirmseite hinauf	Shift Page Down	Markiere e. Bildschirmseite hinunter
Shift Ctrl Page Up	Markiere bis Dateianfang	Shift Ctrl Page Down	Markiere bis Dateiende
Shift Insert	Markiere Wort beim Cursor	Esc	Aufheben der Markierung
← (BS)	Zeichen nach links löschen	Delete	Zeichen nach rechts löschen
Shift ← (BS)	Bis Zeilenanfang löschen	Shift Delete	Bis Zeilenende löschen
Ctrl ← (BS)	Eine Zeile löschen		
Alt R	Download und Programmstart	Alt Q	Verlassen des Editors
Alt L	Laden einer Datei von Platte	Alt S	Speichern einer Datei auf Platte
Alt X	Markierten Text ausschneiden und in den Zwischenspeicher kopieren	Alt C	Markierten Text in den Zwischenspeicher kopieren
Alt V	Zwischenspeicher einfügen	Alt P	Potentiometer kalibrieren
Alt F	Text suchen	Alt N	Weitersuchen

6.2 Variablenkonzept

Es gibt nur 16 Byte RAM, daher können keine Variablen frei definiert werden, sondern nur fix vorhanden Variable verwendet werden. Um „sprechendere“ Variablennamen zur Verfügung zu haben, können mittels des SYMBOL-Kommandos eigene Variablenamen vorhandenen zugeordnet werden.

B13								B12							
W5															
B11								B10							
W4															
B9								B8							
W3															
B7								B6							
W2															
B5								B4							
W1															
B3								B2							
W0															
B1								B0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORT															
DIRS								PINS							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Die einzelnen Speicherstellen können entweder als 16-Bit-Wortvariable (PORT, W0..W6) oder als Bytevariable (DIRS, PINS, B0..B13) verwendet werden. Die untersten zwei 16-Bit-Worte können auch als Bitvariable verwendet werden, dabei gelten folgende Namen für die Bits:

Bytevariable	Bitvariable
PINS	PIN0 .. PIN7 oder PINS.0 .. PINS.7 oder PORT.0 .. PORT.7
DIRS	DIR0 .. DIR7 oder DIRS.0 .. DIRS.7 oder PORT.8 .. PORT.15
B0	BIT0 .. BIT7 oder B0.0 .. B0.7 oder W0.0 .. W0.7
B1	BIT8 .. BIT15 oder B1.0 .. B1.7 oder W0.8 .. W0.15

Bit-, Byte- und Wortvariablen können gleichzeitig verwendet werden, dabei ist aber immer darauf zu achten, daß nicht versehentlich ein noch benötigter Speicherinhalt überschrieben wird.
Die Variable PORT ist das Ein-/Ausgabe-Wort, daher kann es nur eingeschränkt verwendet werden.
Die PINn sind direkt (gepuffert) mit den Ein-/Ausgabepins des Prozessors verbunden, die DIRn geben die Richtung der PINn an (0=Eingabe, 1=Ausgabe).
Die Variable W6 wird als vierstufiger Stack für GOSUB/RETURN verwendet und steht daher ebenfalls nur eingeschränkt zur Verfügung.
Zahlen werden standardmäßig dezimal angenommen, können aber mit einem „$“ davor auch hexadezimal oder mit einem „%“ davor auch binär verarbeitet werden. Zeichen oder Zeichenketten stehen zwischen doppelten Anführungszeichen (“).

6.3 Operatoren und Funktionen

Operatoren

+	Addition
-	Subtraktion
*	Multiplikation (Niederwertige 16 Bit)
**	Multiplikation (Höherwertige 16 Bit)
/	Division (Quotient)
//	Division (Rest)
MAX	Ergebnis ist kleiner oder gleich
MIN	Ergebnis ist größer oder gleich
&	Logisches Und
|	Logisches Oder
^	Logisches Ausschließendes Oder
&/	Logisches Und Nicht
|/	Logisches Oder Nicht
^/	Logisches Ausschließendes Oder Nicht

Vergleichsoperatoren

=	gleich
<>	ungleich
>	größer
=>	größer gleich
<	kleiner
=<	kleiner gleich

Funktionen werden in STAMP-Basic nicht angeboten, an einigen Stellen kann nur ein „#“ vor einer Variablen oder Konstanten stehen. Dies bewirkt dann eine Konversion von einer Zeichenkette in eine Zahl oder umgekehrt.

6.4 Kommandos

DEBUG cls,"Text",cr,var,$var,%var,#var,#$var,#%var
: Öffnet ein eigenes Fenster im Stampeditor am PC, in dem während des Programmablaufes Variableninhalte angezeigt werden können. „Text“ wird unverändert ausgegeben, „Cls“ Löscht den Inhalt des Debugfensters und „Cr“ beginnt eine neue Zeile. Der Inhalt der angegeben Variablen wird dezimal ausgegeben, ein „$“ vor dem Variablennamen führt zu hexadezimaler, ein „%“ zu binärer Ausgabe. Bei vorangestelltem „#“ unterbleibt die Ausgabe des Variablennamens.

SYMBOL name = variable
SYMBOL name = constant
: Zuweisen eines logischen Namens „name“. Wird vor dem Laden in den PIC in die jeweiligen Werte übersetzt. Dient allein zur besseren Lesbarkeit des Programmes.

EEPROM [location,](data,data...)
: Lädt den Inhalt des EEPROM´s vor dem Laden des Programmes.
  - location 0..255 (Wenn nicht angegeben, dann 0)
  - data 0..255

  Programme werden in obersten Speicherbereich geladen.

6.5 Befehle (Statements)

LOW pin
: Setzt den I/O-Pin pin auf Low (0) und auf Ausgang (mit „pin“ 0..7)

HIGH pin
: Setzt den I/O-Pin pin auf High (0) und auf Ausgang (mit „pin“ 0..7)

TOGGLE pin
: Invertiert den I/O-Pin pin (Low wird zu High, ...) und auf Ausgang (mit „pin“ 0..7)

OUTPUT pin
: Setzt den I/O-Pin pin auf Ausgang (mit „pin“ 0..7)

INPUT pin
: Setzt den I/O-Pin pin auf Eingang (mit „pin“ 0..7)

REVERSE pin
: Invertiert die Richtung (Eingang wird zu Ausgang, ...) des I/O-Pin pin (mit „pin“ 0..7)

BUTTON pin,downstate,delay,rate,bytevariable,targetstate,address
: Liest und entprellt Taste, wobei bei einem angegebenen Zustand verzweigt wird
  - pin I/O-Pin (0..7)
  - downstate Zustand bei gedrückter Taste (0,1)
  - delay Verzögerung bevor Autorepeat aktiv (0 Kein Autorepeat und kein Entprellen, 1..254 Verzögerungszeit, 255 Nur Entprellen)
  - rate Autorepeatrate in BUTTON-cycles (0.255)
  - bytevariable Hilfsvariable (muß an Anfang 0 sein), die vom Befehl benötigt wird
  - targetstate Zielzustand bei dem verzweigt werden soll (0=Taste nicht gedrückt, 1)
  - address Marke zu der verzweigt werden soll

SOUND pin,(note,duration,note,duration...)
: Spielt den angegebenen Ton für die angegebene Dauer
  - pin I/O-Pin (0..7), der mit dem Pluspol eines 10µF-Kondensators verbunden ist, der Minuspol des Kondensators ist mit einem 40 Ω Lautsprecher verbunden, der zweite Anschluß des Lautsprechers liegt auf Masse.
  - note Tonhöhe, 0..Stille, 1..127 aufsteigende Töne, 128..255 aufsteigend mit weißem Rauschen
  - duration Dauer (0..255)

PWM pin,duty,cycles
: Gibt ein PWM-Signal aus, das für die Ausgabe eines Analogsignales verwendet wird

pin I/O-Pin (0..7), der mit einem Widerstand (10 kΩ) verbunden ist; der Widerstand ist mit einem Kondensator (1 µF) verbunden, dessen anderer Anschluß auf Masse liegt. An der Verbindung zwischen Widerstand und Kondensator liegt die analoge Spannung an.
duty Analogspannung (0..255 ➏ 0..5 V)
cycles Zeitdauer (0..255, 1➏ 5ms)

POT pin,scale,variable

Liest einen Widerstand im Bereich von 5 bis 50 kΩ ein
pin I/O-Pin (0..7), der mit dem Widerstand verbunden ist; der Widerstand ist mit einem Kondensator (0,1 µF) verbunden, dessen anderer Anschluß auf Masse liegt.
scale Skalierung des internen 16-Bit Ergebnisses mit scale/256. Die beste Einstellung kann mit Alt-P gefunden werden.
variable Bytevariable, in der das Ergebnis abgelegt wird

PULSOUT pin,period

Generiert einen Impuls
pin I/O-Pin (0..7)
period Zeitdauer (0..65535, 1 ➏ 10 µs)

PULSIN pin,state,variable

Mißt einen Impuls
pin I/O-Pin (0..7)
state Welche Flanke startet die Messung (0,1)
variable gemessene Zeitdauer (1..65535, 1 ➏ 10 µs; 0 wenn der Impuls zu lange ist)

SEROUT pin,baudmode,({#}data,{#}data...)

Serielle Ausgabe (nach RS-232 Methode, allerdings mit 0 und 5 V) mit 8 Datenbits, keiner Parität und einem Stopbit
pin I/O-Pin (0..7)
baudmode Baudrate und Modus

T2400, T1200, T600, T300	Nicht invertierter (true) Ausgang
N2400, N1200, N600, N300	Invertierter Ausgang
OT2400, OT1200, OT600, OT300	Nicht invertierter und „open drain“
ON2400, ON1200, ON600, ON300	Invertierter und „open source“

data Variable oder Konstante, die übertragen werden soll, „#“ davor bewirkt die Umwandlung in ASCII-Codes vor der Übertragung

SERIN pin,baudmode,(qualifier,qualifier...)
SERIN pin,baudmode,(qualifier,qualifier...),{#}variable,{#}variable...
SERIN pin,baudmode,{#}variable,{#}variable...

Serielle Eingabe (nach RS-232 Methode, allerdings mit 0 und 5 V, daher einen 22 kΩ Widerstand zwischen echter serieller Schnittstelle und dem I/O-Pin verwenden) mit 8 Datenbits, keiner Parität und einem Stopbit
pin I/O-Pin (0..7)
baudmode Baudrate und Modus

T2400, T1200, T600, T300	Nicht invertierter (true) Eingang
N2400, N1200, N600, N300	Invertierter Eingang

qualifier Optionale Variablen oder Konstanten, die angeben auf welchen Daten gewartet werden muß, bevor weitergemacht wird
variable Variable, in die empfangen werden soll, „#“ davor bewirkt die Umwandlung von ASCII-Codes in Zahlen.

READ location,variable

Einlesen eines EEPROM-Inhaltes in die Variable
location EEPROM-Adresse (0..255, 255 enthält die höchste freie Adresse -1)
variable Variable, in der Inhalt des EEPROMs kopiert werden soll

WRITE location,data

Füllen eines EEPROM-Inhaltes mit
location EEPROM-Adresse (0..255, Programme werden an die höchsten Adressen kopiert)
data Variable oder Konstante, die in das EEPROM kopiert werden soll

PAUSE milliseconds
Pause für milliseconds Millisekunden
milliseconds 0..65535

{LET} variable = {-}value ?? value ...
Zuweisung, die strikt von links nach rechts ausgeführt wird
variable Variable, in die das Ergebnsi gespeichert wird
?? Operator (+ - * / ** // MIN MAX & | ^ &/ |/ ^/)
value Variable oder Konstante

LOOKUP offset,(data0, data1,..., dataN),variable
Kopiert den durch offset bezeichneten Wert aus der Liste in die Variable
offset Variable oder Konstante, die angibt welcher Wert genommen werden soll (0..N)
data Variable oder Konstante
variable Ergebnis, wenn vorhanden
LOOKDOWN value,(value0,value1...),variable
Sucht den Wert value in der Liste und kopiert den Offset (0..N) in die Variable
value Wert, der mit den Werten aus der Liste verglichen wird
value# (0..N) Variable oder Konstante
variable Ergebnis, wenn vorhanden

RANDOM wordvariable
Genertiert eine Zufallszahl und speichert sie in wordvariable ab

FOR variable = start TO end [STEP [-]increment]
...
NEXT [variable]
Zählschleife
variable Zähler
start Startwert
end Endwert
increment Schrittweite (wenn nicht angegeben, dann 1)

IF variable ?? value {AND/OR variable ?? value ...} THEN address
Vergleich und mögliche Verzweigung
variable Variable, die verglichen werden soll
?? Vergleichsoperator (= <> > < => <=)
value Variable oder Konstante
address Marke, zu der bei wahrer Bedingung verzweigt wird

BRANCH offset,(address0,address1,..., addressN)
Verzweigung zu der von Offset bezeichneten Marke

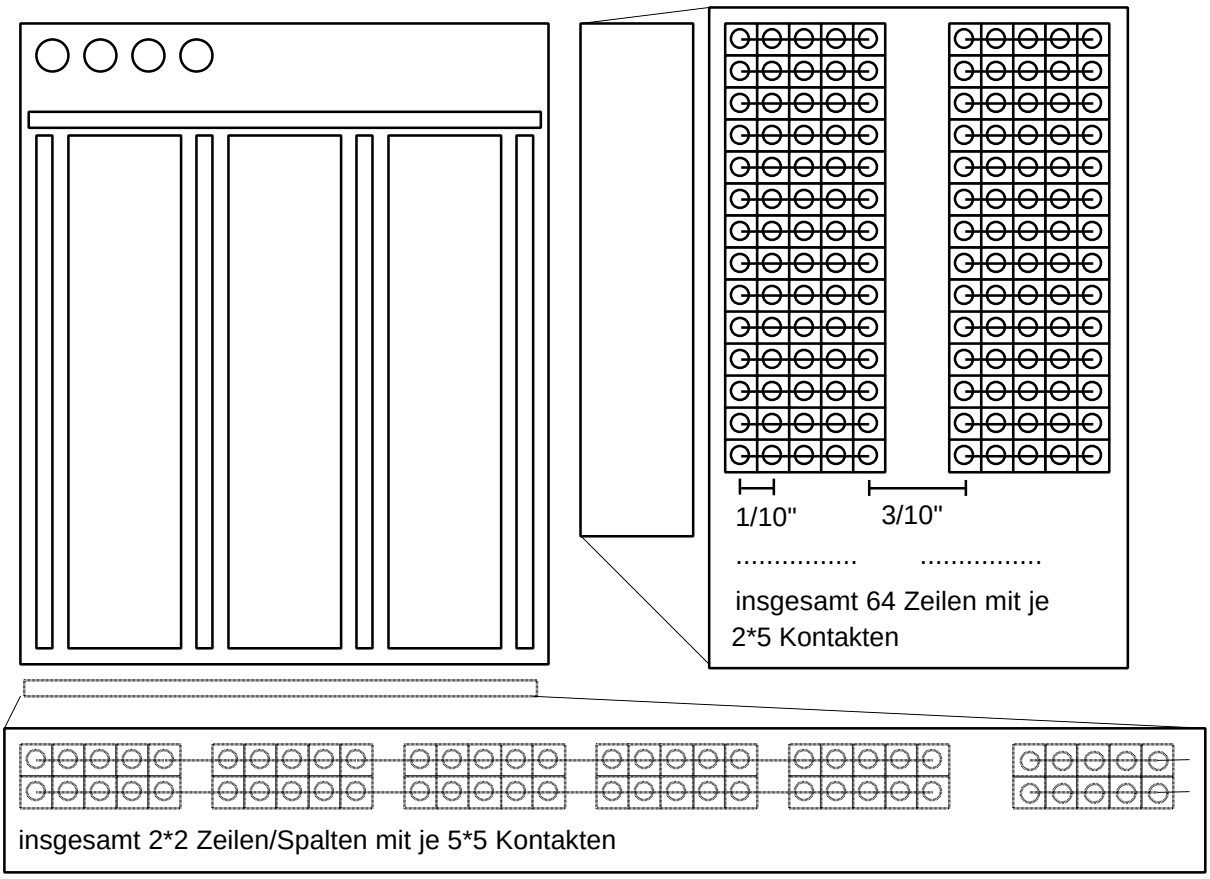
GOTO address
Verzweigung zur angegebenen Marke
GOSUB address
RETURN
Verzweigung zum Unterprogramm an der angegebenen Marke (maximal 16 GOSUBs in vier Ebenen sind möglich). Das Ende des Unterprogrammes wird mit RETURN bezeichnet.

NAP period
Wechsle für period in einen Stromsparmodus (abhängig von der Zeitdauer)
period Angabe der Zeitdauer (0..7; 2^{period} * 18 ms)
SLEEP seconds
Schalte für einige Zeit in den Stromsparmodus (1% des normalen Wertes)
seconds Sekunden (0..65535; Auflösung ca. 2,3s)
END

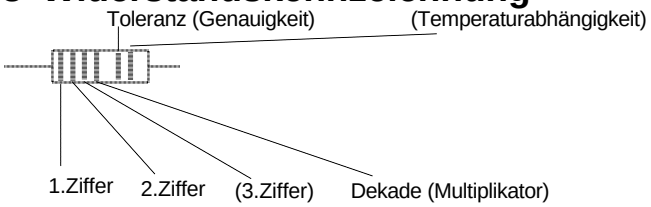
Schalte in den bestmöglichen Stromsparmodus, solange bis entweder die Versorgungsspannung aus- und eingeschaltet, oder der PC angeschlossen wird.

7 Steckbretter

Für den schnellen Aufbau vor allem von Prototypschaltungen sind die Steckbretter besonders geeignet. Die interne Verdrahtung unserer Steckbretter ist der folgenden Skizze zu entnehmen:



8 Widerstandskennzeichnung



Farbcodes für diese Ringe

Farbe	Ziffer	Dekade	Toleranz	Temperatur
schwarz	0 (außer 1.)	*1	-	-
braun	1	*10	± 1 %	-
rot	2	*100	± 2 %	50ppm/°C
orange	3	*1000	-	-
gelb	4	*10000	-	-
grün	5	*100000	± 0,5 %	-
blau	6	*1000000	-	-
violett	7	*10000000	-	-
grau	8	-	-	-

weiß	9	-	-	-
gold	-	*0.1	± 5 %	-
silber	-	*0.01	± 10 %	-

Bsp:

gelb	violett	braun	gold			470 Ω	5 %
braun	grün	schwarz	schwarz	braun	rot	150 Ω	1 %
braun	grün	braun	gold			150 Ω	5 %

Widerstände gibt es üblicherweise nur in bestimmten Größen den sogenannten E-Reihen:
E-6, E-12, E-24, E-48, E-96, dabei bedeutet E-n, daß pro Dekade n Werte existieren, die logarithmisch über die Dekade verteilt sind. Je genauer die Widerstände gefertigt werden, desto umfangreicher werden die E-Reihen.
z.B: Reihe E-24

1	1,5	2,2	3,3	4,7	6,8
1,1	1,6	2,4	3,6	5,1	7,5
1,2	1,8	2,7	3,9	5,6	8,2
1,3	2,0	3,0	4,3	6,2	9,1

dabei entspricht die erste Zeile der Reihe E6, die erste und dritte Zeile der Reihe E-12

9 Dezimale Vielfache und Teile

10^{18}	Exa	E	10^{-1}	Dezi	d
10^{15}	Peta	P	10^{-2}	Centi	c
10^{12}	Tera	T	10^{-3}	Milli	m
10^{9}	Giga	G	10^{-6}	Mikro	µ
10^{6}	Mega	M	10^{-9}	Nano	n
10^{3}	Kilo	k	10^{-12}	Piko	p
10^{2}	Hekto	h	10^{-15}	Femto	f
10^{1}	deka	da	10^{-18}	Atto	a